UiT Norges arktiske universitet

Fakultet for naturvitenskap og teknologi

Institutt for matematikk og statistikk

Kan bruk av programmering i matematikkundervisning fremme algoritmisk tenkning hos elever?

En mixed methods case-studie av 1T elever

Mathea Fennefoss Johnsgård

Masteroppgave i Lektor i realfag trinn 8-13, MAT-3907 juni 2023

Forord

Først og fremst rettes en stor takk til elevene som ikke bare har stilt opp som informanter i forskningsprosjektet, men som det har vært gøy å være sammen med hver time, og har vist engasjement og interesse til tross for at de har synes programmering har vært utfordrende.

Takk til veilederne mine Jonas Oskarsson og Hilja Lisa Huru for gode diskusjoner, positiv energi og tilliten dere har vist meg i prosessen.

Takk til mine romkamerater og beste venner som de anonymiserte elevene er oppkalt etter, som alltid stiller med kos, trøst og gode vitser. Takk til Tromsø som har sørget for 40 sammenhengende dager med regn i masterinnspurten og fantastiske konsentrasjonsforhold. Takk til CrossFit Nordaførr, Godtlevert matkasse, Brynjar Saus som har korrekturlest hele oppgaven, og mamma Birthe og pappa Stein som tar imot enhver eksistensiell krise.

Tromsø, 7. juni 2023
Mathea Fennefoss Johnsgård

Sammendrag

Høsten 2020 ble det med fagfornyelsen innført nye læreplaner, der det i matematikk settes økt fokus på utforskning og problemløsning, samt innføring av følgende kompetansemål i 1T:

«formulere og løse problemer ved hjelp av algoritmisk tenkning, ulike problemløsningsstrategier, digitale verktøy og programmering» (Kunnskapsdepartementet, 2020).

Hensikten med studien er å undersøke overføringsverdien mellom arbeid med programmering i matematikkundervisning og utvikling av matematiske ferdigheter og kompetanse. Basert på relevant teori, tidligere forskning og resultatene fra datainnsamlingen i denne studien er følgende problemstilling forsøkt besvart:

«Kan bruk av programmering i matematikkundervisning bidra til å fremme algoritmisk tenkning hos elever»

For å besvare problemstillingen ble det utviklet et forskningsdesign der en 1T-klasse utførte en prestasjonstest i forkant og etterkant av et 2 uker langt undervisningsopplegg med programmering i matematikk. Prestasjonstesten ble analysert og vurdert kvalitativt i henhold til et rammeverk for å vurdere nivå av elevers matematiske resonnementer. Oppgavene i testen var oppgaver som tilrettela for demonstrasjon av algoritmisk tenkning. I tillegg besvarte elevene et åpent og lukket spørreskjema, og 4 elever ble invitert til intervju i etterkant av undervisningsopplegget. Målet var å avdekke om elevene utviklet sine ferdigheter innen algoritmisk tenking som hadde overføringsverdi til andre områder innen matematikk.

Funnene i studien indikerer at elever kan utvikle sine abstraksjonsevne, matematiske resonnementer og problemløsningsferdigheter i matematikk gjennom arbeid med programmering. Funnene avdekker også faktorer som begrenser algoritmisk tenkning i problemløsningsprosessen. Disse faktorene er begrenset kreativitet, tid og terskelbegreper. Det vil være interessant å se om disse begrensende faktorene blir mindre signifikante etter hvert som læreplanen har vært i virksomhet og undervisningspraksisen stadig utvikles for å nå målene definert i LK20.

Innholdsfortegnelse

Figurliste

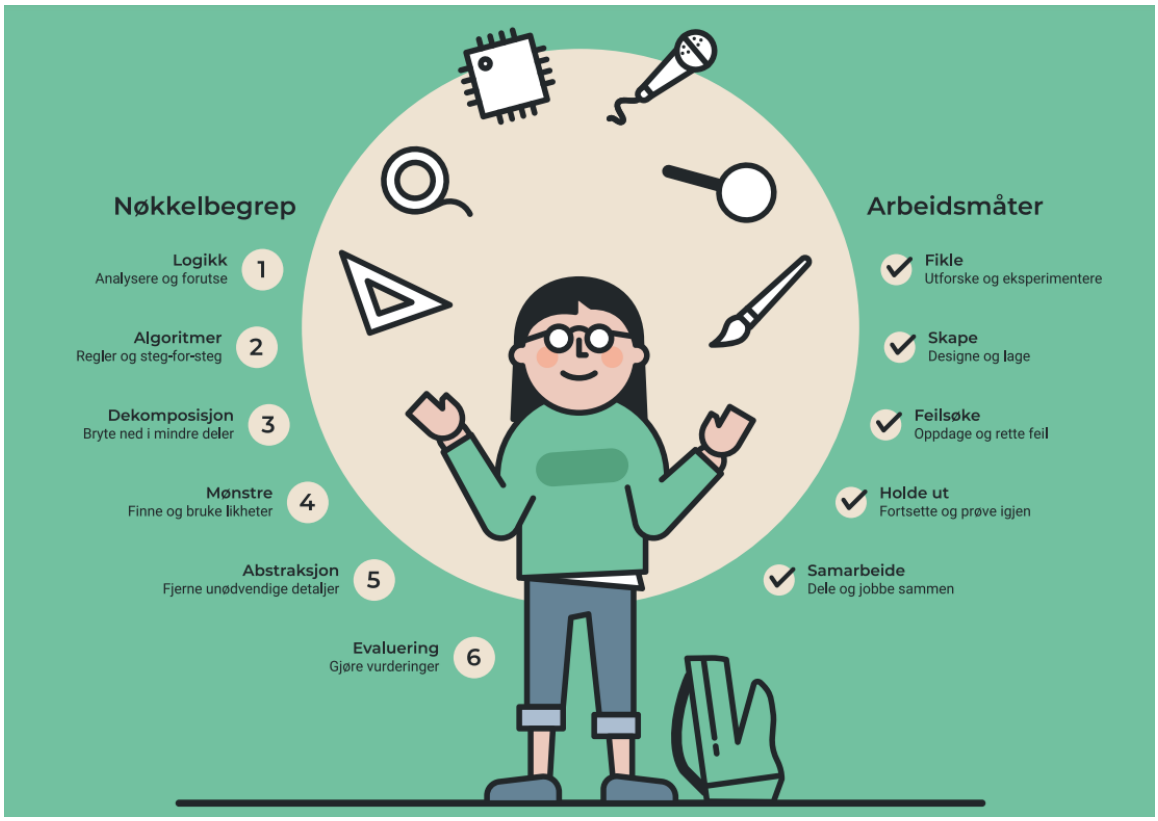
Tabelliste

1 Introduksjon

Med Kunnskapsløftet 2020 ble matematikkfaget renovert; det ble satt økt fokus på både problemløsning, resonnering og algoritmisk tenkning i læreplanen. Samtidig ble også programmering innført som et av kompetansemålene både i grunnskole og videregående opplæring (Kunnskapsdepartementet, 2019). I fellesfaget matematikk 1T står følgende læreplanmål:

«Målet for opplæringen er at eleven skal kunne formulere og løse problemer ved hjelp av algoritmisk tenkning, ulike problemløsningsstrategier, digitale verktøy og programmering» (Kunnskapsdepartementet, 2019).

Dette kompetansemålet danner grunnlag for flere spørsmål, for eksempel om hva algoritmisk tenkning innebærer og på hvilken måte det kan knyttes til programmering. Utdanningsdirektoratet har utarbeidet en definisjon av algoritmisk tenkning, med utgangspunkt i Barefoot Computing's definisjon av det engelske begrepet *computational thinking.*



Figur 1: Algoritmisk tenkning (Utdanningsdirektoratet, 2019) utviklet på bakgrunn av definisjon av Computational thinking (publisert med åpen lisens fra Barefoot Computing).

Slik UDIR definerer begrepet *algoritmisk tenkning* kan det tolkes som en systematisk tilnærming til problemløsning. Illustrasjonen i figur 1 om algoritmisk tenkning nevnes det ulike evner som er nyttige for å løse matematiske problemer, og ulike arbeidsmetoder som skal stimulere til *algoritmisk tenkning.* Det er en klar sammenheng mellom UDIRs definisjon av arbeidsmetodene innen algoritmisk tenkning og arbeid med programmering som vil bli diskutert senere i oppgaven. I artikkelen til Grover og Pea (2013) trekkes det frem mange likhetstrekk mellom algoritmisk tenkning og informatikk. Eksempler på disse er mønstergjenkjenning, abstrahering og generalisering, systematisk prosessering av informasjon, dekomponering av problemet, systematisk feilsøking, symbolbruk og representasjon.

Videre argumenteres det for at programmering ikke bare er en grunnleggende ferdighet i informatikk, men også et nøkkelredskap for å fasilitere algoritmisk tenkning. Det er likevel diskusjoner rundt overføringsverdien mellom programmering og matematisk læring i fagmiljøet på bakgrunn av sprikende empiriske forskningsresultater, og manglende konsensus om undervisningsmetoder. Er det slik at programmering kan bidra til læring i matematikk eller er matematisk kompetanse en forutsetning for å lære programmering?

I den nye læreplanen foreligger det en forutinntatthet om at programmering vil ha en direkte overføringsverdi til matematisk læring, problemløsningsstrategi og algoritmisk tenkning. Selv om programmering er anerkjent for å bistå til de kognitive prosessene som er involvert i algoritmisk tenkning (Grover & Pea, 2013), debatteres det på hvilket pedagogisk grunnlag programmering er innført som en del av matematikkfaget (Forsström & Kaufmann, 2018). Innføringen av programmering i læreplanen medfører at lærere må renovere sin undervisningspraksis, uten at det finnes konsensus om hvordan programmering på best mulig måte skal implementeres i matematikkfaget.

Forsström og Kaufmanns (2018) artikkel er basert på en litteraturstudie hvor 15 utvalgte (av 917 relevante) artikler ble analysert, med mål om å finne forskningsbaserte argumenter for innføringen av programmering i matematikkfagene i flere europeiske land. Artiklene ble sortert inn i 3 hovedgrupper basert på tema; effekt på elevers motivasjon til å lære matematikk, elevers prestasjon i matematikk og samspill mellom elevene og lærerens endrede rolle. Studien konkluderte med at for noen grupper, noen undervisningsmetoder og noen matematiske tema, medførte inkludering av programmering i matematikkundervisningen en positiv effekt på elevenes motivasjon i faget og elevenes prestasjoner i matematikk.

På den andre siden ble fransk studie utført på 109 4.- og 5. klasser ønsket å se på effekten på læring i matematikk ved å sammenligne to grupper; én som brukte programmering i opplæringen, én som fikk tradisjonell matematikkundervisning. Studien er basert på datainnsamling fra 7 tester; én test før iverksettelse av prosjektet, deretter en test både før og etter hvert av de 3 matematiske temaene som inngikk i datainnsamlingen. Programmeringsgruppen bestod av 1519 individer, mens kontrollgruppen bestod av 953 individer. Studien konkluderte med at elevgruppen som hadde brukt programmering i undervisningen hadde en liten, men negativ utvikling i matematisk læring sammenlignet med tradisjonell matematikkundervisning (Laurent et al., 2022).

En annen studie utført på 93 barneskoleelever konkluderte derimot med det motsatte; det ble observert en signifikant forbedring i elevenes matematiske kompetanse ved inkludering av programmering og robotikk i matematikkundervisningen (Sáez-López et al., 2019).

Det er med andre ord sprikende resultater fra empirisk forskning på hvorvidt bruk av programmering fører til læring innen matematikk. Dette kan mulig sees i sammenheng med funnene i studien til Forsström og Kaufmann (2018) , som understeker et behov for konsensus om pedagogisk tilnærming til hvordan programmering undervises og læres i matematikkundervisningen.

Det nordiske forskningsprosjektet MASCOT begynte i 2021 arbeidet med å samle kunnskap om hvordan det undervises i algoritmisk tenkning i skolen. Prosjektet har som mål å utvikle nye undervisnings- og vurderingspraksiser basert på datainnsamling fra de nordiske samarbeidsskolene i prosjektet (Lavoll, 2021).

En tilnærming til å lære programmering er PRIMM-metoden (Predict, Run, Investigate, Modify, Make). Denne tilnærmingen til problemløsningen er forenelig definisjonen av algoritmisk tenkning som diskutert tidligere i denne oppgaven. Stegene i PRIMM kan sees på som en problemløsninsstrategi som inviterer til algoritmisk tenkning. *Predict* stimulerer logikken, hvor det handler om å analysere informasjon og forutse hva koden i et program gjør, før man i *run* kjører koden. Deretter kommer *investigate*, som innebærer å dekomponere problemet, anvende algoritmer, evaluere koden. Deretter kommer steget *modify*, som handler om å utbedre koden basert på feilsøket gjort i de tidligere steg. *Make* er steget hvor en ny kode lages, slik at abstraheringene som er blitt gjort og mønstrene som er blitt avdekket i utforskningen kan benyttes til å løse en annen programmeringsoppgave.

Til sammen har disse stegene involvert elevene i alle arbeidsmåtene UDIR nevner som aktiviteter som stimulerer til algoritmisk tenkning; utforske og eksperimentere gjennom å forutse og tolke koden for så å kjøre den, oppdage og rette feil gjennom feilsøking, designe og lage gjennom å utbedre og tilpasse koden, utholdenhet gjennom å prøve og feile, og samarbeid med både lærer og medelever hvor deling av kode også er en del av prosessen.

En studie publisert i 2019 konkluderte med at elever som hadde blitt undervist i programmering ved hjelp av PRIMM-metoden presterte betydelig bedre i en test etter 10 undervisningsøkter sammenlignet med kontrollgruppen som ikke hadde lært programmering ved hjelp av PRIMM-metoden (Sentance et al., 2019).

På bakgrunn av funnene fra forskning på PRIMM som undervisningsmetode vil denne studien benytte PRIMM-metoden for å inkludere programmering i matematikkundervisningen. Valg av en undervisningsmetode som brukes konsekvent i studien er for å øke studiens reliabilitet. Reliabiliteten til en studie avhenger av hvor reproduserbare resultatene fra forskningen er, og sier noe om kvaliteten på studien (Gleiss & Sæther, 2022) og vil bli ytterligere diskutert i metodedelen av oppgaven.

1.1 Problemstilling

Programmering som aktivitet krever algoritmisk tenkning (Grover & Pea, 2013), og PRIMM-metoden inviterer til læringsaktiviteter som legger til rette for algoritmisk tenkning (Utdanningsdirektoratet, 2019). Denne studien tar sikte på å si noe om overføringsverdien av problemløsningsstrategier, dekomponering, mønstergjenkjenning, kreativitet og generalisering brukt i programmering til andre deler av matematikken.

På bakgrunn av disse betraktningene ble følgende forskningsspørsmål definert:

«Kan bruk av programmering i matematikkundervisning bidra til å fremme algoritmisk tenking hos elever?»

For å besvare problemstillingen og det overordnede forskningsspørsmålet, er flere støttende forskningsspørsmål blitt definert. Hvert av de underordnede forskningsspørsmålene er ment å være operasjonaliseringer som hjelper med å besvare hovedforskningsspørsmålet.

Operasjonalisering innebærer konkretisering av begreper som ikke er direkte observerbare, som for eksempel algoritmisk tenkning (Gleiss & Sæther, 2022). Ved operasjonalisering av slike begreper defineres tydelige kriterier eller observerbare kjennetegn på det teoretiske begrepet som ikke kan direkte måles eller observeres (Gleiss & Sæther, 2022).

Den teoretiske forankringen for operasjonaliseringene som gjøres i oppgaven vil diskuteres nærmere i metodedelen av oppgaven.

- På hvilken måte kan programmering brukes som et verktøy til å fasilitere algoritmisk tenkning hos elever?
- Hvilke faktorer kan hindre algoritmisk tenkning hos elever?
- Hvilke evner kan elever utvikle gjennom programmeringsarbeid som har overføringsverdi til matematisk kompetanse?

Denne masteren har som formål å undersøke hvorvidt bruk av programmering som et verktøy til problemløsning og algoritmisk tenkning i matematikkundervisningen kan bidra til matematisk læring og økt kompetanse i faget.

Denne masteroppgaven tar sikte på å si noe om hvorvidt inkludering av programmering i matematikkundervisningen er produktivt med hensyn på å fremme matematisk læring og nå kompetansemålet om algoritmisk tenkning. Kan problemløsningsstrategier, anvendelse av algoritmer, dekomponering av et problem, gjenkjenning av mønstre og generalisering og anvendelse av kode på nye problemer overføres til andre deler av matematikken?

1.2 Oppgavens struktur

Dette forskningsprosjektet vil benytte et casedesign, som betyr et studie av det spesifikke innenfor rammene av noen få enheter (Johannesen & Christoffersen, 2012). Forskningsdesignet er av typen mixed methods, hvor både kvalitative og kvantitative datainnsamlingsmetoder benyttes for å anskaffe et størst mulig datagrunnlag (Gleiss & Sæther, 2022).

Datainnsamlingen vil bli utført med en prestasjonstest i forkant og etterkant av et planlagt undervisningsopplegg på 2 uker. Deretter vil individene i studien ble invitert til å besvare et spørreskjema med forhåndsformulerte svaralternativer. Til slutt vil intervjuer benyttes til å følge opp interessante funn fra både prestasjonsmålingen og spørreskjemaet.

Teori og valg av metode vil redegjøres for i teori- og metodekapitlene (kap. 2 og 3). Funnene fra datainnsamlingen presenteres i analysen. Til slutt vil funn drøftes i henhold til relevant teori og tidligere forskning, som leder til en konklusjon.

2 Teori

For å kunne besvare problemstillingen

«Kan bruk av programmering i matematikkundervisning bidra til å fremme algoritmisk tenking hos elever?»

må relevante begreper som algoritmisk tenkning og læring i matematikk grundig redegjøres for, i tillegg til å belyse programmerings rolle i skolen og bakgrunn for implementering av programmering i matematikkfaget.

Deretter vil litteratur som beskriver sammenhengen mellom algoritmisk tenkning og programmering beskrives, og sees i sammenheng med tidligere forskning på effekten av programmering i matematikkundervisning.

Algoritmisk tenkning er assosiert med matematisk kreativitet, som er et begrep som må defineres og redegjøres for. Sammenhengen mellom algoritmisk tenkning og matematisk kreativitet vil også presenteres med relevant litteratur og tidligere forskning.

For å lage et forskningsdesign som kan benyttes for å besvare problemstillingen vil teorien om PRIMM som undervisningsmetode for å lære programmering bli brukt som utgangspunkt for utarbeidelse av undervisningsopplegg med programmering i matematikk.

I sammenheng med utarbeidelsen av et undervisningsopplegg med programmering som er ment å medføre læring i algoritmisk tenkning, må teorien om utforskende undervisning inkluderes.

Til slutt vil det redegjøres for teoretiske rammeverk som danner utgangspunkt for metodevalg og hvordan måle utbyttet av undervisningsopplegget, og et teoretisk rammeverk for vurdering av elevers matematiske resonnementer.

2.1 Algoritmisk tenkning

Det norske begrepet algoritmisk tenkning er oversatt fra det engelske computational thinking (Utdanningsdirektoratet, 2019). Aho (2012) gir en kort og presis definisjon av et komplekst og sammensatt begrep med mange ulike definisjoner; computational thinking kan defineres som tankeprosessen som kreves for å formulere et problem slik at løsningene til problemet kan presenteres som anvendbare algoritmer eller stegvise instruksjoner.

Slik utdanningsdirektoratet oversetter *computational thinking* til det norske begrepet *algoritmisk tenkning*, er det norske begrepet helt i samsvar med definisjonen av *computational thinking*, da det beskrives som en systematisk tilnærming til problemløsning med fokus på mønstergjenkjenning, kreativitet og generalisering (Utdanningsdirektoratet, 2019).

Gjøvik og Torkildsen (2019) problematiserer at begrepene har litt ulik betydning på ulike språk, hvor det norske begrepet *algoritmisk tenkning* fort danner assosiasjoner til betydningen av *algoritmer* i matematikken, som handler om å anvende regler og fremgangsmåter for å komme frem til løsningen. På grunn av begrepets ordlyd er det ikke perfekt egnet til å omsette betydningen av *computational thinking*, på tross av at selve definisjonen av begrepet samsvarer godt.

I Sverige bruker de begrepet datalogiskt tänkande som oversettelse for computational thinking, et begrep som mer intuitivt kan forstås som en tenkemåte lik en datamaskins måte å løse problemer på, ved å analysere tilgjengelig informasjon og bruke den til å lage modeller eller algoritmer som kan brukes til å generere nye løsninger (Kilhamn et al., 2021). Samtidig handler ikke computational thinking om å tenke som en datamaskin, datamaskiner tenker jo ikke, det er det mennesker som gjør (Wing, 2006). Med andre ord handler computational thinking om en tankeprosess som både er systematisk og kreativ, og resulterer i algoritmer som kan brukes av datamaskiner for å generere løsninger eller behandle datamateriale og oppgaver som blir for store for mennesker (Wing, 2006). Denne oppgaven vil dog benytte begrepet algoritmisk tenkning, ettersom det er det norske begrepet som brukes i læreplanen, og refererer til samme betydning som computational thinking selv om ordlyden er ulik.

Wing poengterer også at algoritmisk tenkning er en fundamental ferdighet for alle, ikke bare informatikere og andre tekniske yrkesgrupper (Wing, 2006). Algoritmisk tenkning har mange likheter med å programmere, men innebærer likevel mer enn å programmere noe på en datamaskin (Wing, 2006; Grover & Pea, 2013). *Algoritmisk tenkning* innebærer å systematisk

løse problemer ved å identifisere mål, bryte en kompleks problemstilling ned til delproblemer, evaluere både løsning og prosess (Wing, 2006; Utdanningsdirektoratet, 2019; Grover & Pea, 2013).

2.2 Problemløsning og algoritmisk tenkning

Utdanningsdirektoratet (2019) beskriver *algoritmisk tenkning* som en systematisk tilnærming til problemløsning. Problemløsning er et hverdagsbegrep med en selvforklarende betydning, men kan også defineres i matematisk kontekst.

George Polya (1945) har definert 4 punkter for hva matematisk problemløsning innebærer:

1) Forstå problemet
2) Utarbeid en plan
3) Iverksett planen
4) Se tilbake

Det første punktet handler om å forstå hva oppgavebeskrivelsen egentlig ber om. Polya stiller følgende spørsmål; Hvilken informasjon har vi tilgjengelig? Hvilke betingelser gjelder? Har du fått tilstrekkelig informasjon til å finne løsningen til problemet? Polya foreslår også å tegne en skisse for å fremstille informasjonen en har fått, og for å danne et oversiktlig bilde av problemet, betingelsene og den tilgjengelige informasjonen.

Steg 2 handler om å legge en plan for hvordan finne løsningen til problemet. Strategier som kan involveres er å lete etter et mønster, tenke over om en har jobbet med et lignende problem tidligere og om en kan bruke resultatet eller metoden til å ta fatt på problemet. Kanskje en kjenner til et teorem eller en modell en kan anvende på problemet, eller kanskje en klarer å løse deler av problemet?

Når en når det 3. punktet må en bruke sine matematiske ferdigheter for å jobbe seg gjennom planen, og sjekke at hvert steg er korrekt utført. Polya skriver at en kan sjekke løsningen ved å bevise at løsningen er korrekt.

Polyas fjerde steg handler om å evaluere strategi, løsning og argumentasjon brukt for å komme frem til løsningen. Kunne det vært gjort annerledes, og kan du benytte den samme metoden til å løse andre problemer?

Det definisjonen av algoritmisk tenkning innebærer har flere fellestrekk med Polyas 4 steg for problemløsnining. Både Polyas oppskrift og algoritmisk tenkning begynner med å analysere

og forstå problemet (2019). I steget der planen skal legges, nevnes mønstergjenkjenning, dekomponering, anvendelse av regler som mulige strategier, som også inngår i algoritmisk tenkemåte. Det å se tilbake og evaluere arbeidet er definert både i Polyas problemløsningssteg og i definisjonen av algoritmisk tenkning (redegjort for i kapittel 2.1).

2.3 Programmering og algoritmisk tenkning

I læreplanen for faget matematikk T nevnes begrepene algoritmisk tenkning og programmering eksplisitt i samme kompetansemål (Kunnskapsdepartementet, 2020):

- formulere og løse problemer ved hjelp av algoritmisk tenkning, ulike problemløsingsstrategier, digitale verktøy og programmering

Wing (2006) hevder at algoritmisk tenkning er en fundamental ferdighet som kreves av alle mennesker for å kunne navigere seg i en moderne verden og en forutsetning for menneskers analytiske evne. Samtidig påpeker Grover og Pea (2013) at programmering også er blitt en påkrevd ferdighet for å kunne ta del i samfunnet, i en stadig mer digitalisert verden.

Programmering handler om å skape en fremgangsmetode for å løse et problem steg for steg. Programmering innebærer å formulere et sett med instrukser slik at en datamaskin kan benytte tilgjengelig informasjon til å utføre bestemte oppgaver eller løse problemer (Forsström & Kaufmann, 2018). Til sammen utgjør et sett med instrukser et *program,* som bestemmer hvordan datamaskinen fungerer når vi kjører programmet (Rossen, 2022).

Koding er den tekniske formuleringen av disse instruksene, og tar i bruk bestemte programmeringsspråk (Gjøvik & Torkildsen, 2019). Koding er selve aktiviteten som utføres når instruksene til datamaskinen formuleres, og det tas da i bruk ulike programmeringsspråk. Et programmeringsspråk er et språk med egne grammatikkdeler og syntaks forbundet med spesifikke kommandoer som kan tolkes av datamaskiner ("Programmeringsspråk," 2019). Det finnes mange ulike typer programmeringsspråk, som Python, C, C+ og Java. I denne studien vil programmeringsspråket Python brukes i undervisningsopplegget som inngår i en del av datainnsamlingen.

Mens *koding* innebærer å formulere instruksene som inngår i programmet, krever programmering flere kognitive prosesser. Utarbeidelsen av et program krever abstraksjonsevne på flere nivåer (Wing, 2006). **Abstraksjon** handler om å oppdage mønster eller sammenhenger i noe konkret og fysisk, som et datamateriale (Gjøvik & Torkildsen,

2019). Videre må disse sammenhengene og strukturene formuleres korrekt og presist, slik at det dannes visse kriterier som gjør det mulig å sortere og skille ulike typer data fra datamaterialet. På denne måten får datamaskinen beskjed om hvordan den skal differensiere ulike typer data, og hvordan behandle dem.

Abstraksjonsevne er nøkkelen til å lykkes med å håndtere komplekse problemer ved å bryte dem ned til mindre, mer angripelige delproblemer (Wing, 2011). **Dekomponering** er også en forutsetning for å lykkes med å formulere algoritmer en datamaskin kan anvende da en datamaskin må få presise instrukser for å lykkes med å utføre en oppgave (Gjøvik & Torkildsen, 2019).

Algoritmer innebærer et sett med steg for steg regler eller prosedyrer som genererer et ønsket resultat. For å kunne lage et sett algoritmer som kan appliseres på et datasett hører det til prosessen at vi må identifisere likheter mellom observasjonene i datasettet, samt beskrive **mønstre og sammenhenger**. Mønstere og sammenhenger mellom spesifikke observasjoner kan så videreutvikles til et generelt uttrykk. **Generalisering** gjør det mulig å formulere et sett med instrukser, som til sammen utgjør en algoritme.

I programmeringsaktivitet innebærer dette at en utforsker ulike metoder, tester ut programmet underveis, og **evaluere** hvorvidt programmet ga ønsket resultat ved kjøring (Gjøvik & Torkildsen, 2019).

Programmering er ikke bare en grunnleggende ferdighet innen informatikk, men også en aktivitet som inviterer til den kognitive virksomheten som inngår i algoritmisk tenkning (Grover & Pea, 2013). De ulike tankeprosessene og arbeidsmetodene som benyttes i utarbeidelsen av et program, som abstraksjon, dekomponering, mønstergjenkjenning, generalisering, algoritmeutvikling og evaluering, kan direkte knyttes til definisjonen av algoritmisk tenkning. Begrepet *algoritmisk tenkning* definert i delkapittel 2.1 kan defineres som tankeprosessen involvert i å formulere problemer slik at løsningene til problemene kan presenteres som stegvise prosedyrer og algoritmer (Aho, 2012). I tankeprosessen inngår en problemløsningsmetode som handler om å systematisk identifisere mål, dekomponere problemstillingen, og evaluere både løsning og prosess (Wing, 2006; Utdanningsdirektoratet, 2019; Grover & Pea, 2013).

2.4 Programmering i skolen

Et av læreplanmålene for matematikk T på videregående skole i Norge som sier at elever skal formulere og løse problemer ved hjelp av algoritmisk tenkning, ulike problemløsingsstrategier, digitale verktøy og programmering (Kunnskapsdepartementet, 2020). Som argumentert for i kapittel 2.1 og 2.2, er det en klar sammenheng mellom algoritmisk tenkning og programmering, der programmering inviterer til bruk av tankeprosesser og arbeidsmåter som inngår i algoritmisk tenkning. Programmering kan rett og slett sees på som et pedagogisk virkemiddel for å fremme algoritmisk tenkning og utvikling av matematisk kompetanse. Hva matematisk kompetanse innebærer vil diskuteres i delkapittel 2.7.

Den overordnede delen av læreplanen har som formål å utdype grunnskolens overordnede verdier, prinsipper og samfunnsmandat. I den den overordnede delen av læreplanen står det at formålet med opplæringen i grunnskolen er at

«Elevane og lærlingane skal utvikle kunnskap, dugleik og holdningar for å kunne meistre liva sine og for å kunne delta i arbeid og fellesskap i samfunnet. Dei skal få utfalde skaparglede, engasjement og utforskartrong» (Kunnskapsdepartementet, 2017).

Videre utdypes det hva det innebærer å rustes med de kunnskaper og ferdigheter som kreves for å mestre ferdsel i eget liv, arbeids- og samfunnsliv. Kritisk tenkning trekkes frem som e av de overordnede formålene med opplæringen, som innebærer at elevene skal bli i stand til å møte praktiske problemer og ulike informasjonskilder med fornuft og refleksjon (Kunnskapsdepartementet, 2017).

For å være i stand til å møte utfordringer i både kjente og ukjente situasjoner, kreves 5 grunnleggende ferdigheter som defineres i læreplanverket; lesing, skriving, regning, muntlige kommunikasjonsferdigheter og digitale ferdigheter (Kunnskapsdepartementet, 2017). Ikke bare er dette essensielle ferdigheter for å tilegne seg kunnskap og forståelse i utdanningssammenheng, men påkrevde kunnskaper for å kunne ta del i samfunnet, samt orientere seg selvstendig og kritisk i møte med ulike informasjons- og nyhetskilder (Kunnskapsdepartementet, 2017).

I en artikkel publisert av Jeanette Wing (2011) skriver hun at algoritmisk tenkning er det 21. århundrets *literacy*. I 2005 foreslo UNECO følgende definisjon av literacy:

«Literacy is the abilty to identify, understand, interpret, create, communicate and compute, using printed and written materials associated with varying contexts. Literacy involves a continuum of learning in enabling individuals to achieve their goals, to develop their knowledge and potential, and to participate fully in their community and society» (UNESCO, 2005).

Literacy ikke bare handler om lese- og skrivekyndighet, men å kunne uttrykke seg, orientere seg og delta ubegrenset i samfunnet. Det handler om tilgjengeliggjøring av informasjonen og kunnskapen som finnes i verden, og å være i stand til å vurdere den selvstendig og kritisk.

Literacy fanger på mange måter målene formulert i den overordnede delen av læreplanen, og kan sees på som en forutsetning for kunne ferdes i verden som et fritt og velfungerende medmenneske.

Wing argumenterer for at *algoritmisk tenkning* er vår tids literacy, da det er det som gjør et menneske i stand til å anvende og orientere seg i de tilgjengelige digitale hjelpemidlene som preger både hverdags- og arbeidsliv(Wing, 2011). Wing (2006) hevder at *algoritmisk tenkning* ikke kun er relevant for mennesker som er sysselsatt innenfor realfag, men en forutsetning for analytisk og kritisk tenking, som i vår tid er en påkrevd ferdighet på lik linje med lesing, skriving, regning og kommunikasjonsferdigheter. I den digitale tiden vi lever i må alle kunne nyttiggjøre seg av tilgjengelig teknologi. Det innebærer å være i stand til å programmere en PC (i forstand av å gi den instrukser for å få utført et arbeid, ikke nødvendigvis kode et program), gjenkjenne hvilke metoder som kan brukes til å løse ulike problemer, og være i stand til å gå komplekse utfordringer i møte ved å dekomponere dem til mindre delproblemer (Wing, 2011).

Programmering er anerkjent å fasilitere algoritmisk tenking (Grover & Pea, 2013), og har de siste årene blitt implementert i skolen på ulike vis i ulike land (Kilhamn et al., 2021). I Norge er programmering implementert som en del av matematikkfaget, det er tydelig overlapp mellom kjerneelementene i læreplanen og de arbeidsmetodene og tankeprosessene som anvendes i programmeringsarbeid. Kjerneelementene i læreplanen oppsummerer de overordnede læremålene og ferdighetene elevene skal utvikle gjennom opplæring i matematikkfaget. Kjerneelementene i faget 1T er som følger (Kunnskapsdepartementet, 2020):

- Utforsking og problemløsing
- Modellering og anvendelse

- Resonnering og argumentasjon
- Representasjon og kommunikasjon
- Abstraksjon og generalisering
- Matematiske kunnskapsområder

I kjerneelementet *utforsking og problemløsning* nevnes algoritmisk tenkning eksplisitt som en nøkkelferdighet for systematisk tilnærming til problemløsning og avdekke mønstre og sammenhenger (Kunnskapsdepartementet, 2020).

Modellering og anvendelse handler om å være i stand til å bruke matematikk til å forklare fenomener, trender og utvikling i hverdags, arbeids- og samfunnsliv, samt vurdere gyldighet og begrensninger av modeller som skal fremstille virkeligheten (Kunnskapsdepartementet, 2020). Dette kjerneelementet kan knyttes til *literacy*, der blant annet Wing argumenterer for at algoritmisk tenkning er vår tids *literacy* og en nødvendighet for å kunne kritisk vurdere presentasjoner og fremstillinger av verden og nyhetsbildet (Kunnskapsdepartementet, 2020). Denne måten å evaluere en modell på, ved kritisk tenkning og feilsøking, er også et av de definerte nøkkelbegrepene i algoritmisk tenkning (Utdanningsdirektoratet, 2019).

Kjerneelementet *representasjon og kommunikasjon* kan også knyttes til feilsøking og evalueringskomponenten av algoritmisk tenkning, da det handler om å være i stand til å vurdere gyldigheten av matematiske resultater ved å følge resonnementer og formulere matematiske bevis (Kunnskapsdepartementet, 2020). For å være i stand til det kreves evnen til å *abstrahere og generalisere* spesifikke observasjoner, som er en del av algoritmisk tankeprosess (Gjøvik & Torkildsen, 2019). I læreplanen presiseres også *abstraksjon og generalisering* som et kjerneelement i seg selv, og er som tidligere nevnt en essensiell ferdighet for algoritmisk tenkning og programmering (Gjøvik & Torkildsen, 2019; Grover & Pea, 2013; Wing, 2011; Wing, 2006). Det hører til dette også med at elevene må være i stand til å formulere sammenhenger og mønstre med korrekt matematisk notasjon. Dette inngår som en del av abstrahering og generalisering, men er også definert som et eget kjerneelement, *representasjon og kommunikasjon.* Dette kjerneelementet jobbes kontinuerlig med når elevene bedriver programmeringsarbeid, der en datamaskin er sensitiv på notasjon og syntaks, og den gir feilmeldinger med en gang noe ikke er korrekt eller presist.

Det siste kjerneelemente *matematiske kunnskapsområder* handler om selve faginnholdet (Kunnskapsdepartementet, 2020), altså ulike matematiske temaer som inngår i læreplanen, som for eksempel funksjoner og algebra. I den norske læreplanen er ikke programmering

knyttet til noe spesielt matematisk tema (Kunnskapsdepartementet, 2020), og kan dermed inkorporeres i alle deler av pensum.

Oppsummert foreligger det stort potensiale for utvikling av elevers algoritmiske tenkning gjennom arbeid med programmering. I Norden er det enighet om at implementering av algoritmisk tenkning og programmering i skolen har potensiale til å fremme elevers problemløsningsevne, logisk tenking og digital kompetanse (Bocconi et al., 2018). Til tross for det er det likevel mangel på konsensus om hvordan programmering og trening i algoritmisk tenkning skal implementeres og undervisningsmetoder som er best egnet (Bocconi et al., 2018; Forsström & Kaufmann, 2018).

2.5 Tidligere forskning om effekten av programmering i matematikkundervisning

Artikkelen *Engendering Problem Solving Skills and Mathematical Knowledge via Programming* publisert i 2017 tar sikte på å besvare hvorvidt programmering kan brukes til å fremme matematisk læring og problemløsningsferdigheter. Artikkelen er basert på et case-studie der 95 elever fra 5 ulike barneskoler deltok i et undervisningsopplegg med blokkprogrammering, med en prestasjonstest før og etter undervisningsopplegget der kreativitet, problemløsningsferdigheter, samt bruk av algoritmer og notasjon . Resultatene viste at gjennomsnittsskåren hadde økt fra 29% til 52%, som er en signifikant forbedring (Husain et al., 2017).

I tillegg rapporterer artikkelen om at elevene ble bedre til å analysere og forutse utfallet av et program, og argumenterer for at programmering kan bidra til å gjøre elever i stand til å bryte ned komplekse problemer og utvikle matematisk kunnskap (Husain et al., 2017).

Artikkelen løfter også frem at det i den moderne skole forventes mer av elevene enn at de kan utføre beregninger og demonstrere prosesser, de skal også være i stand til å anvende sin matematiske kunnskap i nye situasjoner, som også krever systematiske problemløsningsstrategier, kreativitet og nytenkning (Husain et al., 2017). Artikkelen argumenterer for at programmering kan være et verdifullt verktøy å inkludere i matematikkundervisningen for å utvikle nettopp disse ferdighetene gjennom praktisk arbeid med faget (Husain et al., 2017).

I tillegg rapporterer artikkelen om at elevene som deltok i studien viste økt interesse og engasjement for både programmering og matematikk (Husain et al., 2017), som kan bidra til økt motivasjon for læring (utdypes i delkapittel 2.11

Kaufmann og Stenseth har også publisert en artikkel der bruken av programmering i matematikkundervisning undersøkes. Resultatene fra undersøkelsen viste at elevene som deltok i studien viste fremgang i hvordan elevene bygde opp argumenter og demonstrerer abstraksjonsferdigheter da de skulle løse et matematisk problem ved hjelp av programmering (Kaufmann & Stenseth, 2021). Kaufmann og Stenseth observerte også at elevene var svært engasjerte i oppgaven de arbeidet med, og opplevde økt motivasjon og interesse for både matematikk og programmering (Kaufmann & Stenseth, 2021).

