UiT Norges arktiske universitet

Fakultet for humaniora, samfunnsvitenskap og lærerutdanning

Læreres syn på programmering i matematikk

Kvantitativ studie av læreres syn på programmering og algoritmisk tenkning i matematikk på mellomtrinnet etter innføring av den nye læreplanen

Håvard Salamonsen

LER-3903 – Masteroppgave i matematikkdidaktikk

Mai 2024

Sammendrag

Etter kunnskapsløftet 2020 har programmering blitt en del av læreplanen i matematikk, som medfører at lærere er pålagt å implementere programmering i deres undervisningspraksis. Denne masteroppgaven har som hensikt å undersøke lærernes syn på innføringen av programmering i matematikkfaget, derav er problemstillingen for oppgaven: Hva synes matematikklærere på mellomtrinnet om å inkludere programmering i matematikkfaget etter fagfornyelsen? De tre forskningsspørsmålene retter fokus mot bruken og nytten av programmering i matematikk, lærernes forståelse av begrepet algoritmisk tenkning og til slutt lærernes holdninger til innføringen av programmering i matematikkfaget.

Det teoretiske rammeverket som har vært grunnleggende for oppgaven består av forskningen til Misfeldt et al. (2019), Korhonen et al. (2022) og Bocconi et al. (2016). For å kunne besvare problemstillingen har det vært gjennomført en spørreundersøkelse med kvantitativ tilnærming med 45 respondenter. Nettskjema har blitt tatt i bruk for datainnsamling og SPSS har blitt brukt som dataanalyseverktøy.

Lærerne ser en rekke fordeler og muligheter innenfor programmering i matematikkfaget. Blokkprogrammering gjør det mulig å undervise programmering på mellomtrinnet. Koblingen mellom programmering og matematikk er noe utydelig, og økt kompetanse blant lærerne kan være en viktig faktor for å skape denne koblingen for elevene. Lærerne har en god forståelse av begrepet algoritmisk tenkning og hvordan det kan anvendes i undervisningssammenheng. Lærerne har en litt annen forståelse av algoritmisk tenkning enn begrepets opprinnelige definisjon, ved at bidraget til digitale midler ikke er like viktig på mellomtrinnet. Som en homogen gruppe er lærerne entusiastiske for å undervise programmering i matematikk, men de føler på mangler i deres kompetanse og evne til å undervise. Etterutdanning og kurs virker til å være en viktig faktor for lærernes motivasjon, kompetanse og ressursbruk.

Forord

Masteroppgaven markerer slutter på den femårige lærerutdanninga ved UiT. Det siste året har bydd på en rekke utfordringer som har vært både krevende og interessant å løse, og det er flere som fortjener en stor takk fra meg.

Tusen takk til min veileder Odd Tore Kaufmann for et trivelig og godt samarbeid, det har vært interessant å diskutere rundt forskningsprosessen med noen så dyktig og kunnskapsrik. Jeg vil også takke min familie for støtten gjennom utdanningsløpet. Dere er alltid behjelpelig, enten jeg spør eller ikke, særlig takk til min søster Mari som har korrekturlest oppgaven. Takk til kolleger på jobb som gir meg en så spennende og fortreffelig hverdag jeg kunne ha håpet på.

Sist, men ikke minst, takk til alle som har delt og besvart spørreskjemaet. Det er de som har gjort forskningen mulig å gjennomføre.

Innholdsfortegnelse

Tabelliste

Figurliste

1 Innledning

I den nye læreplanen for matematikk har programmering blitt implementert i kompetansemål for grunnskolen (Kunnskapsdepartementet, 2020). Denne endringen av læreplanen stiller en rekke krav til lærernes kompetanse, i et fagfelt som ikke har vært en del av læreplanen for matematikk før nå. Er de norske lærerne motivert, kompetent og forberedt på utfordringene de står ovenfor, og hva er det som kreves av dem? Dette masterprosjektet vil forhåpentligvis så godt som mulig besvare disse spørsmålene og bidra til å tette deler av kunnskapshullet, samt gi ideer til videre forskning på feltet.

1.1 Bakgrunn for valg av tema

Bakgrunnen for mitt valg av tema er basert på mine egne erfaringer og interesser fra skolegang og utdannelse. Min erfaring etter to år med informasjonsteknologi på videregående skole og noe undervisning ved på UiT innen programmering, er at det kan være krevende å knytte programmering opp mot matematikkundervisningen i grunnskolen, noe Kaufmann & Stenseth (2020) trekker fram i sin forskning. Programmering og algoritmisk tenkning har blitt implementert i den nye læreplanen for matematikk på mellomtrinnet (Kunnskapsdepartementet, 2020). Denne implementeringen stiller krav til læreres evne til å drive opplæring hvor elevene tilegner seg kompetansen fra læreplanen[1]. I denne oppgaven ønsker jeg å finne ut hva forskning sier om disse begrepene og undersøke hvilke holdninger lærerne har til matematikkundervisningen som omhandler programmering og algoritmisk tenkning. Jeg ønsker å undersøke lærerne både som en homogen gruppe, men også se om alder og deltakelse i etterutdanning og kurs har påvirket deres holdninger.

1.2 Tidligere forskning

Siden programmering i skolen er svært nytt sammenlignet med mange andre forskningsområder, er det i kontrast lite forskning som er gjennomført. Mye av forskning på masternivå innenfor programmering har blitt gjort på valgfag i ungdomsskolen og videregående opplæring. Det er stor enighet om at nytten av programmering i skolen er relevant for den fremtidige digitale verden (Neverdahl, 2019; Stenlund, 2021). Thorsnes

[1] Hentet fra prosjektskisse

(2020) viser til at lærere ikke trenger å være gode programmerere for å kunne undervise, mens Mæland (2021) viser til litt av det motsatte, hvor lærerne ga uttrykk for usikkerhet og mangel på kompetanse for å kunne undervise. Selv om programmering alt er innført som en del av andre fag, ikke bare matematikk, er det påpekt en nytte av å ha programmering som et eget fag (Neverdahl, 2019; Stenlund, 2021).