En annen observasjon de gjorde var når elevene benyttet «prøve og feile»-metoden mens de arbeidet med oppgaven, forsvant den matematiske diskusjonen, fordi terskelen for å prøve og feile senkes ved bruk av digitale hjelpemidler og programmering (Kaufmann & Stenseth, 2021). Når det ikke koster noe ekstra arbeid å bare prøve seg frem, oppleves det lettere å bare teste ulike løsninger i stedet for å bruke tid på å analysere, forutse, lete etter mønster og matematisere observasjoner. Sa forsvinner også den matematiske læringen i bruk av programmering (Kaufmann & Stenseth, 2021).

I konklusjonen stiller de spørsmål ved hvorvidt oppgaven elevene fikk var designet slik at de gjennom programmeringsaktivitet forbedret sin matematikkforståelse, eller om matematikk er virkemiddelet elevene bruker for å løse programmeringsoppgaven (Kaufmann & Stenseth, 2021). De besvarer spørsmålet med at begge deler sannsynligvis stemmer, og advarer mot å anta at det er en naturlig sammenheng mellom programmering og problemløsningsferdigheter, da elevenes valg av «prøv og feile» strategi forhindret dem i å løse oppgaven de ble tildelt i forskningsprosjektet (Kaufmann & Stenseth, 2021). Kaufmann og Stenseth understreker at det krever kompetanse og konsensus om undervisningsmetode for programmering i matematikk for optimalt læringsutbytte.

På den andre siden ble fransk studie utført på 109 4.- og 5. klasser ønsket å se på effekten på læring i matematikk ved å sammenligne to grupper; én som brukte programmering i opplæringen, én som fikk tradisjonell matematikkundervisning. Studien er basert på datainnsamling fra 7 tester; én test før iverksettelse av prosjektet, deretter en test både før og etter hvert av de 3 matematiske temaene som inngikk i datainnsamlingen. Programmeringsgruppen bestod av 1519 individer, mens kontrollgruppen bestod av 953 individer. Studien konkluderte med at elevgruppen som hadde brukt programmering i

undervisningen hadde en liten, men negativ utvikling i matematisk læring sammenlignet med tradisjonell matematikkundervisning (Laurent et al., 2022)

En annen studie utført på 93 barneskoleelever konkluderte derimot med det motsatte; det ble observert en signifikant forbedring i elevenes matematiske kompetanse ved inkludering av programmering og robotikk i matematikkundervisningen (Sáez-López et al., 2019).

Forsström og Kaufmanns artikkel er basert på en litteraturstudie hvor 15 utvalgte (av 917 relevante) artikler ble analysert, med mål om å finne forskningsbaserte argumenter for innføringen av programmering i matematikkfagene i flere europeiske land. Artiklene ble sortert inn i 3 hovedgrupper basert på tema; effekt på elevers motivasjon til å lære matematikk, elevers prestasjon i matematikk og samspill mellom elevene og lærerens endrede rolle. Studien konkluderte med at for noen grupper, noen undervisningsmetoder og noen matematiske tema, medførte inkludering av programmering i matematikkundervisningen en positiv effekt på elevenes motivasjon i faget og elevenes prestasjoner i matematikk (Forsström & Kaufmann, 2018).

De sprikende forskningsresultatene fra ulike empiriske studier om hvorvidt programmering medfører matematisk læring, understreker Forsstrømm og Kaufmanns poeng om at det er behov for konsensus om hvordan programmering skal implementeres i undervisningen (Forsström & Kaufmann, 2018).

2.6 PRIMM som undervisningsmetode i programmering

PRIMM er en undervisningsmodell utviklet a Sue Sentance som danner et rammeverk for hvordan programmeringsøkter kan struktureres og inkluderes i undervisningen. PRIMM står for *predict, run, modify og make (Sentance & Waite, 2017).*

PRIMM kan sees på som en stegvis problemløsningsstrategi, som skal hjelpe elever til å utvikle en tenkemåte og arbeidsmetode for å løse problemer ved hjelp av programmering (Sentance et al., 2019). De ulike stegene gjennomgås nedenfor:

Predict (forutse) handler om at elevene først blir presentert et eksempel på et program, og blir bedt om å analysere og tolke koden for å kunne beskrive hva koden gjør, og hvilke resultater programmet gir når det kjøres (Sentance et al., 2019).

Run (kjør) er steget der elevene selv kjører eksempelprogrammet, og får bekreftet eller avkreftet om det de forutså stemte. I dette steget kan elevene gjøre observasjoner av at

programmet gjorde noe annet enn de forutså, som er et godt utgangspunkt for å diskutere observasjonene i fellesskap eller i mindre grupper.

Investigate (utforske) gir elevene tid og rom til å ta for seg programmet linje for linje, og avdekke eventuelle feil eller notere seg forslag til hvordan koden kan forbedres. Elevene kan også få veiledende oppgaver som hjelper dem i gang med å jobbe med forståelse av koden (Sentance et al., 2019).

Modify (tilpasse) er den delen av undervisningsopplegget hvor elevene blir bedt om å utvide eller forandre programmet slik at det oppfyller en annen funksjon (Sentance et al., 2019). Med eksempelkoden som skjelett, skal elevene nå produsere noe eget. Et eksempel på dette kan for eksempel være at elevene skal endre koden fra å behandle partall til oddetall.

Make (skape) er det siste steget i PRIMM-modellen, og her skal elevene anvende det de nettopp har lært til å løse et annet problem ved hjelp av programmering, der de lager sin egen kode fra bunn (Sentance et al., 2019).

I artikkelen *Teaching Computer Programming with PRIMM: a Sociocultural Persepctive* knytter Sentance, Waite og Kallia PRIMM-modellen til Vygotskys sosiokulturelle læringsteori om stilasbygging (Sentance et al., 2019).

Vygotskys læringsteori baserer seg på at læring er en sosial prosess som oppstår i samhandling med andre mennesker, der språket er tankevirksomhetens verktøy (Lyngsnes & Rismark, 2016). Vygotsky mente at læring skjer når eleven trer inn i sin *nærmeste utviklingssone*, ved assistanse eller veiledning (Lyngsnes & Rismark, 2016). Kompetansen eleven innehar akkurat nå kalles det aktuelle utviklingsnivået, og definerer hva eleven er i stand til å løse helt selvstendig (Lyngsnes & Rismark, 2016). Den nærmeste utviklingssonen er det neste nivået av forståelse som eleven kan nå, om eleven bare får litt veiledning, hint, tilbakemelding. Denne typen veiledning kalles *stilasbygging*, og handler om å støtte elevene til rett tid og i riktig omfang, slik at de blir i stand til å løse stadig vanskeligere oppgaver på egenhånd (Lyngsnes & Rismark, 2016). I henhold til Vygotskys læringsteori er det i den nærmeste utviklingssonen at læring skjer, og lærerens oppgave å bygge stilas for elevene slik at de utvikler seg i takt med potensialet sitt (Lyngsnes & Rismark, 2016).

PRIMM-modellen har også klare paralleller til algoritmisk tenkning og relaterte arbeidsmetoder. Stegene i PRIMM inviterer til mange av de samme kognitive prosessene som inngår i algoritmisk tenkning. *Predict* stimulerer logikken, hvor det handler om å analysere informasjon og forutse hva koden i et program gjør, før man i *run* kjører koden. Deretter

kommer *investigate*, som innebærer å dekomponere problemet, anvende algoritmer, evaluere koden. *Modify* handler om å utbedre koden basert på feilsøket gjort i de tidligere steg. *Make* er steget hvor en ny kode lages, slik at abstraheringene som er blitt gjort og mønstrene som er blitt avdekket i utforskningen kan benyttes til å løse en annen programmeringsoppgave.

Til sammen har disse stegene involvert elevene i alle arbeidsmåtene UDIR nevner som aktiviteter som stimulerer til algoritmisk tenkning; utforske og eksperimentere gjennom å forutse og tolke koden for så å kjøre den, oppdage og rette feil gjennom feilsøking, designe og lage gjennom å utbedre og tilpasse koden, utholdenhet gjennom å prøve og feile, og samarbeid med både lærer og medelever hvor deling av kode også er en del av prosessen (Utdanningsdirektoratet, 2019).

Studien til Sentance, Waite og Kallia (2019) gjorde flere interessante funn. Et av funnene var at PRIMM, som benytter sosiokulturell læringsteori, bidro til at elevene ble mer engasjerte og aktivt deltakende i undervisningen (Sentance et al., 2019). Engasjement og deltakelse er indikatorer på trivsel og motivasjon, som i sin tur danner et godt utgangspunkt for læring (Wæge & Nosrati, 2018).

Resultatene fra datainnsamlingen i studien viste også at elever som hadde blitt undervist i programmering ved hjelp av PRIMM-metoden presterte betydelig bedre i en test etter 10 undervisningsøkter sammenlignet med kontrollgruppen som ikke hadde lært programmering ved hjelp av PRIMM-metoden (Sentance et al., 2019).

Videre trekker studien frem at lærerens oppmuntring til samarbeid og deling var essensielt for å skape et trygt og inkluderende arbeidsmiljø, og for at PRIMM-modellen skulle oppnå sitt fulle potensiale som en sosiokulturell læringsmodell (Sentance et al., 2019). Gitt dette konkluderte studien med at PRIMM-modellen er en effektiv undervisningsmetode i programmering, spesielt for elever som hadde lite erfaring med programmering fra før av (Sentance et al., 2019).

På bakgrunn av funnene fra forskning på PRIMM som undervisningsmetode vil dette masterprosjektet benytte PRIMM-metoden for å inkludere programmering i matematikkundervisningen. Valg av en undervisningsmetode som brukes konsekvent i studien er for å øke studiens reliabilitet. Reliabiliteten til en studie avhenger av hvor reproduserbare resultatene fra forskningen er, og sier noe om kvaliteten på studien (Gleiss & Sæther, 2022) og vil bli ytterligere diskutert i metodedelen av oppgaven.

2.7 Fischbeins teori om læring i matematikk

Fischbeins (1994) syn på matematikk er at matematikk er en todelt disiplin; én formell, vitenskapelig matematikkpraksis og én kreativ, aktivitetsbasert matematikkpraksis. Den første formen for matematisk praksis er assosiert med deduktiv metode, som handler om å nærme seg en absolutt sannhet om naturen og verden. Den kreative matematikkdisiplinen omfatter det Fischbein beskriver som *menneskelig aktivitet,* og et samspill mellom 3 komponenter; formell, algoritmisk og intuitiv.

Ifølge Fischbein dette samspillet mellom komponentene avgjørende for matematisk læring, og hva det vil si *kunne* matematikk. I *The Interaction Between the Formal, the Algorithmic, and the Intuitive Components in a Mathematical Activity* skriver Fischbein at det er nettopp samspillet som skaper forutsetningene for et velutviklet matematisk resonnement, og en helhetlig matematisk forståelse (Fischbein, 1994).

Dette reflekteres i både læreplanen og kjerneelementene for matematikkfaget som definerer et formelt faginnhold, samtidig som det er formulert ferdigheter, arbeidsmetoder og egenskaper elevene skal tilegne seg gjennom matematikkopplæringen. I skole- og utdanningssammenheng er det særlig relevant å diskutere, da målet med undervisning er nettopp *læring.*

Den formelle komponenten innebærer matematiske definisjoner, aksiomer, symbolbruk, teoremer og beviser (Fischbein, 1994). Den formelle komponenten utgjør den teoretiske kjernen i matematikkfaget som læringen tar utgangspunkt i, men Fischbein påpeker at selv ikke om en kan alle verdens aksiomer og beviser, vil en være i stand til å løse ethvert matematisk problem.

Matematisk kunnskap inkluderer også prosedyrer, teknikker og regneferdigheter, som utvikles gjennom mengdetrening. Teoretisk forståelse er ikke alene tilstrekkelig for å kunne anvende kunnskap på nye problemstillinger, da matematikk er både praktisk og teoretisk. Fischbein understreker at algoritmiske ferdigheter alene heller ikke strekker til, og først når den teoretiske begrunnelsen og praktiske utførelsen kombineres, oppnår vi produktiv matematiske resonnering (Fischbein, 1994).

Den siste komponenten som beskriver matematikk som menneskelig aktivitet er intuisjon. Intuisjon beskrives som en umiddelbar erkjennelse, og krever ingen rettferdiggjøring eller forklaring (Fischbein, 1994). Hvert menneskes intuisjon påvirker hvert individs umiddelbare tolkning av en situasjon, og vil følgelig ha innvirkning på hvordan hvert enkelt menneskes

tilnærming til et matematisk problem. Intuisjonen vil påvirke hvordan ulike mennesker velger strategier, gjør antakelser og resonnerer i møte med et matematisk problem (Fischbein, 1994). Den intuitive komponenten gjør også matematikk til et kreativt fag, hvor det er rom for å utforske ulike tilnærminger til samme løsning.

Med mål om at elevene skal utvikle en helhetlig matematikkforståelse gjennom opplæringen, må undervisningen tilrettelegge for stimuli av både formell, algoritmisk og intuitiv komponent (Fischbein, 1994). Kjerneelementene er utarbeidet for å oppsummere ferdighetene og kunnskapen elevene skal utvikle gjennom opplæringen i faget. Som argumentert for i kapittel 2.4 er programmeringsarbeid en matematisk virksomhet som jobber med samtlige kjerneelementer, samtidig som det tilrettelegger for utvikling av algoritmisk tenkning. Det foreligger et stort læringsutbytte ved programmering i matematikkundervisning, såfremt det i praksis viser seg å være like stor overføringsverdi som litteraturen forslår (Gjøvik & Torkildsen, 2019; Grover & Pea, 2013; Utdanningsdirektoratet, 2019; Wing, 2011).

2.8 Matematisk kreativitet

Haylock og Lithner har begge utviklet rammeverk for observasjon av kreativitet i elevers matematikkarbeid.

Kreativitet er et begrep som favner bredt, hvor det både assosieres med musikk, kunst, nyskapning, utforskende tenkemåter og vitenskap (Cropley, 1992). Kreativitet i utdanningssammenheng kommer til uttrykk på et helt annet vis enn det gjør i kulturlivet, og det er derfor hensiktsmessig å definere begrepets betydning.

Cropley skriver at elevers kreativitet kommer til uttrykk gjennom kognitive prosesser, der eleven tør utforske ulike løsningsstrategier og ulike løsninger (Cropley, 1992). Med utangspunkt i denne definisjonen av kreativitet, definerer Haylock to hovedkjennetegn på matematisk kreativitet. Haylock skriver at kreativ tenkning nesten uten unntak involverer fleksibilitet i resonneringen (Haylock, 1997), og deler matematisk kreativitet inn i 2 hoveddeler:

- Overkomme fikseringer
- Divergerende tenkning

En fiksering kan være et oppheng på en spesiell metode, et tema, strategi eller algoritme, som eleven tidligere har opplevd suksess med (Haylock, 1997). Dersom en elev henger seg opp i en fiksering i møte med en oppgave, vil denne fikseringen hindre eleven i å lykkes med å løse oppgaven. Haylock (1997) beskriver dette som et rigid tankemønster, og viser til at matematikere som er anerkjent kreative har et fleksibelt tankemønster.

Divergerende tenkning kommer gjerne til uttrykk gjennom divergerende produksjon. Divergerende produksjon er å finne flere gyldige løsninger til et åpent problem, og er et uttrykk for matematisk kreativitet (Haylock, 1997).

Haylocks teori om matematisk kreativitet kan sees i sammenheng med Skemps (1976) teori om *instrumentell* og *relasjonell* forståelse. Den instrumentelle forståelsen innebærer læren om regler, algoritmer og formler (Skemp, 1976). Denne typen kunnskap kan knyttes til tradisjonell undervisning, hvor læreren gjennomgår en matematisk regel på tavla med eksempler, etterfulgt av at elevene regner oppgaver og i stor grad *reproduserer* lærerens løsninger. Den *relasjonelle* kunnskapen stiller større krav til forståelse av begreper, og evnen til å se sammenhengen og strukturen mellom begrepene (Skemp, 1976).

I artikkelen *Habits of mind* defineres det ulike typer undervisningsaktiviteter som legger til rette for at elevene skal oppnå relasjonell kunnskap i matematikk. Eksempler som nevnes i artikkelen er for eksempel å legge til rette for at elevene kan oppdage skjulte mønstre (Cuoco et al., 1996). Å avdekke mønstre er en komponent som inngår i algoritmisk tenkning, og en viktig del av programmeringsaktiviteter (Grover & Pea, 2013). Elevene vil også oppleve glede og mestring når de lykkes med å avdekke disse mønstrene (Cuoco et al., 1996).

I motsetning til læring er mønstergjenkjenning noe som kan observeres, for eksempel ved at elevene gis en utforskende oppgave som innebærer nettopp mønstergjenkjenning. Det å kjenne igjen mønstre danner et viktig utgangspunkt for den relasjonelle kompetansen, som kreves for å være i stand til å gå fra observasjon av det spesifikke til det generelle (Skemp, 1976). På bakgrunn av disse teoriene er det rimelig å anta at mønstergjenkjenning, evnen til å generalisere funn og anvende kunnskapen på nye problemer er uttrykk for algoritmisk tenkning.

2.9 Kreativitet og algoritmisk tenkning

I 2021 publiserte Israel-Fischelson og Hershkovitz m.fl. en artikkel som tok sikte på å si noe om sammenhengen mellom kreativitet og algoritmisk tenkning, som argumentert for tidligere begge er sentrale ferdigheter for matematisk kompetanse. Studien kartla 62 empiriske undersøkelser som handlet om kreativitet og algoritmisk tenkning utført mellom 2011-2022.

Israel-Fischelson og Hershkovitz har utført en litteraturstudie basert på 62 empiriske forskningsartikler, med formål om å kartlegge på hvilket teoretisk grunnlag og i hvilken sammenheng forskning på algoritmisk tenkning og kreativitet (Israel-Fishelson & Hershkovitz, 2022). Studien bemerker at det meste av forskningen på sammenheng mellom kreativitet og algoritmisk tenkning ble utført etter 2016, der det har vært en kraftig økning i forskning på temaet de siste årene (Israel-Fishelson & Hershkovitz, 2022). Videre poengterer studien at det fremdeles er mangelfull forskning på feltet.

Til tross for et behov for videre forskning på temaet, konkluderes det fra flere hold med at det finnes en sammenheng mellom kreativitet og algoritmisk tenkning. Som nevnt tidligere i teoridelen av denne oppgaven, beskriver Grover og Pea (2013) programmering som en kreativ prosess der en benytter algoritmisk tenkning.

Det er også utført flere empiriske studier som konkluderer med det samme. I 2019 utførte Israel-Fischelson et al. en studie der 124 sjetteklassinger deltok, fordelt på to ulike skoler i

nord-Spania. Elevene som deltok i studien var med på en teknologiworkshop med programmering og robotikk. Et av funnene i studien var at kreativitet fremmer algoritmisk tenkning, og har overføringsverdi på tvers av ulike disipliner, kunst som matematikk (Israel-Fishelson et al., 2021).

Shell et al. (2013) utførte en studie som tok sikte på å kartlegge avgjørende faktorer for realfag- og teknologistudenters motivasjon, kreativitet, selvreguleringsevne, engasjement, målsettinger og læringsutbytte. Studien gjorde flere funn som støttet det annen forskningslitteratur også sier om at positive følelser tilknyttet faget og motivasjon for faget er assosiert med gode resultater, strategisk selvregulering i arbeid med faget og læring (Shell et al., 2013). Et annet interessant funn som ble gjort i den empiriske studien, var at kreativ kompetanse ble assosiert med selvreguleringsstrategi, læringsstrategier og tilegning av langtidskunnskaper (Shell et al., 2013). Studien konkluderte også at kreativitet kan fremme algoritmisk tenkning.

Shell et. al (2014) et al. utførte senere en ny empirisk studie for å se på hvordan bruk av kreativitetsøvelser kunne fremme læring av algoritmisk tenkning i et IT-emne på universitetsnivå. I denne studien ble det gjort samme funn av at arbeid med å øke den kreative kompetansen medfører positiv effekt på læring av algoritmisk tenkning, og styrker funnene gjort i studien fra 2013.

Det er viktig å understreke at selv om det er utført empiriske studier som foreslår en sammenheng mellom kreativitet og algoritmisk tenkning, er det ennå et generelt lite forskningsgrunnlag på algoritmisk tenkning (Forsström & Kaufmann, 2018; Grover & Pea, 2013), og dermed vanskelig å trekke en bastant konklusjon om sammenhengen mellom kreativitet og algoritmisk tenkning. Problemløsning og utforskende matematikk

For å legge til rette for at elevene skal benytte seg av problemløsningsstrategi og algoritmisk tenkning skal finne sted, er det interessant å ta i betraktning hva slags matematiske problemer elevene får servert. Algoritmisk tenkning er assosiert med *relasjonell kunnskap*, som innebærer å anvende kunnskap en allerede besitter på nye måter og nye problemer (Skemp, 1976). I møte med rene regnetekniske oppgaver, som for eksempel tekniske multiplikasjons- eller derivasjonsoppgaver, vil ikke elvene trenge å tenke så mye da dette er eksempler på oppgaver som kan løses ved å reprodusere løsninger til problemer de har løst før. Slike oppgaver legger til rette for demonstrasjon av instrumentell kunnskap (Skemp, 1976).

I den sammenheng er det relevant å trekke inn begrepet *undersøkende matematikk* (oversettelse av inquiry based mathematics), som Blomhøj (2019) knytter til matematisk læring, problemløsnings- og modelleringskompetanse. Med utgangspunkt i Dewey sitt arbeid med en utdanningsteori basert på utforskende undervisning, har Blomhøj og Artigué (2013) analysert og konseptualisert begrepet utforskende matematikkundervisning. Funnene ble senere oppsummert i 7 punkter (Blomhøj, 2019):

- Mennesket forsøker å forstå og beherske sin omverden gjennom utforskning og problemløsning, og gjennom å dele sin forståelse gjennom sosiale interaksjoner
- Denne grunnleggende erkjennelsesinteressen danner utgangspunktet for kunnskap om verden, som etter raffinering og kultivering blir vitenskapelig kunnskap
- Gyldig kunnskap er effektiv for forståelse av fenomener og løsning av problemer. Det er derfor viktig at elevene opplever at kunnskapen de utvikler er både nyttig og meningsfull i deres verden.
- Utdannelse skal utvikle den enkelte elev til å lære gjennom utforskning og refleksjon i sosiale fellesskap
- Kunnskapen generaliseres og settes i system i undervisningen gjennom refleksjon over felles erfaringer
- Det overordnede målet er å utdanne elever til å ta en aktiv og kritisk rolle i utviklingen av det demokratiske samfunn

Videre kommenterer Blomhøj (2019) at filosofien bak utforskende undervisning ikke kan direkte oversettes til matematikkundervisningen, men gir gode argumenter for hvorfor tilrettelegge for mer utforskende matematikkundervisning.

Som en mal for å utarbeide utforskende undervisningsopplegg har Blomhøj (2019) dannet en 3-punktsliste med 3 faser som må finne sted for et vellykket utforskende undervisningsopplegg:

1) Iscenesettelse
2) Elevene bedriver utforskende arbeid
3) Felles refleksjon og faglig læring

Iscenesettelse handler om at det må etableres et problem som gjøres tilgjengelig og relevant for elevene, slik at elevene forstår hva de skal undersøke og finner en egen inngang til problmeløsningen. I tillegg til dette må det formidles klare rammer om tidsbruk,

forventninger til hva det utforskende arbeidet skal resultere i og hvilke kriterier arbeidet skal bedømmes med.

For at selve utforskningen skal bli vellykket, må elevene få tilstrekkelig tid, frihet og veiledning slik at de er i stand til å arbeide selvstendig med utforskningen. I tillegg må det tilrettelegges for samarbeid mellom elever, utfordring og veiledning gjennom dialog.

Til slutt skal de resultatene, oppdagelsene og erfaringene fra utforskningsarbeidet oppsummeres, og settes i system i gjennom felles diskusjon. Her kan elevenes oppdagelser knyttes til relevant fagteori, og slik utvikle kunnskap og forståelse for fagbegreper.

Noen typiske elevaktiviteter for utforskende undervisning er å stille faglige spørsmål, klassifisere, måle og kvantifisere, utvikle definisjoner, innføre og bruke symboler, bruke algebra, resonnere og bevise, representere og visualisere, danne og undersøke hypoteser, eksperimentere, tolke og vurdere resultater, samt kommunisere faglig (Blomhøj, 2019).

Mange av disse elevaktivitetene kan knyttes til arbeidsmetoder og konsepter forbundet med *algoritmisk tenkning*, hvor også arbeidsmetodene innebærer det å utforske og eksperimentere, designe en fremgangsmåte for å løse problemet, feilsøke, samarbeide og diskutere med medelever (Utdanningsdirektoratet, 2019). Andre nøkkelbegrep innen *algoritmisk tenkning* er å analysere og forutse (danne hypotese), deretter bryte ned problemet, formulere algoritmer (gjerne ved hjelp av algebra), gjøre observasjoner og gjenkjenne mønstere, abstrahere og generalisere (gjerne ved bruk av algebra), og til slutt evaluere.

Det er en tydelig overlapp mellom algoritmisk tenkning og utforskende undervisning, og det kan derfor argumenteres for at et utforskende undervisningsopplegg inviterer til algoritmisk tenkning, eller at algoritmisk tenkning utvikles gjennom utforskende arbeid. Dermed kan utforskende undervisningsopplegg og utforskende oppgaver være gode utgangspunkt for å fremme læring innen algoritmisk tenkning.

2.10 Terskelbegreper

Terskelbegreper er den norske oversettelsen av begrepet *threshold concepts* (Pettersson & Brandell, 2017). Terskelbegreper kjennetegnes ved at de er begreper som er avgjørende for videre utvikling av kunnskap. I matematisk kontekst er slike begreper essensielle for at elevene skal få tilgang på relevante matematikkferdigheter og kunnskaper for å effektiv kunne ta del i problemløsning (Pettersson & Brandell, 2017).

Terskelbegreper innehar et potensial for dypere innsikt i temaet når nettopp denne terskelen for å forstå, mestre og anvende begrepet er oppnådd. Når en først har overkommet denne terskelen og behersker begrepet vil en få nye perspektiver på gitt tema, og forkaste tidligere, mindre presise oppfatninger av begrepet.

Dette er en av årsakene til at terskelbegreper kan være strevsomme å beherske, da de krever forkasting av tidligere kunnskaper og oppfatninger (Pettersson & Brandell, 2017). Et eksempel på dette kan være når barn lærer seg å telle fra 20 og oppover. De har forstått og lært at de øker en på enerplassen om gangen, og teller oppover *tjueåtte, tjueni, tjueti…* Her kan *tretti* være et terskelbegrep, og å forstå at når en når neste tier, må vi telle oppover på tierplassen og enerplassen nullstilles (Haugereid, O. A., personlig kommunikasjon, 06.06.2023).

Meyer og Land (2005) beskriver terskelbegreper som transformative, irreversible og integrative. Det *transformative* handler om at når eleven tilegner seg begrepet vil det medføre en endring i elevens syn og oppfattelse av det aktuelle temaet. En transformasjon tar tid, og det oppstår sjelden gjennom umiddelbar erkjennelse. Terskelbegreper kjennetegnes også av å være *irreversible,* som innebærer at når transformasjonen først er skjedd vil en sjelden gå tilbake til den opprinnelige oppfattelsen av begrepet. I tillegg er terskelbegreper gjerne integrative, som betyr at kunnskap om terskelbegrepet vil gjøre at tidligere skjulte sammenhenger kommer til syne. Slik bidrar terskelbegrepet til en dypere innsikt med større oversikt. Dette medfører at terskelbegrepene er de mer avgjørende begrepene for videre utvikling av kompetanse på et felt.

2.11 Motivasjon

Motivasjon kan defineres som et individs incentiv til å utføre et arbeid eller en handling. Motivasjon lar seg ikke observere direkte, og er dermed et begrep som må operasjonaliseres. Det finnes mange ulike arenaer hvor motivasjon er interessant, men i klasseromssammenheng kan elevers motivasjon komme til uttrykk gjennom tanker, følelser og handlinger (Wæge & Nosrati, 2018).

Motivasjon har innvirkning på elevenes handlinger i klasserommet, som innebærer i hvilken grad de engasjerer seg, hvor hardt de arbeider og utholdenheten de utviser i arbeid med vanskelige oppgaver. Motivasjon har med andre ord stor innvirkning på elevenes deltakelse i undervisningen (Wæge & Nosrati, 2018), som i sin tur får følger for hver enkelt elevs læringsutbytte. Også Lyngsnes & Rismark (2016) understreker at motivasjon er en vesentlig forutsetning for at elevene skal gå i gang med arbeidsoppgaver. Det er dermed interessant å ta motivasjon i betraktning i undervisningsplanleggingen, for å best mulig tilrettelegge elevene opplever glede, mestringsfølelse og engasjement ved å delta i undervisningen.

2.11.1 Indre og ytre motivasjon

Motivasjon kan deles inn i to hovedtyper; indre og ytre motivasjon (Wæge & Nosrati, 2018). Den indre motivasjonen anses som høyerestående enn den ytre motivasjonen. Om en elev er indre motivert for matematikkfaget vil den legge ned tid og innsats fordi eleven opplever det meningsfylt i seg selv å arbeide med matematikk. Elever som er ytre motivert har derimot utenforliggende incentiver for å arbeide med faget, som frykt for en dårlig karakter eller ønske om anerkjennelse fra lærer eller foresatte. Ved ytre motivasjon finner ikke eleven det verdifullt i seg selv å arbeide med matematikkfaget.

Det finnes dog ulike grader av ytre motivasjon. Ryan & Decis selvbestemmelsesteori handler om at elevens opplevelse av autonomi definerer den ytre motivasjonen, hvor høyere grad av autonomi medfører sterkere motivasjon, mens motivasjonen svekkes sammen med elvens opplevelse av selvbestemmelse (Ryan & Deci, 2000) .

Den svakeste formen for ytre motivasjon oppstår når elevens handlinger kontrolleres av en konkret straff eller belønning. Eksempler på dette kan være redsel for dårlig karakter, tilsnakk, anmerkning eller annen negativ oppmerksomhet.

En annen form for kontrollert ytre motivasjon er når eleven arbeider med faget for å unngå negative følelser som dårlig samvittighet, bekymring eller skam over å ikke mestre faget

(Wæge & Nosrati, 2018). Det er ubehagelig å føle seg dårlig forberedt eller å oppleve at en får til mindre enn sine medelever. Frykten for slikt ubehag kan gi en form for motivasjon til å arbeide med faget, men vil ikke være lystbetont og meningsfylt for eleven.

Den ytre motivasjonen oppleves derimot sterkere dersom eleven drives av et mål som oppleves viktig og meningsfylt. Dersom eleven har ambisjoner om å bli for eksempel sivilingeniør eller realfagslektor, innebærer det at eleven opparbeider seg gode fagkunnskaper og solid forståelse innen matematikkfaget. Selv om dette er en form for ytre motivasjon, vil også denne typen motivasjon være sterk da eleven har høy grad av autonomi. Eleven arbeider med matematikk på bakgrunn av egne drømmer og ambisjoner. Med et konkret mål å nå, oppleves arbeid med matematikk mer verdifullt enn ved kontrollert form for ytre motivasjon (Wæge & Nosrati, 2018).

2.11.2 Mestringsforventninger

Bandura definerer *self-efficacy* som et individs selvtillit egen utførelseskapasitet og evnen til å mobilisere sine kognitive, sosiale, emosjonelle ferdigheter til å utføre en spesifikk oppgave (Bandura, 1997). Det engelske begrepet s*elf-efficacy* kan oversettes til *mestringsforventning* på norsk.

Mestringsforventning bestemmes i stor grad av tidligere erfaringer fra liknende situasjoner, og har stor innvirkning på hvordan vi går nye utfordringer i møte (Bandura, 1997). I løpet av livet møter vi på ulike mennesker og situasjoner som gjør oss klar over hva vi mestrer eller ikke. Dette gjelder på mange ulike arenaer, deriblant i skolesammenheng og i matematikkundervisningen. Det har vist seg at elever med lave mestringsforventninger har lavere terskel for å gi opp når de møter på et problem, eller la være å begynne på en oppgave fordi de ikke tror at de kommer til å få den til (Wæge & Nosrati, 2018). Når eleven forventer å mislykkes, vil det oppleves enklere å la være å prøve enn å bli gjennomskuet eller å få seg en selvtillitsknekk av å ikke få til. Elever med høy mestringsvilje viser derimot større utholdenhet og kampvilje når de møter på vanskelige oppgaver, og orker å stange i oppgaven lenger (Wæge & Nosrati, 2018).

Bandura definerer 4 faktorer som sammen definerer en elevs mestringsforventninger; egne mestringserfaringer, vikarierende mestringserfaringer, oppmuntring og støtte, samt elevens psykologiske og fysiske tilstand (Bandura, referert i Wæge & Nosrati, 2018).

Hovedkilden til en elevs mestringsforventninger er elevens egne mestringserfaringer (Bandura, 1997). Når elevene arbeider med diverse oppgaver vil de vurdere eget arbeid, og

bedømme hvorvidt de mestret oppgaven eller ikke. På bakgrunn av dette justerer elevene oppfatningen de har av egen kompetanse og ferdigheter (Wæge & Nosrati, 2018). Når elevene opplever at de lykkes, vil det reflekteres i mestringsforventningene. Gode mestringsopplevelser gir elevene selvtillit og tro på egen kompetanse, og som dermed gjør at elevene forventer at de vil være i stand til å løse liknende oppgaver. På den andre siden vil elevenes tiltro til egen kapabilitet svekkes om de har lagt ned innsats i en oppgave de ikke klarer å løse. Når elevene samler opp erfaringer der de mislykkes, svekkes mestringsforventningene, som igjen kan gjøre det lett å gi opp i møte med motgang (Wæge & Nosrati, 2018).

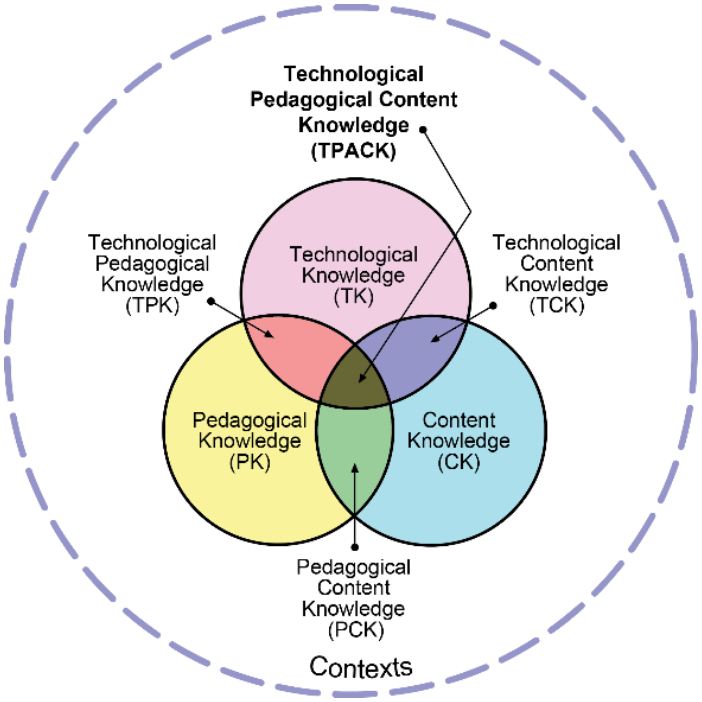
En annen bidragsytende faktor til en elevs oppfatning av egen kompetanse er vikarierende erfaringer (Bandura, 1997). Vi mennesker sammenligner oss kontinuerlig med hverandre, og er hverandres korrektiver. Elever som er på et tilsvarende faglig nivå, kan elevenes mestringsforventninger påvirkes av hverandre. Dersom en elev observerer en medelev jobbe hardt med en vanskelig oppgave for så å lykkes, vil det øke mestringsforventningene til den observerende eleven (Wæge & Nosrati, 2018). Da er terskelen for å prøve seg på den utfordrende oppgaven lavere, og eleven er forberedt på å klare det når den jevngode klassekameraten klarte det. De vikarierende mestringsforventningene kan også slå andre veien, dersom en elev observerer en jevngod medelev slite og gi opp på en oppgave, kan den observerende eleven fort tenke at *«da klarer sikkert ikke jeg det heller»*, og vil ha lettere for å gi opp (Wæge & Nosrati, 2018).

En elev mestringsforventning kan også påvirkes av *oppmuntring og støtte* den får av medlever, foreldre eller lærere. Tydelig uttrykk for at disse tillitspersonene har tro på at eleven er kapabel til å løse den kan styrke elevens mestringsforventning (Wæge & Nosrati, 2018).

Til slutt har også elevens *psykologiske og fysiologiske tilstand* innvirkning på mestringsforventning. Hvordan eleven har det i arbeid med matematikk er relevant for mestringsforventningene, hvor følelser som angst og stress vil ha negativ påvirkning, mens opplevelse av glede og nysgjerrighet kan øke mestringsforventningene til eleven (Bandura, referert i Wæge & Nosrati, 2018, s. 47).

2.12 TPACK

TPACK er en forkortelse for Technological Pedagogical Content Knowledge. TPACK beskriver den overlappende, tverrfaglige kunnskapen en lærer besitter på bakgrunn av sin pedagogiske kunnskap, faglige kunnskap og teknologiske kunnskap (Koehler et al., 2012). TPACK danner dermed utgangspunktet for at læreren skal lykkes med å lage et undervisningsopplegg som kombinerer både programmeringskunnskap, pedagogisk kunnskap matematikkfaglig kunnskap.



Figur 2: TPACK illustrert (Publisert med åpen lisens fra utgiver, © 2012 av tpack.org)

Koehler, Shin & Mishra Niess (2012) gjennomførte en litteraturstudie som analyserte 303 relevante artikler utgitt mellom 2006 og 2010 som omhandlet TPACK.

66 av de relevante artiklene ble inkludert i studien, hvorav resten av artiklene ble forkastet da de ikke oppfylte studiens kriterier. Formålet med studien var å besvare hvordan TPACK kan måles. Basert på den empiriske dataen fra de 66 inkluderte artiklene, kunne studien definere 4 metoder som kan benyttes for å måle TPACK (Koehler et al., 2012):

1) Selvrapportering: et spørreskjema hvor deltakerne bes rangere hvor enige de er med et ferdigformulert utsagn

2) Åpne spørreskjema: Ber deltakerne besvare spørsmål med egne ord

3) Prestasjonsvurdering: Direkte vurdering av deltakernes evne til å løse oppgaver som er designet for å måle kunnskap innen et spesifikt felt

4) Intervju

TPACK er en metode for å måle lærerens teknologisk-pedagogiske kompetanse, og teorien om TPACK danner et utgangspunkt for valg av metode og forskningsdesign i denne masteroppgaven. Selv om TPACK er en metode for å måle lærens teknologisk-pedagogiske kompetanse, er det rimelig å anta at vellykketheten i et utarbeidet og gjennomført undervisningsopplegg vil kunne gjenkjennes i elevenes utbytte av det.