De fire masterprosjektene som henvises til ovenfor er alle kvalitative studier, men det finnes også kvantitativ forskning om hva lærere mener om at programmering skal innføres i matematikkfaget. For eksempel har Misfeldt et al. (2019) gjennomført en studie blant lærere i Sverige. Deres forskning viser til at lærerne som mangler kompetanse innenfor programmering ikke ser en like stor nytte av å undervise dette i matematikkfaget, og heller ikke ser sammenhengen mellom programmering og matematikk[2]. Korhonen et al., (2022) har gjort lignende forskning i Finland, hvor lærerne også ytrer en rekke bekymringer rundt implementeringen av programmering i deres læreplan. Varierende entusiasme, motivasjon, ressurser, kompetanse og holdninger noen av funnene blant lærere som har deltatt på kurs om programmering.

I forskningen trekkes det frem en rekke viktige grunner til å implementere programmering i den nye læreplanen. Blokkprogrammering har gjort det mulig å lære elever i yngre alder konsepter, som før ikke har vært mulig med tekstprogrammering. Blokkene gjør det mulig for elevene å utvikle interesse, kreativitet og problemløsningsegenskaper (Laurent et al, 2022; Resnick, 2019). Mens det fortsatt er noe usikkert hvilken sammenheng det er mellom matematikk og programmering, er utviklingen av algoritmisk tenkning trukket frem som en viktig problemløsningsmetode (Bocconi et al., 2016; Kaufmann & Stenseth, 2021; Kilhamn et al, 2021). En rekke modeller viser til forskjellige kunnskaper og ferdigheter som kreves av den algoritmiske tenkeren og hvilken virkning det har å kunne tenke algoritmisk (Bocconi et al., 2016; Shute et al, 2007; Wing, 2006).

Etter å ha lest meg opp på forskningsfronten, så virker det til å være lite forskning på norske mellomtrinnslærere og deres syn på implementeringen av programmering. Det er i overgangen fra småtrinnet til mellomtrinnet elevene møter på programmering, derfor er det

[2] Delvis hentet fra prosjektskisse

viktig å finne ut hva disse lærerne tenker om programmering i matematikk. Dette er kunnskapshullet i forskningsfeltet og danner grunnlag for problemstillingen.

1.3 Problemstilling og forskningsspørsmål

Kunnskapshullet i forskningsfeltet og min egen nysgjerrighet for lærernes holdninger til et fagfelt jeg selv er begeistret for, var utgangspunktet for masterprosjektet. Selv føler jeg med ikke trygg på hvordan programmering kan undervises på mellomtrinnet, og min tanke var dersom jeg ikke er trygg på det, er det med stor sannsynlighet en rekke andre lærere som er deler noen av usikkerhetsmomentene ved å undervise i dette. Den nye læreplanen danner et behov for å forskning på virkningen av endringene fra den gamle læreplanen. Siden forskningsfeltet er rimelig ferskt, vil prosjektet ikke gå så mye i dybden, men heller gi et overblikk over lærernes holdninger.

Problemstillingen for mitt masterprosjekt er dermed:

> **Hva synes matematikklærere på mellomtrinnet om å inkludere programmering i matematikkfaget etter fagfornyelsen?**

Med forskningsspørsmål som gjør hensikten bak prosjektet tydelig:

I. Hvilket syn har lærerne på undervisning av programmering i matematikk?
II. Hva er læreres forståelse av algoritmisk tenkning?
III. Hvilke holdninger har lærere til programmering?

Forskningsspørsmålene deler oppgaven i tre deler. Det første forskningsspørsmålet vil besvares ved å se på hva lærernes synes om bruken av programmering, hvordan blokkprogrammering gjør det mulig å undervise på mellomtrinnet og hvilken sammenheng programmering har med matematikk. Det andre forskningsspørsmålet viser til en problemløsningsmetode som er nært tilknyttet programmering, algoritmisk tenkning, og hvilken forståelse lærerne har av dette begrepet. Det tredje forskningsspørsmålet utforsker hvilke holdninger lærerne har til deres egen interesse, kompetanse og ressurser innenfor programmering.

Jeg har valgt å gjennomføre en spørreundersøkelse med kvantitativ tilnærming for å kunne gi et bedre bilde av norske læreres syn på programmering i matematikk på mellomtrinnet (Gleiss & Sæther, 2021). Spørreskjemaet vil kunne gi et stort datasett med verdier av

operasjonaliserte begreper, og via statistiske analyser vil potensielt kunne avdekkes signifikante funn. En viktig del av analysen har vært å se på hvordan de uavhengige variablene alder og etterutdanning påvirker de avhengige variablene. Prosjektet vil forhåpentligvis tette noe av det eksisterende kunnskapshullet, selv ved begrensninger i muligheter for generaliseringer i mitt masterprosjekt[3].

1.4 Struktur for oppgaven

Strukturen for oppgaven baserer seg på en del av masteroppgaven jeg har lest om kvantitativ forskning. Jeg har hentet inspirasjon fra andre oppgaver som var strukturert og oversiktlig. Kapittel 3 tar for seg den metodiske tilnærmingen for oppgaven, hvor det reflekteres rundt valg som har blitt gjort for å øke kvaliteten på forskningen. Teori- og analysekapitlene er strukturert etter ett forskningsspørsmål hver, 2.2 og 4.1 handler om programmering, 2.3 og 4.2 handler om algoritmisk tenkning og 2.4 og 4.3 handler om lærernes holdning til programmering og algoritmisk tenking i matematikkfaget. Jeg har slått sammen analyse og diskusjon i kapittel 4, ettersom dette ga mest ryddig struktur for oppgaven.

Eksamen for metode til masteremnet besto av å skrive en prosjektskisse for masteroppgaven. Jeg har tatt i bruk noe av min egen tekst fra dette arbeidskravet og lagt til en fotnote som presiserer om teksten er likt eller delvis hentet fra prosjektskissen, for å ikke plagiere meg selv.

[3] Delvis hentet fra prosjektskisse

2 Teori

Teoridelen vil være strukturert i en lik rekkefølge som spørreskjemaet som ble sendt i forbindelse med denne oppgaven. Endringer i læreplaner og samfunnsinteresser vil først danne et grunnlag for hvorfor programmering har blitt, og kommer til å bli en større del av undervisningen i skolen.

Videre vil det redegjøres for innføringen av programmering i skolen og hvilke muligheter og utfordringer som finnes innen dette emnet som motivasjon, vanskelighetsgrad og læring. Blokkprogrammering er en form for programmering som åpner for å undervise programmering til elever på lavere trinn i grunnskolen, og dekker for noen av de tidligere utfordringene som gjorde det vanskelig for elever som ikke hadde gode forutsetninger for å lære tekstprogrammering. Det er også mye diskutert hvilket fagområde programmering tilhører, og siden det er en del av læreplanen i matematikk er det viktig å se på hvilke sammenhenger det finnes mellom programmering og matematikk. Er programmering en del av matematikken eller er det et hjelpemiddel eller undervisningsmetode for å lære matematikk?

Noe som også er nylig innført i den nye læreplanen er begrepet algoritmisk tenkning, på engelsk *computational thinking* (CT). Begrepet ble definert av Jeanette Wing (2006) og har de siste årene gjennomgått en rekke endringer i hvordan begrepet defineres innenfor ulike fagfelt og hvilke komponenter begrepet består av. Algoritmisk tenkning er beskrevet under det matematiske kjerneelementet «utforskning og problemløsning» og har en rekke likheter med hvordan det defineres internasjonalt. Både Norge og andre land har ofte fem eller seks punkter for ulike former for kunnskaper som kreves for å kunne tenke algoritmisk, samt omtrent like mange punkter for egenskaper og arbeidsmåter som legger til rette for algoritmisk tenkning (Bocconi et al., 2016; Utdanningsdirektoratet, 2019).

Til slutt vil det være viktig å gjøre rede for hvilke holdninger grunnskolelærerne har, for å undervise emner de ikke har undervist før. Verken programmering eller algoritmisk tenkning har eksplisitt vært en del av læreplanen matematikklærerne forholder seg til. Dette medfører at det kan være behov for videreutdanning av lærerne, og for å kunne gjøre dette på en best mulig måte er det viktig å vite hvilke holdninger lærerne har til deres egen motivasjon og kompetanse.

2.1 Hva er nytt i læreplanen og behovet for programmering

Når man skal se på bruken av programmering i skolen vil det være nyttig å vite hvilke rammeverk som er satt for undervisning i skolen. I denne oppgaven vil lærerne som responderer på spørreundersøkelsen ha læreplanen for matematikk som de er lovpålagt å følge. Punktene i læreplanen som er aktuell for studien vil være kompetansemålene for 5., 6. og 7. trinn samt kjerneelementene som «er det elevene må lære for å kunne mestre og anvende faget» (Utdanningsdirektoratet, 2019). Hvert eneste kompetansemål har sammenhengende kjerneelementer presisert av Kunnskapsdepartementet under «støtte til læreplanen» (Kunnskapsdepartementet, 2020).

Dette er kompetansemålene som eksplisitt nevner programmering:

- 5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker
- 6. trinn: bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre
- 7. trinn: bruke programmering til å utforske data i tabeller og datasett[4]

Utdanningsdirektoratet har satt kjerneelementene «utforskning og problemløsning» som felles for alle tre og «abstraksjon og generalisering» for kompetansemålene på 5. og 6. trinn. Kjerneelementet «utforskning og problemløsning» beskriver hvordan elevene skal klare å skape en forståelse av hvordan de selv kan løse problemer ved en egnet fremgangsmåte. En rekke av egenskapene for å finne gode løsninger beskrives av problemløsningsmetoden algoritmisk tenkning, ofte brukt av informatikere og matematikere (Kunnskapsdepartementet, 2020). «Abstraksjon og generalisering» er også to av nøkkelbegrepene fra algoritmisk tenkning, og begge disse begrepene er beskrevet under kjerneelementet «utforskning og problemløsning» i likhet med hvordan de er beskrevet i kapittel 3 i denne oppgaven. «Abstraksjon og generalisering» knyttes bare til kompetansemålene på 5. og 6. trinn, hvor grunnen til at det ikke er koblet til kompetansemålet på 7. trinn er utydelig. Basert på hvordan kjerneelementene er formulert er abstraksjon og generalisering viktige egenskaper for å utforske tabeller og datasett. CSTA (Computer Science Teachers Association) og ISTE (International Society for Technology in Education) laget en definisjon av begrepet algoritmisk tenkning, hvor et av kjennetegnene var å kunne *logically organize and analyze*

[4] Hentet fra prosjektskisse

data (Bocconi et al. 2016). Slik både Kunnskapsdepartementet (2020) og annen forskning i kapittel 2.3, definerer begrepet algoritmisk tenkning, vil det være logisk at begge kjerneelementene støttet opp mot kompetansemålene under programmering i matematikk. De nye kompetansemålene er laget slik at de åpner for tolkning av læreren når det kommer til hvordan de ønsker å undervise, men i de aller fleste situasjoner når elevene programmerer, vil de har bruk for kompetansen disse to kjerneelementene beskriver.

2.1.1 Nasjonale offentlige utredninger

Det er tydelig at den nye læreplanen i større grad vektlegger programmering og utvikling av elevers digitale ferdigheter blir viktigere i tiden fremover (Kunnskapsdepartementet, 2020). Nasjonale offentlige utredninger (NOU) drøfter disse verdiene og hvordan samfunnet og skolen er i utvikling. Både samfunnet og skolen er stadig i utvikling og det krever endring i skolens innhold for å oppfylle dets samfunnsmandat. Digitalt kompetansebehov nevnes både for den enkelte og samfunnet som et behov i NOU – *fremtidige kompetanse behov 1* (2018: 2). På grunn av økt digitalisering og automatisering endres kompetansebehovet, og elever må tilegne seg den kompetansen som kreves i eksisterende og nye yrker. NOU – *hindre for digital verdiskaping* (2013: 2) peker på at skolen i for liten grad vektlegger digital kompetanse, og at det ikke er tilstrekkelig å kun lære elever digitale ferdigheter tverrfaglig. Digitalutvalget peker også på at manglende programmeringsferdigheter blant elevene i skolen, medfører at skolen utvikler mennesker som konsumerer og ikke har forståelse for digitale midler. Dette er på tross av at «bruke og forstå» er at av underpunktene i digitale ferdigheter. Det påpekes behov for økt digital kompetanse blant de som har IKT som en sentral del av sitt yrke, og de som arbeider i andre sektorer «for å sikre en forståelse av hva som skjer bak skjermen» (NOU, 2013 :2). NOU – *fremtidens skole* (2015: 8), peker også på like behov for utvikling av digital kompetanse på tvers av fagene i skolen, spesielt innenfor matematikk og naturfag.