2.13 Vurdering av elevbesvarelser

Stylanides påpeker at argumentasjon basert på empiriske observasjoner og konkrete eksempler også kan deles inn i 2 kategorier; naiv empirisme og avgjørende eksperiment.

Naiv empirisme kjennetegnes ved at eleven benytter enkeltstående eksempler som grunnlag for å trekke generelle konklusjoner (Stylianides, 2009). Eksempler på dette i elevbesvarelser er at de tester en påstand med vilkårlige tall. Etter å ha testet en påstand med 2 gjeldende eksempler trekker eleven konklusjonen at påstanden gjelder for alle tilfeller.

Avgjørende eksperiment er derimot en mer avansert form for empirisk argumentasjon. I et slik tilfelle har eleven en strategisk tilnærming til tallene som velges ut for å bevise eller motbevise en påstand (Stylianides, 2009). Argumentasjon ved *avgjørende eksperiment* trenger heller ikke være gyldig eller riktig, men eleven anerkjenner at bekreftelse fra et enkelt eksempel ikke medfører en generell gyldighet for alle tilfeller (Stylianides, 2009). Eksempler på dette i elevbesvarelser kan være at eleven argumenterer for test med ulike typer tall, som oddetall og partall, eller gjør et bevisst og strategisk valg av eksempel for å motbevise at en påstand er gyldig.

Med utgangspunkt i dette rammeverket vil elevenes testbesvarelser vurderes slik at de enklere kan kvantifiseres for å avdekke trender i datamaterialet. I *Undervisningskunnskap i matematikk* (Hovik, 2016*)* defineres 4 nivåer av matematisk resonnering, der 1 er lavest og 4 er høyest:

1) Begrunnelse ved å referere til autoriteter (lærer, lærebok, foreldre, mm)
2) Begrunnelse gjennom konkrete eksempler
3) Matematisk resonnering basert på en visuell representasjon (konkreter/tegning, tekst eller regnefortelling)
4) Bevis ved bruk av algebraisk notasjon og bruk av regnelovene

Det å prøve ut spesifikke tilfeller er et viktig utgangspunkt for å avdekke skjulte mønstre og sammenhenger, og kan være det enklere å bevege seg videre til generalisering (Hovik, 2016). På samme måte kan ulike former for representasjon gjøre det enklere å gå videre fra de spesielle oppdagelsene til de generelle tilfellene (Hovik, 2016). Dette er også et teoretisk rammeverk som gjør det mulig å kvantifisere og vurdere besvarelsene i prestasjonsmålingene (redegjøres for i metodedelen).

3 Metode

3.1 Valg av metode

Metode er et ord av gresk opprinnelse, og betyr *«veien frem til et bestemt mål»* (Gleiss & Sæther, 2022). Valg av metode baseres på hva slags kunnskap en ønsker å utvikle, og hva slags del av virkeligheten en ønsker å presentere.

Denne masteroppgaven tar sikte på å utforske sammenheng og overføringsverdi mellom programmering ved bruk av PRIMM-metoden og algoritmisk tenkning. Algoritmisk tenkning er ikke en direkte målbar kvantitet, og derfor vil det være avgjørende å gjøre grundige operasjonaliseringer basert på relevant teori.

Problemstillingen involverer både matematiske fagkunnskaper, teknologisk kompetanse i form av programmering og en didaktisk metode for å bistå læringen. Kombinasjonen av disse 3 kalles TPACK (Technological Pedagocial Content Knowledge) (Koehler et al., 2012)

I henhold til teorien om hvordan måle TPACK, er det naturlig å velge de foreslåtte metodene for å samle inn datamaterialet på feltet. Koehler et al. (2012) foreslår

1) Selvrapportering: et spørreskjema hvor deltakerne bes rangere hvor enige de er med et ferdigformulert utsagn

2) Åpne spørreskjema: Ber deltakerne besvare spørsmål med egne ord

3) Prestasjonsvurdering: Direkte vurdering av deltakernes evne til å løse oppgaver som er designet for å måle kunnskap innen et spesifikt felt

4) Intervju

Datainnsamlingsmetodene som foreslås for å danne kunnskapsgrunnlag om TPACK er både kvantitative og kvalitative metoder, og det er da naturlig å benytte et forskningsdesign som heter *mixed methods.* Denne masteroppgaven vil både benytte seg av den kvantitative datainnsamlingsmetoden *selvrapporteringsskjema,* i tillegg til de foreslåtte kvalitative datainnsamlingsmetodene *åpne spørreskjema* og *intervju. Prestasjonstest* kan utføres som både kvantitativ og kvalitativ datainnsamlingsmetode, men i denne studien vil prestasjonstesten behandles og analyseres kvalitativt. Dette utdypes under delkapittel 3.2 og 3.3 om forskningsdesign, og egne delkapitler for hver datainnsamlingsmetode (3.4 og 3.5).

For å avgrense oppgaven og omfanget av datainnsamlingen tilpasset en masteroppgave, vil denne studien være en casestudie, der datainnsamlingen og prosjektet utføres 1T-matteklasse på en videregående skole. Mer om utvalget under delkapittel 3.6.

3.2 Forsknings design: Case studie

Dette forskningsprosjektet vil benytte seg av et casedesign, som betyr et studie av det spesifikke innenfor grensene av noen få enheter (Johannesen & Christoffersen, 2012). Dette er en type forskningsdesign som kan ta mange ulike former, hvor selve tilfellet som studeres kan variere i natur, i tillegg til at det er mulig å anvende flere datainnsamlingsmetoder (Johannesen & Christoffersen, 2012). Det casestudier har til felles, er at de tar for seg et spesielt tilfelle, som er avhengig av tid og sted.

Casestudier egner seg også til å bidra til utvikling av teorier (Flyvbjerg, 2010). Denne studien har som formål å undersøke om programmering kan fremme algoritmisk tenkning i matematikk ved bruk av PRIMM-modellen. Denne studien kan dermed være et bidrag til forskning om undervisningsmetoder for programmering i matematikkfaget .

Studien vil være en mixed methods casestudie, der det kvantitative datamaterialet består av et lukket spørreskjema, og de kvalitative kildene til datamateriale er åpnet spørreskjema, prestasjonstester og intervju. Denne mixed methods studien vil derfor være kvalitativt dominert. Dersom virkeligheten er for kompleks til å representere gjennom tall, bør en forske kvalitativt (Postholm & Jacobsen, 2018). Dette utdypes nærmere i kapittel 3.3.

3.3 Mixed methods

I en casestudie vil det være fordelaktig å benytte seg av flere ulike datainnsamlingsmetoder for å få et detaljert og størst mulig datagrunnlag om enheten(e) som studeres (Johannesen & Christoffersen, 2012). Dette står i tråd med teorien om TPACK og hvordan måle teknologisk pedagogisk kompetanse, hvor 4 datainnsamlingsmetoder oppgis.

Et forskningsdesign som kombinerer både kvalitative og kvantitative datainnsamlingsmetoder kalles *mixed methods* (Gleiss & Sæther, 2022). Fordelen med mixed methods er at det gir muligheten til å få oversikt over et større utvalg og potensielt avdekke trender gjennom kvantitativ datainnsamling. Samtidig kan kvalitativ datainnsamling berike forskningen med ulike perspektiver, oppfølging av avvikende funn, samt gi innsyn i aspekter som ikke fremkommer i en kvantitativ undersøkelse (Gleiss & Sæther, 2022).

I Shorten og Smiths (2017) artikkel klassifiserer de ulike typer mixed methods-studier, basert på om det er det kvantitative eller kvalitative datamaterialet som dominerer studien, i hvilken rekkefølge datamaterialet samles inn og analyseres og studiens formål.

I denne studien vil det kvantitative og det kvalitative datamaterialet samles inn samtidig, og resultatene bli satt i sammenheng med hverandre under analysearbeidet i etterkant av datainnsamlingen (Shorten & Smith, 2017).

Det blir også formulert noen spørsmål til evaluering av et forskningsdesign som benytter mixed methods (Shorten & Smith, 2017).

Et av spørsmålene er om problemstillingen rettferdiggjør bruk av mixed methods. Formålet med masteroppgaven er å undersøke hvorvidt programmering kan fremme algoritmisk tenkning hos elever. Dette er et komplekst tema, som involverer både teknologisk kunnskap, fagkunnskap, anvendelse av relasjonell kunnskap innen to fagområder samtidig, i tillegg til utvikling av problemløsningsstrategier. Dette kan kategoriseres som TPACK, der forskningen som finnes fra før av på temaet er samlet inn ved bruk av flere metoder, som redegjort tidligere for i teoridelen. Artikkelen om TPACK som henvises til er en litteraturstudie der 303 studier på temaet ble analysert og kategorisert, og på bakgrunn av det anbefales bruk av flere datainnsamlingsmetoder, både kvantitative og kvalitative (Koehler et al., 2012). På bakgrunn av tidligere forskning vil det derfor benyttes et mixed methods forskningsdesign i denne masteroppgaven.

Et annet spørsmål verdt å ta i betraktning er om de ulike datainnsamlingsmetodene er likestilte, eller om visse dominerer datamaterialet. I denne studien vil det først gjennomføres en spørreundersøkelse med ferdigstilte svaralternativer (kvantitativ) for å danne et overblikk av elevenes opplevelser av arbeidsmetoder, mestringsfølelse og bruk av programmering i matematikkundervisning. Som tidligere nevnt er tematikken i studien kompleks, og utvalget er relativt lite (n=21). Det er derfor rimelig å anta at det er for få individer til å avdekke trender som er gjeldende for andre 1T-mattegrupper eller lignende utvalg. I tillegg til at hver klasseromskultur varierer stort og er unike, som er typisk for casestudier der tid, sted og kontekst spiller en avgjørende rolle. Dermed er kvalitativ forskningsmetode bedre egnet til å gjengi presis kunnskap om den studerte virkeligheten (Postholm & Jacobsen, 2018), og det vil i denne studien være de kvalitative observasjonene innhentet gjennom åpne spørreskjema, elevbesvarelsene i prestasjonstest og intervjuer som dominerer denne studien.

3.4 Kvantitativ datainnsamling

Kvantitativ metode kjennetegnes av datamateriale som kommer i tallform, hvor det gjør det mulig å si noe om mengdeforhold mellom observasjoner, og bruke datasettet som grunnlag for å beregne statistiske størrelser og finne trender (Gleiss & Sæther, 2022). Som den kvantitative datainnsamlingen vil det utføres et spørreskjema med forhåndsformulerte svaralternativer, basert på teorien om hvordan TPACK måles.

3.4.1 Spørreskjema

Når spørreundersøkelsen gjennomføres som en selvrapportering med forhåndsformulerte svar, er spørreundersøkelsen og datamaterialet sortert på forhånd, som vil si at resultatene kan kvantifiseres (Gleiss & Sæther, 2022). Målet med spørreskjemaet er å samle informasjon om elevenes opplevelser, holdninger og syn på programmerings rolle i matematikkundervisningen.

For å sikre validitet i spørreskjemaet ble det stilt flere spørsmål som ønsket å avdekke det samme, for å sikre at svarene elevene ga samsvarte med hverandre, og at de ikke bevisstløst klikket seg gjennom spørreskjemaet.

Et eksempel på dette er følgende to spørsmål hentet fra spørreskjema (se vedlegg 3):

«Jeg forstår ikke hva jeg skal bruke programmering til»

«Jeg synes det er nyttig å lære programmering»

Disse to spørsmålene er formulert med motsatt forutinntatthet, der vinklingen av det første spørsmålet stiller seg negativ til nytteverdien av programmering, mens det andre spørsmålet har en mer positiv vinkling. Dette er en måte å sikre at elevene har forstått hva de er blitt spurt om, og at de har svart konsekvent i spørreskjemaet.

I spørreskjemaet ble det brukt 4 svaralternativer; helt enig, litt enig, litt uenig, helt uenig. Formålet med dette var å tvinge elevene til å ta standpunkt til hvert av utsagnene, da det ikke er mulig å svare nøytralt. Slik kan en tilegne variabler verdier, og det blir mulig å sammenligne besvarelsene fra ulike informanter (Gleiss & Sæther, 2022).

Til forskjell fra intervjuer der det er mulig å legge til eller endre på ordlyden av et spørsmål slik at informanten oppfatter spørsmålet slik det er ment, må det brukes mye tid på å formulere spørsmål i spørreskjema for å sikre at de ulike informantene oppfatter spørsmålene så likt som mulig (Gleiss & Sæther, 2022).

3.5 Kvalitativ datainnsamling

Kvalitativ metode kjennetegnes av at datamaterialet kommer i tekstformat, ikke tall (Gleiss & Sæther, 2022). Den kvalitative dataen er mest ment å gi et dypere innblikk av elevenes oppfatning av undervisningsopplegget og bruk av programmering i undervisning (Gleiss & Sæther, 2022).

Elevbesvarelsene fra prestasjonsmålingene vil bli analysert i forhold til teoretisk rammeverk redegjort for i kapittel 2.13, og knyttet til relevant teori om hvordan algoritmisk tenkning kommer til uttrykk. Formålet med en pretest og en retest i etterkant av et undervisningsopplegg er å avdekke om det kan observeres økt bruk av resonnementer og arbeidsmetoder assosiert med algoritmisk tenkning.

3.5.1 Åpent spørreskjema

Mens et lukket spørreskjema med forhåndsformulerte svaralternativer er en forutsetning for å kunne fremstille datamaterialet statistisk, har åpne spørreskjema den fordel at de kan fange opp informasjon fra korrespondentene som ikke forskeren forutså i utarbeidelsen av spørreskjemaet (Gleiss & Sæther, 2022).

For å få et mer helhetlig datagrunnlag å basere drøfting av problemstilling på, ble det benyttet både lukket og åpent spørreskjema. Det åpne spørreskjemaet gir korrespondentene anledning til å ytre seg fritt, og gi mer nyanserte besvarelser enn det er mulig i et lukket spørreskjema (Gleiss & Sæther, 2022).

3.5.2 Prestasjonsmåling: pretest og retest

Prestasjonsmålinger kan gjøres kvantitative om oppgavene lar seg oversette til tall. Opprinnelig var det tenkt å poengsette prestasjonsvurderingene i henhold til de matematiske resonneringsnivå observert i elevbesvarelsene, og kvantifisere eventuelle endringer fra pretest til retest.

I etterkant av datainnsamlingen har det blitt vurdert som mer hensiktsmessig å analysere prestasjonstestene kvalitativt, da oppgavene i pretest og retest ikke er helt like, og det dermed er vanskelig å sammenlikne direkte. Algoritmisk tenkning er som tidligere nevnt et sammensatt begrep, og for å lykkes med å fange kompleksiteten av det ble det vurdert som mer hensiktsmessig å betrakte utviklingen av hvert individs resonneringsnivå fra pretest til retest, og kunne følge opp interessante funn. Algoritmisk tenkning involverer blant annet

bevisst problemløsningsstrategi, analyse, dekomponering, mønstergjenkjenning generalisering og vurdering (Wing, 2006). Observasjon og forbedring av dette fra pretest til retest hos enkelte individer er interessante funn, og vil kunne knyttes til spørreundersøkelse og intervju for å oppnå mer helhetlig kunnskap om den observerte utviklingen.

3.5.3 Semistrukturert intervju

Det vil derfor gjennomføres intervjuer med noen utvalgte individer fra 1T-gruppen. Målet er å avdekke aspekter og oppfatninger ved bruk av programmering i matematikkundervisningen som ikke fanges opp av prestasjonsmåling og selvrapporteringsspørreskjema. Det kan i tillegg være interessant å følge opp hvordan elevene har tenkt når de har løst ulike oppgaver i prestasjonstesten, for å få et innblikk i for eksempel elevens problemløsningsstrategier, mønstergjenkjenning og abstrahering.

Intervjuene vil være av typen semistrukturert intervju, hvor intervjuet tar utgangspunkt i forhåndsformulerte spørsmål, men rekkefølgen på spørsmål kan variere fra intervju til intervju, og det gis rom for spontane oppfølgingsspørsmål underveis (Gleiss & Sæther, 2022). Dette gir både sammenheng i datamaterialet, men også frihet til å følge opp interessante funn.

3.6 Om utvalget

Ettersom algoritmisk tenkning er et begrep som først introduseres i læreplanen på 1. trinn på videregående i faget matematikk T (Kunnskapsdepartementet, 2020), var det interessant å bruke en 1T-gruppe som informanter i studien.

Det første året på videregående velger elevene mellom praktisk (P) eller teoretisk (T) matematikk. Matematikk T anses å være et mer teoritungt og dermed arbeidskrevende fag, da det er ment å danne grunnlaget for realfaglig studieretning. Det vil si at de som velger matematikk T ofte har større motivasjon for faget, enten fordi elevene opplever faget som meningsfylt i seg selv, opplever mestring eller har personlige målsettinger om f.eks en studieretning eller yrke som krever realfag (Wæge & Nosrati, 2018).

Elevene som begynte på VGS i 2022, gikk i 9. trinn da den nye læreplanen ble innført for skoleåret 2020/2021. Det vil si at de skal ha vært borti programmering fra før av, men det er rimelig å anta varierende kompetanse da det var første skoleår med læreplan fra Kunnskapsløftet 2020, sammen med manglende konsensus om hvordan programmering skulle inkorporeres i matematikkfaget (Forsström & Kaufmann, 2018). I tillegg til dette var skoleårene 2020/2021 preget av koronapandemien, der skolehverdagen for mange har vært

preget av nettundervisning, delvis nedstengning, sykefravær og avlyste eksamener (Nøkleby et al., 2021).

Den utvalgte 1T-gruppen som fikk invitasjon til å delta i forskningsprosjektet har selv rapportert at de hadde lite kjennskap til programmering før 1T (kapittel 4). Utvalget er dermed en relativt homogen gruppe ved at de har begrenset erfaring med programmering, i tillegg til at de har valgt T-matematikk og det dermed kan antas at de er motiverte og interesserte i faget.

Dette kalles et dermed et strategisk utvalg, der informantene er valgt basert på spesifikke kriterier, i dette tilfellet erfaring, alder og faglig motivasjon (Gleiss & Sæther, 2022). Formålet med forskningsprosjektet er å avdekke hvorvidt et programmeringsbasert undervisningsopplegg kan fremme algoritmisk tenkning, og på det viset ha overføringsverdi til problemløsning innen andre områder i matematikk. Dermed var det mer hensiktsmessig å bruke et strategisk utvalg, enn et sannsynlighetsutvalg der formålet er å avdekke trender i et utvalg som skal være representativt for den generelle befolkningen (Gleiss & Sæther, 2022).

Den utvalgte 1T-gruppen består av 21 individer. Ideelt sett skulle casestudien vært utført i flere 1T-klasser, da 21 individer er et lite utvalg der individuelle forskjeller vil gi betydelig utslag i datamaterialet. Dermed vil det være vanskelig å overføre observasjonene fra dette casestudiet til lignende utvalg, for eksempel 1T grupper ved andre skoler. Observasjonene og funnene som blir gjort i denne studien vil derfor være kunnskap om dette spesifikke utvalget, som er typisk for casestudier (Johannesen & Christoffersen, 2012).

Til tross for at et større utvalg ville styrket både studiens validitet og reliabilitet, måtte forskningsprosjektet måtte tilpasses rammene av en mastergradsoppgave, der tid og omfang er begrensende faktorer. Det var dermed ikke mulig å utføre den samme studien i flere 1T-klasser over samme tidsperiode. Dersom forskningsdesignet skulle blitt reprodusert og utført i en annen T-matteklasse på et senere tidspunkt, ville en også fått feilkilden av at elevene hadde arbeidet med faget i lengre tid, og kunne ha utviklet evner innen programmering og algoritmisk tenkning i takt med progresjon i faget.

Som faglærer til denne 1T-gruppen var rolleavklaring essensielt for både forskningsetikk og studiens kvalitet (De nasjonale forskningsetiske komiteene, 2021). Det er flere aspekter ved denne situasjonen som har innvirkning på resultatene i studien. Ettersom det er etablerte relasjoner mellom lærer og elever i 1T-gruppen vil det gjøre det vanskeligere å reprodusere resultatene i en vilkårlig klasse. Dette svekker studiens reliabilitet. På den andre siden vil

disse etablerte relasjonene danne trygge rammer for elevene, og de vil oppleve at forskningen finner sted innenfor deres vante klasseromssituasjon (Gleiss & Sæther, 2022). Det er dermed rimelig å anta at elevenes adferd vil påvirkes mindre av stressfaktoren det kan være å delta i et forskningsprosjekt enn om det var utført av noen som elevene ikke hadde kjennskap til. I tillegg vil en som lærer for klassen være bedre egnet til å tolke og forstå elevenes utsagn, både verbalt og gjennom kroppsspråk. Dette vil styrke validiteten av studien og resultatene. På den andre siden kan relasjonen gjøre at elevene føler på et press til å delta eller yte ekstra fordi de ønsker å være støttende og positive overfor en lærer de har relasjon til. Dette vil diskuteres ytterligere i 3.11 etiske betraktninger.

3.7 Undervisningsopplegg

Formålet med denne masteroppgaven er å undersøke om bruk av programmering i matematikkundervisning kan fremme algoritmisk tenkning hos elever. At elever skal kunne løse problemer ved hjelp av algoritmisk tenkning er en del av læreplanen for faget matematikk 1T (Kunnskapsdepartementet, 2020), men skiller seg fra andre kompetansemål ved at det er mindre konkret enn for eksempel å bruke trigonometri til å løse teoretiske og praktiske problem med lengder, vinkler og areal. Algoritmisk tenkning er bedre beskrevet som en ferdighet elevene skal utvikle gjennom matematikkfaget, som handler en bevisst og systematisk tilnærming til problemløsning og de kognitive prosessene som inngår. Eksempler på kognitive prosesser som inngår i problemløsningsstrategien algoritmisk tenkning er analyse av problemet, mønstergjenkjenning, abstrahering, generalisering og vurdering (Wing, 2006).

Disse kognitive prosessene kommer gjerne ikke til uttrykk i arbeid med oppgaver som kun krever å *imitativ resonnering,* som handler om å være i stand til å reprodusere løsninger basert på tidligere erfaringer (Lithner, 2007). Eksempler på slike oppgaver er prosedyreorienterte oppgaver som kun krever at elevene skal anvende algoritmer, teknikker eller aritmetikk for å komme frem til riktig løsning.

Det må dermed tilrettelegges for at elevene utvikler og anvender *relasjonell kunnskap* gjennom både undervisningsopplegget med programmering som datamaterialet baseres på, og også i oppgavevalg i prestasjonstesten. Et utvalg av oppgavene i prestasjonstesten vil redegjøres for senere 3.8, fullstendige oppgavesett fra prestasjonstest i vedlegg 5 og 6.

Relasjonell kunnskap er igjen forbundet med *utforskende matematikk.* Basert på Blomhøjs (2019) kriterier for utforskende undervisning må følgende 3 punkter oppfylles for at et undervisningsopplegg skal kunne klassifiseres som utforskende:

1) Iscenesettelse
2) Elevene bedriver utforskende arbeid
3) Felles refleksjon og faglig læring

Oppsummert blir elevene først introdusert for et problem, og det gjøres relevant og tilgjengelig for dem. Deretter får elevene anledning til å bedrive selve utforskningen, innen forhåndsavklarte rammer for tidsbruk og forventninger. Til slutt samles trådene, og elevenes observasjoner blir i fellesskap diskutert i henhold til relevant fagteori.

Disse kriteriene for et utforskende undervisningsopplegg er forenelige med teorien om PRIMM, som er en undervisningsmodell utviklet av Sue Sentance (2017) som kan brukes til å lære elever programmering.

PRIMM kan sees på som en stegvis problemløsningsstrategi, som skal hjelpe elever til å utvikle en tenkemåte og arbeidsmetode for å løse problemer ved hjelp av programmering (Sentance et al., 2019). PRIMM er deles opp i de 5 stegene: Predict, run, investigate, modify, make. Som argumentert for gjennom kapittel 2 overlapper PRIMM-metoden godt med arbeidsmetoder og de kognitive prosesser involvert i algoritmisk tenkning.

På bakgrunn av det, og for å sikre reliabilitet i studien vil PRIMM-metoden bli brukt som mal for undervisningsopplegget som inngår i denne studien. En mal for utarbeiding av undervisningsopplegg vil gjøre det enklere å reprodusere forskningsdesignet, og å gjenskape de samme funnene.

Undervisningsopplegget består av 10 skoletimer der elevene arbeidet med programmering. Da undervisningsopplegget skulle gjennomføres holdt elevene på med temaet *funksjoner*. Det ble derfor naturlig å arbeide med programmering som var relevant for temaet *funksjoner*.

Med unntak av noen svært få, rapporterte elevene at de hadde liten erfaring med programmering, og ingen erfaring med programmeringsspråket Python som ble benyttet i undervisningsopplegget. De fleste hadde gjort litt blokkprogrammering på ungdomsskolen, men de aller fleste hadde begynt å lære å programmere med Python høsten de begynte med faget matematikk T. Tidligere samme høst hadde de arbeidet litt med grunnleggende programmering, som å bruke matematiske operatorer, definere variabler, skrive til skjerm,

ulike datatyper (heltall, desimaltall, skrift), samt å lage enkle programmer som å bruke programmering til å utføre regneoperasjoner.

Nedenfor kommer to eksempler på hvordan PRIMM-metoden ble benyttet til å utarbeide undervisningsopplegg i perioden. I løpet av perioden arbeidet elevene med å lage programmer som beregner stigningstall og konstantledd for lineære funksjoner basert på at brukeren av programmet oppgir to punkter fra grafen til funksjonen, beregne røttene til et andregradsuttrykk, finne nullpunktene ved hjelp av symmetriaksen, definere funksjoner i Python, beskrive figurtall ved å bruke funksjoner, bruke løkker til å differensiere hvordan et program behandler ulik informasjon, legge til verdier i lister og bruke det til å plotte grafen til en funksjon. Vanskelighetsgraden på programmeringen økte gradvis.
I begynnelsen av hver undervisningsøkt ble ordet «PRIMM» skrevet på tavla, og i fellesskap ble det repetert hva de ulike bokstavene stod for. Formålet var bevisstgjøring og konkretisering av problemløsningsstrategien.

3.7.1 Stigningstall og konstantledd til en lineær funksjon med PRIMM

Den første programmeringsøkten handlet om å lage et program som hentet inn 2 punkter til en funksjon, ved hjelp av input-funksjonen i Python (en funksjon som tilegner en variabel verdi basert på informasjon fra brukeren). Deretter må elevene være i stand til å bruke relevant matematikkunnskap, hente inn de riktige variablene og benytte regler for regnerekkefølge, og til slutt gi brukeren informasjon tilbake.

Undervisningsopplegget fulgte PRIMM-metoden, som forklart stegvis nedenfor.

```
1  print("Dette programmet vil beregne stigningstallet a til en lineær funksjon ved hjelp av topunktsformelen.")
2
3  x_1 = float(input("Oppgi x-koordin. til punkt 1: "))
4  y_1 = float(input("Oppgi y-koordin. til punkt 1: "))
5  x_2 = float(input("Oppgi x-koordin. til punkt 2: "))
6  y_2 = float(input("Oppgi y-koordin. til punkt 2: "))
7
8  a = (y_2 - y_1)/(x_2 - x_1)
9
10
11 print("Stigningstallet til funksjonen er", "a=" , a)
12
13 b = y_1 - x_1*a
14
15 print("Konstantleddet er b=", b)
16
17 print("Funksjonen er", "y=",a,"x","+",b)
```

Figur 3: Stigningstall og konstantledd til en lineær funksjon i Python

Predict:

Elevene ble først introdusert for koden på linje 1-11 (figur 3) på storskjerm. Elevene ble bedt om å parvis analysere og forutsi betydningen av hver linje med kode. Elevene samarbeidet med å tolke koden, og brukte egne ord til å forklare hverandre hva som kom til å skje.

Da klasserommet stilnet, ble linje for linje gjennomgått i fellesskap. Elevene delte sine tolkninger med resten av klassen, og læreren fylte inn og oppsummerte der forklaringene var mangelfulle. Det å analysere og forutse er første steg i algoritmisk tenkemåte, og samarbeid og deling er en av arbeidsmetodene som stimulerer algoritmisk tenkning (Utdanningsdirektoratet, 2019; Wing, 2006).

Run:

I dette steget ble elevene bedt om å finne frem egne PC'er, skrive inn kodelinjene og kjøre koden. Når elevene skulle skrive inn koden selv, ikke bare åpne et ferdig dokument, ble de nødt til å være nøye med notasjonen.

Investigate:

Dersom de ikke hadde skrevet inn alt riktig fikk elevene enten feilkode eller feil svar, og da måtte de gjøre feilsøk. Dette ga elevene mer eierskap til programmet, i tillegg til at feilsøking og rette opp i feil er en av arbeidsmetodene innen algoritmisk tenkning (Utdanningsdirektoratet, 2019) (Wing, 2006). I tillegg blir elevene tvunget til å dekomponere problemet, da de må feilsøke linje for linje for å få et program til å fungere som ønsket.

Modify:

Etter at elevene hadde fått programmet til å fungere på egne PC'er, og jobbet seg inn i programmet fikk de i oppgave om å utvide programmet. I dette steget fikk elevene i oppgave å utvide programmet slik at det kunne beregne konstantleddet til funksjonen. I tillegg ble elevene bedt om å få programmet til å skrive funksjonsuttrykket til skjerm.

Her diskuterte elevene hvordan de skulle løse det, både matematisk, hvordan de skulle bygge videre på de allerede definerte variablene, formelomskrivning, og hvordan benytte ulike datatyper til å skrive funksjonsuttrykket til skjerm.

Elevene brukte noen minutter på å legge en plan før de gikk i gang med selve programmeringen.

Make:

I dette steget må elevene iverksette planen de har lagt. De har utforsket et program, feilsøkt og designet en utvidelse, og nå skal de utvide programmet ved å produsere en ny funksjon.

I dette steget må elevene prøve seg frem, utforske, feilsøke, gå tilbake til tegnebrettet, holde ut og samarbeide om å utvikle et program som oppfyller ønskede kriterier. Dette er arbeidsmetoder som er inngår i prosessen algoritmisk tenkning.

Mens elevene arbeidet med programmet fikk de støtte og utfordring etter behov, i henhold til sosiokulturell læringsstil som PRIMM-metoden bygger på (Sentance et al., 2019).

Oppsummering og faglig læring

I tråd med Blomhøjs kriterier for utforskende undervisning, ble til slutt trådene samlet i plenum. Gjennom dialogbasert undervisning ble linje 13, 15 og 17 i figur 3 formulert. Det ble diskutert ulike feil som elevene hadde gjort, og hvordan de rettet dem opp. Elever som hadde løst oppgaven på andre måter fikk vise frem sitt program i plenum. Slik ble elevenes egne erfaringer og observasjoner knyttet til både matematikk- og programmeringskunnskap.

3.7.2 Flere eksempler fra programmeringsøkter

Alle undervisningsøktene foregikk etter prosedyren redegjort for i kapittel 3.7.1, med eksempel fra stigningstall og konstantledd for en lineær funksjon. Nedenfor presenteres flere eksempler fra programmeringsøkter i 1T gruppen, som også ble arbeidet med ved hjelp av PRIMM-metoden.

Andregradsformelen:

I en annen undervisningsøkt var målet å programmere 2. gradsformelen for å beregne røttene til et andregradsuttrykk (figur 4). Elevene ble introdusert for linje 1-11, hvor hver linje ble analysert først parvis, deretter i plenum. Særlig linje 11 ble det lagt vekt på å forstå.

```
1  from pylab import *
2
3  print("Programmet skal løse andregradslikningen ")
4  print("ax^2 + bx + c = 0 ved hjelp av abc-formelen.")
5  print("Skriv inn verdiene for a, b og c.")
6
7  a = float(input("a = "))
8  b = float(input("b = "))
9  c = float(input("c = "))
10
11 d = b**2 - 4*a*c
12
13 x_1 = (-b + d**(1/2))/(2*a)
14 x_2 = (-b - d**(1/2))/(2*a)
15
16 print("Løsningen på andregradslikningen er:")
17 print("x1 =", x_1, "og x2 =", x_2)
```

Figur 4: Beregning av røtter ved hjelp av andregradsformelen i Python

Elevene fikk deretter i oppgave å utvide programmet slik at de to røttene ble beregnet, og skrevet til skjerm.

Dette programmet ble igjen utvidet til å finne nullpunktene, ved å inkludere beregning av symmetriaksen basert på røttene til andregradsfunksjonen. Et eksempel på hvordan dette kunne løses er illustrert ved figur 5 nedenfor.

```
16
17 #Utvider med symmetriaksen for å finne ekstremalpunktene
18 x_s= -b/2*a
19
20 print("Symmetriaksen er i x=", x_s)
21
22 y_s=a*x_s**2 + b*x_s + c
23
24 print("Ekstremalpunktet til funksjonen er (", x_s, ",", y_s, ")")
25
```

Figur 5: Bestemme ekstremalpunkt ved symmetriaksen i Python

Noen elever mestret dette ganske kjapt, og synes ikke det var noe problem å hente aktuelle variabler, koble det til matematikken og anvende korrekte matematiske operatorer og notasjon

for å få programmet til å gjøre som de ønsket. Det var mange elever som fikk feilmelding når de kjørte programmet selv om de ikke hadde gjort noen feil rent kodemessig. Da måtte den matematiske forståelsen kobles til observasjonen, da det viste seg at programmet ikke fikk til å kjøre dersom løsningene var imaginære.

Da ble elevene utfordret til å definere ulike kriterier for ulike verdier. Det er i hovedsak 3 ulike aktuelle scenarier i denne sammenheng; to reelle løsninger, to like (men reelle) løsninger, og ingen reelle løsninger.

Elevene måtte først analysere hvordan de ulike tilfellene oppstod, deretter formulere dem og bruke løkker til å få programmet til å differensiere de ulike typene informasjon.

Figurtall

Elevene arbeidet også med å definere funksjoner i Python, og eksperimenterte med ulike anvendelser. En av måtene å arbeide med funksjoner var å lage funksjoner som beskrev ulike figurtall, og kunne generere en hvilken som helst n'te figurtall. Et eksempel på en funksjon som genererer et bestemt figurtall er illustrert i figur 6 nedenfor.

```
1
2  n=int(input("Skriv inn figurnummer: "))
3
4  def antall_ruter(n):
5      F_n = n**2 + 2*(n+1)
6      return F_n
7
8  print("Antall ruter i figur nummer ", n, "=", antall_ruter(n))
```

Figur 6: Pythonfunksjon som bestemmer antall ruter i et n'te figurtall

Her får elevene trening i å gjenkjenne mønstre, gå fra spesifikke observasjoner til et generelt uttrykk, og formulere algoritmen som funksjonen skal anvende for å generere et gitt figurtall. Underveis i prosessen inngår samarbeid, feilsøking, utforskning og vurdering av eget arbeid.

Plotte en funksjon som en samling av punkter

I arbeid med lister og å legge til nye elementer i lister virket det på elevene som at de synes det var helt overkommelig så lenge de jobbet med å legge til konkrete elementer som navn (se eksempel nedenfor, figur 7)

```
navneliste=[]
print("Foreløpig navneliste: ", navneliste)

for i in range(10):



    nyttnavn=str(input("Skriv inn et navn du vil legge til i lista: "))

    navneliste.append(nyttnavn)

    print("Oppdatert navneliste: ", navneliste)
```

Figur 7: Legge til navn i liste ved for-løkke i Python

Når bruk av lister skulle overføres til å bruke en liste med x-verdier til å generere tilhørende y-verdier som kunne plottes, ble det vanskelig. Det viste seg å være svært utfordrende for elevene å generalisere for en variabel, når variabelen skulle være hvert element i en liste. Det blir en form for dobbel abstrahering, der elevene både må formulere hva som er variabelen og hvilke algoritmer som skal appliseres, men også generalisere variabelen slik at programmet kan iterere gjennom alle x-variablene. Dette viste seg å være utenfor den proksimale utviklingssonen, som er grensen for hva eleven er i stand til å løse med støtte og veiledning (Lyngsnes, 2016).

```
from pylab import *

x_verdier=[-5, -3, 0, 1, 2, 10]
y_verdier=[]


for item in x_verdier:
    y=item**2 + 2*item
    y_verdier.append(y)

plot(x_verdier, y_verdier)
show()
```

Figur 8: Generere en liste med y-verdier ved å iterere gjennom en liste med x-verdier ved bruk av en for-løkke

Eksempelet i figur 8 demonstrerer et program som bruker en liste med x-verdier til å generere y-verdier ved å sende x-elementene gjennom funksjonen $f(x) = x^2 + 2x$. Deretter blir x-verdiene og y-verdiene plottet mot hverandre og tegner grafen basert på listene med x- og y-verdier. Akkurat mekanismen der for-løkker itererer gjennom hver *item* i en liste, var for mange elever vanskelig å forstå. Dette eksempelet er hentet fra den siste programmeringsøkten i undervisningsopplegget som varte 2 uker.

Eksemplene demonstrert ovenfor er ment å forklare hvordan PRIMM-metoden er brukt som mal for utvikling av undervisningsopplegg som benytter programmering i arbeid med et matematisk tema. Som redegjort for i teoridelen har PRIMM-metoden gitt gode resultater som undervisningsmetode for å lære elever programmering (Sentance & Waite, 2017), mens programmering er anerkjent å stimulere til algoritmisk tenkning (Aho, 2012). Algoritmisk tenkning er forbundet med relasjonell kunnskap, som handler om å være i stand til å anvende kunnskap på nye problemer fremfor å reprodusere løsninger, og for å gi elevene anledning til å utvikle relasjonell kunnskap er det i studien benyttet et utforskende undervisningsopplegg (Blomhøj, 2019).

3.8 Prestasjonstest

For å besvare forskningsspørsmålet

Har problemløsning med programmeringsoppgaver overføringsverdi til andre typer problemløsningsoppgaver innen matematikk?

ble det utarbeidet en prestasjonstest der elevene gjennomførte én i forkant av undervisningsopplegget og én i etterkant av undervisningsopplegget.

I henhold til Fischbeins (1994) teori om læring, består matematisk kunnskap av 3 komponenter; den formelle, algoritmiske og intuitive. Dermed var det ønskelig at oppgavene i prestasjonstesten la til rette for at elevene fikk demonstrere ulike komponenter av matematikk.

Prestasjonstesten inneholder 4 oppgaver hver, der ingen av oppgavene er like for å styrke validiteten. Dersom oppgavene var for like i pretest og retest risikerer en at elevene er i stand til å løse oppgavene fordi de har arbeidet med lignende før (instrumentell kunnskap), ikke fordi de har utviklet sin matematiske kompetanse. Mer om dette i *begrensninger*.

Oppgavene i prestasjonstesten gir elevene anledning til å demonstrere *formell kunnskap* ved korrekt bruk av matematiske definisjoner, teoremer, notasjon, aksiomer eller beviser. Den

algoritmiske komponenten kommer til uttrykk gjennom regneferdigheter og teknikker, som aritmetikk og anvendelse av algebra.