2.2 Programmering

Programmering har gjort sitt inntog i læreplanen i matematikk. Dagens grunnskoleelever er den generasjonen som bruker digitale midler mest, men ikke mange av dem er vant med å programmere, skape, animere eller designe (Resnick et al., 2019). En vanlig definisjon av programmering er, å skape en rekke av datainstruksjoner ved bruk av et programmeringsspråk for å utføre en oppgave (Cheng, 2019). De store pådriverne for programmering i skolen er, blokkprogrammering som gjør programmering mer tilgjengelig på lavere trinn, interesse for

teknologi gjennom hele samfunnet og økt verdien av å kunne tenke algoritmisk (Benton et al., 2018; Bocconi et al., 2016; Cheng, 2019; Resnick et al., 2019; Shute et al., 2017). Sammenhengen mellom elevens kompetanse i programmering og matematikk er det lite forskning som tyder til som veldig positiv for elever i grunnskolealder (Laurent et al., 2022). Programmering vises derimot til å ha en sterk sammenheng mellom kompetanse i programmering og algoritmisk tenkning[5].

Siden innholdet i kompetansemålene om programmering er nytt fra forrige læreplan, K06, vil det være et behov for en begrepsavklaring. Både for oppgavens skyld og som det beskrives i kapittel 2.3.3, er det viktig å skape en felles forståelse av hvordan begrepet defineres, anvendes og kjennetegnes, slik at man ikke skaper en konflikt. Begrepene som skal defineres fra læreplanen er programmering, algoritmer, variabler, vilkår, løkker og algoritmisk tenkning (Tabell 1)(Kunnskapsdepartementet, 2020).

Ved innføringen av den nye læreplanen LK20 skal elevene fra 5. klasse, lære seg å programmere. Elevene lærer på småtrinnet grunnleggende matematikk som er viktig for å mestre programmeringen de skal lære. I de neste kapitlene blir det redegjort for kompetansen elever utvikler ved programmering, hvordan virkning programmering i undervisning kan ha, hvordan blokkprogrammering gjør det mulig å undervise på mellomtrinnet og hvilken sammenheng programmering har med matematikk.

Tabell 1: Begrepsavklaring

Begrep	**Betydning**
Programmering	Programmering er å «bryte ned komplekse problemer ned til mindre problemer og deretter gi en fullstendig og nøyaktig beskrivelse av fremgangsmåten for løsningen av problemet» (Statped, 2021). Programmering gjøres ofte ved hjelp av et digitalt middel, men dette er ikke et krav.
Koding	Koding er instruksjonen som gis til et program, enten ved tekst eller blokker. Koding er en del av programmering, men omfatter ikke alle tankene og refleksjonene et menneske gjør.
Variabler	En variabel er et element som kan inneholde en kjent eller ukjent verdi, ofte i form av tall eller tekst.

[5] Hentet fra prosjektskisse

Vilkår	Koden i programmet står overfor å måtte ta en beslutning hvilke linjer med kode som skal kjøres gjennom programmet. De vanligste vilkårene er hvis, hvis-ellers og ellers.
Løkker	Løkker er et verktøy i programmering som gjør at en linje med kode kan gjentas flere ganger.
Funksjoner	Funksjoner har ikke samme betydning som ellers i matematikk, hvor en funksjon beskriver relasjonen mellom to tall. Funksjoner i programmering er et stykke med kode som tydelig deler opp programmet i mindre deler. Disse funksjonene kan i likhet med vilkår kjøres flere ganger, men funksjonene viser til tydelige oppgaver som skal utføres.
Algoritmer	En algoritme er en oppskrift eller plan som steg for steg beskriver en kode som skal kjøres eller en handling som skal utføres. Algoritmer gir en konkret fremgangsmåte for løsningen på et problem.
Algoritmisk tenkning	Algoritmisk tenkning er en problemløsningsmetode som stammer fra informatikken og er en måte å tenke på som gjør det enklere å systematisk løse problemer. Ofte brukes denne problemløsningsmetoden ved digitale midler grunnet at den er basert på hvordan datamaskiner er bygget opp, men metoden har nylig blitt mere brukt innenfor flere fagfelt.

2.2.1 Programmering i skolen

Programmering i skolen er nytt både i Norge og internasjonalt, utenom bruk av Logo på slutten av 1900-tallet (Cheng, 2019). Logo var designet for å hjelpe barn med å utvikle deres problemløsningskompetanse ved å dekomponere problemer, planlegge og finne løsninger på problemer. Elever som lærer seg å programmere har i noen studier fått bedre problemløsningskompetanse og kreativitet, samtidig som det finnes forskning hvor denne effekten ikke har blitt vist (Cheng, 2019). Logo ble mindre brukt siden det var meget tidkrevende å lære seg alle syntaksene i programmet (Laurent et al. 2022). Blokkprogrammering er i dag en egnet måte å undervise, ettersom blokkene erstatter syntaksene som elevene tidligere måtte memorere, og dermed resulterte i mindre tid til læring.

Programmering kan være en svært sosial aktivitet og åpner for mange måter å samarbeide, men kildebruk kan være krevende. Det er en sentral del av kulturen innenfor programmering å publisere produserte tidligere løsninger på nett (Kaufmann & Stenseth, 2021). Ved at det finnes tilgjengelige løsninger på nett, kan elever enkelt finne noe de kan kopiere rett fra nettet.

Dette har i nyere tid blitt tydeligere ved tilgjengeligheten til kunstig intelligens, som enkelt kan omgjøre en elevs abstraksjoner i form av en bestilling til et fult fungerende program. Lærerrollen i programmering blir en annen enn vanlig matematikkundervisning. Læreren må kunne vise elevene hvordan de skal jobbe med å forstå og bearbeide tidligere løsninger og gjøre dem om til sine egne. Enkle muligheter for å kopiere i dataprogram kan være krevende for elever å motstå, og dersom elever tar snarveier når de jobber med programmering, er undervisningen mot sin hensikt. Lærerens rolle er i større grad preget av å gi elever gode fremgangsmåter og forståelse hvordan de selv kan innhente informasjon fra nettet og samarbeide med medelever (Kaufmann & Stenseth, 2021)(Resnick, 2009).