Oppgavesettet inkluderer også problemer som har flere riktige løsninger, og der det er mulig å bruke ulike strategier for å komme frem til løsningen(e). Dette betegnes som divergerende tenkning, eller divergerende produksjon som innebærer å generere flere korrekte løsninger til et problem. Analyse, tolkning, valg av strategi, mønstergjenkjenning og utforskning viktige komponenter innen algoritmisk tenkning, og inngår i det Fischbein (1994) betegner som den *intuitive komponenten.* Dette er også en demonstrasjon av matematisk kreativitet (Haylock, 1997).

Oppgave 1:

Vurder følgende påstand, og prøv å bevise eller motbevise den:

1) Summen av tre påfølgende tall er alltid delelig med tre
   (3 påfølgende tall = 3 tall som kommer etter hverandre)

2) Vurder også disse påstandene:
   - Summen av fire påfølgende tall er alltid/aldri/noen ganger delelig med 4
   - Summen av fem påfølgende tall er alltid/aldri/noen ganger delelig med 5
   - Summen av seks påfølgende tall er alltid/aldri/noen ganger delelig med 6

Ser du noen mønster mens du utforsker påstandene? Kan du lage en hypotese og undersøke den?

Figur 9: Oppgave 1 prestest - 3 påfølgende tall

Oppgaven i figur 9 ovenfor er hentet fra oppgaveheftet elevene fikk utdelt *før* undervisningsopplegget med programmering. Denne oppgaven kan løses med ulike tilnærminger, for eksempel ved å bruke algebra til å gi et generelt bevis, eller forklare ved

hjelp av tegning. Elevene kan forsøke å bevise/motbevise påstanden på flere ulike nivåer, for eksempel ved å føre et generelt bevis eller ved å prøve seg frem med spesifikke eksempler. I denne oppgaven får elevene demonstrert både problemløsningsstrategi, mønstergjenkjenning, gjøre observasjoner av det spesifikke, gjøre generaliseringer, utarbeide og anvende algoritmer, vurdere løsningen. Alle disse er essensielle komponenter i algoritmisk tenkemåte.

Oppgave 4:

Miriama har et kvadratisk vindu som består av 9 små glassruter inni. Hun ønsker å sette inn noen røde glassruter og noen blanke.

Hvordan kan Miriama sette inn ruter i vinduet sitt **uten** at 3 røde er på rad, samtidig som det **alltid** vil bli 3 røde på rad dersom hun setter inn en rød rute til? (Diagonal teller også som 3 på rad)

1) Hva er det minste antall røde ruter hun kan sette inn?
2) Hvor mange løsninger klarer du å finne? Tegn alle!

Figur 10: Oppgave 4 pretest - 9 glassruter

Oppgave 4 fra pretesten (fig. 10) er en åpen oppgave der det finnes mange gyldige løsninger. Den mest effektive metoden er at elevene oppdager et mønster, og systematisk genererer de mulige løsningene. Denne oppgaven gir elevene anledning til å demonstrere blant annet mønstergjenkjenning, utforming og anvendelse av en algoritme, systematisk problemløsning og divergerende tankegang.

Disse tenke- og arbeidsmetodene skal elevene få trening i gjennom programmeringsøktene, og det er interessant å se om det er observerbar utvikling fra pretest til retest. Se fullstendige oppgavesett i vedlegg 5 og 6.

3.9 Pilot

I forkant av datainnsamlingen vil det bli gjennomført en pilot av prestasjonsmålingene for å få en indikasjon på hva slags svar elevene ville komme frem til, om det var oppgaver i heftet som var uklart formulert og eventuelle andre problemer.

Piloten som ble gjennomført var pretesten, altså det oppgaveheftet elevene vil bli tildelt i forkant av undervisningsopplegget. Piloten ble gjennomført i en R2-klasse ved samme skole som 1T-klassen resten av datainnsamlingen vil basere seg på.

Årsaken til at piloten ble gjennomført i en R2-klasse var i hovedsak på grunn av tilgjengelighet og praktiske muligheter for gjennomføring på skolen. Utfordringene med å kjøre pilot i en R2 klasse er at elevene i T har mindre matematisk modning enn elevene i R2. De har mindre trening i å uttrykke seg ved hjelp av matematisk notasjon og har kommet kortere i utdanningsløpet.

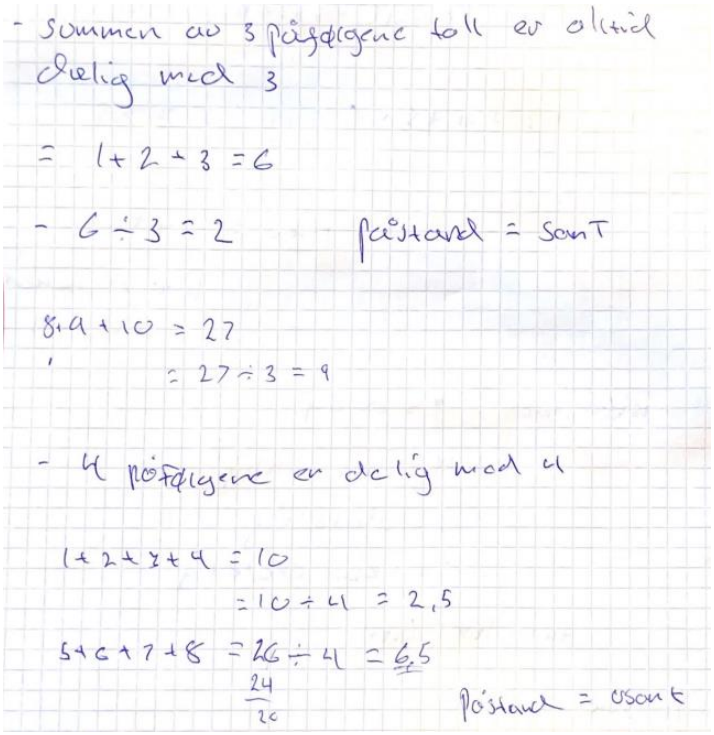
På den annen side finnes det også fordeler med å gjennomføre piloten i en R2-klasse i stedet for i en 1T-klasse. For eksempel blir det tydelig at testen er for lang eller for vanskelig dersom R2-elevene sliter med å gjennomføre innen tidsrammen eller sliter med å tyde oppgavene. I tillegg har R2-elevene mer erfaring både med matematikkfaget, ulike arbeidsmetoder og testsituasjoner. Det er derfor rimelig å anta at R2-elever er bedre rustet til å gi tilbakemelding på oppgavene og selve testen, om det var uklare formuleringer eller andre utydeligheter i oppgaveheftet.

3.9.1 Resultater: Pilot

Oppgave 1

I den første oppgaven i oppgaveheftet ble elevene bedt om å bevise eller forsøke å motbevise følgende utsagn:

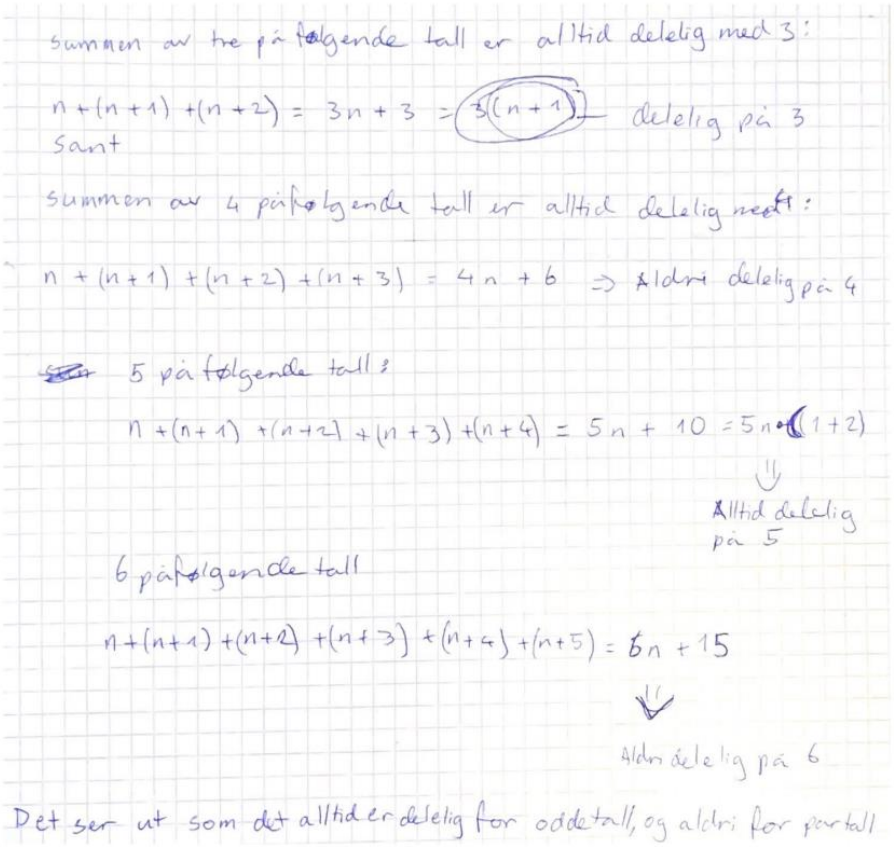
«Summen av 3 påfølgende tall er alltid delelig med 3».



Figur 11: Eleveksempel 1 oppgave 1 pilot

I eleveksempelet i figur 11 har eleven benyttet konkrete tall for å vurdere om påstandene er sanne eller usanne. Dette er et tilfelle av *naiv empirisme*, hvor elever bruker enkeltstående eksempler til å trekke generelle konklusjoner om at dette er gjeldende for alle tilfeller.

Et annet eleveksempel fra den samme oppgaven viste derimot mer avansert argumentasjonsform (figur 12).



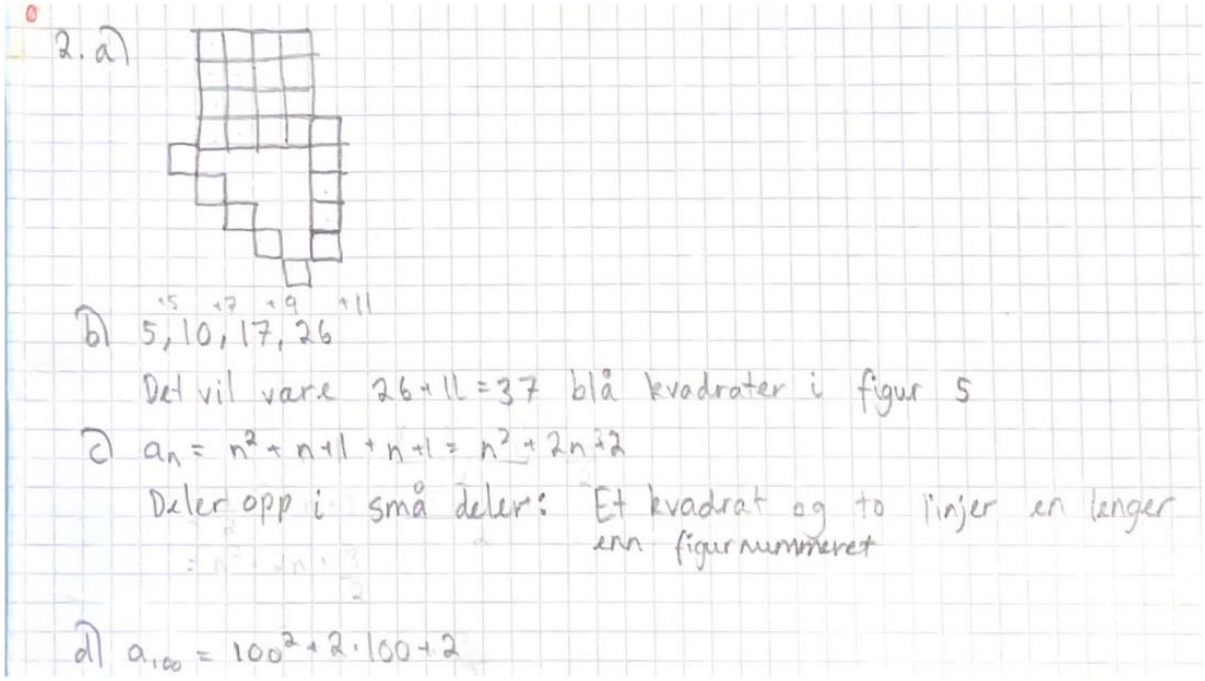
Figur 12: Eleveksempel 2 oppgave 1 pilot

I eleveksempelet ovenfor har eleven derimot lyktes med å konstruere et generelt bevis som forklarer at 3 påfølgende tall alltid vil være delelig med 3, i likhet med at 5 påfølgende tall vil være delelig med 5. Eleven argumenterer også algebraisk for hvorfor ikke 6 påfølgende tall alltid er delelig med 6. Til slutt presenterer eleven en hypotese om at påstanden er gjeldende for et odde antall påfølgende tall, men ikke for partall. Dette er igjen naiv empirisme, basert på empiriske observasjoner gjort for 4 ulike tilfeller av påfølgende tall.

Eleven har beveget seg fra det spesielle tilfellet illustrert i oppgaveteksten med de 3 påfølgende tallene 6 + 7 + 8 = 21 som er delelig med 3, til å lage et generelt uttrykk som viser at hvilke som helst 3 påfølgende tall er delelig med 3. Dette er en demonstrasjon av algoritmisk tenkning, som er nettopp det studien har som mål å undersøke.

3.9.2 Oppgave 2

I oppgave 2 ble elevene bedt om å tegne den neste figuren basert på 3 figurer med samme mønster, og deretter utlede et generelt uttrykk som kunne brukes til å bestemme antall ruter i figur n.

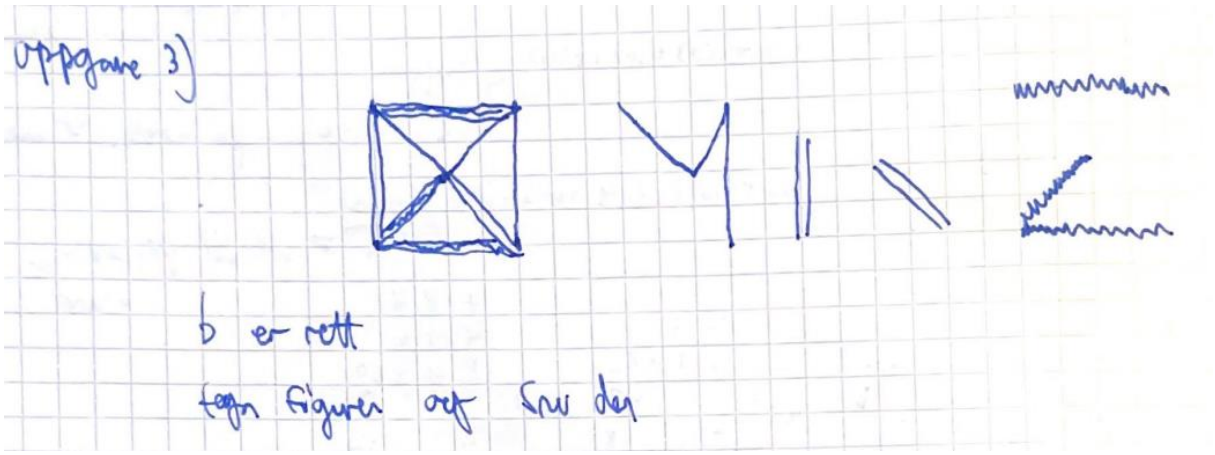


Figur 13: Eleveksempel oppgave 2 pilot

I eleveksempelet i figur 13 forklarer eleven hvordan dekomponert figuren for å kunne lage et generelt uttrykk som beskriver den. Dekomponering av et problem er en del av en problemløsningsstrategi som inngår i algoritmisk tenkning.

3.9.3 Oppgave 3:

I oppgave 3 ble elevene bedt om å gjenkjenne en figur først sett fra siden når den ble vist ovenfra.

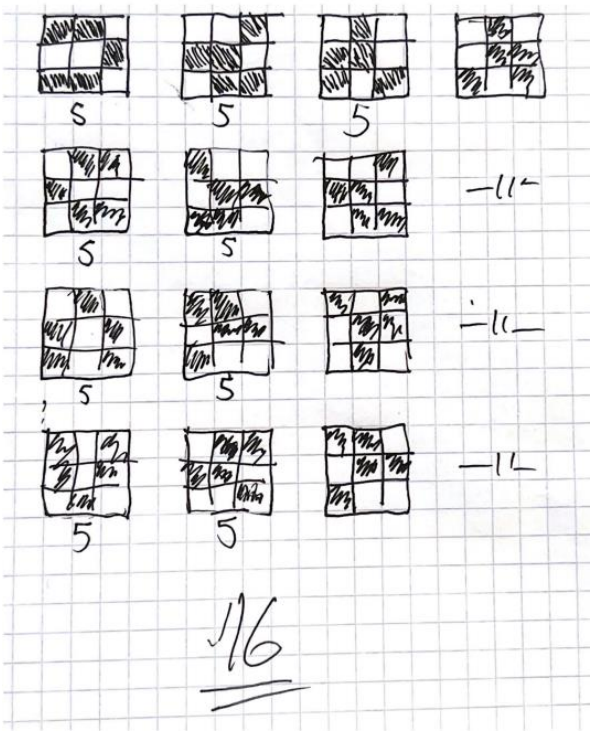


Figur 14: Eleveksempel oppgave 3 pilot

I dette eleveksempelet (fig 14) har eleven dekomponert selve figuren, og fokusert på stillingen og tilkoblingen av de sorte linjene, de hvite og de grå hver for seg. Dette resonnementet viser at eleven har brutt ned problemet i mindre delproblemer, for å forenkle oppgaven. Dette er en kognitiv aktivitet som inngår i algoritmisk tenkning.

3.9.4 Oppgave 4

Oppgave 4 gikk ut på å plassere røde glassruter i et kvadratisk vindu bestående av 9 ruter. De røde rutene må plasseres uten at det er 3 røde på rad, samtidig som det **alltid** vil bli 3 røde ruter på rad dersom de setter inn en rute til. Elevene bedt om å tegne alle løsningene de kunne finne, og hva som var minste antall røde ruter de kunne bruke.



Figur 15: Eleveksempel oppgave 4 pilot

En av elevene demonstrerer i besvarelsen en strategi for å finne flest mulige løsninger (figur 15). Eleven finner én løsning som oppfyller betingelsene, og roterer deretter den samme strukturen for å få alle utgavene av løsningen. Deretter går eleven videre til en annen sammensetning som oppfyller kriteriene. Her har eleven lagd egne regler for hvordan løse problemet. Dette er å anvende en algoritme, som også er en del av UDIRs definisjon av algoritmisk tenkemåte.

3.9.5 Oppsummering

Pilotgjennomføringen av pretesten ga flere interessante funn. Elevene ga praktiske tilbakemeldinger på selve oppgaveteksten, og spørsmålene de stilte underveis mens de gjennomførte testen avdekket flere ting i oppgaveteksten som kunne justeres og formuleres tydeligere.

En annen ting piloten avdekket var at i elevenes besvarelser var det flere ulike tegn til algoritmisk tenking og anvendelse av problemløsningsstrategier som kjennetegner algoritmisk tenkning i henhold til Utdanningsdirektoratets (2019) modell som denne oppgaven bruker som base for å definere begrepet algoritmisk tenkning.

Aktuelle justeringer ble gjort i oppgaveheftet, og det konkluderes med at de valgte oppgavene har potensiale til å skaffe datamateriale som gjør det mulig å observere elevenes resonnering, hvordan de argumenterer, mønstergjenkjenning, problemløsningsstrategier, abstrahering og generalisering. På bakgrunn av dette vil endringen i elevers algoritmiske tankevirksomhet bli undersøkt i denne studien.

3.10 Studiens kvalitet

For å sikre en masteroppgave med god kvalitet er det to begreper som er viktig å ta i betraktning: *reliabilitet* og *validitet.*

Reliabiliteten til en studie og et forskningsresultat avhenger av kvaliteten på selve forskningsprosessen (Gleiss & Sæther, 2022). Som forsker må en være klar over og kunne redegjøre for hvordan datamaterialet blir påvirket av måten det er blitt samlet inn på. Ideelt sett skal det være mulig for en annen forsker å gjenskape de samme resultatene med datainnsamling på et annet utvalg. For å oppnå dette må forskeren være bevisst på å være så objektiv som mulig for at ikke datainnsamlingen skal farges av *bias* (Gleiss & Sæther, 2022). Et tiltak som kan gjøres for å styrke reliabiliteten til en studie er å samle inn data ved hjelp av ulike metoder for å kontrollere at funnene er konsistente. Dette vil diskuteres ytterligere i metodedelen av oppgaven.

Validiteten av en studie er også med på å definere dens kvalitet. Validiteten defineres ut ifra kvaliteten på det innsamlede datamaterialet, sammen med de fortolkninger og konklusjoner forskeren trekker (Gleiss & Sæther, 2022). For å sikre god validitet i et forskningsprosjekt kreves det gode avgjørelser i valg av metode og forskningsdesign for å lykkes med å faktisk måle det studien har som formål å undersøke.

I sammenheng med validitet er det naturlig å diskutere *operasjonalisering*, som handler om hvor vellykket oversettelsen av teoretiske begreper er til konkrete faktorer som lar seg observere (Gleiss & Sæther, 2022).

Eksempler på to sentrale teoretiske begreper i dette forskningsprosjektet er *algoritmisk tenkning* og *læring*. Ingen av disse begrepene kan observeres direkte eller kvantifiseres, og må derfor *operasjonaliseres* for å gjøre datainnsamling mulig. I denne oppgaven er det gjort avgrensninger og operasjonaliseringer som vil ha påvirkning på både validiteten og reliabiliteten til resultatene.

Et eksempel på en viktig antakelse gjort som vil påvirke **validiteten** til masteroppgaven er at elevene ikke har sett oppgavene i prestasjonsmålingen tidligere. Det antas at elevene ikke har sett oppgavene før, og gjennomfører sin egen utforskning og bruker egne løsningsstrategier. Dersom elevene har sett oppgavene før måler ikke testen algoritmisk tenkning, men om elevene har lært fra å ha gjort liknende oppgaver tidligere. I så fall vil en positiv utvikling i resultatene ikke være annet enn en kausalitet.

Et eksempel på et aspekt som vil ha betydelig påvirkning på studiens reliabilitet er **bias.** Studien er en casestudie som utføres på få individer, som vil si at hvert individ vil ha stor påvirkning på datamaterialet. Bias kan forekomme i form av endret atferd på grunn av forskerens tilstedeværelse, eller for eksempel på grunn av måten spørsmål blir stilt på i intervju (Gleiss & Sæther, 2022). Dermed er det ikke rimelig å anta at funnene ville være reproduserbare i en vilkårlig 1T-gruppe.

3.11 Etiske betraktninger

Forskningsetikk handler om å ivareta både *sannhetsnormen*, som innebærer sannhetsforpliktelse, åpenhet og redelighet om forskningsprosess og resultater (De nasjonale forskningsetiske komiteene, 2021). Et viktig aspekt ved forskning som involverer mennesker er å sørge for en trygg forskningsprosess som ivaretar menneskenes trygghet og velferd. Det innebærer blant annet at alle deltakere i et forskningsprosjekt skal ha gitt et informert og frivillig samtykke.

Dette forskningsprosjektet er innmeldt og godkjent av NSD (norsk senter for forskningsdata). NSD anbefaler at barn under 15 som gir sitt samtykke også må ha samtykke fra foresatte. Ettersom alle elevene som ble invitert til å delta i studien hadde fylt 16 år ble samtykke samlet direkte fra elevene.

Et frivillig samtykke krever at deltakelse i studien må være basert på et autonomt valg, fritatt fra ytre press eller begrensning av valgfriheten (De nasjonale forskningsetiske komiteene, 2021). Som både forsker og 1T-klassens faglærer var det avgjørende å tydelig kommunisere at deltakelsen i studien var frivillig og at forskningsprosjektet var organisert helt uavhengig av 1T faget. Det innebar å avklare at om elevene valgte å delta i studien eller ikke ville det på ingen måte påvirke relasjon mellom lærer og elev, læringsutbytte i faget eller prestasjon og utvikling i faget. Det ble understreket at samtykket var frivillig, som innebar at de på hvilket som helst tidspunkt kunne trekke samtykket sitt og be om å få opplysninger slettet såfremt det ikke allerede var publisert.

For å kunne gi et informert samtykke må elevene være i stand til å forstå hva forskningsprosjektet og konsekvensene av deltakelse innebærer, blant annet i henhold til personvernsopplysninger (De nasjonale forskningsetiske komiteene, 2021). Først ble informasjon avgitt muntlig om bakgrunn for studien, oppbygning av selve forskningsprosjektet, samt hvordan opplysningene ville bli håndtert og oppbevart underveis og i etterkant av forskningsprosjektet. Deretter fikk elevene utdelt samtykkeskjemaet med samme informasjon skriftlig (vedlegg 2).

Anonymitet er som avtalt ivaretatt for å beskytte deltakernes identitet og integritet (De nasjonale forskningsetiske komiteene, 2021). Ingen elever er identifiserbare, og det er heller ikke mulig å spore testresultater eller besvarelser til spesifikke elever. Under databehandlingen var opplysningene pseudoanonymiserte, som vil si at forskeren og biveileder kunne identifisere elever gjennom en koblingsnøkkel, som slettes når forskningsprosjektet er avsluttet.

4 Analyse

Med formål om å besvare forskningsspørsmålene formulert i problemstillingen (kap 1.1) er det blitt samlet inn datamateriale i en 1T-gruppe på ulike måter; spørreskjema, åpent spørreskjema, prestasjonstest og intervju.

Teorien om TPACK foreslår mixed methods med både kvantitativ og kvalitativ datainnsamlingsmetode (Koehler et al., 2012). På grunn av avgrensningene som måtte gjøres for å tilpasse studien til en mastergradsavhandling er dette en casestudie, med 21 deltakende individer. I spørreskjemaet var det kun 20 elever som deltok, da det var en elev var fraværende dagen spørreundersøkelsen ble gjennomført.

Utvalget i spørreskjemaene består av 20 individer, som er et relativt lite utvalg. Det vil si at 1 elev tilsvarer 5%, som medfører at hver enkelt elevs besvarelse vil utgjøre et betydelig utslag i det samlede datamaterialet. Dette gjør det vanskelig å trekke konklusjoner om trender i utvalget som vil være gjeldende for andre utvalg, til tross for at dataen er innsamlet ved kvantitativ metode. Spørreundersøkelsen kan likevel brukes til å danne et oversiktsbilde av klassens samlede opplevelser og holdninger knyttet til programmering i matematikk.

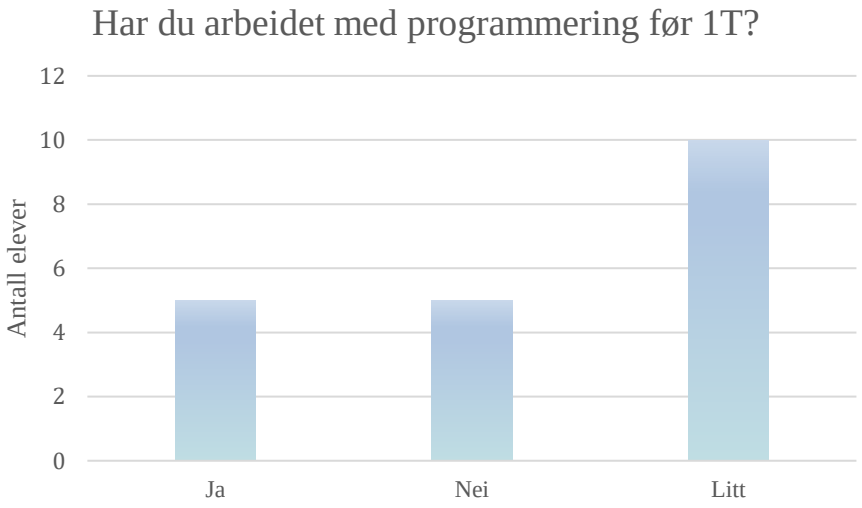
Det ble gjennomført en induktiv koding av besvarelsene i det åpne spørreskjemaet, som innebærer koding av observasjoner i datamaterialet som var relevante for forskningsspørsmålene. Eksempler på interessante observasjoner er uttrykk for både positive og negative følelser tilknyttet programmering, problemløsningsstrategier, dekomponering, generalisering, feilsøking, kreativitet og samarbeid.

Som diskutert i metodedelen var prestasjonstestene laget ulike for å unngå at elevene demonstrerte instrumentell kunnskap fordi de hadde sett oppgavene/oppgavetypen tidligere. Det er dermed vanskelig å poengsette og sammenligne pretest og retest direkte. Derfor viste det seg å være mer hensiktsmessig å kvalitativt analysere prestasjonstestene, og se etter indikatorer på algoritmisk tenkning og forbedringer i elevenes matematiske resonnementer. For å besvare problemstillingen om bruk av programmering kan fremme algoritmisk tenkning hos elever, ble elevbesvarelsene fra pretest og retest sammenlignet og vurdert i henhold til Hoviks (2016) rammeverk for matematisk argumentasjonsnivå.

Spørreskjemaene (åpent og lukket), prestasjonstest og intervju vil sees i sammenheng og forskningsspørsmålene forsøk besvart så nært sannheten om virkeligheten som mulig. For å

kunne gå i dybden med analysen er det blitt valgt ut 4 individ er som ble invitert til å delta i intervju basert på besvarelsene i spørreskjema og prestasjonstest, samt at det ble gjort observasjoner under gjennomføring av undervisningsopplegget. Intervjuene ble transkribert og lest gjennom flere ganger. Deretter ble det foretatt en induktiv koding der relevante observasjoner ble klassifisert, og tolket i sammenheng med øvrig datamateriale.

Fra besvarelsene i spørreskjemaet kommer det frem at programmering er nytt for elevene, som presentert i figur 16.



Figur 16: Elevers erfaring med programmering før 1T

Fra resultatene presentert i figur 16 fremkommer det at flertallet i klassen har liten erfaring med programmering fra tidligere. 50% (n=10) av elevene svarer at de har arbeidet litt med programmering før, mens 25% (n=5) oppgir at de ikke har arbeidet med programmering før 1T. Kun 25% (n=5) har arbeidet med programmering tidligere. Det vil si at flertallet av elevene i klassen lærer seg programmering ordentlig for første gang gjennom 1T-kurset, og vil ha betydning for funnene i studien.

Elevene som ble intervjuet vil først bli introdusert med anonymiserte navn, deretter vil funnene fra intervjuene presenteres sammen med relevante funn fra resten av datamaterialet.

Jonas var den i klassen som hadde lengst erfaring med programmering, hovedsakelig fra egen interesse og initiativ. Jonas har erfaring med programmering fra kodeklubb, Vitensenteret og erfaring fra Lego League som både deltaker og veileder. Jonas gir uttrykk for at han liker å jobbe med programmering, og synes det er gøy å utfordre seg med programmering.

Sofie har kun jobbet litt med blokkprogrammering på ungdomsskolen og synes det var mye nytt å sette seg inn i og synes ofte det er mer strev enn utbytte når det kommer til programmering. Thea har kun jobbet litt med programmering den siste uken på ungdomsskolen, men har lyst til å lære seg å bruke det og ser at det kan være et nyttig hjelpemiddel. Magnus har heller ikke vært borti Python før 1T, men synes det er gøy å jobbe med det og ser overføringsverdien det kan ha til matematikk.

Følgende funn ble gjort i analysen av datamaterialet:

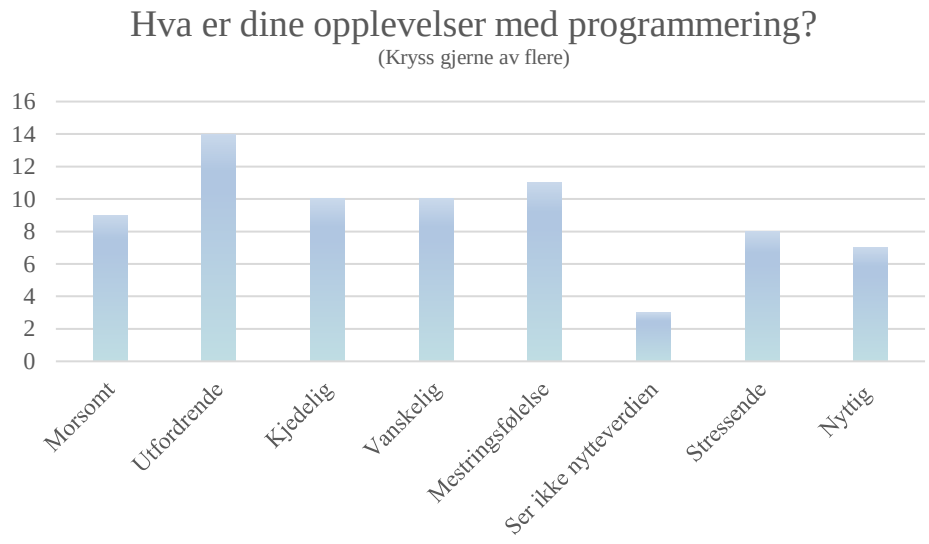
1. Elevens syn på programmering
    - Basert på intervju, åpent og lukket spørreskjema
2. Hva gjør elevene når de programmerer
    - Hva sier elevene selv i åpent spørreskjema, intervju og hva kommer frem gjennom observasjon
3. Hva hjelper elevene når de programmerer
    - Kreativitet og samarbeid basert på åpent spørreskjema og intervju
4. Hva hindrer elevene når de programmerer
    - Begrenset kreativitet, tid og terskelbegrep basert på åpent spørreskjema og intervju
5. Elevenes syn på sammenheng mellom programmering og matematikk
    - Basert på intervju, åpent og lukket spørreskjema
6. Hvordan algoritmisk tenkning viser seg i arbeid med matematikk
    - Argumentasjon og resonnering i prestasjonstestene

I analysen vil disse funnene redegjøres for med observasjon og relevant teori. Validiteten og reliabiliteten til resultatene vil diskuteres i diskusjonsdelen av oppgaven.

4.1 Elevenes syn på programmering og opplevelser med programmering i skolen - Basert på intervju, åpent og lukket spørreskjema

Først vil resultatene fra det lukkede spørreskjemaet presenteres for å danne et oversiktsbilde av elevenes opplevelser tilknyttet programmeringsarbeid, undervisningsopplegget de deltok i, følelser og holdninger overfor programmering, hvorvidt de ser nytteverdien og opplever mestring.

Fra resultatene presentert i figur 16 fremkommer det som nevnt at flertallet i klassen har liten erfaring med programmering fra tidligere. Det vil si at flertallet av elevene i klassen lærer seg programmering ordentlig for første gang gjennom 1T-kurset.



Figur 17: Elevers opplevelser med programmering fra lukket spørreskjema

Figur 17 ovenfor viser en oversikt over de ulike opplevelsene elevene har knyttet til programmering. Elevene fikk muligheten til å krysse av flere svaralternativer for å få uttrykt alle følelsene de har tilknyttet undervisningsopplegget. De fleste elevene i spørreundersøkelsen har valgt flere svaralternativer, som kanskje sier noe om at programmering krever mye av dem.

De ulike svaralternativene deles inn i 3 kategorier:

1. Utfordrende, vanskelig og stressende:

De mest fremtredende resultatene i spørsmålet er at 70% (n=14) av elevene svarer at de opplever programmering som *utfordrende*, og *vanskelig* og *stressende* har også høy svarandel. men det er flere beskrivende ord som har høy svarandel.

Blant elevene som har programmert litt eller ikke i det hele tatt før 1T (75%, n=15), svarer 93.3% (n=14) at de opplever programmering som enten utfordrende, stressende eller vanskelig. Dette indikerer en sammenheng mellom liten erfaring med programmering og en opplevelse av programmering som krevende.

2. Morsomt, mestringsfølelse og nyttig

Mestringsfølelse og morsomt har også ganske høy oppslutning i spørreskjemaet, med henholdsvis 50% (n=10) og 45% (n=9).

Diagrammet i figur 18 fremstilles oppslutningen av de ulike opplevelsene med programmering, gitt at elevene har svart at de opplever mestringsfølelse:

Figur 18: Elevers opplevelser med programmering gitt mestringsfølelse (lukket spørreskjema)

Som fremstilt ved diagrammet oppgir elevene at de har både positive og negative assosiasjoner til programmering til tross for mestringsfølelse. Blant de 10 elevene som oppgir mestringsfølelse oppgir 7 av dem at de synes programmering er morsomt, og 8 av dem synes det er utfordrende. Henholdsvis 5 og 4 elever synes programmering er kjedelig og stressende, selv om de opplever mestringsfølelse.

3. Kjedelig og ser ikke nytteverdien

De eneste observasjonene som ikke sammenfaller er *«ser ikke nytteverdien»* og *«morsomt»*. Blant de 3 elevene som har oppgitt at de ikke ser nytteverdien av programmering, er det ingen av dem som har valgt alternativet «morsomt».

Det er ikke noen klare trender blant ord som skiller seg ut da elevene rapporterer at de har flere ulike opplevelser tilknyttet arbeid med programmering. For eksempel er det flere elever som oppgir at de opplever programmering som både morsomt og kjedelig. Det resultatene derimot kan tolkes som, er at det er enighet blant elevene om at programmeringsarbeid krever mye av dem. Resultatene indikerer at opplevelsene med programmering er sammensatt, tidvis positive og tidvis negative. Kun 2 av elevene har valgt kun ett alternativ, mens de resterende 18 har valgt flere alternativ, uten at det avdekkes noen tydelige grupperinger.

Disse sammensatte opplevelsene elevene har kan sees i sammenheng med at programmering krever flere kognitive prosesser og flere nivåer av abstraksjon (Wing, 2006). Elevene må både oppdage mønster eller sammenhenger i det de arbeider med, formulere sammenhengene presist slik at det er mulig å lage et program som kan skille de ulike datatypene. I tillegg må elevene bryte problemet ned i flere delproblemer for å kunne stegvis lage algoritmer som datamaskinen kan anvende for å løse problemet (Gjøvik & Torkildsen, 2019).

I tillegg til at programmering er nytt for de fleste elevene er det flere komponenter som skal mobiliseres på likt, både matematikkfaglig kunnskap, algoritmisk tenkning og kodeferdigheter. Dette gjenspeiles i at det store flertall av elever i spørreundersøkelsen oppgir at de opplever programmering som f.eks både utfordrende, frustrerende, morsomt og kjedelig. Det er mange faktorer som avgjør elevenes oppfatning av et undervisningsopplegg, både sosiale og faglige, som gjør det vanskelig å isolere årsak og virkning.

Tabell 1 viser den prosentvise fordelingen på svaralternativene på noen utvalgte utsagn (fullstendig spørreskjema i vedlegg 3). Resultatene presentert i tabellen vil i kommende delkapittel (4.2) brukes for å utdype hva elever gjør når de arbeider med programmering, og på hvilken måte det krever mye av dem.