Økt bruk av programmering kan ses på som et verktøy og en kompetanse elevene kan får bruk for, men det er avhengig av flinke lærere. Programmering som et verktøy i undervisning forutsetter at lærerne har lyst å undervise emnet og selv ser læringens verdi (Rich, 2020). Det er funnet en nær sammenheng mellom lærerens oppfattede verdi, og hvor mye de strever for å lære det bort. Dette gjelder også lærere som er flinke undervisere i faget, hvis de ikke ser nytten av å lære det bort, bruker de mindre tid på det. Dette kan ses på som en utfordring med den nye læreplanen hvor lærerne har større frihet til hvordan de vil sørge for at elevene utvikler sin kompetanse i faget. Smylie (1998, referert i Rich, 2020) fant at den største indikatoren til hvor villig en lærer var til å endre sin undervisning, er deres holdninger og oppfattede verdi av emnet. Programmering har ikke tidligere vært en del av læreplanen i grunnskolen og det er viktig at lærerne tror at de kan lære bort og at de tror elevene deres kan lære. Dette viser til hvor viktig det kan være å gi lærere god etterutdanning og programmeringskompetanse, slik at elevene får undervisning med kvalitet. Lærerne som ikke ser nytten av å lære bort faget, overfører holdningen til sine elever, og er svært lite ønskelig (Rich, 2020). Eksempelvis har lærere i Sverige utrykt motivasjon for å undervise programmering, men med noen mangler i deres kompetanse (Misfeldt et al., 2019). Elever i Sverige har også beskrevet programmering som engasjerende og anvendelig (Kilhamn et al., 2021).

2.2.2 Blokkprogrammering

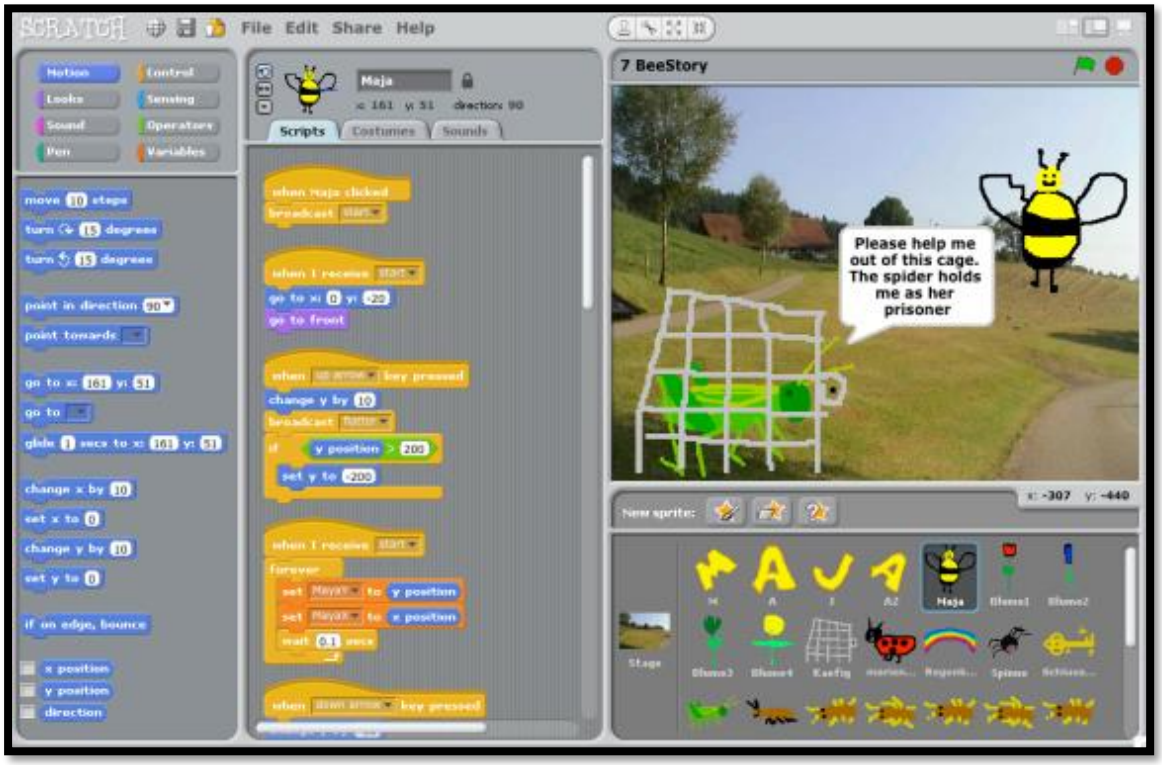
Blokkprogrammering er et visuelt programmeringsspråk (*visual programming language)* som egner seg til å bli brukt av barn. Programmering har tidligere ikke blitt undervist i skolen, både grunnet et manglende behov og vanskene med å undervise tekstprogrammering (Benton, 2017; Cheng, 2019; Laurent; 2022; Resnick, 2019). Blokkprogrammering har også en rekke

andre fordeler, sammenlignet med løsninger som har blitt brukt før i skolen. Blokkprogrammering gir stadig tilbakemeldinger til brukeren om hvordan deres endringer i koden, påvirker hvordan programmet funker (Laurent, 2022). Tilbakemeldingene gjelder både de synlige endringene elevene kan oppfatte selv og konkrete tilbakemeldinger fra programmet, som har vist økt evne til å tenke algoritmisk (Shute et al., 2017). En vanlig løsning som blant annet programmeringsverktøyet Scratch bruker, viser over halve skjermen de forskjellige blokkene og hvordan brukeren har satt sammen sin kode (*block palette & script),* mens den andre halvdelen viser grafiske muligheter og kjøres for å vise hva koden gjør (*sprite pane & stage*). Blokkprogrammering skal være så likt vanlig programmering som mulig, men det skal være en lav inngangsterskel for å begynne (Resnick, 2019). Når eleven begynner å programmere lærer de seg er rekke nye begreper som er viktig å forstå for å kunne programmere.

Resnick (2019) peker på at det lærere implisitt tenker om nåværende generasjon av elever som digitalt innfødte, det er en forventning at de mestrer bruk av digitale verktøy. Problemet er at barns bruk av digitale verktøy har et litt ensidig formål, ofte i form av underholdning. Elever har ikke nok erfaring og kompetanse ved å bruke digitale verktøy for å skape, animere eller programmere, «det er som om de kan lese, men ikke skrive» (Resnick, 2019). Scratch er en av mange blokkprogrammeringsprogram som legger til rette for at elever skal kunne utrykke seg med digitale verktøy. Scratch har også som hensikt å lære elever å tenke algoritmisk samtidig som de får øvelse å designe sine egne strategier og fremgangsmåter, men det er varierende forskning som viser til ulik grad økning av elevers algoritmiske tenkning eller matematikk ved bruk av Scratch (Laurent et al., 2022; Shute et al., 2017).