Tabell 1: Prosentvis svarfordeling av elevers opplevelser med programmering i matematikkundervisning

	Utsagn	Helt uenig (%)	Litt uenig (%)	Litt enig (%)	Helt enig (%)
1	Jeg opplever mestring i arbeid med programmering	15	10	65	10
2	Jeg opplever ingen mestringsfølelse i arbeid med programmering	30	40	20	10
3	Programmering gjør meg mer motivert til å arbeide med matematikk	35	40	20	5
4	Jeg synes programmering er frustrerende	10	25	30	35
5	Jeg er usikker på hvordan jeg skal gå i gang med programmeringsoppgaver	10	20	45	25
6	Jeg har en god strategi for å arbeide med programmering	25	45	25	5
7	Å jobbe med programmering gir meg muligheten til å tenke kreativt	10	45	15	30
8	I programmering er det mange ulike løsninger som er riktige	5	5	60	30
9	Programmering passer bra inn i 1T-faget	5	20	35	40
10	Jeg ser ikke sammenhengen mellom matematikken vi lærer og programmeringen vi gjør	35	50	10	5
11	Programmering kan brukes i alle delemnene av 1T	0	35	60	5
12	Når jeg sliter med en programmeringsoppgave prøver jeg ut ulike strategier for å komme i mål	5	30	35	30
13	Jeg synes det er nyttig å lære programmering	15	30	30	25
14	Jeg føler ikke jeg lærer noe av å arbeide med programmering	20	50	25	5
15	Jeg foretrekker andre arbeidsmåter innen matematikk	5	30	30	35
16	Jeg synes programmering tar for stor plass i matematikkfaget	10	50	35	5
17	Jeg kan bruke programmering til å løse matematiske problemer	10	5	50	35

4.2 Hva gjør elevene når de programmerer: - Hva sier elevene selv i åpent spørreskjema, intervju og hva kommer frem gjennom observasjon

I påstand 5 og 6 (tabell 1) oppgir henholdsvis 70% av elevene oppgir at de er usikre på hvordan de skal gå frem med en programmeringsoppgave, og 70% oppgir at de mangler en god strategi for å løse programmeringsoppgaver. Til forskjell fra prosessorienterte regneoppgaver der det holder med instrumentell kunnskap, og elevene kun skal reprodusere en løsning, krever programmeringsoppgaver langt mer av dem. Det finnes ikke et sett med regler elevene kan anvende for å komme i mål med en kompleks problemløsningsoppgave.

Når elevene skal løse en programmeringsoppgave krever det anvendelse av relasjonell kunnskap, og et samspill mellom de 3 komponenter som Fischbein (1994) definerer som kreativ, aktivitetsbasert matematikkpraksis. I møte med problemet kreves den intuitive komponenten for å analysere og forstå problemet, velge problemløsningsstrategi og legge en plan for hvordan løse problemet.

Videre må elevene benytte sin formelle matematikkunnskap, for å være i stand til å korrekt løse problemet presentert i oppgaven. I undervisningsopplegget med programmering elevene har arbeidet med var temaene likninger, algebra, faktorisering og funksjoner sentral kunnskap. I tillegg til formell og intuitiv komponent, må elevene mobilisere den algoritmiske komponenten, som det matematiske håndtverk. Elevene må kunne benytte ulike teknikker og regneferdigheter, og klare å bryte ned problemet i mindre delproblemer, slik at de kan formulere et program som besvarer oppgaven.

Slik det som kreves av elevene i arbeid med programmering er beskrevet, krever programmering både algoritmisk tenkning og matematisk kreativitet. Der algoritmisk tenkning fungerer som en problemløsningsstrategi som mobiliserer de 3 komponentene Fischbein definerer som *matematisk kompetanse*, er matematisk kreativitet viktig for fremdrift i arbeidet.

Noen av arbeidsmåtene innen algoritmisk tenkning er å holde ut, prøve nye tilnærminger og løsninger, utforske og eksperimentere. I arbeid med programmering må elevene prøve seg frem med ulike metoder for å løse problemet, feilsøke koden sin, tilpasse feilene, prøve noe

nytt. Evnen til å overkomme fikseringer er en form for matematisk kreativitet. 65% av elevene rapporterer at de prøver ut ulike strategier for å komme i mål med en oppgave (påstand 12), som vil si at flertallet av elevene rapporterer at de forsøker å ikke bli fiksert på en spesiell metode, tema eller strategi. Dette er en form for fleksibel tankegang og evne til å overkomme fikseringer, som indikerer matematisk kreativitet (Haylock, 1997). Dette er også et tegn på algoritmisk tenkning, i form av at elevene feilsøker og vurderer løsningene sine, og holder og ut og fortsetter arbeidet med å utvikle programmene sine.

En annen interessant observasjon er at hele 90% av elevene sier seg enige i at det i programmering finnes mange ulike gyldige løsninger (påstand 8). Dette er en indikasjon på divergent tankegang, der elevene har oppdaget at det finnes mange riktige løsninger til samme problem, enten fra egne erfaringer eller i dialog og samarbeid med andre elever. Divergent produksjon er også en form for matematisk kreativitet (Haylock, 1997).

Likevel er det færre som opplever at de tenker kreativt gjennom programmering, hvor kun 30% er helt enig, og 15% litt enig med utsagnet (påstand 7). Dette strider imot det elevene har oppgitt i påstand 8 og 12. Det kan være mange årsaker til dette, en mulig forklaring kan være at elevene har tolket spørsmålet basert på at de har andre assosiasjoner til ordet *kreativitet* enn det brukes i matematisk sammenheng.

4.2.1 Elevenes fremgangsmåter i arbeid med programmering

I dette delkapittelet vil det bli redegjort for de ulike kognitive prosessene elevene beskriver at de benytter i arbeid med programmering. De kognitive prosessene elevene oppgir at de engasjerer seg i gjennom åpne spørreskjema, intervju og som kommer frem i observasjoner inngår i algoritmisk tenkning. For å presentere datamaterialet som argumenterer for dette funnet vil observasjonene deles inn i kategorier som er komponenter av algoritmisk tenkning.

- Strategi
- Analyse
- Algoritmer
- Dekomposisjon
- Mønstergjenkjenning
- Abstraksjon
- Feilsøking og vurdering

Funnet om at programmering krever algoritmisk tenkning er basert på resultatene fra spørreskjemaene (åpent og lukket), intervjuene og observasjoner av elevarbeid i det 2 uker lange undervisningsopplegget.

Resultatene fra prestasjonstesten er bedre egnet til å si noe om det matematiske læringsutbyttet, da resultatene ikke kan knyttes direkte til programmering. Dette diskuteres ytterligere i delkapittel 5.4.

Fra det lukkede spørreskjema kommer det tydelig frem at elevene synes selve problemløsningen er utfordrende. Som fremstilt i figur 19 oppgir flertallet av elevene at de mangler en god strategi for å jobbe med programmering.

Figur 19: Elevers oppfatning av egen strategi i arbeid med programmering

Både i det åpne og lukkede spørreskjema, intervju og kommer det frem at elevene synes terskelen for å komme inn i programmeringsarbeidet er høy.

I motsetning til matematikkoppgaver som er prosessorienterte og kun krever at elevene demonstrerer instrumentell kunnskap, krever programmering at elevene benytter sin relasjonelle kunnskap. Elevene må mobilisere både formell, algoritmisk og intuitiv komponent for å komme i mål med programmeringsoppgaven.

I arbeid med programmering og utforskende oppgaver i matematikk finnes det ikke noe sett med regler som kan brukes for å komme frem til løsningen. Det nærmeste en kommer er en problemløsningsstrategi.

I undervisningsopplegget ble PRIMM-modellen brukt for å strukturere undervisningen og for å øke elevenes bevissthet rundt problemløsningsstrategi. I det åpne spørreskjemaet blir elevene spurt om hvordan de går frem når de skal løse en oppgave ved hjelp av programmering. Mange av elevene nevner PRIMM eksplisitt, eller beskriver hele eller deler av problemløsningsstegene. En elev skriver følgende:

«Først ser jeg hva som man må løse. Videre skriver jeg kanskje ned koden i et vanlig språk, for å få et bilde av hva som må skje i koden. Deretter skriver jeg koden i programmeringsspråket.»

Eleven iverksetter flere av de kognitive prosessene som inngår i algoritmisk tenkning, først analyse og tolkning av oppgaven. Deretter dekomponerer eleven problemet, formulerer en algoritme som kan brukes av en datamaskin for å utføre en oppgave. Deretter prøver eleven å oversette sin matematiske løsning til et program som kan kjøres av en datamaskin.

Den samme eleven svarer at det de synes er vanskeligst med programmering er *«når man har en idé, og man ikke klarer å finne ut hvordan man kan løse det.»* Fra det åpne spørreskjemaet og observasjonene gjort av elevene underveis i undervisningsopplegget og i arbeid med programmering, virker dette å være en utfordring for mange.

Mange av elevene har forstått at de må tolke hva oppgaven ber dem om og se for seg hva som er målet med koden de skal lage. De forstår at de må gi datamaskinen konkrete, stegvise instrukser, og de forstår matematikken bak programmeringsoppgaven.

Likevel opplever mange elever at de stagnerer, og ikke klarer å jobbe seg videre. Dette kan forklares med at algoritmisk tenkning, og de kognitive prosessene analysering, dekomponering, mønstergjenkjenning, abstrahering og feilsøking er en del av matematisk tenkemåte og matematiske ferdigheter som utvikles over tid. Å gjøre elevene bevisst på problemløsningsstrategien PRIMM som inviterer til algoritmisk tenkning (Sentance & Waite, 2017) er noe annet enn å lære elevene hvordan de beregne nullpunkter ved hjelp av andregradsformelen. Førstnevnte krever læring gjennom utforskning, samarbeid, dialog og rom til å prøve og feile (Blomhøj, 2019), mens sistnevnte kun krever instrumentell kunnskap.

4.2.2 Forstå problemet

Det er flere av elevene som forklarer at det først prøver å forstå hva oppgaven ber dem om, altså analysere og tolke problemet. En elev beskriver at *«jeg begynner å tenke på hvordan sluttproduktet skal se ut og hva jeg må gjøre for å nå det (...)».*

Det er mange elever som gir lignende beskrivelser i både åpent spørreskjema og intervju, og det er tydelig at mange elever har forstått viktigheten av å tolke og *forstå* oppgaven før de kan løse den. De må legge en plan for hvordan de skal besvare oppgaven, og identifisere relevant matematikk de må bruke og hvordan formulere det til instrukser datamaskinen kan lese. Denne måten å analysere, planlegge og forutse er en viktig del av algoritmisk tenkning og nødvendig for å komme i gang med utforskning av mulige løsninger (Wing, 2011).

En annen elev sier at det de liker minst med programmering er *«at spørsmålene er upresise. Det gjør at det tar lengre tid å starte på oppgavene, siden jeg må bruke så lang tid på å tolke hva oppgaven vil at jeg skal gjøre»*.

Det kommer tydelig frem at det utforskende undervisningsopplegget med programmering krever at elevene analyserer oppgavene. I motsetning til regnetekniske oppgaver som kun krever instrumentell kunnskap og anvendelse av et sett med regler, har ikke utforskende oppgaver samme tydelige instruks. Eleven gir uttrykk for slike åpne, utforskende oppgaver krever både mer kognitiv virksomhet og tid.

Disse observasjonene understreker at *analyse* finner sted når elevene jobber med programmering, og at elevene tvinges til å påbegynne en algoritmisk tankeprosess.

4.2.3 Kreativitet

En annen interessant observasjon er at flere elever nevner kreativitet i spørsmål om hva de liker best og minst med programmering.

Thea beskriver at *«det jeg liker minst er å være kreativ, tenke utenfor boksen for å finne forskjellige løsninger. Jeg er ikke kreativ, jeg får ikke til å tenke utenfor boksen»*.

Det er interessant at Thea uoppfordret trekker inn manglende kreativitet som en utfordring i arbeid med programmering. Matematisk kreativitet er en viktig del av det å *kunne* matematikk, og inngår i det Fischbein (1994) beskriver som den intuitive komponenten som blant annet omhandler hvordan en tolker et matematisk problem og hvordan en planlegger å

løse det. Matematisk kreativitet bidrar til å hjelpe oss med å overkomme fikseringer når en har gått seg fast i en løsning, og å produsere flere typer gyldige løsninger (divergent produksjon) (Haylock, 1997).

Uten matematisk kreativitet vil det derfor bli utfordrende å analysere et problem, gjenkjenne mønster, abstrahere, evaluere arbeidet og se for seg nye løsninger om en ikke kommer i mål ved første forsøk. Dette er essensielle komponenter i algoritmisk tenkning (Utdanningsdirektoratet, 2019), som kreves i arbeid med programmeringsarbeid.

Thea forteller også at *«jeg liker best å jobbe med min sideparter fordi da kan vi komme med forskjellige ideer og finne ut hvilken idé som er best. (…) Det er lett for meg å gi opp hvis jeg jobber alene.»*

I det åpne spørreskjemaet oppgir 14 av elevene at de foretrekker å samarbeide, 5 foretrekker selvstendig arbeid og 1 elev sier at de liker begge deler. Mange av elevene som sier at de foretrekker samarbeid forklarer at de synes det er lettere å holde ut og lettere å jobbe seg gjennom utfordringer sammen.

Samarbeid er en arbeidsmåte som fremmer algoritmisk tenkning (Utdanningsdirektoratet, 2019), og flere av elevene opplever at det gjør det lettere for dem å overkomme fikseringer, som er et uttrykk for matematisk kreativitet (Haylock, 1997).

På den andre siden er det flere elever som trekker frem kreativitet som et positivt aspekt ved programmering. I det åpne spørreskjemaet har noen av elevene kommet med utsagn *som «jeg liker å bli utfordret og måtte tenke kreativt i problemløsning (…)». En annen elev sier at det beste med programmering er at «(…) Det er gøy å være kreativ, men vanskelig (…)».*

Flere elever anerkjenner at kreativitet er nødvendig for å forstå hva oppgaven ber om, og å se for seg hvordan de skal komme frem til løsningen. Det er vanskelig for elevene å følge en problemløsningsstrategi som PRIMM dersom de mangler den intuitive matematikkomponenten komponenten som behøves for å komme i gang med og videreutvikle et program når står fast.

4.2.4 Dekomponering

Dekomponering er en av de kognitive prosessene som inngår i algoritmisk tenkning, og kan hjelpe med å gjøre komplekse problemer mer håndterlige (Wing, 2011). Det er gjort funn i både intervju, spørreskjema og prestasjonstest som viser hvordan elevene bryter ned et problem på. Ettersom funnene fra prestasjonstesten viser dekomponering i matematikk, vil disse observasjonene bli analysert i delkapittel 4.6.

I undervisningsopplegget ble PRIMM-metoden benyttet som en mal av flere grunner. For det første vil en slik mal øke studiens reliabilitet ved at programmeringsarbeidet er konsekvent, og at det blir enklere å reprodusere forskningsdesignet. For det andre vil bevisst bruk av og fokus på PRIMM-metodens 5 steg bidra til å gjøre elvene bevisste på hva det vil si å ha en problemløsningsstrategi. Polya (1945) definerer 4 punkter som fanger essensen av matematisk problemløsning, som er særlig viktig innen utforskende matematikk:

1) Forstå problemet
2) Utarbeid en plan
3) Iverksett planen
4) Se tilbake

En problemløsningsstrategi vil hjelpe elvene med tolkning og sikre fremdrift i matematiske problemløsningsoppgaver. Predict-steget i PRIMM handler om å analysere et stykke kode og forutse hvordan dataen leser hver linje, og hva resultatet blir. Gir koden ønsket resultat?

Ved første blikk kan et program virke uforståelig for elevene. Det første steget i PRIMM tilrettelegger for at elevene lærer seg å bryte ned koden linje for linje, slik at det blir enklere å forstå hva som skjer og i hvilken rekkefølge. I kapittel 4.2.2 ble det diskutert ulike observasjoner av hvordan elevene analyserer og tolker oppgaven før de går i gang. Slik bruker elevene problemløsningsstrategi for å bryte ned et komplekst problem.

I intervjuet som ble utført i etterkant av undervisningsopplegget forteller Jonas hvordan han fortsetter med å løse en oppgave i programmering etter at han har analysert hva oppgaven spør om:

«Jeg lager ideer for hvordan type programmering jeg skal bruke, om jeg må lage en funksjon, eller om jeg skal bruke if/for/while, jeg må tenke hvordan verktøy og kommando jeg skal bruke. Når jeg har funnet ut av det, så har jeg et mål jeg må jobbe meg mot (…)»

Slik Jonas beskriver sin fremgangsmåte med programmeringsoppgaver kommer det tydelig frem at han dekomponerer problemet i tråd Polyas 4 steg ved å først analysere og tolke problemet, deretter legge en plan for hvordan han skal komme frem til løsningen. Videre svarer han at

«(...) Når jeg har definert alle variablene mine, så kan jeg begynne å sette all informasjonen sammen, sånn at jeg får ut det jeg ønsker.»

Jonas begynner med å definere variabler, altså legge inn informasjonen han har tilgjengelig i programmet. Deretter forteller han at han begynner å sette inn variablene der de hører hjemme i programmet han har laget en plan for. Dette er steg 3 av Polyas (1945) punkter, som handler om å utføre selve planen .

«Så går jeg tilbake og finpusser koden min, ser at resultatene blir riktige og fremstilt på en god måte» oppsummerer Jonas, som nå har forklart hvordan han evaluerer eget arbeid og egne løsninger.

Slik Jonas beskriver sin fremgangsmåte med programmeringsoppgaver kommer det tydelig frem at han dekomponerer problemet i tråd Polyas 4 steg ved å først analysere og tolke problemet, deretter legge en plan og utføre den, og til slutt evaluere arbeid og løsninger.

PRIMM-metoden med stegene *predict, run, investigate, modify, make* er forenelige med de samme problemløsningsprosessene Polya definerer, men er spesifisert mot programmering. I det åpne spørreskjemaet er det 3 elever som spesifikt nevner PRIMM ved spørsmål om hvordan de tenker når de går i gang med en programmeringsoppgave, mens flere elever beskriver selve problemløsningsmetoden helt eller delvis.

I intervju med Sofie fortalte hun at etter å ha forstått hva oppgaven ber henne om, pleier hun å gjøre følgende for å finne ut hvordan hun skal komme frem til løsningen:

«Tenker først hvordan jeg ville gjort det på ark, så prøve å få det inn i Spyder.»

Sofie dekomponerer ved at hun først finner ut hvordan hun matematisk skal løse oppgaven, og deretter formulerer hun algoritmen hun har laget som et program som kan kjøres i Spyder. Sofie husket ikke hva bokstavene i PRIMM stod for, men hun har funnet en metode å bryte ned komplekse programmeringsoppgaver ved å først hente frem relevant matematisk teori, se for seg hvordan hun må strukturere programmet for å få ønsket resultat fra datamaskinen, og

deretter utføre koden. Dette er et eksempel på dekomponering, som inngår i algoritmisk tenkning (Grover & Pea, 2013).

4.2.5 Abstraksjon og generalisering

Abstraksjon i matematikk handler om at elevene er i stand til å formulere tanker, strategier, mønster gjennom matematisk språk. Når elevene oppdager et mønster gjennom utforskning, kan de generalisere mønsteret ved å beskrive sammenhengen mellom observasjonene de gjør (Gjøvik & Torkildsen, 2019) . Å kunne beskrive disse sammenhengene matematisk er essensielt for å kunne formulere et program som kan leses av en datamaskin (Wing, 2006).

Et eksempel på hvordan elever må abstrahere når de programmerer er ved at de identifiserer ulike størrelser og sammenhenger i oppgaven, også må de abstrahere ved å definere variabler og formulere sammenhenger mellom variablene slik at de kan formulere en generell algoritme som datamaskinen kan bruke på datamateriale den får oppgitt.

I intervju med Thea forteller hun at det hun synes er vanskeligst med programmering er

«Det er vanskelig å bygge et program, og se for seg hvordan strukturen må være. Jeg føler jeg skjønner hva oppgaven ber om og klarer å skrive opp informasjonen, men det er vanskelig å se for seg hvordan man skal bygge det opp».

Som Wing (2006) forklarer, krever programmering abstraksjon på flere nivåer, noe som gjør det krevende for elevene. Thea opplever at hun mestrer å abstrahere ved å gjøre om tall til variabler datamaskinen kan hente, og hun forstår hvilken relevant matematisk teori hun må bruke for å komme frem til riktig løsning. Utfordringen er neste nivå av abstraksjon, som handler om å formulere en algoritme som datamaskinen kan lese.

Dette vil bli demonstrert med et eksempel fra et undervisningsopplegg hvor det ble tydelig at abstraksjon på flere nivåer gjør programmering vanskelig for elevene. Eksempelet handler om lister og er fra undervisningsoppleggets siste økt, der elevene hadde arbeidet med programmering i 2 uker. Koden som ble brukt som utgangspunkt i denne undervisningsøkten er presentert i figur 20 nedenfor.

```
5  navneliste=[]
6  print("Foreløpig navneliste: ", navneliste)
7
8  for i in range(10):
9
10     nyttnavn=str(input("Skriv inn et navn du vil legge til i lista: "))
11
12     navneliste.append(nyttnavn)
13
14     print("Oppdatert navneliste: ", navneliste)
```

Figur 20: Programmeringseksempel med liste og for-løkke

Figur 20 ovenfor viser et eksempel på et program elevene fikk presentert til å tolke og diskutere med hverandre (*predict*). Programmet ber brukeren om å skrive inn et navn i konsollen, som blir lagt til i en tom liste. Programmet bruker en for-løkke, slik at programmet gjentas et forhåndsbestemt antall ganger (10). Etter å ha diskutert hver linje, forutsett hva koden gjør og kjørt den, fikk elevene flere deloppgaver om å tilpasse (*modify*) programmet på ulike måter. En av deloppgavene var å endre programmet slik at det ville fortsette å legge til navn fra brukeren for alltid.

Alle elevparene klarte å endre programmet slik at det kjørte uten begrensninger. Elevene forstod at de måtte lage en grense programmet aldri ville nå. Et eksempel på hvordan noen elever løste det, var ved å definere variabelen $x = 1$, og lage en while-løkke med betingelsen «while x==1:». Når programmet aldri gjør noe for å endre verdien av variabelen x, vil $x = 1$ for alltid og programmet kjører evig.

Dette er en abstraksjon elevene gjør, der de går fra å tenke at de må lage en betingelse som alltid vil være oppfylt, til å formulere det med et sett instrukser datamaskinen kan tolke.

Elevene har i ulike omganger arbeidet med å lage funksjoner i Python, lage lister der de legger til nye verdier, og å plotte to lister med x- og y-verdier mot hverandre. Da elevene fikk i oppgave å lage et program som kombinerer kunnskapen de hadde fra disse tre programtypene, stoppet det opp for mange.

Elevene ble bedt om å lage et program med en funksjon som henter x-verdier fra en liste, sender dem gjennom funksjonen for å lage en tilhørende y-verdi og legge denne til i listen. Dette krever enda et nivå med abstraksjon, da elevene må finne ut hvordan de skal strukturere en kode som henter variabler ved å iterere gjennom listen, og for hver x-verdi sender tilbake

en ny y-verdi i en tom liste. Figur 21 nedenfor viser et eksempel på hvordan denne oppgaven kunne løses.

```
31 #Program som plotter oppgitte x-verdier med y-verdier generert gjennom funksjon
32 from pylab import *
33
34 x_verdier=[1,2,3,4,5,6,7]
35 y_verdier=[]
36
37 for x in x_verdier:
38     y= x**2 + 2*x
39     y_verdier.append(y)
40
41 print(x_verdier, y_verdier)
42
43 plot(x_verdier, y_verdier)
44 show()
```

Figur 21: Programmeringseksempel der en for-løkke skal iterere gjennom en liste med x-verdier for å generere en liste med tilhørende y-verdier

Programmet i figur 21 ovenfor bruker en for-løkke til å iterere gjennom verdiene som ligger i listen for x-verdier. Deretter defineres en variabel $y = x^2 + 2x$, som legges til i den tomme y-listen for programmet kjøres på nytt for neste x-verdi i listen. Deretter blir begge listene skrevet til skjerm, og til slutt plottes x-verdier og y-verdier mot hverandre og et plot av en 2.gradsfunksjon skrives til skjerm.

Observasjonene som ble gjort underveis i undervisningsopplegget tydet på at et ekstra nivå med abstraksjon gjorde det veldig utfordrende for elevene. Dette var eneste gang gjennom undervisningsopplegget at stilasbygging i henhold til PRIMM-metoden ikke var tilstrekkelig veiledning, og dette var helt tydelig utenfor den proksimale læringssonen til så godt som hele klassen.

Disse observasjonene indikerer at programmering krever at elevene må gjøre abstraksjoner og generaliseringer når de jobber med matematiske problemløsningsoppgaver og programmering. I undervisningsopplegget nådde klassen en tydelig grense hvor oppgaven ble for vanskelig til at de klarte å løse den med støtte og veiledning da en ekstra grad av abstrahering ble introdusert. Abstraksjonsevne er en ferdighet som må trenes og utvikles over tid, og i denne studien er tid klart en begrensende faktor. Dette vil diskuteres ytterligere i diskusjonsdelen av studien.

4.2.6 Feilsøking og vurdering

Et funn som forekommer både det åpne spørreskjemaet, i intervjuene og i observasjoner underveis av hvordan elevene arbeidet og diskuterte underveis i undervisningsopplegget, er at elevene systematisk bruker feilsøking som en strategi for å sikre fremdrift i arbeidet.

I spørreskjemaet svarer en av elevene følgende til hva de liker best med programmering:

«Det jeg liker best er feilene jeg gjør. Grunnen til det er at jeg må da tenke på nytt og prøve ulike ting. Programmering er vanskelig, så når jeg gjør feil, bruker jeg det som en mulighet til å lære og forbedre meg.»

Denne eleven synes at programmering er vanskelig, men har funnet en strategi der den prøver seg frem, og utnytter at en får feilmelding i konsollen som spesifiserer hva slags type feil som er i koden, og på hvilken linje problemet er.

10 av elevene beskriver en form for feilsøking i spørreskjemaet når de blir bedt om å forklare hvordan de går frem med programmeringsoppgaver. Feilsøking er en viktig del av algoritmisk tenkning, og mange av elevene trener seg kontinuerlig på å evaluere egne løsninger når de arbeider med programmering.

I intervjuet fokuserer Jonas mer på evaluering enn feilsøking, og forklarer hvordan han vurderer løsningene sine:

«Så går jeg tilbake og finpusser koden min, ser at resultatene blir riktige og fremstilt på en god måte.»

Han forklarer at han kontrollerer at løsningene han får er gyldige, og vurderer om det finnes bedre, mer effektive måter å komme frem til samme løsning på. Dette er en del av den algoritmiske tankeprosessen, som funnene tyder på at finner sted når elevene arbeider med programmering. I tillegg til det er måten Jonas evaluerer og utbedrer programmet sitt på en demonstrasjon av divergent produksjon, som er en form for matematisk kreativitet, som diskutert i kapittel 4.2.3. Dette funnet støtter opp om Grover og Pea (2013) sin beskrivelse av programmering som en kreativ prosess.

4.3 Hva hjelper elevene

Samarbeid er en arbeidsmåte som fremmer algoritmisk tenkning (Utdanningsdirektoratet, 2019). I undervisningsopplegget med programmering har elevene arbeidet parvis med oppgavene. Elevene har fått spørsmål tilknyttet samarbeid både i spørreskjema og intervju, som vil presenteres og analyseres i dette delkapittelet.

Til spørsmål om elevene foretrekker å samarbeide eller arbeide selvstendig, svarer 5 av elevene at de foretrekker selvstendig arbeid, 1 elev sier at de liker begge deler, og 14 av elevene svarer at de foretrekker å jobbe sammen med noen andre.

Elevene rapporterer at de synes det er hjelpsomt å ha noen å diskutere med, og at de synes det er enklere å holde ut og komme seg videre når de står fast med oppgaven enn når de arbeider alene. Dette kan indikere at samarbeid fremmer algoritmisk tenking, der elevene sammen kan evaluere og forbedre hverandres ideer.

I spørreskjemaet blir elevene spurt om de foretrekker å jobbe selvstendig eller sammen med andre når de jobber med programmering. Elevene svarer for eksempel:

«jeg liker best å jobbe sammen med andre fordi da er det større sjans for at vi klarer det. Vi kan alle komme med ideer til hvordan vi skal løse det.»

Slik eleven beskriver det vil diskusjon med en partner bidra til fremdrift i arbeidet. Når elevene står fast med oppgaven er det en trygghet å dele det med noen, og de kan bistå med ulike kunnskaper og perspektiver. Eleven beskriver at det gjør det lettere å holde ut, noe som er et av nøkkelordene innen algoritmisk tenkning (Utdanningsdirektoratet, 2019).

En annen elev svarer at

«Jeg liker å jobbe sammen med noen andre når det kommer til programmering. Da kan man få en annen synsvinkel på et problem, hvis man sitter fast.»

Eleven forklarer at det er lettere å jobbe seg videre når en står fast dersom en har noen å diskutere med. Et ekstra sett med øyne vil være hjelpsomt i feilsøking, og kan effektivisere prosessen med å designe og redesigne et program som gir ønsket resultat. Programmering

krever evaluering, utbedring, prøving og feiling, som er en del av den algoritmiske tankeprosessen (Utdanningsdirektoratet, 2019).

Et annet perspektiv er at samarbeid kan hjelpe elevene med å stå lenger i motgang. En elev svarer følgende i spørreskjemaet:

«Jeg liker best å jobbe med min sidepartner fordi da kan vi komme med forskjellige ideer også finne ut hvilken idé som er best. Jeg føler at jeg lærer bedre hvis jeg jobber med noen andre. Det er lett for meg å gi opp hvis jeg jobber alene.»

For å lykkes med å løse en programmeringsoppgave kreves det utholdenhet av elevene, og at de fortsetter å prøve når programmet ikke funger, gir dem feil resultat eller de får diverse feilmeldinger. Flere av elevene har uttrykt at de gir lettere opp når de jobber alene enn når de samarbeider. Det er trygt for elevene å ha en støttespiller når de opplever å ikke lykkes. Som tidligere argumentert for i både teori- og resultatdel påvirkes motivasjon av mestringsfølelse.

Fra observasjonene i klasserommet har det virket som at elevene har hatt det gøy mens de har jobbet med programmering. Til tross for at de fleste elevene har uttrykt at de synes programmering er utfordrende gjennom både spørreskjema, intervju og i muntlig tilbakemelding, var det både latter og godt arbeidstrykk under de 2 ukene med programmering i matematikktimene.

Magnus utrykte i intervju at *«det var litt morsomt også, man kunne snakke i lag og hjelpe hverandre»*, som tyder på at elevene har hatt det gøy, trivelig og sosialt mens de har utforsket og jobbet med programmering. Positive følelser, opplevelser og en følelse av at man ikke er alene om å synes det er vanskelig har positiv påvirkning på elevers motivasjon, som igjen vil bidra til økt utholdenhet med programmeringsarbeidet (Wæge & Nosrati, 2018).

På samme måte som det krever trening og innsats for å bli god på ski, må matematiske ferdigheter som algoritmisk tenkning, kreativitet og problemløsning trenes opp gjennom øving. Elevene får større læringsutbytte dersom de har lenger utholdenhet med arbeidet, særlig når de møter på motgang (Wæge & Nosrati, 2018).

Et til sitat til fra en elev er

«Liker som oftest best å jobbe med andre. Det er bedre å samarbeide og diskutere problemet ditt enn å sitte alene helt frustrert og forvirret».

Utifra elevenes tilbakemeldinger ser det også ut som at samarbeid gjør det lettere å overkomme fikseringer. Å overkomme fikseringer er en del av matematisk kreativitet (Haylock, 1997), og programmering er en kreativ prosess (Grover & Pea, 2013). Når elevene samarbeider har de flere hoder som kan jobbe sammen om å analysere, feilsøke, vurdere løsningene og komme med innspill til nye løsninger. Den matematiske kreativiteten inngår i det Fischbein (1994) definerer som den intuitive komponenten av matematikk, som er viktig for å kunne gjøre analyse, vurderinger og oppdage mønster som inngår i algoritmisk tenkning (Grover & Pea, 2013).

Det er et intrikat nettverk mellom algoritmisk tenkning, matematisk kreativitet og matematisk kompetanse, og en utfordring ved denne studien er å skille ulike årsaker og virkninger. Dette vil drøftes ytterligere i kapittel 5, men et funn i studien er at samarbeid stimulerer algoritmisk tenkning.

4.4 Elevenes rapportering av konkrete utfordringer når de jobber med programmering:

Hva hindrer dem

I studien ble det avdekket noen utfordringer som kan hindre læringsutbytte i arbeid med programmering:

- Matematisk kreativitet
- Tid
- Terskelbegrep

Gjennom spørreskjema, intervju og interaksjoner i klasserommet har disse to faktorene utpekt seg som mulige begrensende faktorer for læringsutbyttet i programmering.

4.4.1 Matematisk kreativitet

I spørreskjemaet ble elevene spurt om hva de liker best og minst med programmering. Det er flere elever som nevner kreativitet, både som et positivt og negativt aspekt ved programmering. Eksempler på positive utsagn er at det beste med programmering er:

«Å kunne være kreativ i løsningene sine»

«Er gøy å være kreativ, men vanskelig»

Elevene trekker frem kreativitet som et aspekt ved programmeringen uten at det spesifikt er nevnt for dem. Elevene opplever programmering som en kreativ prosess, som står i samsvar med teorien om at kreativitet er en nøkkelkomponent i arbeid med programmering (Grover & Pea, 2013).

I spørreskjemaet skriver Thea at det hun liker minst med programmering er nettopp det å måtte være kreativ:

«Det jeg liker minst er å være kreativ, tenke utenfor boksen for å finne forskjellige løsninger. Jeg er ikke kreativ, jeg får ikke til å tenke utenfor boksen».

Når elevene skal løse programmeringsoppgaver må de skape noe basert på et definert mål og tilgjengelig informasjon. Utfordringen er at de ikke har noen mal eller oppskrift som de kan bruke på samme måte som når de gjør funksjonsdrøfting eller løser likninger. Det er ikke slik

at funksjonsdrøfting ikke krever relasjonell kompetanse, men på 1T-nivå kan elevene komme i mål med mange oppgaver ved at de lærer seg metoden for å finne for eksempel nullpunkter, stigningstall, skjæringspunkter med y-akse og andre funksjoner.

Det er mange deler av pensum i 1T der elevene kan få til mange oppgaver ved hjelp av instrumentell kunnskap, men i programmering må elevene lage en løsning uten hjelp av en mal.

I intervjuet forteller Thea at det vanskeligste med programmering er *«å se for seg hvordan man skal bygge det opp»*. Thea forteller at hun begynner med å tolke informasjonen og hva oppgaven ber henne om, men at hun ofte opplever at hun kommer til et punkt i arbeidet der hun stagnerer og *«det er vanskelig å jobbe meg ut fra det når jeg står helt fast»*.

Det Thea forteller kan indikere mangel på matematisk kreativitet, slik hun selv foreslår i besvarelsen sin i spørreskjemaet. Matematisk kreativitet kommer til uttrykk gjennom evne til å overkomme fikseringer og divergent produksjon (Haylock, 1997). Mangel på matematisk kreativitet vil gjøre det vanskelig å jobbe seg forbi fikseringer, og gjøre det vanskelig å tenke seg til andre måter å løse oppgaven på. Dette funnet foreslår at begrenset kreativitet vil være et hinder for algoritmisk tenkning og progresjon i programmeringen.

Shell et. al.s (2014) empiriske studie konkluderer med at kreativitetsøvelser fremmer læring innen algoritmisk tenkning. Funnet i denne studien om at manglende kreativitet er et hinder når elevene skal løse programmeringsoppgaver er forenelig med tidligere empirisk forskning. Kreativitet og et fleksibelt tankemønster er en ferdighet som må trenes over tid, som innleder til den neste utfordringen som ble avdekket i denne studien; tid.

4.4.2 Elevene må få tilstrekkelig tid til å utforske

Blomhøj (2019) poengterer at et viktig aspekt ved utforskende undervisning er at elevene må få tilstrekkelig tid til å utforske. Arbeidsmåter som fremmer algoritmisk tenkning er å planlegge og designe, fikle og eksperimentere, utholdenhet i arbeidet, diskusjon og samarbeid, feilsøking og evaluering (Utdanningsdirektoratet, 2019). Denne arbeidsmetoden krever at elevene får anledning til å jobbe frem og tilbake med programmene sine, avdekke feil og mangler, prøve på nytt.

Særlig i intervjuene kommer det frem at elevene opplever tid som en begrensende faktor.

I intervju med Jonas forteller han om hva han synes var bra og hva han synes var dårlig med det 2 uker lange undervisningsopplegget:

«Det var bra at vi fikk jobbe mye fritt og fikk tid til å utforske selv hvordan vi ville lage koden for å besvare oppgaven, i stedet for at vi bare sitter der og skriver ned en kode fra tavla. Men jeg skulle gjerne hatt enda mer tid til å jobbe med programmene og utvikle koden mer».

Jonas fikk da oppfølgingsspørsmålet om han opplevde tid som en viktig faktor i programmering, og da svarte han:

«Ja, tid er en stor faktor i programmering, man trenger god tid til å tenke, og når du slipper å stresse kan du få en bedre forståelse av det du holder på med. Det er aldri nok tid med programmering, man blir aldri ferdig»

Det Jonas beskriver er i tråd med Blomhøjs (2019) kriterier for et vellykket utforskende undervisningsopplegg. Jonas poengterer at en aldri blir ferdig med en programmeringsoppgave, som tyder på at han har erfart at programmering er en kreativ prosess som krever algoritmisk tenkning, der analyse, feilsøking, redesigning og fleksibel tankegang for å forbedre programmet er viktige kodeord. Dette funnet understreker at elevene må gis nok tid i undervisningen til å jobbe med nettopp denne prosessen, der de lager en plan for hvordan de skal løse problemet, og deretter får tid til å gå tilbake og evaluere og forbedre arbeidet.

Et annet aspekt ved tilstrekkelig tid er at den relasjonelle matematikkunnskapen, som innebærer et samspill av formell, algoritmisk og intuitiv komponent (1994), må bygges over *tid*. Dette leder inn til funn av terskelbegrep.

4.4.3 Terskelbegrep

Thea får også spørsmålet om hun opplever at hun har hatt nok tid til å utforske og lære seg programmering. Til det svarer hun at

«Jeg har klart å lære basics, men det er vanskelig å utvikle det» og utdyper videre at *«jeg følte jeg hadde god tid i de 2 ukene, men jeg tror man trenger litt mer tid på å lære det (...)».* Hun fikk da et oppfølgingsspørsmål om hun hadde noen forslag til noe som kunne vært gjort annerledes for å gjøre det lettere å lære seg, og *svarer «det hadde hjulpet å ha litt mer programmering på ungdomsskolen, for plutselig på videregående kom det, og da var det*

veldig mye nytt på likt. Det hadde vært bedre å starte med det enkle på ungdomsskolen slik at vi hadde fått tid til å synke det inn.»