Variabler, vilkår, løkker og funksjoner er brukt i læreplanen (Kunnskapsdepartementet, 2020) og elevene blir fort kjent med disse begrepene, samt mange andre i Scratch. Variablene, vilkårene, løkkene og funksjonene har alle forskjellige farger og former i programmet og tydeliggjør for elevene hvilke virkninger bruken av de forskjellige blokkene kan gjøre i deres kode (Figur 1). Et av målene da Scratch ble laget var å gjøre programmering enklere å fikle med (*tinkerable* på engelsk). Blokkene i blokkprogrammering kan sammenlignes med legobrikker, som enkelt kan flyttes og byttes og gi en helt annen funksjon enn tidligere. Blokkprogrammering gir muligheten for å lære seg programmering, uten den tradisjonelt vanskelige tegnsettingen og syntaks i tekstprogrammering (Laurent, 2022). Elevene kan enkelt endre verdien til en variabel og få en illustrasjon av endringen i verdi, bruke forskjellige vilkår og finne den som egnes mest, prøve ut forskjellige løkker eller gjøre to

funksjoner om til én. Dette er eksempler på læringssituasjoner som ved bruk av tekstprogrammering kan være vanskelig og tidkrevende. Elever som har blitt undervist i blokkprogrammering viderefører sin programmeringskompetanse til når de begynner å jobbe med tekstprogrammering (Shute et al., 2017). Tekstprogrammering er også blitt en del av pensum i ungdomsskolen i matematikk, så det er bare en fordel for elevene å lære seg blokkprogrammering på mellomtrinnet (Kunnskapsdepartementet, 2020).



Figur 1: Tidlig versjon av kjent blokkprogrammeringsprogram Scratch

Programmering har tidligere vært undervist i grunnskolen, men hadde kanskje for stort fokus på individets utvikling av lite konkret programmering. Når Logo var i bruk appellerte det ikke til elever i ung alder slik man skulle ønske. Resnick (2019) mener programmet ofte ble brukt for å programmere utenfor elevers interesser og erfaringer. Scratch og andre blokkprogrammeringsprogrammer ble laget for å løse disse problemene. Skaperne av Scratch gjorde dette ved å skape et mangfold av muligheter for brukeren i form av ulike prosjekter som f.eks. bilde, video, spill, historiefortelling og animasjoner. Brukeren har også mulighet til å importere filer fra nett, noe som gjør deres program mere personlig. De mange mulighetene elevene har for problemløsning ved blokkprogrammering skaper det Resnick (2019) kaller, «vide vegger» og er en viktig ferdighet innenfor algoritmisk tenkning. «Vide vegger» gjør at elever har mange muligheter for å løse oppgaver, samtidig som de skal være åpne for andres

løsninger (Bocconi et al., 2016). Opplevelsen som elevene får av å programmere i Scratch består av kontekster som er viktig for dem og gjør programmering nesten til en lek, samtidig som de lærer programmering, design og algoritmisk tenkning.

Muligheten for å samarbeide på felles prosjekt har også gjort Scratch populært. Samarbeid er en av ferdighetene som er sentral for den algoritmiske tenkeren. Elever kan løse deloppgaver hver for seg, og når de er ferdig har de mulighet til å dele deres personlige løsninger med sine medelever. Elever som samarbeider utvikler også sin algoritmiske tenkning i større grad enn elever som jobber individuelt (Shute et al., 2017). Programmering har utviklet seg til en sosial aktivitet som åpner for egne løsninger i kontekster som er viktig for elevene.

En vanlig tanke blant lærere og elever er at programmering og algoritmisk tenkning er vanskelig å lære, sammenlignet med andre deler av pensum i skolen (Cabrera, 2019). At både algoritmisk tenkning og programmering kan være vanskelig, er viktig å anerkjenne. En av de viktige ferdighetene som kreves av den algoritmiske tenkeren er å kunne jobbe med komplekse oppgaver (Bocconi et al., 2016). Som tidligere nevnt er blokkprogrammering godt egnet for å gi elever en lav inngangsterskel for å starte med programmering, og det er viktig å finne gode programmer som egner seg til sitt formål. For at elevene skal kunne lære seg programmering kreves lærere med tro på deres egen mestring til å undervise programmering. Disse lærerne egner å gi gode oppgaver, veiledning og tilbakemeldinger til elevene, både for elever som sliter med programmering og har behov for en lav inngangsterskel, samt de elevene som drar nytte av den høye takhøyden programmering byr på (Resnick, 2019). Det er et vanskelig steg å ta for de lærerne som skal undervise programmering eller blokkprogrammering for første gang, og det krever etterutdanning og kursing for å få til (Cabrera, 2019). Det er et behov for lærere som kan lære elever programmering i kontekster som er viktig for dem, samtidig som de blir utfordret til å videreutvikle sine eksisterende løsninger og problemløsningsevner.

2.2.3 Programmering og matematikk

Ved innføringen av programmering som kompetansemål i matematikk og naturfag på mellomtrinnet vil det være viktig å utforske hvordan lærerne oppfatter sammenhengen mellom kunnskapene elevene danner, og om det faktisk finnes koblinger mellom fagfeltene (Kunnskapsdepartementet, 2020). Ikke bare er det viktig å se på hva lærerne synes, men også den målte effekten av elevenes kunnskaper. Vil elever bli bedre i matematikk når de jobber med programmering og vil de bli bedre i programmering når de jobber med matematikk?

I Sverige er det gjennomført forskning på lærernes oppfatning om virkningen av programmering og dets sammenheng med matematikk. Kilhamn et al (2021) gjennomførte en kvalitativ studie som forsket på lærernes meninger om inkluderingen av programmering i matematikkfaget. Funnene i studien viser til forskjellige syn på virkningen av å inkludere programmering i matematikkfaget og hvilken sammenheng som kan finnes mellom de programmering og matematikk. Funnene deles i fire kategorier, hvor to viser til hvordan programmering kan fungere som et pedagogisk virkemiddel og to om sammenhengen med matematikk.

Kategori 1 beskriver programmering som et nyttig verktøy i undervisningen. Programmering beskrives av noen lærere som et virkemiddel i deres undervisningspraksis i likhet med f.eks. matematikkprogrammet GeoGebra, og elevene vil etter hvert være rustet til å bruke programmering for å løse problemer når de får tilstrekkelig med kompetanse. Kategori 2 viser til lærernes inntrykk av at elever opplever programmering som engasjerende og tilkoblet virkeligheten. Lærerne tror elever er mer motivert for å lære matematikk ved programmering siden det er en form for matematikk de aldri har jobbet med før, samtidig som det er tydelig for elevene at dette er noe de kan få som jobb i fremtiden (Kilhamn et al., 2021).