Thea uttrykker at det kan være overveldende med programmering da det er mye nytt. Programmering var nytt for det store flertall i klassen da de begynte med programmering i 1T, og flere elever har gitt tilbakemelding både muntlig og i spørreskjemaet om at de synes det er mye nytt å forholde seg til på likt. For elever som ikke har vært borti programmering før og aldri har brukt en Pythoneditor før er det kanskje nok i første omgang til å få tid til å utforske hvordan datamaskinen tolker ulike datatyper som skrift og desimaltall, ulike kommandoer, og hvordan skrive variabler og resultater til skjerm.

Magnus forteller i intervjuet at *«jeg fikk tid til å begynne å forstå det, men jeg fikk ikke tid til å bygge noe særlig på det, bare startkunnskap liksom, det sitter ikke skikkelig»* og at en utfordringe er *«å lære seg alt og huske alt sånn at ting etter hvert går litt mer automatisk»*.

Dersom elevene er usikre på språk og kommandoer vil det være vanskelig for dem å gå videre til de mer avanserte delene av programmering som handler om selve utarbeidelsen av et program. Det er i denne prosessen kreativitet og algoritmisk tenking kommer inn. Om elevene må bruke mye tid på det rent kodetekniske som språk og kommandoer, kan dette være til hinder for læring innen algoritmisk tenkning.

Typisk for terskelbegrep er at de er essensielle byggesteiner for videre utvikling av kunnskapen på området (Meyer & Land, 2005). I dette tilfellet indikerer resultatene at selve kodespråket (som inkluderer syntaks, kommandoer og notasjon) er et terskelbegrep. Det å få tid til å la det *«synke inn»* som Thea sier, er også typisk for terskelbegrep som karakteriseres av å være tidkrevende å internalisere (Pettersson & Brandell, 2017). Funnet vil diskuteres nærmere i kapittel 5.3.4.

4.5 Elevens syn på sammenhengen mellom programmering og matematikk (nytteverdi)

Som redegjort for i 4.2 har elvene i studien demonstrert ulike kognitive prosesser som til sammen utgjør algoritmisk tenkning. I arbeid med programmering har elevene tilnærmet seg oppgaven ved å først analysere og forstå både koden og hva oppgaven ber dem om, elevene har dekomponert problemet ved å bryte ned koden linje for linje. Videre har de brukt ulike strategier for å produsere gyldige løsninger, og gått problemet i møte på ulike vis. Det er også tydelig at programmering krever at elevene abstraherer, slik at de er i stand til å formulere sammenhengene og mønstrene de avdekker som generelle algoritmer en datamaskin kan lese. Til slutt har også elevene vist at de bruker feilsøking for å sikre fremdrift i arbeidet, i tillegg til at noen elever evaluerer arbeidet sitt og fortsetter arbeidet med å forbedre eller tilpasse programmet.

Elevene benytter algoritmisk tenkning når de programmerer, men i hvilken grad kan ferdighetene overføres til andre matematikkoppgaver? Det er dette studien tar sikte på å undersøke.

Fra det lukkede spørreskjemaet blir elevene spurt om sine holdninger og opplevelser tilknyttet programmering i matematikkfaget.

4.5.1 Klassens selvrapportering: Programmering og læring

Et av forskningsspørsmålene i oppgaven er hvorvidt elevene kan utvikle evner gjennom programmering som har overføringsverdi til andre deler av læreplanen. Kan elever lære matematikk gjennom programmering, eller er programmeringsarbeid best egnet til å lære nettopp programmering?

30% av elevene sier seg enig i at de ikke føler de lærer noe ved å arbeide med programmering, mens det store flertall (70%) sier seg uenig (påstand 14). I påstand 9 oppgir 75% av elevene oppgir at de mener at programmering passer inn i 1T faget, og 60 % er helt eller litt uenig i at programmering tar for stor plass i matematikkfaget (påstand 16). Likevel er 65% prosent av elevene enige om at de foretrekker andre arbeidsmåter i matematikk, mens de resterende 35% angivelig foretrekker programmering (påstand 15).

Når det kommer til utsagn 17 om at programmering kan brukes til å løse matematiske problemer er 85% av elevene enige i det, og 75% rapporterer at de synes programmering passer bra inn i 1T faget (påstand 9). Kun 15% oppgir at de ikke ser sammenhengen mellom matematikken og programmeringen som arbeides med i undervisningen (påstand 10).

At elevene opplever programmeringsopplegget som meningsfylt i forhold til matematikken de skal lære er et positivt utgangspunkt for motivasjon, innsats og læring (Wæge & Nosrati, 2018). Mer spesifikt *hva* elevene lærer gjennom undervisningsopplegget vil diskuteres i delkapittel 4.6.

Som presentert i tabell 1 sier hele 85% av elevene seg uenige i påstand 10 *«jeg ser ikke sammenhengen mellom matematikken vi lærer og programmeringen vi gjør»*. Det er altså en tydelig overvekt av elever i klassen som anser programmeringen relevant for matematikken, eller matematikken relevant for programmeringen. 75% av elevene sier seg litt eller helt enig i at programmering passer inn i faget 1T (påstand 9), og 65% av elevene er helt eller litt enig i at programmering kan brukes i alle delemner (påstand 11). Dette vil diskuteres nærmere sammen med funn fra intervju og prestasjonstester.

4.5.2 Elevers perspektiver på matematisk læring og programmering

I læreplanen for matematikk 1T er problemløsning ved hjelp av algoritmisk tenkning spesifikt nevnt som et kompetansemål (Kunnskapsdepartementet, 2019). Programmering er en aktivitet som er anerkjent å kreve algoritmisk tenkning og kreativitet (Grover & Pea, 2013; Wing, 2006), som begge er komponenter som inngår i hva det vil si å ha en relasjonell matematikkunnskap (Fischbein, 1994). Denne studien har som formål å undersøke denne sammenhengen, og overføringsverdien mellom algoritmisk tenkning i programmering og matematikk.

Som tidligere nevnt ble 4 elever invitert til intervju i etterkant av undervisningsopplegget der elevene brukte programmering som verktøy for å jobbe med det matematiske temaet *funksjoner*.

I intervju med Thea og Sofie trekker de hver for seg frem at den største overføringsverdien mellom programmering og matematikk er at programmering kan være et nyttig hjelpemiddel. Thea legger til at det er mulig å programmere store deler av pensum i 1T, og trakk frem et

eksempel fra da hun laget et program som beregnet røttene til en andregradsfunksjon ved hjelp av abc-formelen.

Dette kan en riktignok gjøre både i GeoGebra og på kalkulator også, så programmering må kunne tilby noe mer ettersom læreplanen etter Kunnskapsløftet 2020 spesifikt nevner programmering som et av kompetansemålene i læreplanen. Ifølge teori om programmering og de kognitive prosessene det krever, foreligger det et stort læringspotensiale innen algoritmisk tenkning og problemløsning i matematikk (Wing, 2006).

I intervju med Magnus og Jonas kommer det frem flere interessante betraktninger som knytter sammen programmering og matematikk. Jonas har mye erfaring med programmering og en stor egeninteresse, mens Magnus for første gang er blitt introdusert skikkelig for programmering i 1T på lik linje med resten av klassen.

Begge elevene ble spurt om de hadde lært noe de hadde fått nytte av i andre matematikkoppgaver. Jonas svarer at

«Ja, det er jo litt sånn steg for steg tenking, tenker jeg. (…) Det kan du bringe litt over til matte; jeg starter med den her informasjonen, så bruker jeg den her algoritmen for å få ut den her informasjonen, og du kan bruke den samme tankegangen når du skal gjøre matte, at du får en systematisk måte å gjøre ting på.»

Jonas beskriver deler av en algoritmisk tankeprosess der han analyserer tilgjengelig informasjon, bryter ned problemet i mindre delproblemer, abstraherer og formulerer en algoritme som gir ønsket resultat. Algoritmisk tenkning er et nyttig verktøy i matematikk, da det er en problemløsningsmetode med ulike kognitive prosesser som kan hjelpe i arbeid med særlig utforskende matematikkoppgaver der elevene ikke når frem med instrumentell kunnskap. Utforskende oppgaver krever at elvene kombinerer både algoritmisk, formell og intuitiv komponent i henhold til Fischbeins (1994) teori om hva matematisk kompetanse er.

Jonas legger også til at han følte han fikk økt forståelse for likninger, funksjoner, røtter og 2. gradsformelen med de ulike tilfellene som gir 2 unike reelle røtter, to like reelle røtter og ingen reelle løsninger. Som tidligere nevnt har Jonas en del erfaring med programmering, og er komfortabel med både programmeringsspråket og selvstendig skrive linjer med kode. Dette gir ham overskudd til å fokusere på matematikken og programmeringen, ikke bare det rent kodetekniske, som mange av elevene synes var en utfordring (mer om dette i 1.7)

Magnus hadde kun vært borti litt blokkprogrammering før 1T, og synes programmering var tidvis gøy og tidvis vanskelig. Da Magnus ble spurt om han hadde lært noe gjennom programmering som han kunne videreføre til matematikken svarte han at

«Ja, at det er viktig å ta det veldig sånn trinn for trinn. Hvertfall jeg er sånn som lett kan skippe over ting, ofte i matte gjør jeg slurvefeil fordi jeg skal gå fort frem og glemt å gjøre minus eller pluss eller noe greier, minus og minus, ja.»

Det er interessant at Magnus også tar opp dette med den stegvise tilnærmingen til problemløsning, og hvordan det kan gi en bedre struktur i oppgaveløsningen og gjøre det lettere å evaluere om løsningene er gyldige. Både Jonas og Magnus beskriver prosesser som inngår i algoritmisk tenkning, som de kan videreføre til andre matematikkoppgaver.

Elevene ble blant annet spurt om hva de synes om måten de hadde jobbet på gjennom de 2 ukene med programmering med PRIMM-modellen som mal for undervisningen.

«D*et var litt morsomt også, man kunne snakka sammen og hjelpe hverandre, og vi så at det hadde sammenheng med matematikken, det var ikke bare programmering bare for å gjøre det*» svarte Magnus. Magnus trekker frem samarbeid som en positiv opplevelse, og at programmeringen opplevdes som fornuftig bruk av matematikktimene.

Det Magnus forteller stemmer overens med funnene om at flertallet av elevene synes programmering passer inn i 1T-faget og at de ser sammenhengen mellom matematikken de lærer og programmeringen de gjør i timene (henholdsvis 75% og 85%). Samsvaret mellom funn i ulike typer datainnsamlingsmetode styrker funnets validitet.

Elevene blir også spurt om hva de synes om hva de synes om oppgavene de hadde løst i testen før og etter undervisningsopplegget. Da svarte Jonas at

«det var jo relatert til programmering, det var blant annet figurtall og det å kjenne igjen mønster. Det var jo ikke nøyaktig det vi har brukt i pensum, det var jo litt sånn generell matematisk og logisk tenkning. For eksempel de vinduene med røde ruter, det var jo ikke veldig relevant for mattepensum i 1T, men litt mer generelt logisk tenkning, og det kan jo hjelpe deg i matte ved at du får en bedre tankegang.»

Til det samme spørsmålet svarer Magnus

«Det var liksom litt sånn tenkeoppgaver, ikke sånne du bare klarer hvis du er veldig god i matte, men sånne der du må tenke logisk».

Både Jonas og Magnus beskriver oppgavene de løste i prestasjonstesten som oppgaver som krever *logisk tenkning*. Logisk tenkning assosieres med å forklare eksisterende sammenhenger eller fenomener ved å strukturere holdbare argumenter. Verken Jonas eller Magnus bruker begrepet algoritmisk tenkning, og det er uvisst om de kjenner til begrepet. Likevel kan det tenkes at det de beskriver som *logisk tenkning* passer inn under begrepet *algoritmisk tenkning* tatt i betraktning det elevene trekker frem. Jonas trekker frem mønstergjenkjenning, mens Magnus beskriver problemløsningsstrategi, evnen til å tolke og forstå en oppgave og være kreativ fremfor å anvende avanserte matematiske prosedyrer.

De matematiske gevinstene fra programmering som Jonas og Magnus trekker frem indikerer at algoritmisk tenkning brukt til problemløsning med programmering har overføringsverdi til problemløsning i matematikk.

4.6 Hvordan elevers bruk av algoritmisk tenkning viser seg i arbeid med matematikk

I utarbeidelsen av prestasjonstestene var det et poeng at de skulle inneholde ulike oppgaver der elevene fikk demonstrere lignende ferdigheter, som mønstergjenkjenning, dekomponering, abstraksjon, formulering av algoritmer og strategi for å løse oppgaven. Dette for å styrke datamaterialets validitet, og sørge for at elevene viste forbedring i disse ferdighetene heller enn at de demonstrerte instrumentelle ferdigheter på lignende oppgaver de hadde løst tidligere.

Det var forutsett at oppgavesett 2 skulle være litt vanskeligere enn oppgavesett 1, da beskrivelse av et av figurtallene inkluderte en sum, og oppgave 4 med diagonaler i mangekanter er krevende å avdekke mønsteret i og enda mer krevende å generalisere (se vedlegg 5 og 6).

Dette resulterte i at det ble vanskelig å kvantifisere besvarelsene og poengsette, da for det første testene er ulike og retesten var ansett som litt vanskeligere. Et enda viktigere aspekt er at målet med prestasjonstestene var å se etter indikatorer på algoritmisk tenkning, og hvorvidt elevene har forbedret sine problemløsningsferdigheter og matematisk resonnering gjennom det 2 uker lange programmeringsopplegget de deltok i. Prestasjonstestene ble vurdert ikke bare etter gyldige løsninger, men også resonnementer elevene gjorde for å komme frem til løsningen, og resonnementer de startet uten å fullføre.

For å vurdere besvarelsene ble det utført en kvalitativ analyse i henhold til Hoviks (2016) rammeverk for matematisk resonneringsnivå i som redegjort for i teoridelen:

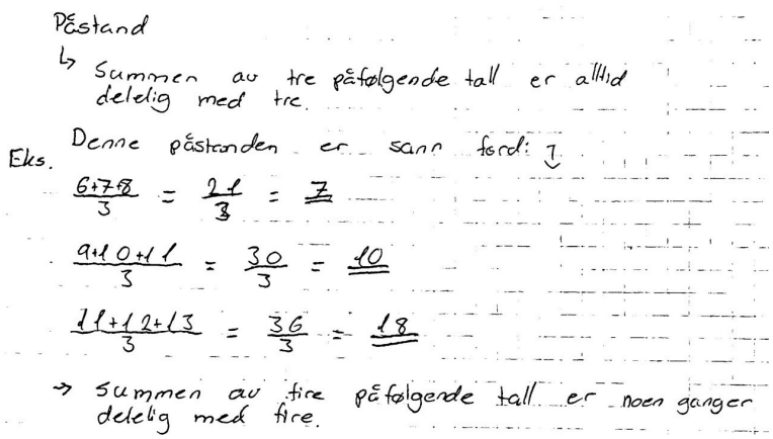
1) Begrunnelse ved å referere til autoriteter (lærer, lærebok, foreldre, mm)
2) Begrunnelse gjennom konkrete eksempler
3) Matematisk resonnering basert på en visuell representasjon (konkreter/tegning, tekst eller regnefortelling)
4) Bevis ved bruk av algebraisk notasjon og bruk av regnelovene

Blant de 21 individene som gjennomførte prestasjonstestene viste 13 elever forbedringer i resonnementer, problemløsningsstrategier og produksjon av gyldige løsninger. 6 elever

presterte likt i begge testene, og 2 elever gjorde det best i den første testen. 2 av elevene som presterte likt viste høyt nivå på både matematisk resonnering og kompetanse.

Tilnærmet 62 % av elevene forbedret seg i løpet av perioden. Det kan være mange årsaker til at flertallet av elevene har forbedret seg fra pretest til retest, som vil bli diskutert i kapittel 5.

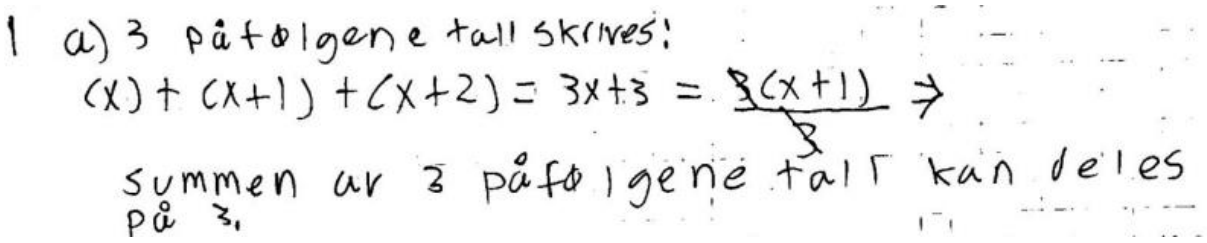
I figur 22 og 23 nedenfor presenterer to eksempler på ulike nivåer av matematisk resonnering. Begge er hentet fra oppgave 1 fra pretest der elevene ble bedt om å ta stilling til påstanden om at 3 påfølgende tall alltid vil være delelig med 3.



Figur 22: Eleveksempel matematisk resonnering nivå 2

Figur 22 ovenfor viser et eksempel på et matematisk resonnement der eleven bruker konkrete eksempler for å bevise at påstanden er sann, som klassifiserer som matematisk resonnering på nivå 2.

En annen elev bruker algebra og regnelover for å bevise at påstanden er sann (figur 23). Her har eleven beveget seg fra konkrete eksempler til et generelt uttrykk som er gyldig for alle heltall. Dette er et mer avansert matematisk resonnement som bruker algebra og regneregler, og klassifiserer som nivå 4 etter Hoviks (2016) rammeverk for vurdering av matematiske resonnement.

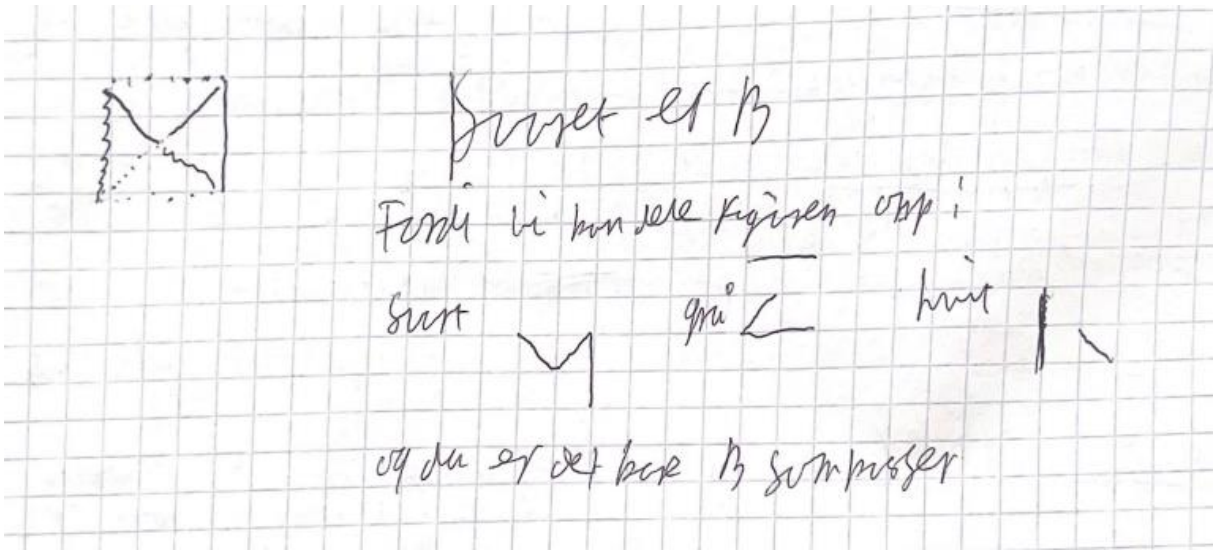


Figur 23: Eleveksempel matematisk resonnering nivå 4

Dette resonnementet viser at eleven har analysert oppgaven, tolket og forstått tilgjengelig informasjon, og basert på det laget en algoritme for å undersøke påstanden. Eleven produserer et generalisert uttrykk, der den viser at 3 påfølgende tall alltid vil være delelig med 3. Eleven bruker videre samme algoritme for å undersøke om det samme gjelder for 4, 5 og 6 påfølgende tall. De kognitive prosessene eleven har brukt for å komme frem til løsningen demonstrerer algoritmisk tenkning (Utdanningsdirektoratet, 2019).

Andre eleveksempler som indikerer algoritmisk tenkning

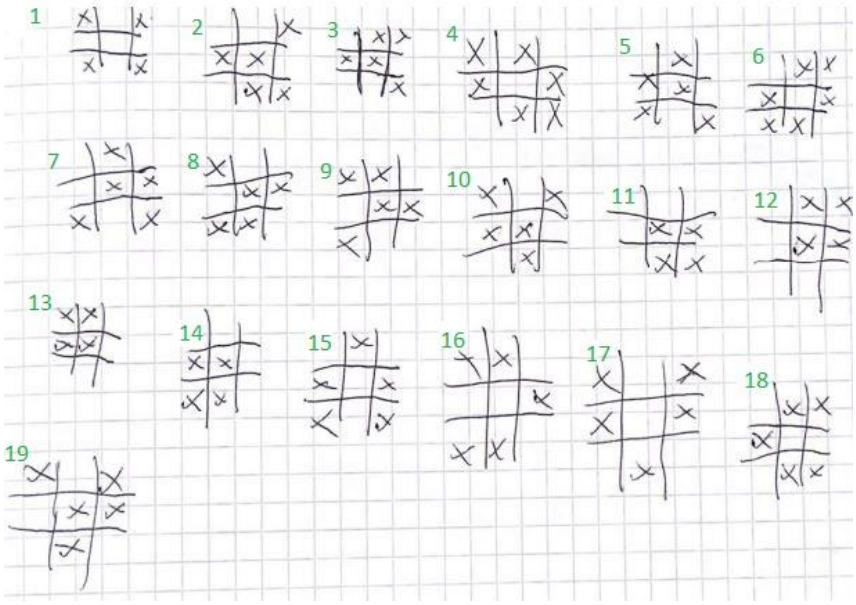
Dekomponering



Figur 24: Eleveksempel med demonstrasjon av dekomponering

Figur 24 ovenfor viser tydelig at eleven brutt problemet ned i mindre deler for å gjøre det mer håndterlig. I dette eksempelet har eleven delt opp figuren ved å tegne de svarte, grå og hvite linjene for seg, for å gjøre det lettere å sammenligne figurene og bestemme hvordan figuren ser ut ovenfra. Denne dekomponeringen bidrar til å lage en stegvis metode for å komme frem til løsningen, og er en demonstrasjon av algoritmisk tenkning.

Kreativitet: divergent produksjon



Figur 25: Eleveksempel med demonstrasjon av divergent produksjon

I den siste oppgaven på pretesten viser en av elevene at de er i stand til å finne mange korrekte løsninger, som klassifiseres som divergent produksjon (figur 25). Divergent produksjon er en form for matematisk kreativitet, som assosieres med både algoritmisk tenkning (Israel-Fishelson & Hershkovitz, 2022) og programmering (Grover & Pea, 2013). Elevens løsninger er nummerert for å enkelt kunne referere til løsningene som blir analysert.

Fra løsningen ser det også ut som at eleven har vekslet mellom å prøve seg frem, og å lage en algoritme for å generere flere riktige løsninger. Eleven finner først en løsning ved å sette inn 4 røde ruter (markert 1). Deretter produserer eleven 2 løsninger med 5 ruter, som har samme mønster men med ulik rotasjon (2 og 3). I løsning 4 produserer eleven et nytt mønster som også er en gyldig løsning, før eleven i nummer 5 lager en ny utgave av mønsteret den fant i løsning 2 og 3. I løsning 6 lager eleven en rotasjon av løsning 4, før den i løsning 7-10 går tilbake til å rotere mønsteret funnet i 2.

I løsning 11-14 har eleven funnet en løsning med 4 ruter, og roterer systematisk for å finne alle gyldige løsninger med mønster av denne typen. Eleven gjør det samme for 15-18. Her er

det tydelig at eleven har laget en algoritme, og anvendt den for å generere alle løsninger av denne typen.

Løsning 19 er en ny variant av mønsteret eleven fant i nummer 2, slik at eleven til slutt lykkes med å finne alle 19 unike gyldige løsninger. Eleven demonstrerer at de er i stand til å oppdage mønster, lage en stegvis prosedyre for å komme frem til riktige løsninger, og være kreativ innen matematikk, som alle er en viktig del av algoritmisk tenkning.

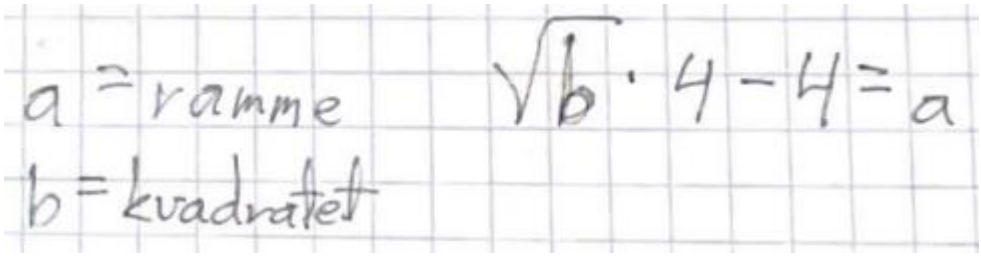
4.6.1 Eksempler på forbedringer

Som nevnt innledningsvis er det ulikheter i oppgavesettet som gjør det vanskelig å sammenlikne besvarelsene direkte. Oppgavene som best lar seg sammenlikne er oppgave 2 i begge testene som handlet om figurtall (se vedlegg 5 og 6). Ellers er det interessant å se etter endringer i nivået på elevenes matematiske resonnementer, endringer i om elevene avdekker mønster eller i det hele tatt leter etter dem, om de er i stand til å generalisere observasjonene sine, om de viser matematisk kreativitet eller gjør rede for strategien de bruker for å komme frem til løsningen. Alt dette er indikatorer på algoritmisk tenkning, som er det denne oppgaven har som formål å undersøke.

4.6.1.1 Elevforbedring 1

En interessant observasjon er gjort hos en elev som ikke gjør noen generaliseringer i det hele tatt i pretesten. Eleven løser flere oppgaver ved hjelp av tall, uten å gjøre rede for noe mønster eller formulere et oppdaget mønster med ord eller algebra.

På oppgave tilsvarende oppgave i retesten (oppgave 1) regner eleven først ut hvor mange ruter rammer av ulike kvadrater består av, før de formulerer et generelt uttrykk som kan beregne antall ruter i hvilken som helst ramme i et kvadrat (figur 26).



Figur 26: Elevforbedring 1

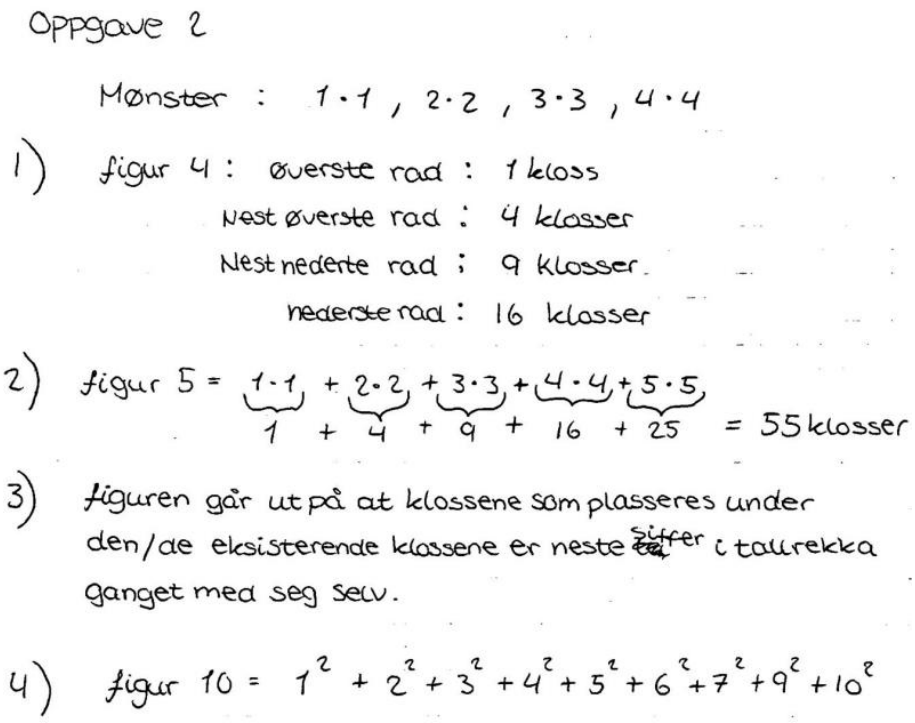
Eleven har gjenkjent et mønster, og definert to variabler. Eleven viser hvordan de beregne rammestørrelsen, gitt at en kjenner kvadratets størrelse i antall ruter. Eleven trekker fra 4 hjørner som blir telt dobbelt i hver side av kvadratets ramme.

Eleven viser styrket matematisk resonnement, og demonstrerer algoritmisk tenkning gjennom mønstergjenkjenning, generalisering, formulering av en algoritme og bruk av den.

4.6.1.2 Elevforbedring 2

Begge oppgavesettene inneholdt et figurtall, der elevene skulle avdekke mønsteret i figurtallet, lage et generelt uttrykk for å beskrive det, og bruke det til å finne et gitt figurtall. Det er flere elever som har vist bedring i nivå på resonneringen fra pretest til retest, enten ved at de har lyktes bedre med å gjenkjenne mønsteret, eller i større grad har lyktes med å formulere et generelt uttrykk som kan brukes til å generere et n'te figurtall.

Eleven i dette eksempelet besvarte ikke oppgaven om figurtall i pretesten, men var i stand til å avdekke mønsteret i tilsvarende oppgave i retesten. Eleven forklarer mønsteret ved hjelp av ord, og bruker tall, tekst og piler som representasjon (figur 27). Dette klassifiserer som nivå 3 i henhold til Hoiviks nivådeling på matematiske resonnementer, og er en tydelig forbedring fra ikke besvart oppgave i pretesten.

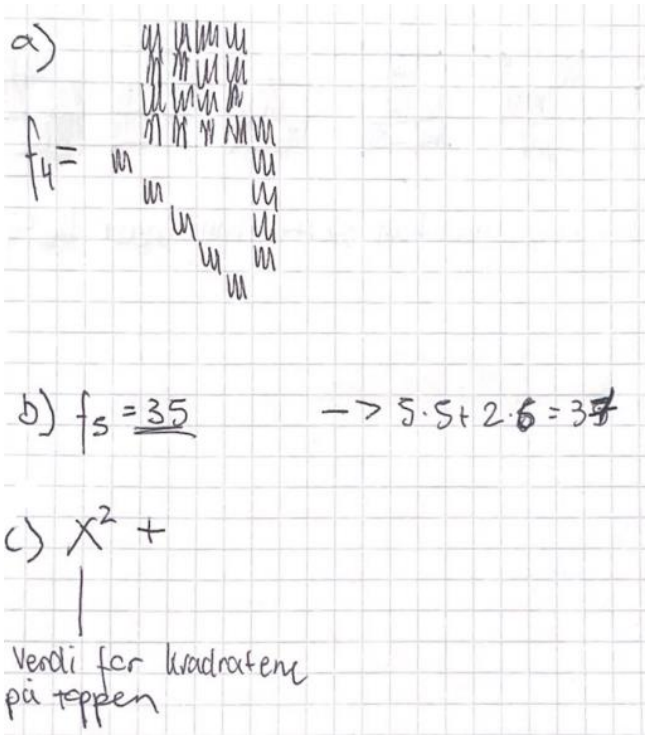


Figur 27: Elevforbedring 2

Analyse av oppgaven og tilgjengelig informasjon, mønstergjenkjenning og formulering av en algoritme er demonstrasjon av algoritmisk tenkning (Gjøvik & Torkildsen, 2019; Utdanningsdirektoratet, 2019; Wing, 2006).

4.6.1.3 Elevforbedring 3

Figur 28 viser et eksempel på en elevbesvarelse i pretesten. Her viser eleven at den klarer å finne neste figurtall med en numerisk tilnærming, og eleven har begynt å formulere et generelt uttrykk som beskriver et hvilket figurtall av denne typen. Eleven beskriver kvadratet på toppen av figuren som x^2, men klarer ikke formulere et uttrykk for de to linjene. Det kommer likevel frem av besvarelsen at eleven har lett etter et mønster for å regne ut hvor mange klosser neste figurtall består av, og deretter begynt å dekomponere figuren for å kunne beskrive figurtallet på generell form.

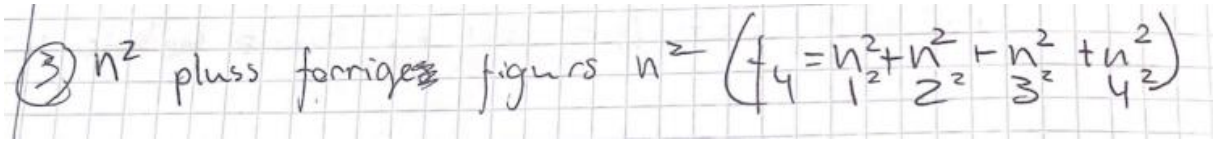


Figur 28: Elevforbedring 3 – generalisering i pretest

På den tilsvarende oppgaven i retesten viser den samme eleven hvordan den finner neste figurtall nummer 5 i sekvensen ved å summere de første 5 kvadrattallene, som eleven også gjorde i pretesten.

Til forskjell fra pretesten, kommer eleven lenger i arbeidet med å lage et generelt uttrykk for figurtallet. Som vist i figur 29 nedenfor har eleven beskrevet mønsteret delvis med ord, tall og

algebra. Eleven kommer ikke helt i mål med å formulere uttrykket algebraisk da det skulle vært $f_4 = n^2 + (n-1)^2 + (n-2)^2 + (n-3)^2$, men generaliserer i større grad enn i pretesten.



Figur 29: Elevforbedring 3 – generalisering i retest

4.6.1.4 Elevforbedring 4

Magnus er en av elevene som viser en stor forbedring i problemløsningsstrategi, mønstergjenkjenning, abstrahering og matematisering mellom pretest og retest. I oppgave 1 i pretesten bruker han konkrete eksempler som

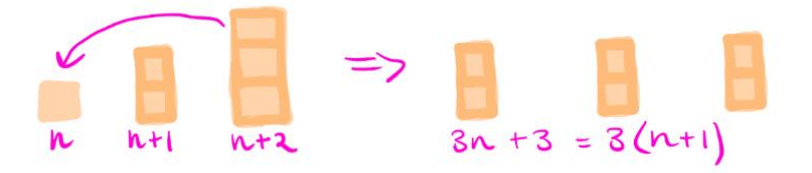
$$\frac{6+7+8}{3} = 7$$

Til å forklare at summen av 3 påfølgende tall alltid er delelig med 3. Han forklarer også at

«Du har 3 tall som stiger med 1 for hvert tall, det gjør at det tredje tallet er 2 mer enn det første tallet. Om du trekker 1 fra det siste tallet og legger det til det første tallet blir alle tre tallene like store, som beviser at det er sant.»

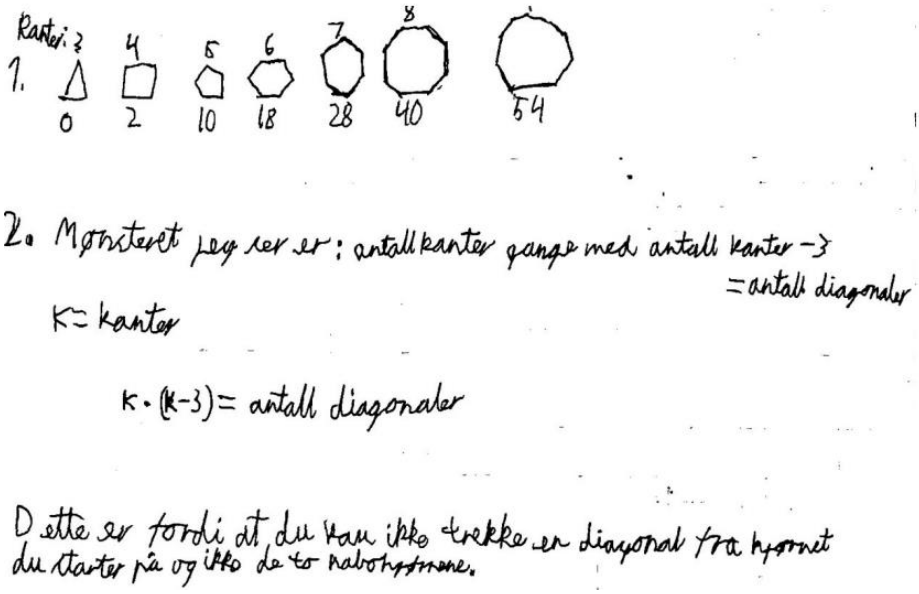
Magnus klarer å visualisere og forklare ved hjelp av ord hvorfor påstanden må være sann for hvilke som helst 3 påfølgende tall. Magnus har analysert og funnet et mønster, som han generaliserer ved å formulere en algoritme som vil gjelde for 3 vilkårlige påfølgende tall. Han prøver seg ikke på å formulere algoritmen ved hjelp av algebra og regneregler, og resonnementet klassifiserer som nivå 3.

Magnus sitt resonnement er illustrert ved tegning og algebra figur 30 nedenfor.



Figur 30: Elevforbedring 4 – illustrasjon av resonnement i pretest

I retesten derimot (oppgave 4) bruker Magnus både tegning, ord og algebra for å beskrive mønsteret han har oppdaget (figur 31).



Figur 31: Elevforbedring 4 - mønstergjenkjenning og generalisering i retest

Han kommenterer at det ikke er mulig å trekke diagonaler til de to nabohjørnene, og heller ikke det hjørnet en står i. Dermed kan hvert hjørne k trekke diagonal til alle hjørner $k-3$. Magnus kommer ikke helt i mål, og glemmer å dele uttrykket på 2 for å eliminere de dupliserte diagonalene som kommer fra at en diagonal kan trekkes begge retninger.

I både pretesten og retesten viser han at det foregår kognitive prosesser som passer med algoritmisk tenkning, som mønstergjenkjenning og generalisering. Han hever likevel nivået på sin matematiske resonnering ved å uttrykke seg ved hjelp av algebra.

4.6.2 Oppsummering

Det er vanskelig å trekke tråd mellom årsak og virkning i elevenes forbedringer, da det er mange elementer i undervisningsopplegget som kan medføre matematisk læring. Det kan tenkes at elevene har fått trening i å abstrahere og generalisere når de definerer variabler og lager algoritmer i utarbeidelsen av et program som en datamaskin kan lese. Samtidig har elevene deltatt i et utforskende undervisningsopplegg der de har blitt presentert for et problem, fått tid og rom til å utforske problemet ved hjelp av programmering, og til slutt har elevenes funn og observasjoner blitt diskutert i plenum der den nye kunnskapen er blitt formalisert og satt i system. Undervisningsopplegget ble utarbeidet etter Blomhøjs (2019) modell for utforskende undervisningsopplegg, som tilrettelegger for relasjonell kunnskap.