Kategori 3 handler om algoritmisk tenkning og hvordan elevene utvikler sin algoritmiske tenkning ved bruk av programmering. Innføringen av programmering er nytt og engasjerende for elevene, men programmering krever også endringer i hvordan elevene tenker problemløsning i matematikk. De står overfor uvante problemer som må løses, blant annet nevner de svenske lærerne dekomponering og algoritmebehandling som kunnskaper elevene ikke har hatt behov for å lære tidligere, samt en rekke ferdigheter innen algoritmisk tenkning (Bocconi et al., 2016). Å holde ut og *debugging* nevnes som viktige ferdigheter, siden programmering skaper nye utfordringer trenger elevene å lære seg nye ferdigheter. De svenske lærerne aksepterer at elevene deres vil prøve og feile mange ganger, for at de skal kunne gjøre de små endringene i programmet sitt som er nødvendig (Kilhamn et al., 2021).

Kategori 4 er litt likt kategori 1, hvor programmering ses på som et verktøy i undervisningen. I denne kategorien brukes programmering som et verktøy for å tydeliggjøre matematikken elevene lærer mens de programmerer. Programmering kan gi visuelle modeller av koden, dette kan vise hva som skjer når man endrer en variabel i programmet sitt (Benton, 2017). Det finnes en rekke programmer som tydeliggjør viktige matematiske konsepter mer effektivt enn hva en lærer kan forklare, vise på tavla eller illustreres i en bok.

Én av lærerne i studien nevner at programmering i seg selv er matematikk. Ved å finne mønstre og løse matematiske puslespill er elevene når elevene kjernen at matematikken (Kilhamn et al., 2021). Det kan være bekymrende at bare én av tjue lærere tenkte på programmering som matematikk i seg selv og det var få lærere som klarte å se nytten av programmering i en matematisk kontekst. Kilhamn et al (2021) viser til hvordan programmering tidligere har blitt brukt for grafiske illustrasjoner, som medførte at lærerne så nytten av programmering i mange andre fag enn matematikk. Kaufmann & Stenseth (2021) konkluderte også i sin studie at det er en mulighet at overføringen fra programmering til matematikk er sterk, slik annen forskning viser til, men lærerne ikke evner å lære dette bort på en effektiv måte.

Lærernes syn på koblingen mellom matematikk og programmering er også forsket på kvantitativt i Sverige, basert på endringene i svensk læreplan (Misfeldt et al., 2019). Lærerne svarte frivillig på et spørreskjema koblet opp mot et nasjonalt seminar om programmering for svenske lærere. Spørsmålene som ble stilt omhandlet både lærernes og elevenes utbytte av forskjellige kompetanser av programmering og matematikk. Fordelingen av svarene bikket mere mot «enig» ved alle spørsmål og det er ikke gjort noen analyser av dataen. De fire faglige kompetansene i spørreskjemaet var «overføringen fra matematisk kompetanse til programmering», «man kan gjøre mer matematikk med programmering», «bedre forståelse av matematiske konsept ved programmering» og «bedre forståelse av prosedyrer og algoritmer med programmering» (Misfeldt et al., 2019). Disse kompetansene relaterer til kategori 4 fra Kilhamn et al. (2021) sin studie ved at programmering ses på som et verktøy for å jobbe med matematikk. Programmering er dermed et verktøy for å tydeliggjøre andre fagområder og konsepter som tidligere har vært en del av læreplanen.

Den faktiske overføringen mellom matematikk og programmering viser til mange forskjellige resultater og konklusjoner avhengig av hvilke kompetanser som er målt, med en rekke begrensninger for mye av forskningen (Laurent et al., 2022). Det er lite forskning som tyder på at programmering er en effektiv måte å lære matematikk som kunne blitt lært på andre måter. Selv om relasjonen mellom matematikk og programmering argumenteres for, kan de kognitive ferdighetene være krevende for barn i skolealder. Duval (2000, referert i Laurent et al., 2022) nevner at «difficulty of recognition hinders conversion and is at the root of learning difficulties in mathematics». Dersom programmering skal implementeres i matematikk er det i størst grad egnet som et verktøy for å utvikle algoritmisk tenkning, programmeringsferdigheter eller som variasjon i undervisning. Som variasjon i

undervisningen er det sentralt at lærerne bruker tid på å konkret forklare generaliseringene, elevene trenger for å forstå matematikken når de programmerer (Laurent et al., 2022).

2.3 Algoritmisk tenkning

Algoritmisk tenking er en del av kjerneelementet «Utforskning og problemløsning» og defineres som «å vurdere hvilke steg som skal til for å løse et problem, og å kunne bruke sin teknologiske kompetanse for å få en datamaskin til å løse (deler av) problemet. I dette ligger også en forståelse av hva slags problemer/oppgaver som kan løses med teknologi og hva som bør overlates til mennesker» (Kunnskapsdepartementet, 2020; Utdanningsdirektoratet, 2019) [6]. En finner også en rekke lignende synonymer i forskningen som algoritmisk tenking, algoritmisk tankegang og algoritmisk resonnering, hvor de tre begrepene er mere spesifikke (Gjøvik, 2016). Algoritmisk tenkning er dermed det begrepet som passer best inn i dagens skole som i større grad er rettet mot å forstå faginnholdet, enn å kunne gjøre oppgaver steg for steg uten å vite hvorfor (Utdanningsdirektoratet, 2020).

Algoritmisk tenkning (*computational thinking/CT*) er en problemløsningsmetode som har blitt svært aktuell siden det ble definert av Wing i 2006 slik «Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science» (Wing, 2006). Begrepet algoritmisk tenkning oversettes til computational thinking, algoritmisk tankegang eller algoritmebehandling oversettes til algorithmic thinking, som tidligere nevnt er mer spesifikk bruk av en bestemt algoritme. Algoritmebehandling er et av underpunktene som kjennetegner algoritmisk tenkning (Gjøvik, 2019). Forståelsen av begrepet har endret seg mye i engelsk litteratur samtidig som oversettelsen av begrepet kan virke forvirrende, og øker dermed behovet for en tydelig introduksjon av begrepet i norsk skole.