I tillegg har elvene samarbeidet og diskutert mens de har arbeidet med programmering og matematikk, som er anerkjent som en arbeidsmetode som fremmer algoritmisk tenkning (Utdanningsdirektoratet, 2019). Dette vil diskuteres ytterligere i delkapittel 5.3.2.

5 Diskusjon

I dette kapittelet vil funnene presentert i analysen brukes til å diskutere forskningsspørsmålene introdusert i problemstillingen. Diskusjonen er ment å belyse styrker og svakheter ved funnene, og føre til en konklusjon som besvarer problemstillingen

«Kan bruk av programmering i matematikkundervisning fremme algoritmisk tenkning hos elever?»

5.1 Overordnet

Det er noen elementer ved denne studien som er gjeldende for alle funn gjort på bakgrunn av datainnsamlingen. Det første og mest åpenbare er at studien er en casestudie basert på en 1T-klasse bestående av 21 individer. Formålet med å utføre en casestudie var å kunne gå i dybden på utforskning av problemstillingen, og samtidig holde studien innenfor grensene av en masteravhandling. Studien er dog basert på et lite utvalg, som begrenser studiens reliabilitet. Ettersom datainnsamlingen er basert på kun 21 individer vil hvert individ ha stor utslagskraft på det samlede datamaterialet, og det er rimelig å anta at funnene gjort i studien ikke vil være reproduserbare i en vilkårlig skoleklasse.

I et klasserom er det en kompleks dynamikk som er unik for hver klasse. Det avhenger av eksempelvis samspill mellom lærer og elever, relasjoner blant elevene, holdninger til læring innad i klassen, for ikke å snakke om at hvert menneske påvirkes av både positive og negative hendelser i sine personlig liv. Dette gjør at dersom det samme undervisningsopplegget og samme datainnsamling hadde blitt utført på samme tid i en annen klasse, kan det forventes at resultatene hadde sett annerledes ut.

Det er rimelig å anta 1T-elever har generelt høy grad av motivasjon for matematikk, da de frivillig har valgt et krevende matematikkfag på bakgrunn av eksempelvis egen indre motivasjon eller en ytre motivasjon om at de trenger realfag for komme inn på et bestemt studie. Dette reflekteres i elevenes innsats og deltakelse i undervisningsopplegg, prestasjonstest, spørreskjema og intervjuer. Elevene har vært positive, gjort sitt beste både i læringssituasjon og under testene, og tatt sine roller som informanter med største alvor.

Det var avklart at besvarelsene til elevene i spørreskjema, prestasjonstest og intervju ikke hadde noen slags påvirkning på karakteren i faget. Elevene kan likevel ha blitt preget av at det var en forskningssituasjon, som kan ha gjort at det har følt ekstra på en slags vurdering og at elevene har følt et slags prestasjonspress (Gleiss & Sæther, 2022).

Dersom elevene har følt på et slags krav om at de må prestere, kan det ha påvirket elevene til å ha lagt ned en ekstra innsats eller vært ekstra motiverte for å jobbe med programmering disse ukene, enn de ville vært i en læringssituasjon innenfor helt normale rammer.

Det må også nevnes at selv om det har vært tydelig kommunisert at alt elevene svarer i sammenheng med studien ikke vil påvirke verken karakteren deres eller forholdet mellom elevene og lærer. Det kan likevel tenkes at denne rollekonflikten mellom å være elevenes faglærer og innta en forskerrolle med egne elever som informanter kan ha påvirket elevenes atferd, holdninger og besvarelser. Elevene er blitt forsikret at studien ikke har noe med deres faglige prestasjon og vurdering å gjøre, og at de ikke må være redde for å komme med negative tilbakemeldinger til både undervisningsopplegg og programmering i matematikk. Det kan likevel tenkes at det er vanskeligere for dem å være negative og ikke ville delta i undervisningsopplegget når det er faglæreren deres som gjennomfører studien.

Det har dog virket på elevene som at de har genuint trivdes med å jobbe med programmering ved PRIMM-metoden, som tilrettelegger for diskusjon og utforskning. Mange elever har gitt muntlige tilbakemeldinger på at de har hatt det gøy i timen og å få så mye rom til å prøve seg frem selvstendig, at det har vært hyggelig å samarbeide mye og at de synes oppgavene i prestasjonstesten var morsomme og litt annerledes. Dette kan for all del også forklares med at de ønsker å være positive og støttende til forskningsprosjektet til faglæreren sin, men temperaturen i klasserommet har vært god, fylt med latter, gode diskusjoner og engasjerte elever. Slike uttrykk for positive følelser og opplevelser er likevel en indikasjon på et trygt og godt læringsmiljø og motivasjon for det de arbeider med, som er et godt utgangspunkt for læring uavhengig av tema (Wæge & Nosrati, 2018).

Andre begrensninger i studien er at det er lagt til grunn at teorien om TPACK er overførbar til å måle elevers teknologisk-pedagogiske fagkunnskap. Teorien er basert på lærere, og det er i denne studien gjort en antakelse om at lærerens TPACK vil gjenspeiles i elevenes læringsutbytte etter deltakelse i undervisningsopplegg. Dette er en antakelse som svekker studiens validitet.

Til slutt må en som forsker være klar over sitt bias. Dette er en mixed-methods studie som domineres av det kvalitative datamaterialet, der dataanalysen i stor grad er preget av induktiv koding. Det vil si at fokus har vært på å lete etter relevante funn for forskningsspørsmål og problemstilling, og det påvirker både tolkning av observasjoner og utvalg av relevante observasjoner. Dette er også en feilkilde som både svekker studiens validitet, og også reproduserbarheten av resultatene (reliabilitet). For å minimere svekkelsen av resultatene og studien har det derfor vært benyttet både et rammeverk for utarbeidelse av undervisningsmetode (PRIMM (Sentance et al., 2019)) og vurdering av elevbesvarelser (nivåer av matematisk resonnering (Hovik, 2016)).

5.2 Kan programmering brukes som et verktøy til å fasilitere algoritmisk tenkning hos elever?

I dette delkapittelet vil funnene presentert i 4.1 *Elevenes syn på programmering* og 4.2 *Hva gjør elevene når de programmerer* drøftes i henhold til relevant teori for å forsøke å besvare forskningsspørsmålet

«Kan programmering brukes som et verktøy til å fasilitere algoritmisk tenkning hos elever?»

På bakgrunn av funnene gjort i analysen er svaret på forskningsspørsmålet **ja, funnene gir en indikasjon på at programmering kan brukes som et verktøy til å fasilitere algoritmisk tenkning hos elever.** Funn og begrensninger diskuteres i dette delkapittelet.

Som presentert i analysen, krever arbeid med programmering at elevene iverksetter flere av de kognitive prosessene som inngår i algoritmisk tenkning. Algoritmisk tenkning kan sees å som en systematisk tilnærming til problemløsning. Der Polya (1945) formulerer 4 steg for matematisk problemløsning

1) Forstå problemet
2) Utarbeid en plan
3) Iverksett planen
4) Se tilbake

defineres algoritmisk tenkning ved konkrete kognitive prosesser som vil bistå med å komme frem til en løsning på problemet.

Algoritmisk tenkning innebærer å **analysere** og tolke oppgaven, forstå hva oppgaven ber om og hvilken informasjon som er tilgjengelig. Elevene definerer et mål, og legger en plan for hvordan de skal nå den. **Dekomponering** er en strategi hvor en bryter ned komplekse problemer til mindre og mer håndterlige delproblemer. **Mønstergjenkjenning** handler om å avdekke sammenhenger og forhold som kan brukes til å beskrive det du undersøker. Deretter må elevene definere variabler og **abstrahere** oppdagelsene sine, slik at de kan formuleres til en **generell algoritme** som kan leses av en datamaskin. Til slutt gjenstår det å **evaluere** arbeid og løsninger (Gjøvik & Torkildsen, 2019; Grover & Pea, 2013; Utdanningsdirektoratet, 2019; Wing, 2006).

Programmering er anerkjent som en kreativ prosess som krever algoritmisk tenkning (Grover & Pea, 2013). For å legge til rette for at elevene skal få anledning og rom til å utforske programmering og utvikle ferdigheter innen algoritmisk tenkning, ble PRIMM-modellen (predict, run, investigate, modify, make) brukt som mal for undervisningsopplegget. Resultater fra tidligere empirisk forskning foreslår at PRIMM-modellen er en effektiv metode for å lære elever programmering, og å hjelpe dem å utvikle problemløsningsferdigheter (Sentance et al., 2019).

PRIMM-modellen dannet rammene for undervisningsopplegget, med formål om å lede elevene inn i en algoritmisk tankeprosess som redegjort for i kapittel 2.

Programmeringsarbeid krever at elevene bruker sin relasjonelle kunnskap, da det oppgavene de får er av utforskende karakter og det er mange gyldige løsninger. Elevene kommer ikke i mål med instrumentell kunnskap, da det ikke finnes noen oppskrift de kan følge.

Etter gjennomført undervisningsopplegg rapporterer flertallet av elevene (14/21) at de opplever at de mangler en god strategi for å arbeide med programmeringsoppgaver. Likevel beskriver mange av elevene problemløsningsstrategier som begynner med å analysere oppgaven, definere et mål for oppgaven og legge en plan for hvordan de skal komme frem til løsningen, generalisere relevante matematiske prosedyrer som en algoritme en datamaskin kan bruke, og benytte feilmeldingene de får i konsollen for å sikre fremdrift i arbeidet.

Mange av elevene beskriver at de delvis eller fullstendig bruker PRIMM-modellen, som inviterer til de kognitive prosessene som inngår i algoritmisk tenkning. Datamaterialet indikerer at elevene har utviklet egne versjoner av PRIMM-modellen og algoritmisk tenkning for å løse programmeringsoppgaver. Elevene opplever likevel at de mangler en god strategi,

som kan tenkes å være fordi det ikke finnes noen oppskrift som vil føre dem til riktig løsning hver gang. Det betyr ikke at de ikke har en god problemløsningsstrategi, men det kan være andre årsaker som gjør at de ikke kommer i mål med oppgavene.

Et av funnene som ble gjort i løpet av perioden med programmering i matematikkundervisningen, var at de ulike nivåene av abstraksjon som programmering krever (Wing, 2006) gjør at elevene går seg fast. Som presentert i kapittel 4 var det tydelig at elevene møtte grensene for sin proksimale sone da de ble introdusert for en ekstra dimensjon med abstraksjon. Elevene har lyktes med å løse oppgaver der de har måtte abstrahere for eksempel figurtall, hvor programmet tar inn informasjon fra bruker om hvilket figurtall den skal produsere, og har brukt informasjon fra bruker som variabel i en funksjon som lager n'te figurtall. Elevene har abstrahert innenfor grensene av hva de får til med hjelp og støtte når de har jobbet med programmering, som er en del av den algoritmiske tankeprosessen (Gjøvik & Torkildsen, 2019; Grover & Pea, 2013; Utdanningsdirektoratet, 2019; Wing, 2006).

Det er flere grunner til at elevene likevel opplever programmering som vanskelig, og ikke kommer i mål med oppgavene. Dette diskuteres i delkapittel 5.3.

I undervisningsopplegget har elevene jobbet med utforskende oppgaver, innenfor rammene av PRIMM. Først har elevene blitt presentert en utgangspunktkode i fellesskap, der elevene har diskutert med hverandre for å forklare og forutsi hva som skjer i koden. Deretter har de blitt presentert for et problem, og får tid og rom til å selvstendig utforske og skape et program som oppfyller nye betingelser eller løser problemet. Til slutt har oppgaven blitt løst i fellesskap, med innspill og forslag fra elevene, og programmet er blitt grundig forklart og gjennomgått underveis. Denne formen for undervisningsopplegg er utforskende, etter Blomhøjs kriterier (Blomhøj, 2019).

Gjennom PRIMM har det blitt tilrettelagt for arbeidsmåter som fremmer algoritmisk tenkning. *Predict* tilrettelegger for diskusjon og samarbeid. *Investigate* gir elevene rom til å utforske og feilsøke. *Modify* tilrettelegger for å designe og skape. *Make* gir elevene anledning til å gjennomføre planen og skape et program, prøve på nytt, samarbeide og holde ut i prosessen. Arbeidsmåtene elevene har brukt i undervisningsopplegget inviterer til algoritmisk tenkning og utvikling av matematisk resonnering.

Elevene rapporterer i både spørreskjema og intervjuer at de har en problemløsningsstrategi i ulik grad. Elevene beskriver helt eller delvis PRIMM-metoden, som fører til arbeidsmåter

som stimulerer algoritmisk tenkning. De færreste elevene nevner PRIMM eksplisitt, men forklarer at de begynner med å analysere oppgaven, dekomponerer gjennom å formulere et mål og hvordan de skal nå det, de bruker feilsøking for å sikre fremdrift i problemløsningen, og vurderer løsnignene sine. Dette er demonstrasjon av algoritmisk tenkning, som oppstår mens elevene arbeider med programmering i matematisk kontekst.

I tillegg til at funnet er kontekstavhengig gitt naturen av en casestudie, er det en annen stor usikkerhet ved dette funnet som er verdt å ta i betraktning. Som tidligere nevnt er undervisningsopplegget *utforskende*, som også assosieres med relasjonell kunnskap og tilrettelegger for arbeidsmetoder som stimulerer nettopp algoritmisk tenkning (Blomhøj, 2019). Dette gjør det vanskelig å trekke slutning mellom årsak og virkning, da den algoritmiske tenkningen elevene beskriver kan teoretisk forklares som et resultat av både programmering og utforskende undervisning. Dette svekker funnets validitet, da forskningsdesignet gjør det usikkert om en har lyktes med å måle ønsket parameter.

Resultatene indikerer at i denne 1T klassen, kan programmering brukes som et verktøy i matematikkundervisning for å fasilitere algoritmisk tenkning. Dette funnet er i tråd med teorien til Wing (2006), Grover & Pea (2013), Gjøvik & Torkildsen (2019) om at programmering er en menneskelig aktivitet som krever algoritmisk tenkning. Resultatene samsvarer også med tidligere empirisk forskning fra Husain et. al. (2017) og Sáez-López et al. (2019), men det finnes også motstridende forskningsresultater fra empiriske studier. Som Forsström & Kaufmann (2018) understreker i sin litteraturstudie er det fremdeles behov for forskning på feltet og konsensus om undervisningsmetode og programmerings rolle i matematikkfaget.

5.3 Hvilke faktorer kan hindre algoritmisk tenkning hos elever?

Analysen av spørreskjema, intervju og observasjoner i klasserommet peker ut noen faktorer som kan hindre elevenes algoritmiske tankeprosess. De begrensende faktorene som ble avdekket er begrenset kreativitet, tid og terskelbegreper.

5.3.1 Kreativitet

I spørreskjemaet kommer det frem at elevene synes det er gøy at programmering er en kreativ prosess (Grover & Pea, 2013), og trekker det frem som en positiv opplevelse med

programmering. Samtidig setter Thea ord på at kreativitet er det som gjør programmering vanskelig synes hun:

«det jeg liker minst er å være kreativ, tenke utenfor boksen for å finne forskjellige løsninger. Jeg er ikke kreativ, jeg får ikke til å tenke utenfor boksen.»

Hun er ikke alene om å rapportere at en av de største utfordringene med programmering er å se for seg hvordan en skal bygge opp programmet. Dette er en utfordring som kan knyttes til matematisk kreativitet i form av divergerende tankegang.

Når elevene skal bygge et program fra start selv kreves det divergerende tankegang (Haylock, 1997), noe som kreves av elevene når de skal bygge et program selv. Det kommer frem i spørreskjemaet at dette er noe mange elever synes er utfordrende, og understrekes i intervju med både Thea, Sofie og Magnus.

En annen utfordring som kan knyttes til matematisk kreativitet er evnen til å overkomme fikseringer. Det var mange elever som uttrykte at de ofte opplevde å sette seg helt fast i arbeid med programmering:

«Jeg liker best å jobbe med programmering med andre, og dette er fordi jeg synes det er lett å sette seg fast når man programmerer.»

Ofte setter man seg fast med oppgaver dersom en enten har hengt seg opp i en spesiell løsningsmetode eller algoritme. Det å overkomme slike fikseringer er et uttrykk for matematisk kreativitet (Haylock, 1997).

Resultatene indikerer at begrenset kreativitet gjør at elevene stagnerer med oppgaven. Dette stemmer overens med teorien om at programmering er en kreativ prosess, og en del av algoritmisk tenkning (Grover & Pea, 2013).

5.3.2 Samarbeid fremmer utholdenhet og kreativitet

På den andre siden viste funnene fra spørreundersøkelse, intervju og observasjon at elevene opplever det lettere å overkomme fikseringer dersom de samarbeider. I spørreskjemaet svarer en av elevene

«Jeg liker å jobbe sammen med noen andre, når det kommer til programmering. Da kan man få en annen synsvinkel på et problem, hvis man sitter fast.»

Denne eleven er ikke alene om å rapportere at det er lettere å jobbe seg videre når en sitter fast dersom en har en partner å diskutere med. Dette tyder på at samarbeid og diskusjon gjør det lettere for elevene å overkomme fikseringer, som er et uttrykk for fleksibel tankegang og matematisk kreativitet (Haylock, 1997).

Et annet aspekt ved samarbeid er at elevene utviser større utholdenhet dersom de gis muligheten til å samarbeide. I spørreskjemaet svarer en av elevene at

«jeg liker best å jobbe med min sidepartner fordi da kan vi komme med forskjellige ideer også finne ut hvilken ide som er best. jeg føler at jeg lærer bedre hvis jeg jobber med noen andre. det er lett for meg å gi opp hvis jeg jobber alene.»

I både spørreundersøkelse, intervju og observasjoner gjort underveis i undervisningsopplegget kommer det frem at elevene opplever det tyngre å stå i utfordrende oppgaver når de jobber alene enn sammen med andre. Utholdenhet i arbeid med oppgaver er en indikator på motivasjon i matematikk (Wæge & Nosrati, 2018), og en viktig komponent i den algoritmiske tankeprosessen (Utdanningsdirektoratet, 2019). Funnene indikerer at samarbeid har en positiv effekt på elevenes utholdenhet.

Et annet aspekt ved samarbeidet er at det øker elevenes trivsel, som også er et positivt utgangspunkt for motivasjon og læring. En elev skriver i spørreskjemaet at de *«liker som oftest best å jobbe med andre. Det er bedre å samarbeide å diskutere problemet ditt, enn å sitte aleine helt frustrert og forvirret»*

Slik eleven beskriver det medfører samarbeid at negative følelser som frustrasjon og forvirring reduseres. Positive følelser og trivsel tilknyttet arbeid med programmering er et positivt utgangspunkt for elevenes motivasjon, utholdenhet og læring.

Disse funnene indikerer at samarbeid hjelper elevene med kreativitet, algoritmisk tenkning og trivsel i arbeid med programmering.

5.3.3 Tid

Tid som en faktor for algoritmisk tenkning deles inn i hovedsakelig 2 observasjoner; tid til å utforske og tid til å lære. I intervju med Jonas forteller han at tid er en viktig faktor i arbeid med programmering, og beskriver programmering som en langvarig prosess som innebærer kontinuerlig feilsøk, tilpassing og utvikling av programmet.

Dette stemmer overens med teorien om utforskende matematikkundervisning, der et av kravene for vellykket undervisningsopplegg er at elevene får tilstrekkelig tid til å utforske konsepter og sammenhenger, og rom til å gjøre egne erfaringer som senere settes i system med relevant teori (Blomhøj, 2019). Denne observasjonen handler om at elevene må få nok tid til å utforske, og være i den algoritmiske tankeprosessen.

Den andre observasjonen av tid som begrensende faktor handler om at elevene må få tid til å lære. Dette funnet leder inn til funn av *terskelbegreper.*

5.3.4 Terskelbegreper

For de fleste elevene i studien var programmering relativt nytt for dem. De fleste hadde vært innom blokkprogrammering, men bare et fåtall hadde kjennskap til Python før 1T. Fra observasjoner underveis i undervisningsopplegget og tilbakemeldinger fra elevene kommer det frem at det ikke bare er kreativitet, problemløsning og abstrahering de synes er vanskelig. Elevene gir også tydelig uttrykk for at de strever med det rent kodetekniske. I det åpne spørreskjemaet skriver en av elevene at det vanskeligste med programmering er

«å ikke få til fordi jeg ikke husker/vet nøyaktig hvordan ord og setninger skal skrives eller settes opp (…)»

Observasjonene tyder på at elevene bruker mye tid på nettopp å lete frem ønsket kommando og å avdekke syntaksfeil. Typiske feil elevene gjør er at de ikke forstår hva som må indenteres, at de ikke forstår i hvilken rekkefølge ulike kodelinjer må stå for å oppnå ønsket resultat, glemmer kolon i løkker og funksjoner, og blander ulike datatyper som string og integer.

Det tyder på at selve kodespråket utgjør et terskelbegrep for elevene. Et av kjennetegnene ved terskelbegreper er at de tar tid å utvikle (Meyer & Land, 2005). Det har vært mye nytt for elevene på en gang, og de har ikke hatt særlig lang inkubasjonstid til å tilegne seg kunnskap om selve kodespråket, som innebærer notasjon, kommandoer og syntaks.

Både Thea og Magnus uttrykker i intervjuene at det var mye nytt å lære på likt, og Thea sier at

«det hadde hjulpet å ha litt mer programmering på ungdomsskolen, for plutselig på videregående kom det, og da var det veldig mye nytt på likt. Det hadde vært bedre å starte med det enkle på ungdomsskolen slik at vi hadde tid til å synke det inn».

Det Thea forklarer er typisk for terskelbegreper, ved at det *faller på plass* når elevene behersker begrepet (Pettersson & Brandell, 2017). Når elevene har tilegnet seg kunnskapen om et terskelbegrep vil det oppleves selvfølgelig, og den nye forståelsen gjør at elevene ikke vil gå tilbake til sine tidligere oppfatninger (Meyer & Land, 2005).

Når elevene må bruke mye tid på å rette opp syntaksfeil og å bruke kommandoer riktig hver gang de programmerer, blir de revet ut av resonnementene sine og det gjør det vanskelig for dem å fokusere på selve problemløsningen og de ulike kognitive prosessene som inngår i algoritmisk tenkning. Dette er også typisk for terskelbegrep, at de er essensielle for byggesteiner for videre utvikling av kunnskap (Meyer & Land, 2005), i dette tilfellet problemløsningsferdigheter innen programmering.

I intervju med Magnus forteller han at han gjerne skulle hatt mer tid til å utvikle kunnskapen slik at *«det sitter skikkelig»*, og at en av utfordringene er *«å lære seg alt og huske alt sånn at ting etter hvert går litt mer automatisk»*.

Dette tyder på at også Magnus opplever at det går en del tid til å få til selve språket, og at det gjør problemløsningsarbeidet mindre effektivt.

5.4 Hvilke evner kan elevene utvikle gjennom programmeringsarbeid som har overføring til matematisk kompetanse?

Som redegjort for i 4.2 og 5.2, beskriver flere elever i spørreskjemaet at de bevisst benytter problemløsningsstrategi i arbeid med programmering. I undervisningsopplegget har PRIMM-modellen vært brukt som ramme for utforskningsarbeidet, og være en støtte for elevene i fremdrift med arbeidet. Ved å følge stegene i PRIMM ledes elevene inn i en algoritmisk tankeprosess der de først analyserer og tolker koden, kjører den, utforsker den og feilsøker, legger en plan for tilpasning og utvidelse av programmet i henhold til oppgaven, før de til slutt gjennomfører planen og stegvis jobber seg frem til ønsket resultat mens de kontinuerlig evaluerer løsningen.

PRIMM-modellen er forenelig med Polyas (1945) 4 steg for matematisk problemløsning; forstå problemet, utarbeide en plan, iverksette planen og se tilbake.

I intervju med Jonas og Magnus kommer det frem at de ser sammenhengen mellom programmering og matematikk, og de foreslår nytteverdi av programmering inn i matematisk problemløsning. De trekker begge frem hvordan den stegvise tilnærmingen til å arbeide med programmering er overførbar til matematikk.

De legger begge fokus på selve tilnærmingen til problemet og tankegang i arbeid med både programmering og matematikk. Jonas legger vekt på at fremgangsmåten han har brukt i programmering der han først har tolket og forstått oppgaven, identifisert tilgjengelig informasjon og relevant matematisk teori, lagt en plan for hvordan komme frem til riktig løsning og utført den er overførbar til matematikk. Han sier at en på den måten kan få en mer systematisk måte å løse problemer på. I intervjuet nevner han også at oppgavene i prestasjonstesten var relatert til programmering ved at det innebar mønstergjenkjenning, beskrivelse av figurtall som innebærer generalisering, og «logisk tenking».

Magnus sier at det var nyttig å ta med seg den stegvise tilnærmingen til problemløsning videre i matematikken, og at det kan gjøre det lettere å unngå slurvefeil eller typiske fortegnsfeil.

De kognitive prosessene Jonas og Magnus beskriver inngår i algoritmisk tenkning, og det er tydelig at de reflekterer over et læringsutbytte fra programmering annet enn selve kodingen,

notasjon, syntaks og kommandoer. Gjennom programmering har de utviklet en problemløsningsstrategi som har overføringsverdi til matematisk problemløsning.

Resultatene fra prestasjonstesten indikerer at ferdighetene elevene utvikler gjennom programmeringsarbeid har overføringsverdi til matematisk resonnering og problemløsningsferdigheter. Prestasjonstesten inneholdt oppgaver som tilrettelegger for demonstrasjon av algoritmisk tenkning, ved at for eksempel dekomponering, mønstergjenkjenning og abstrahering kommer til uttrykk i elevenes besvarelser.

Besvarelsene fra prestasjonstesten ble analysert og vurdert med utgangspunkt i Hoviks (2016) rammeverk for å beskrive nivå av elevers matematiske resonnementer Blant de 21 elevene som gjennomførte prestasjonstesten viste 13 elever forbedring fra pretest til retest, 6 elever presterte likt, og 2 elever gjorde det best i pretesten. 2 av elevene som presterte likt i begge testene demonstrerte høyt nivå på både matematisk resonnering og kompetanse. Det kan derfor argumenteres for at også disse elevene forbedret seg ettersom retesten var ansett som vanskeligere. Da er det totalt 15 av 21 elever som har vist forbedring fra pretest til retest.

Det er vanskelig å si noe om graden av forbedring da besvarelsene ikke er poengsatt, og vurderingen av besvarelsene var basert på resonneringsnivå og ikke bare antall gyldige løsninger. En form for forbedring i elevenes resonnementer var at flere viste forbedret abstraksjonsevne, der flere elever i retesten lyktes med å formulere generelle uttrykk ved bruk av algebra og korrekt matematisk notasjon for å beskrive mønstre og sammenhenger. En mulig forklaring av denne forbedringen er at elevene gjennom programmering har fått trening i å definere variabler, og å videre bruke variablene til å beskrive sammenhenger og hente variabler inn i løkker og funksjoner. Dette er å gjøre generaliseringer og å formulere algoritmer, som kan forklare at elevene er blitt flinkere til å bruke algebra til å beskrive mønstre og sammenhenger i retesten.

Funnet om at bruk av programmering i matematikkundervisningen kan fremme matematisk kompetanse er i tråd med funnene gjort i tidligere empirisk forskning av Husain et al. (2017), Forsström & Kaufmann (2018), Sáez-López et al. (2019) og Forsström & Stenseth (2021). Samtidig må det også nevnes at en fransk studie fra 2022 konkluderer med det motsatte, der de observerte en liten, men negativ utvikling i matematisk læring sammenlignet med tradisjonell matematikkundervisning (Laurent et al., 2022).

En utfordring ved både forskningsdesignet og resultatene er at det flere faktorer til stede i undervisningsopplegget som kan fremme matematisk læring. Resultatene indikerer at elevene har forbedret sine ferdigheter innen algoritmisk tenkning, men det er ikke mulig stadfeste programmering som årsak.

Et av funnene presentert i analysen er at samarbeid fremmer diskusjon og progresjon i elevenes arbeid, og at de opplever det enklere å komme forbi fikseringer og komme på flere mulige løsninger når de samarbeider med hverandre. Det å overkomme fikseringer og produsere ulike løsninger er kjennetegn på matematisk kreativitet (Haylock, 1997). Som redegjort for i teoridelen er matematisk kreativitet både en viktig faktor i løsning av programmeringsoppgaver, men også en ferdighet som inngår i den intuitive komponenten som sammen med formell og algoritmisk komponent utgjør matematisk kompetanse etter Fischbeins (1994) definisjon.

Samspillet mellom de 3 komponentene gjør en i stand til å takle matematiske problemer som krever det Skemp (1976) definerer som relasjonell kunnskap, der en må kunne anvende kunnskapen på nye områder og det ikke er mulig å komme i mål ved å reprodusere en tidligere løsning. Dermed kan også forbedringen sett i prestasjonstesten forklares med at elevene har fått trening i å være kreative og styrket den intuitive komponenten av sin matematikkforståelse. Dette kan skyldes programmeringsarbeidet, eller det utforskende arbeidet i seg selv.

Den relasjonelle matematikkforståelsen er igjen assosiert med utforskende undervisning, der elevene utfordres til å avdekke sammenhenger og mønstre, og gjennom utforskning og diskusjon skal gjøre seg erfaringer som kan settes i system med relevant teori (Artigue & Blomhøj, 2013).

Undervisningsopplegget elevene har tatt del i med 2 uker med programmering er også et utforskende undervisningsopplegg. Samarbeid og diskusjon som er en del av utforskende undervisning (Blomhøj, 2019) og er anerkjent å fremme algoritmisk tenkning (Utdanningsdirektoratet, 2019). Det er derfor vanskelig å si om forbedringen i algoritmisk tenkning og matematisk kreativitet skyldes programmeringen som et verktøy for å fremme algoritmisk tenkning, eller om det er utforskning og diskusjon som forklarer elevenes læringsutbytte. Dette sees på som den største svakheten i studien, da forskningsdesignet svekker studien og resultatenes validitet ved at funnet ikke kan årsaksforklares.

Mulige relevante feilkilder er at forbedringen av resultatene som elevene viser til en viss grad kan skyldes instrumentell kunnskap. Oppgavesettene var ikke like, men begge settene inneholdt en oppgave som handlet om å lage et generelt uttrykk for et figurtall. Elever som har forbedret seg på denne oppgaven kan ha demonstrert instrumentell kunnskap ved at de har kan løse oppgaven ved en prosedyre som ikke kobler inn algoritmisk tenkning. Dette medfører i så fall at prestasjonstesten ikke egner seg til å teste forbedring i algoritmisk tenkning.

Det er også mulig at elevene har opplevd oppgavesett 2 som lettere, til tross for at oppgavesett 2 ble vurdert som litt vanskeligere i forkant av datainnsamlingen. Dette trenger ikke være tilfellet, og det er ikke utenkelig at det blant de 21 individene vil være ulike oppfatninger av vanskelighetsgraden på pretest og retest. Dette er også en mulig feilkilde som svekker studiens validitet.

Det er også verdt å nevne at elevene har holdt et høyt arbeidstrykk i perioden, som kan tenkes å være utslagsgivende for læringsutbyttet. Elevene har vist stort engasjement for å lære programmering, og har deltatt aktivt i undervisningsopplegget. De har hatt gode diskusjoner, fått tid og rom til å utforske oppgavene selvstendig, de har vist utholdenhet i møte med motgang og gjennom diskusjon med medelever kommet frem til gyldige løsninger. Elevenes aktive deltakelse kan også være forklarende årsak til forbedringene observert i prestasjonstesten.

6 Konklusjon

Med Kunnskapsløftet 2020 ble programmering innført som en del av læreplanen i matematikk, og knyttes til problemløsningsferdigheter og algoritmisk tenkning (Kunnskapsdepartementet, 2020). Det finnes et solid teoretisk grunnlag som binder sammen programmering og algoritmisk tenkning, der programmering anerkjennes som en kreativ prosess som krever analyse, dekomponering, generalisering, formuering av algoritmer, og kontinuerlig feilsøking og evaluering av løsningene. Dette er også de kognitive prosesser som assosieres med algoritmisk tenkning (Gjøvik & Torkildsen, 2019; Grover & Pea, 2013; Kaufmann & Stenseth, 2021; Utdanningsdirektoratet, 2019; Wing, 2006).

Det er likevel ikke spesifisert i læreplanen noen undervisningsmetode eller hvilken rolle programmering skal ha i matematikkfaget for å oppnå de ønskede resultatene. Til tross for at programmering og algoritmisk tenkning er høyaktuelt for å kunne ta del i et digitalisert samfunn som en kritisk tenkende og autonom borger (Wing, 2011), kan det virke som innføringen av programmering i læreplaner kan virke prematur, med tanke på sprikende resultater fra empirisk forskning. For eksempel konkluderer en fransk studie fra 2022 med at matematikkundervisning med programmering medførte en liten, men negativ utvikling i matematisk læring sammenlignet med kontrollgruppen (Laurent et al., 2022). På den andre siden konkluderer en studie fra 2017 med at elevene viste en signifikant forbedring i blant annet problemløsningsferdigheter og kreativitet etter å ha deltatt i et undervisningsopplegg med blokkprogrammering (Husain et al., 2017). Forsström og Kaufmann (2018) understreker i sin litteraturstudie at de sprikende forskningsresultatene tyder på at det er behov for konsensus om undervisningsmetode og programmerings rolle i matematikkfaget. Denne studien kan bidra til å klargjøre eventuelle sammenhenger mellom undervisningsmodellen PRIMM og programmerings rolle i matematikkfaget.

Formålet med studien var å få innsikt i overføringsverdien mellom læringsutbytte fra arbeid med programmering og matematisk kompetanse. Studien tar sikte på å besvare følgende problemstilling:

«Kan bruk av programmering i matematikkundervisning bidra til å fremme algoritmisk tenkning hos elever?»

For å forsøke å besvare problemstillingen ble følgende 3 forskningsspørsmål formulert:

- På hvilken måte kan programmering brukes som et verktøy til å fasilitere algoritmisk tenkning hos elever?
- Hvilke faktorer kan hindre algoritmisk tenkning hos elever?
- Hvilke evner kan elever utvikle gjennom programmeringsarbeid som har overføring til matematisk kompetanse?

For å undersøke disse forskningsspørsmålene ble det gjennomført en mixed methods studie, på bakgrunn av teorien om hvordan måle teknologisk-pedagogisk fagkunnskap (Koehler et al., 2012). Studien er basert på datainnsamling fra 21 1T-elever gjennom åpent og lukket spørreskjema, intervjuer og en prestasjonstest i forkant og etterkant av et 2 uker langt undervisningsopplegg med programmering.

Funnene indikerer at programmering kan brukes som et verktøy til å fremme algoritmisk tenkning hos elever. I undervisningsopplegget ble PRIMM-modellen brukt som ramme for undervisningen da empirisk forskning konkluderer med at det er en effektiv metode å for å lære bort programmering (Sentance et al., 2019), i tillegg til at det sikrer kontinuitet og øker studiens reliabilitet.

Gjennom undervisningsopplegget har elevene engasjert seg i ulike kognitive prosesser som inngår i algoritmisk tenkning, som analyse, dekomponering, generalisering, bruke og lage algoritmer, samt evaluere arbeidet (Gjøvik & Torkildsen, 2019; Grover & Pea, 2013; Utdanningsdirektoratet, 2019; Wing, 2006).

Til spørsmålet om hvordan elevene tenker når de skal gå i gang med en programmeringsoppgave beskriver mange av elevene utgaver av problemløsningsstrategier som innebærer analyse ved at de først tolker og forstår oppgaven, og definerer et mål for hvordan resultatet skal være. De dekomponerer gjennom å definere variabler, identifisere relevant matematikk for å komme frem til riktige løsning, samt at mange elever systematisk bruker feilmeldingene i konsollen til å sikre fremdrift i arbeidet. Flere elever beskriver også at de evaluerer løsningene sine, og fortsetter å utvikle koden sin. De kognitive prosessene som den matematiske problemløsningsprosessen utløser, inngår alle i algoritmisk tenkning.

Elevene har også demonstrert abstraksjon for eksempel i arbeid med å programmere figurtall. Elevene avdekket mønster og sammenhenger, som de gjennom generalisering formulerte som en algoritme som kunne anvendes av en datamaskin. Dette er en demonstrasjon av abstraksjon, som også inngår i algoritmisk tenkning (Gjøvik & Torkildsen, 2019; Grover & Pea, 2013; Utdanningsdirektoratet, 2019; Wing, 2006).

Gjennom programmeringsarbeidet ved PRIMM-modellen har det blitt tilrettelagt for algoritmisk tenkning gjennom arbeidsmåter som fremmer samarbeid og fleksibel tankegang. Gjennom *predict* har elevene inngått i diskusjon og samarbeid, fått rom til å utforske og feilsøke under *investigate*, designe og tilpasse programmer gjennom *modify,* og til slutt lage programmet, prøve og feile, holde ut i prosessen gjennom *make.*

Utfordringen med dette funnet er at dette undervisningsopplegget også er et utforskende undervisningsopplegg i henhold til Blomhøjs (2019) kriterier. Elevene får presentert et problem og rammer for utforskning, deretter tid til å utforske selvstendig med støtte og veiledning, før til slutt elevenes observasjoner systematiseres og knyttes til relevante fagkunnskaper gjennom plenumsdiskusjon. Utforskende undervisning er assosiert med relasjonell kunnskap og utvikling av problemløsningsferdigheter som inngår i algoritmisk tenkning (Artigue & Blomhøj, 2013). Funnene indikerer at elevene engasjerer seg i algoritmisk tenkning når de arbeider med programmering, men det er ikke mulig å utelukke at det er det utforskende undervisningsopplegget som medfører resultatene, uavhengig av tema.

Fra resultatene i prestasjonstesten der elevene besvarte matematiske oppgaver som tilrettela for demonstrasjon av algoritmisk tenkning, viste 15 elever forbedring fra pretest til retest, 4 elever presterte likt og 2 elever presterte best i pretesten. Besvarelsene ble vurdert i henhold til Hoviks (2016) rammeverk for nivåer av matematisk resonnering. Resultatene indikerer at elevene utvikler ferdigheter gjennom programmeringsarbeid som har overføringsverdi til matematisk resonnering og problemløsningsferdigheter. Et eksempel på dette er at flere elever har forbedret sine resonnementer ved å gå fra å vise sammenhenger og mønstre ved hjelp av spesifikke observasjoner, til å abstrahere og formulere generelle uttrykk som beskriver sammenhenger ved hjelp av algebra og regneregler. En mulig forklaring av funnet er at elevene gjennom programmering har fått trening i å definere variabler og formulere algoritmer som kan forstås og brukes av en datamaskin.

I intervjuene med to av 1T elevene kommer det frem at de ser overføringsverdien mellom problemløsning i arbeid med programmering og problemløsning i matematikk. Den ene eleven beskriver hvordan arbeid med programmering hjelper ham å holde oversikt, analysere oppgaven og tilgjengelig informasjon og legge en stegvis plan for å komme frem til løsningen. Eleven forteller at denne tankegangen har overføringsverdi til matematikk, og vil gjøre problemløsning mer systematisk.

Den andre av de to elevene forteller også at programmering preges av stegvis problemløsning, som han med fordel kan videreføre til matematisk problemløsning for å få bedre oversikt og potensielt unngå slurvefeil.

PRIMM-modellen som en programmeringsspesifikk problemløsningsstrategi er forenelig med Polyas (1945) 4 steg for matematisk problemløsning. Både teorien og resultatene fra de empiriske studiene til Husain et al. (2017), Saez-López et al. (2019) og Kaufmann & Stenseth (2021) er forenelige med funnene gjort i denne studien, som underbygger styrking av studiens reliabilitet.

Studiens funn indikerer at elevene utvikler sine problemløsningsferdigheter i matematikk gjennom arbeid med programmering. I prestasjonstesten viser 15 av 21 elever forbedring innen abstraksjonsferdigheter og hvordan de bygger opp matematiske resonnement. Det er likevel en signifikant svakhet i studiens validitet ettersom det ikke er mulig å skille om resultatene skyldes programmeringen alene eller undervisningsoppleggets utforskende karakter.

I studien ble det også avdekket faktorer som ser ut til å hindre algoritmisk tenkning:

- Begrenset kreativitet
- Tid til å utforske
- Terskelbegreper

Programmering er anerkjent å være en kreativ prosess (Grover & Pea, 2013). Kreativitet viste seg å være en av faktorene som gjorde det vanskelig for elever å komme i mål med programmeringsoppgavene. Flere elever forklarer hvordan de ikke klarer å jobbe seg videre når de sitter fast. Matematisk kreativitet kjennetegnes ved fleksibel tankegang, som innebærer divergent produksjon og evnen til å overkomme fikseringer (Haylock, 1997). Thea uttrykker også at det eksplisitt er begrenset kreativitet som gjør det vanskelig å se for seg hvordan

programmet skal bygges opp. Disse funnene indikerer at programmering krever kreativitet, og mangel på kreativitet hemmer algoritmisk tenking ved å begrense problemløsningsprosessen.

På den andre siden indikerer funnene at samarbeid senker terskelen for elevene til å overkomme fikseringer, som er et uttrykk for matematisk kreativitet (Haylock, 1997). Gjennom spørreskjema, intervju og observasjon av elevene i undervisningsopplegget kommer det også frem at elevene opplever det enklere å stå i usikkerhet og utfordring når de har en partner å diskutere med og støtte seg på. Utholdenhet i arbeid med oppgaver er en indikator på motivasjon (Wæge & Nosrati, 2018), i tillegg til å være en komponent i algoritmisk tenking (Utdanningsdirektoratet, 2019). Flertallet av elevene rapporterer at de foretrekker samarbeid når de jobber med programmering av de overnevnte grunner, i tillegg til at det øker trivselen deres. I tillegg til at funnene indikerer at samarbeid har positive effekter på elevenes kreativitet og utholdenhet, er også trivsel og motivasjon er et gode utgangspunkt for læring (Wæge & Nosrati, 2018).

En annen faktor som hemmer den algoritmiske tankeprosessen er om elevene ikke gis nok tid. En av elevene forteller i intervju at tid er en viktig faktor for å komme i mål med en programmeringsoppgave, og for å utvikle og forbedre programmene sine. Dette funnet underbygger teorien om at elever må få nok tid og rom til å utforske selvstendig for at et utforskende undervisningsopplegg skal være vellykket (Blomhøj, 2019).

I studien er det også avdekket at selve kodespråket, med syntaks, kommandoer og notasjon, er et terskelbegrep for elevene. I tilbakemeldingene til elevene gjennom både spørreskjema, observasjon og intervju kommer det frem at mange elever bruker mye tid på å lete frem riktige kommandoer, og at de bruker mye tid på å rette og forstå syntaksfeil som manglende intendering.

Terskelbegreper er essensielle byggesteiner for videreutvikling av kunnskapen på området (Meyer & Land, 2005). Flere elever foreslår at det ville blitt lettere å komme i mål med programmeringsoppgaver dersom de var tryggere på det en av elevene kaller *«basics»* og hadde fått lenger tid til å *«synke det inn.»* Dette er karakteristisk for terskelbegreper, som tar tid å internalisere, men som åpner opp for ny innsikt og dypere forståelse når elevene behersker begrepene (Pettersson & Brandell, 2017).

For å besvare problemstillingen, så indikerer funnene i studien at programmering kan brukes som et verktøy i matematikkundervisning for å fremme algoritmisk tenkning. Blant elevene

som har deltatt i studien viser 15/21 elever forbedringer i sine abstraksjonsferdigheter og matematiske resonneringsevne. To av elevene som ble intervjuet forklarte hvordan de gjennom programmeringsopplegget hadde læringsutbytte som kunne videreføres til matematisk problemløsningsstrategi. I både arbeid med programmering og i prestasjonstesten demonstrerer elevene kognitive prosesser som analyse, dekomponering, mønstergjenkjenning, abstrahering og feilsøking, som inngår i definisjonen av algoritmisk tenkning (Grover & Pea, 2013; Wing, 2006).

Det må likevel understrekes at studien er en casestudie basert på 21 individer i en 1T klasse, som derfor antas å være motiverte for matematikkfaget og har relativt høy grad av utholdenhet i møte med utfordringer. Hver klasse har sin egen, komplekse dynamikk bygget på intrikate nettverk av relasjoner og sosiale samspill, som gjør hver klasse til unike utvalg. Dette sammen med at utvalget er lite, gjør at studien har lav grad av reproduserbarhet og reliabilitet.

Casestudier kjennetegnes dog av å kunne gi et detaljert datagrunnlag av utvalget som studeres, og egner seg som utgangspunkt for å utvikle teorier (Flyvbjerg, 2010) . Denne studien vil dermed være et bidrag til pågående forskning om konsensus om både undervisningsmetode og programmerings rolle i matematikkfaget.

Et av funnene som vil være interessant å følge med på er om de terskelbegrep denne studien indikerer kan sammenlignes med terskelbegrep i andre lignende studier. Da den nye læreplanen ble innført i 2020 gikk elevene som deltok i denne studien i 9. klasse, og flertallet elevene rapporterer at de har hatt svært lite programmering før 1T. Det blir interessant å følge effekten av LK20 etter hvert som elever som har hatt programmering som en del av læreplanen fra tidligere alder flytter seg oppover i skolesystemet.

Indikasjoner styrket av denne studie på at kreativitet er en forutsetning for problemløsning i programmering kan være interessant for videre forskning. Læreplanen LK20 setter økt fokus på utforskning og problemløsning, og stiller høyere krav til elevenes matematikkompetanse, så vel som læreres didaktiske undervisningskompetanse (Kunnskapsdepartementet, 2020). Den relasjonelle matematikkunnskapen kan ses bestå av 3 komponenter; formell, algoritmisk og intuitiv. Det vil si at kreativitet er like viktig som aksiomer, teoremer, aritmetikk, notasjon og bevis for en holistisk matematikkompetanse. Kompetansemålene som er definert i læreplaner må gjenspeiles i undervisningspraksisen i skolen, og funnene i denne studien

underbygger Forsström & Kaufmanns (2018) anmodning om mer forskning og konsensus om undervisningsmetode i programmering og hvilken rolle det skal ha i matematikkfaget.

7 Litteratur

Algoritmisk tenkning. [Bilde]. (2019). Hentet fra: https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal*, *55*(7), 832-835. https://doi.org/:10.1093/comjnl/bxs074

Artigue, M., & Blomhøj, M. (2013). Conceptualizing Inquiry-Based Education in Mathematics. *ZDM - Mathematics Education*, *45*, 797-810. https://doi.org/10.1007/s11858-013-0506-6

Bandura, A. (1997). *Self-efficacy: The Exercise of Control*. W. H. Freeman and Company.

Blomhøj, M. (2019). *Fagdidaktikk i matematikk*. Frydenlund.

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introducing Computational Thinking and programming in compulsory education*. http://www.itd.cnr.it/doc/CompuThinkNordic.pdf

Cropley, A. J. (1992). *More ways than one: Fostering Creativity*. Ablex Publishing Corporation. https://archive.org/details/morewaysthanonef0000crop/page/n5/mode/2up

Cuoco, A., Goldenberg, E. P., & Mark, J. (1996). Habits of mind: An Organizing Principle for Mathematics Curricula. *The Journal of Mathematical Behavior*, *15*, 375-402.

De nasjonale forskningsetiske komiteene. (2021). *Forskningsetiske retningslinjer for samfunnsvitenskap og humaniora*. https://www.forskningsetikk.no/globalassets/dokumenter/4-publikasjoner-som-pdf/forskningsetiske-retningslinjer-for-samfunnsvitenskap-og-humaniora

Fischbein, E. (1994). The Interaction between the Formal, the Algorithmic and the Intuitive Components in a Mathematical Activity. *Didactics of mathematics as a scientific discipline*. https://www.ime.usp.br/~dpdias/2016/GEN5711%20-%20Fischbein.pdf

Flyvbjerg, B. (2010). Fem misforståelser om casestudiet (Five Misunderstandings about Case-Study Research). In L. Tanggard (Ed.), *Kvalitative metoder - en grundbog* (pp. 463-487). Hans Reitzels forlag.

Forsström, S. E., & Kaufmann, O. T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18-32. https://doi.org/https://doi.org/10.26803/ijlter.17.12.2

Gjøvik, Ø., & Torkildsen, H. A. (2019). Algoritmisk tenkning. *Tangenten: tidsskrift for matematikkundervisning*, *30*(3), 31-37. http://tangenten.no/wp-content/uploads/2021/12/Tangenten-3-2019-Gjovik-Torkildsen.pdf

Gleiss, M. S., & Sæther, E. (2022). *Forskningsmetode for lærerstudenter*. Cappelen Damm Akademisk.

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Haylock, D. (1997). Recognising Mathematical Creativity in Schoolchildren. *ZDM*, 68-74. https://doi.org/https://www.emis.de/journals/ZDM/zdm973a2.pdf

Hovik, E. K. (2016). *Undervisningskunnskaoe i matematikk* (B. Kleve, Ed.). Cappelen Damm Akademisk.

Husain, H., Kamal, N., Ibrahim, M. F., Huddin, A. B., & Alim, A. A. (2017). Engendering Problem Solving Skills and Mathematical Knowledge via Programming. *Journal of Engineering Science and Technology*, 1-11. http://jestec.taylors.edu.my/Special%20Issue_PEKA_2017/PEKA%202017_01.pdf

Israel-Fishelson, R., & Hershkovitz, A. (2022). Studying Interrelations of Computational Thinking and Creativity: A scoping review (2011–2020). *Computers & Education, 176*. https://doi.org/https://doi.org/10.1016/j.compedu.2021.104353

Israel-Fishelson, R., Hershkovitz, A., Eguíluz, A., Garaizar, P., & Guenaga, M. (2021). A log-based analysis of the associations between creativity and computational thinking. *Journal of Educational Computing Research*, *59*(5), 926-959. https://doi.org/https://doi.org/10.1177/0735633120973429

Johannesen, A., & Christoffersen, L. (2012). *Forskningsmetode for lærerutdanningene*. Abstrakt forlag.

Kaufmann, O. T., & Stenseth, B. (2021). Programming in Mathematics Education. *International journal of mathematical education in science and technology*. https://doi.org/10.1080/0020739X.2020.1736349

Kilhamn, C., Rolandsson, L., & Bråting, K. (2021). Programmering i svensk skolmatematik: Programming in Swedish school mathematics. *LUMAT: International Journal on Math, Science and Technology Education*, *9*(1), 283–312-283–312. https://doi.org/10.31129/LUMAT.9.2.1457

Koehler, M. J., Shin, T. S., & Mishra, P. (2012). How Do We Measure TPACK? Let Me Count the Ways. In *Educational Technology, Teacher Knowledge, and Classroom Impact* (pp. 16-31). IGI Global. https://doi.org/10.4018/978-1-60960-750-0.ch002

Kunnskapsdepartementet. (2017). *Overordnet del - verdier og prinsipper for grunnopplæringen*. Fastsatt som forskrift ved kongelig resolusjon. Læreplanverket for Kunnskapsløftet 2020 Retrieved from https://www.udir.no/lk20/overordnet-del/om-overordnet-del/

Kunnskapsdepartementet. (2019). *Læreplan i matematikk fellesfag vg1 teoretisk, matematikk T (MAT09-01)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020 Retrieved from https://www.udir.no/lk20/mat09-01

Kunnskapsdepartementet. (2020). *Læreplan i matematikk fellesfag, vg1 teoretisk (MAT09-01)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. Retrieved from https://www.udir.no/lk20/mat09-01

Laurent, M., Crisci, R., Bressoux, P., Chaachoua, H., Nurra, C., de Vries, E., & Tchounikine, P. (2022). Impact of Programming on Primary Mathematics Learning. *Learning and Instruction, 82*. https://doi.org/https://doi.org/10.1016/j.learninstruc.2022.101667

Lavoll, H. (2021, 10.02.2021). Research Focus. *MASCOT - Mathematics, Science and Computational Thinking*. https://uni.oslomet.no/mascot/

Lithner, J. (2007). A Research Framework for Creative and Imitative reasoning. *Eucational Studies in Mathematics, 67*, 255-276. https://doi.org/ 10.1007/s10649-007-9104-2

Lyngsnes, K., & Rismark, M. (2016). *Didaktisk arbeid* (3 ed.). Gyldendal Akademisk.

Lyngsnes, K., Rismark, M. (2016). *Didaktisk arbeid* (3 ed.). Gyldendahl.

Meyer, J. H. F., & Land, R. (2005). Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher education, 49*, 373-388. https://doi.org/10.1007/s10734-004-6779-5

Nøkleby, H., Berg, R., Muller, A. E., & Ames, H. M. R. (2021). *Konsekvenser av covid-19 på barn og unges liv og helse: en hurtigoversikt*. https://www.fhi.no/globalassets/dokumenterfiler/rapporter/2021/konsekvenser-av-covid-19-pa-barn-og-unges-liv-og-helse-rapport-2021.pdf

Pettersson, K., & Brandell, G. (2017). Å utvikle elevers begrepsforståelse. *Realfagsløyper*. https://realfagsloyper.no/sites/default/files/2018-04/T3.P1.M2A%20-A%CC%8A%20utvikle%20elevers%20begrepsforsta%CC%8Aelse%20Oversatt_0.pdf

Polya, G. (1945). *How to solve it: A new aspect of mathematical method*. Princeton University Press. https://books.google.no/books?id=z_hsbu9kyQQC&lpg=PP2&ots=oZpOShiUQ4&dq

=how%20to%20solve%20it%20polya%201945&lr&hl=no&pg=PR4#v=onepage&q&f=false

Postholm, M. B., & Jacobsen, D. I. (2018). *Forskningsmetode for masterstudenter i lærerutdanningen*. Cappelen Damm Akademisk.

Programmeringsspråk. (2019). In *Store norske leksikon*.

Rossen, E. (2022). Programmering. In *Store norske leksikon*.

Ryan, R. M., & Deci, E. L. (2000). Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *American Psychologist*, *55*(1), 68-78. https://doi.org/ 10.1037110003-066X.55.1.68

Sáez-López, J.-M., Sevillano-García, M.-L., & Vazquez-Cano, E. (2019). The Effect of Programming on Primary School Students' Mathematical and Scientific Understanding: Educational Use of mBot. *Educational Technology Research and Development*, *67*(6), 1405-1425. https://doi.org/doi.org/10.1007/s11423-019-09648-5

Sentance, S., & Waite, J. (2017, 8.-12. november 2017). *PRIMM: Exploring pedagogical approaches for teaching text-based programming in school* Proceedings of the 12th Workshop on Primary and Secondary Computing Education, Nijmegen, Netherlands.

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, *29*(2), 135-176. https://doi.org/https://doi.org/10.1080/08993408.2019.1608781

Shell, D. F., Hazley, M. P., Soh, L.-K., Ingraham, E., & Ramsay, S. (2013, 23-26 oktober 2013). *Associations of Students' Creativity, Motivation, and Self-Regulation with Learning and Achievement in College Computer Science Courses* 2013 IEEE Frontiers in Education Conference (FIE), Oklahoma CIty.

Shell, D. F., Hazley, M. P., Soh, L.-K., Miller, L. D., Chiriacescu, V., & Ingraham, E. (2014, 22-25 oktober 2014). *Improving learning of computational thinking using computational creativity exercises in a college CS-1 computer science course for engineers* 2014 ieee frontiers in education conference (FIE) proceedings, Madrid.

Shorten, A., & Smith, J. (2017). Mixed methods research: expanding the evidence base. *Evidence Based Nursing*, *20*(3), 74-75. https://doi.org/10.1136/eb-2017-102699

Skemp, R. R. (1976). Relational Understanding and Instrumental Understanding. *Mathematics Teaching*(77), 20-26. http://www.davidtall.com/skemp/pdfs/instrumental-relational.pdf

Stylianides, A. (2009). Breaking the Equation 'Empirical Argument = Proof'. *Mathematics Teaching*(213). https://nrich.maths.org/6664

UNESCO. (2005, 10.-12. Juni 2005). *Aspects of Literacy Assessment* Topics and Issues from the UNESCO Expert Meeting, Paris. https://unesdoc.unesco.org/ark:/48223/pf0000140125

Utdanningsdirektoratet. (2019). Algoritmisk tenkning. Retrieved 7. september 2022, from https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Wing, J. (2011). Research notebook: Computational thinking —What and why. *The link magazine*, *6*, 20-23. https://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wæge, K., & Nosrati, M. (2018). *Motivasjon i matematikk*. Universitetsforlaget.

Vedlegg 1: Godkjenning fra NSD

Vurdering av behandling av personopplysninger

Referansenummer	Vurderingstype	Dato
297588	Standard	22.12.2022

Prosjekttittel
Kan bruk av programmering i matematikkundervisning bidra til å fremme algoritmisk tenkning hos elever?

Behandlingsansvarlig institusjon
UiT Norges Arktiske Universitet / Fakultet for naturvitenskap og teknologi / Institutt for matematikk og statistikk

Prosjektansvarlig
Hilja Lisa Huru

Student
Mathea Fennefoss Johnsgård

Prosjektperiode
29.08.2022 - 01.09.2023

Kategorier personopplysninger
Alminnelige

Lovlig grunnlag

Samtykke (Personvernforordningen art. 6 nr. 1 bokstav a)

Behandlingen av personopplysningene er lovlig så fremt den gjennomføres som oppgitt i meldeskjemaet. Det lovlige grunnlaget gjelder til 01.09.2023.

Kommentar

OM VURDERINGEN
Sikt har en avtale med institusjonen du forsker eller studerer ved. Denne avtalen innebærer at vi skal gi deg råd slik at behandlingen av personopplysninger i prosjektet ditt er lovlig etter personvernregelverket.

FØLG DIN INSTITUSJONS RETNINGSLINJER
Vi har vurdert at du har lovlig grunnlag til å behandle personopplysningene, men husk at det er institusjonen du er ansatt/student ved som avgjør hvilke databehandlere du kan bruke og hvordan du må lagre og sikre data i ditt prosjekt. Husk å bruke leverandører som din institusjon har avtale med (f.eks. ved skylagring, nettspørreskjema, videosamtale el.)

Ved bruk av databehandler (spørreskjemaleverandør, skylagring, videosamtale o.l.) må behandlingen oppfylle kravene til bruk av databehandler, jf. art 28 og 29. Bruk leverandører som din institusjon har avtale med.

Personverntjenester legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til oss ved å oppdatere meldeskjemaet. Se våre nettsider om hvilke endringer du må melde: https://sikt.no/melde-endringar-i-meldeskjema

OPPFØLGING AV PROSJEKTET
Vi vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!

Vedlegg 2: Samtykkeskjema

UiT
NORGES
ARKTISKE
UNIVERSITET

Forespørsel om deltakelse i forskningsprosjektet: «Kan bruk av programmering i matematikkundervisningen fremme algoritmisk tenkning hos elever?»

Dette er et spørsmål til deg om å delta i et forskningsprosjekt med formål om å undersøke elevers opplevelser med programmering i matematikkundervisningen. Dette skrivet inneholder informasjon om prosjektets mål og hva deltakelse i prosjektet vil innebære for deg.

Formål

Programmering er en relativt ny del av læreplanen, og det jobbes ennå med å utarbeide effektive metoder for å best mulig inkorporere programmeringen i matematikkundervisningen. Denne studien har som formål å undersøke hvordan elevene opplever inkludering av programmering i matematikkundervisningen.

Elevene vil bli spurt om erfaringene de har med programmering, foretrukne arbeidsmetoder, problemløsningsstrategi og opplevelser med programmering i matematikkundervisningen.

Forskningsprosjektet er en masteroppgave som avlegges ved Universitetet i Tromsø.

Hvem er ansvarlig for forskningsprosjektet?

Institutt for matematikk og statistikk ved fakultet for naturvitenskap og teknologi ved Norges arktiske Universitet (UiT) er ansvarlig for forskningsprosjektet.

Hovedveileder for masteroppgaven er Jonas Oskarsson (Institutt for lærerutdanning og pedagogikk) og biveileder er Hilja Lisa Huru (Institutt for matematikk og statistikk).

Hvorfor får du spørsmål om å delta?

Utvalget er videregåendeelever i faget matematikk teoretisk (1T), hvor det er naturlig å inkludere programmering i de ulike delene av pensum.

Hva innebærer deltakelse i studien?

Dersom du velger å delta i forskningsprosjektet innebærer det at du må gjennomføre et oppgavesett i forkant og etterkant av et undervisningsopplegg som varer i 2 uker. Undervisningsopplegget og oppgavesettene vil gjennomføres i den ordinære undervisningstiden. Oppgavene løses skriftlig på ark som samles inn. Du vil også bli bedt om å svare på et spørreskjema med ferdigstilte svaralternativer, hvor du blir bedt om å rangere hvor enig/uenig du er med gitt utsagn. Noen elever vil bli spurt om å

delta i et intervju i etterkant av undervisningsopplegget for å kunne følge opp interessante funn i oppgaveløsningene eller i spørreskjemaet. Det vil bli gjort lydopptak av intervjuet. Observasjoner gjort underveis i datainnsamlingen som anses relevant kan også bli brukt som en del av datamaterialet. Alt datamaterialet vil bli **anonymisert.**

Hva skjer med informasjonen om deg?

Alle personopplysninger vil behandles konfidensielt og i samsvar med personvernregelverket. Kun medlemmene av forskningsprosjektet får tilgang på datamaterialet.

- De som vil ha tilgang på opplysningene om deg vil være Mathea Fennefoss Johnsgård og Hilja Lisa Huru. Hovedveileder Jonas Oskarsson vil kun få tilgang til anonymisert og ferdig transkribert materiale da han tilhører HiNN.
- Datamaterialet samlet inn på papir vil oppbevares i et låst skap på UiT. Behandling av datamaterialet vil bli lagret i et lukket område i Office365 innenfor UiTs servere som er under beskyttelse av tofaktor-autentisering. Spørreskjemaet som blir benyttet i forskningsprosjektet er Nettskjema, en sikker datainnsamlingsmetode som driftes av UiO.

Studien planlegges avsluttet 01.09.2023, innen da vil **alt datamaterialet være anonymisert og lydopptak slettet.**

Frivillig deltakelse

Deltakelse i studien er frivillig, og samtykket kan når som helst trekkes uten å måtte oppgi noen årsak. Dersom du ønsker å trekke deg fra studien vil alle personopplysninger og datamateriale om deg fjernes med mindre det allerede er brukt i publikasjoner.

Prosjektet er meldt inn til Norsk samfunnsvitenskapelige datatjeneste (NSD) for å sikre ivaretakelse av personvern i forskning ved UiT.

Ved spørsmål til studien, kontakt Jonas Oskarsson per epost: jonas.oskarsson@inn.no. Grunnet studentprosjekt oppgis kontaktinformasjonen til veileder for prosjektet.

Hva gir oss rett til å behandle personopplysninger om deg?

Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra Institutt for matematikk og statistikk ved Universitetet i Tromsø har Personverntjenester vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Dine rettigheter

Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke opplysninger vi behandler om deg, og å få utlevert en kopi av opplysningene
- å få rettet opplysninger om deg som er feil eller misvisende
- å få slettet personopplysninger om deg
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger

Hvis du har spørsmål til studien, eller ønsker å vite mer om eller benytte deg av dine rettigheter, ta kontakt med:

Institutt for matematikk og statistikk ved Universitetet i Tromsø ved:

- Veileder: Jonas Oskarsson | jonas.oskarsson@inn.no | +47 41 67 16 28
- Biveileder: Hilja Lisa Huru | hilja.huru@uit.no | +47 77 66 02 53
- Masterstudent: Mathea Fennefoss Johnsgård | mjo277@uit.no | +47 48 15 75 58
- Vårt personvernombud: Joakim Bakkvold | personvernombudet@uit.no | +47 77 64 63 22

Dersom du har spørsmål knyttet til Personverntjenester sin vurdering av prosjektet, kan du ta kontakt med:

- Personverntjenester på epost (personverntjenester@sikt.no) eller på telefon: 53 21 15 00.

Med vennlig hilsen

Jonas Oskarsson Hilja Lisa Huru Mathea Fennefoss Johnsgård

UiT
NORGES
ARKTISKE
UNIVERSITET

Samtykke til deltakelse i studien

Navn: _____

□ Jeg samtykker til å delta i forskningsprosjektet.

□ Jeg samtykker til at data som samles inn gjennom prestasjonstest, spørreskjema, observasjon

□ Jeg samtykker til å delta på intervju

Jeg er informert om studien, og er villig til å delta i forskningsprosjektet

---

(Signatur prosjektdeltaker, dato)

Vedlegg 3: Spørreskjema

Nettskjema

Elevers opplevelse av programmering i matematikk

Generert: 2023-06-11 20:50:00.

Spørreskjema: elevers opplevelse av programmering i matematikkundervisningen

Dette spørreskjemaet benyttes i forbindelse med datainnsamling til et masterprosjekt. Spørreskjemaet har som formål å samle inn informasjon om elevers opplevelse av- og erfaringer med programmering i matematikkundervisningen.

Oppgi kandidatnummeret ditt i feltet nedenfor

Har du arbeidet med programmering før 1T?

- □ Ja
- □ Nei
- □ Litt

Hvor har du arbeidet med programmering før 1T?

Dette elementet vises kun dersom alternativet «Ja eller Litt» er valgt i spørsmålet «Har du arbeidet med programmering før 1T?»

- □ Ungdomsskolen
- □ Vitensenteret (eller lignende)
- □ På fritiden
- □ Lego League
- □ Annet

Hva er dine opplevelser med programmering?

Kryss gjerne av flere

- □ Morsomt
- □ Utfordrende
- □ Kjedelig
- □ Vanskelig
- □ Mestringsfølelse
- □ Ser ikke nytteverdien
- □ Stressende
- □ Nyttig

Hvor enig er du med gitt påstand:

Velg et av de følgende fire alternativer for hver påstand

□ Helt uenig

□ Litt uenig

□ Litt enig

□ Helt enig

- Jeg opplever mestring i arbeid med programmering
- Jeg synes programmering er frustrerende
- Jeg forstår ikke hva jeg skal bruke programmering til
- Programmering gjør meg mer motivert til å arbeide med matematikk
- Jeg synes programmering tar for stor plass i matematikkfaget
- Jeg misliker å jobbe med programmering
- Jeg synes det er nyttig å lære programmering
- Jeg kan bruke programmering til å løse matematiske problemer
- Jeg er usikker på hvordan jeg skal gå i gang med programmeringsoppgaver
- Jeg har en god strategi for å arbeide med programmering
- Jeg liker å jobbe med programmering
- Jeg opplever ingen mestringsfølelse i arbeid med programmering
- Å jobbe med programmering gir meg muligheten til å tenke kreativt
- Jeg blir nervøs når vi skal jobbe med programmering
- Jeg føler ikke jeg lærer noe av å arbeide med programmering
- Programmering passer bra inn i 1T-faget
- Jeg ser ikke sammenhengen mellom matematikken vi lærer og programmeringen vi gjør
- Jeg foretrekker andre arbeidsmåter innen matematikk
- Jeg skulle gjerne brukt mer tid på programmering
- Programmering kan brukes i alle delemnene av 1T
- Når jeg møter på utfordringer i forbindelse med programmering gir jeg fort opp
- Når jeg sliter med en programmeringsoppgave prøver jeg ut ulike strategier for å komme i mål
- Jeg opplever å komme inn i flytsonen mens jeg jobber med programmering
- I programmering er det mange ulike løsninger som er riktige
- Jeg synes det er vanskelig å konsentrere seg om programmering
- Hva liker du best med å arbeide med programmering?
- Liker du best å jobbe selvstendig eller sammen med andre når du jobber med
- programmering?
- Hva liker du minst med å arbeide med programmering?
- Beskriv i korte trekk hvordan du går frem når du skal løse en oppgave ved hjelp av
- programmering:
- Hva synes du om måten vi har arbeidet med programmering på i timene?

Vedlegg 4: Intervjuguide

Intervjuguide

Erfaring og holdning

1) Har du vært borti programmering før 1T?
2) Hva forbinder du med programmering?
3) Hva er din motivasjon for programmering?
4) Hvorfor tror du at programmering er blitt en del av mattefaget?
5) På hvilken måte passer det inn/ikke inn?

Opplevelser, følelser, motivasjon

6) Kan utdype litt om hva som er dine opplevelser med programmering?
7) Har du et eksempel på når det kan være nyttig å bruke programmering i matematikk?
8) Når kan det være nyttig å bruke programmering? Når er det ikke nyttig?
9) Hva er de største utfordringene med programmering?
10) Hva er vanskeligst? Hva er lettest?
11) Opplever du mestringsfølelse i programmering? Når?
12) Hva påvirker motivasjonen din positivt og negativt når du jobber med programmering?
13) Hvordan er motivasjonen din i matematikk sammenlignet med programmering?

Problemløsningsstrategi:

14) På hvilken måte tenker du når du skal gå i gang med en programmeringsoppgave?
    a. Liker du best å samarbeide eller jobbe selvstendig?
15) Hva gjør du når du står fast med en programmeringsoppgave?
16) Vet du hva PRIMM står for?
17) Bruker du PRIMM når du jobber med programmeringsoppgaver? Hvorfor/Hvorfor ikke?
18) Hva synes du var bra med måten v i jobbet på?
19) Hva var dårlig? Hva var bra?
20) Lærte du noe gjennom programmering du har fått nytte av i andre matematikkoppgaver?

21) De oppgavesettene vi gjorde før og etter programmeringsopplegget
    a. Hva synes du om oppgavene?
    b. Var de ulike i vanskelighetsgrad?
22) Føler du at du har hatt nok tid til å
    a. lære programmering
    b. bruke og utforske programmering
23) Følte du at du lærte noe gjennom programmeringsopplegget vi hadde?
    a. Hva?
    b. Fikk du noe endret syn på programmering?
24) Er det noe du har lyst til å tilføye til det jeg undersøker i masteren min, som handler om programmering og matematisk læring?
25) Eventuelt: Følge opp interessante funn fra pretest/retest. Hvordan tenkte du om denne oppgaven?

Vedlegg 5: Pretest

Oppgave 1:

Vurder følgende påstand, og prøv å bevise eller motbevise den:

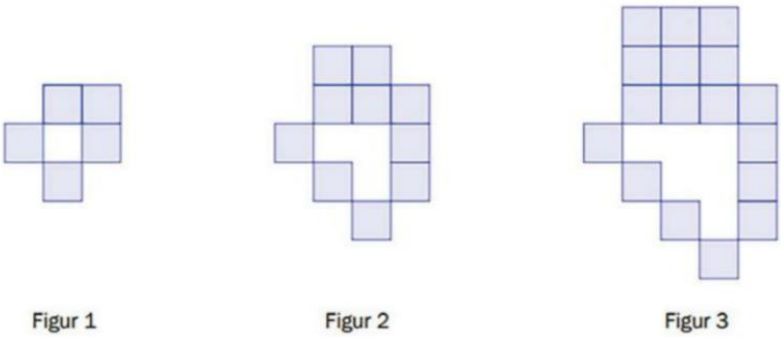
1) **Summen av tre påfølgende tall er alltid delelig med tre**
   (3 påfølgende tall = 3 tall som kommer etter hverandre)

2) Vurder også disse påstandene:
   - Summen av fire påfølgende tall er alltid/aldri/noen ganger delelig med 4
   - Summen av fem påfølgende tall er alltid/aldri/noen ganger delelig med 5
   - Summen av seks påfølgende tall er alltid/aldri/noen ganger delelig med 6

Ser du noen mønster mens du utforsker påstandene? Kan du lage en hypotese og undersøke den?

Oppgave 2:

Nedenfor ser du 3 figurer. Figurene er satt sammen av små, blå kvadrater. Tenk deg at du skal fortsette å lage figurer etter samme mønster.

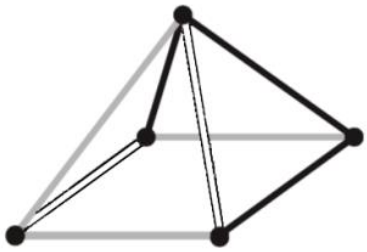


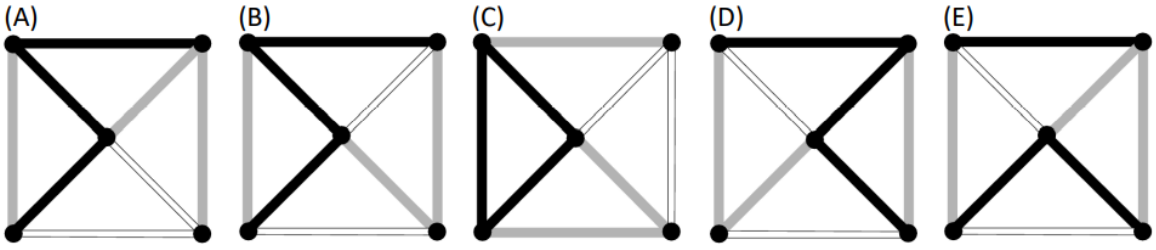
Figur 1 Figur 2 Figur 3

a) Tegn figur 4
b) Hvor mange små, blå kvadrater vil det være i figur 5?
c) Bestem et uttrykk for antall små, blå kvadrater for hvilken som helst figur av denne typen
d) Hvor mange små, blå kvadrater vil det være i figur 100? Du trenger ikke regne det ut, kun beskrive fremgangsmetode.

Oppgave 3:

Bildet viser en figur sett fra siden. Hvordan vil figuren se ut ovenfra? Forklar hvorfor kun ett alternativ er riktig.





Oppgave 4:

Miriama har et kvadratisk vindu som består av 9 små glassruter inni. Hun ønsker å sette inn noen røde glassruter og noen blanke.

Hvordan kan Miriama sette inn ruter i vinduet sitt **uten** at 3 røde er på rad, samtidig som det **alltid** vil bli 3 røde på rad dersom hun setter inn en rød rute til? (Diagonal teller også som 3 på rad)

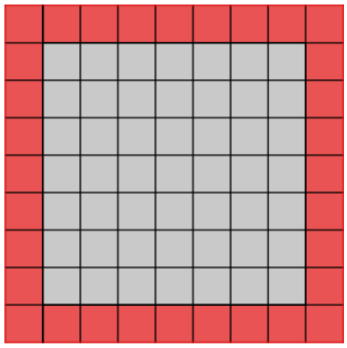
1) Hva er det minste antall røde ruter hun kan sette inn?
2) Hvor mange løsninger klarer du å finne? Tegn alle!

Vedlegg 6: Retest

Oppgave 1: Hvor stor er rammen?

Figuren er et kvadrat med 81 ruter. De røde rutene ytterst kaller vi **rammen** i kvadratet.

Hvor mange brikker er det i rammen? Merk deg hvordan du tenker når du løser denne oppgaven.



1) Finn antall ruter i rammen. Forklar/vis hvordan du har tenkt!

2) Bruk samme måten å tenke på for raskt å finne ut hvor mange ruter det vil være i **rammene** til kvadrater med 16, 25 og 36 ruter.

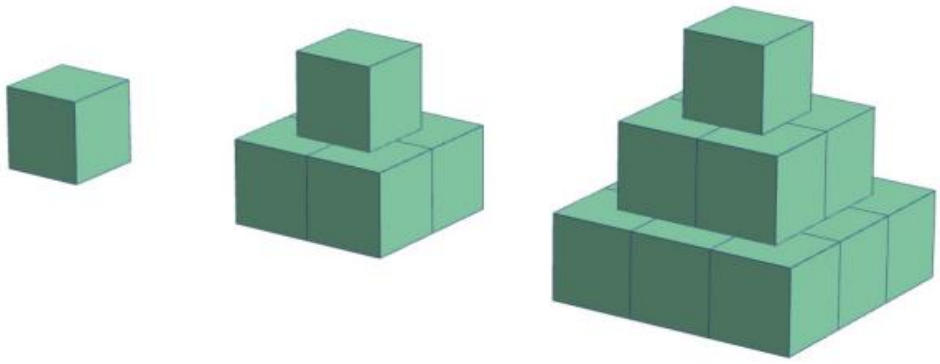
3) Kan du tenke deg flere måter å finne dette antallet på? Kan du illustrere disse også?

4) Hvor mange ruter vil det være i rammen til et kvadrat av hvilken som helst størrelse?

Ser du noe mønster mens du utforsker kvadrater av ulik størrelse? Kan du lage en hypotese og utforske den?

Oppgave 2:

Nedenfor ser du 3 figurer. Figurene er satt sammen av små klosser. Roar vil fortsette å lage



Figur 1 Figur 2 Figur 3

figurer etter samme mønster.

1) Tegn figur 4

2) Hvor mange klosser trenger han for å lage figur 5?

3) Bestem et uttrykk for antall klosser i hvilken som helst figur av denne typen.

4) Hvor mange klosser vil det være i figur 10? Du trenger ikke regne det ut, kun beskrive fremgangsmetode.

Oppgave 3:

Det er skrevet KANGAROO på en paraply. Se bildet.

Hvilket alternativ kan være denne paraplyen? Forklar hvorfor kun ett alternativ er riktig.

a)

b)

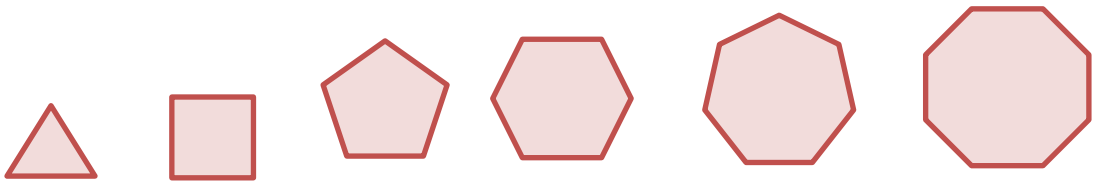
c)

d)

e)

Oppgave 4:

En diagonal er en rett linje som kan trekkes fra et hjørne til et annet hjørne i en mangekant.



1) Hvor mange diagonaler er det i de ulike mangekantene?

2) Ser du noe mønster? Utforsk og forklar!