Algoritmisk tenkning er en problemløsningsmetode som tar utgangspunkt i måten informatikere løser problemer. Algoritmisk tenkning er derimot ikke begrenset i form av et behov for digitale midler, men er en fremgangsmåte på lik linje en rekke andre problemløsningsmetoder (Bocconi et al., 2016; Wing, 2011) [7]. Utviklingen av begrepet har

[6] Hentet fra prosjektskisse

[7] Deler hentet fra prosjektskisse

medført at det ikke lenger kjennetegnes som en metode eksklusivt for informatikere, men et tankesett som kan anvendes i mange situasjoner. Å lære seg å kunne tenke algoritmisk regnes som en grunnleggende ferdighet for det 21. århundre, som gir muligheten for å utrykke og forstå den digitale verden vi lever i ved å analyse problemer og finne løsninger (Bocconi et al., 2016). Algoritmisk tenkning er dermed et begrep som er svært generelt og kan være til nytte utenfor fagfeltet det stammer fra. Felles for tolkningene av begrepet finnes det underpunkter som spesifiserer nøkkelbegrep og arbeidsmåter (Figur 2)(Utdanningsdirektoratet, 2019).

Figur 2: Den algoritmiske tenkeren laget av Udir (Utdanningsdirektoratet, 2019)

I plakaten som brukes av Utdanningsdirektoratet (Figur 2) er det listet en rekke nøkkelbegrep og arbeidsmåter. Plakaten er designet og tilpasset en plakat fra Barefoot Computing fra Storbritannia som lager undervisningsopplegg og veiledningsmateriale for lærere om algoritmisk tenkning. Nøkkelbegrep består av logikk, algoritmer, dekomposisjon, mønstre, abstraksjon og evaluering. Arbeidsmåter består av å fikle, skape, feilsøke, holde ut og samarbeide. Tross mange forskjellige definisjoner og tolkninger av begrepet algoritmisk tenkning er små variasjoner av disse punktene mye brukt i internasjonal forskning (Bocconi et al., 2016) som tydeliggjøres i de neste to kapitlene.

2.3.1 Nøkkelbegrep og kunnskaper

Abstraksjon (abstraction) kan beskrives som det mest sentrale elementet ved å kunne tenke algoritmisk. Wing (2006) som definerte begrepet Computational Thinking definerer abstraksjon som det viktigste nøkkelbegrepet (Wing, 2011). Abstraksjon er å gjøre et problem forståelig og løselig ved å synliggjøre hva som er viktig, og se bort fra unødvendige detaljer (Bocconi et al., 2016; Gjøvik, 2016). Samtidig som det er det viktigste nøkkelbegrepet er det også mest krevende (Wing, 2011). Å skape en algoritme som er effektiv krever abstraksjon av problemer, eksempelvis er alle digitale midler basert på mange abstraksjoner i form av operativsystemer, filsystemer, nettverk og så videre. Ettersom noen andre tidligere har gjort denne abstraksjonen av komplekse system, trenger ikke brukeren å ta hensyn til disse. Slike gode abstraksjoner fører til løsninger som er svært effektive, mens dårlige abstraksjoner gir tidkrevende og unødvendig avanserte løsninger (Wing, 2011)

Algoritmebehandling (algorithmic thinking) er å følge og forklare trinnvise instruksjoner. Som tidligere nevnt må ikke det engelske begrepet direkte oversettes til norsk. Sammenlignet med algoritmisk tenkning er algoritmebehandling mer snevert og spesifikt, og det er en vanlig misoppfatning at å følge en algoritme er det sentrale i algoritmisk tenkning (kilde). Algoritmen som dannes behøver heller ikke være en digital konkret løsning, men er også ment som instruksjoner som kan følges av mennesker (Shute et al., 2017). I læreplanen skal elever lære seg å tenke algoritmisk i matematikk når de starter på skolen ved å «lage og følge regler og trinnvise instruksjoner» (Kunnskapsdepartementet, 2020).

Dekomposisjon (decomposition) er å bryte opp et problem inn i mindre deler (Gjøvik, 2019; Utdanningsdirektoratet, 2019). Hensikten er å dele opp et stort problem opp i mindre deler som kan løses hver for seg, for så og settes sammen til slutt. En slik egenskap er svært nyttig når man skal løse et problem som umiddelbart ikke ligner på noe man har løst før. De mindre problemene vil bli enklere å løse fordi de ofte består av like problem man har løst før (Bocconi et al., 2016).

Generalisering (generalization) som Utdanningsdirektoratet (2019) har valgt å kalle mønstre vil si å gjenkjenne sammenhenger i, og finne løsninger på tidligere problemer (Gjøvik, 2019). Generalisering er dermed nært knyttet mot dekomposisjon, siden problemer som er brutt opp i mindre deler kan ha en løsning brukt tidligere. Man kan bruke sine forkunnskaper ved å spørre seg selv hvordan problemer ligner noe man har løst før, og hva er det som er forskjellig (Bocconi et al., 2016).

Abstraksjon, algoritmebehandling, dekomposisjon og generalisering er de fire tydeligste nøkkelbegrepene, dersom man vektlegger norsk og internasjonal forskning likt (Bocconi et al, 2016; Shute et al. 2017; Utdanningsdirektoratet, 2019). Enkelte nøkkelbegreper er nevnt under hverandre i ulike modeller, samt noen nøkkelbegreper er beskrevet som arbeidsmåter, som diskuteres videre i kapittel 2.3.2. Jeg har valgt å sette evaluering og automatisering som egne nøkkelbegreper og vil redegjøre for hvordan andre modeller inkluderer andre nøkkelbegreper.

Evaluering (evaluation) vil si å stadig gjøre vurderinger av løsninger (Utdanningsdirektoratet, 2019). Dette innebærer å gjøre vurderinger av sin egen og andres abstraksjoner, algoritmer og løsninger. Shute et al. (2017) og Bocconi et al. (2016) beskriver *debugging* som og systematisk analyse og evaluere ved å teste, følge og bruke logisk tenkning for å forutse utfall, hvor *debugging* er én av seks nøkkelbegrep i deres modeller. Shute et al. (2017) har *iteration* som et eget nøkkelbegrep, det vil si å gjentatte ganger gjøre evalueringer av løsninger. Grover & Pea (2013, referert i Bocconi et al., 2016) har langt flere nøkkelbegrep og *efficiency and performance constraints* beskrives i likhet med både evaluering og *debugging*, men er mindre konkret. Utdanningsdirektoratet (2019) har logikk som sitt første nøkkelbegrep ved å analysere og forutse. Siden analysering og å forutse er nevnt under annen forskning sin definisjon av evaluering og *debugging* valgte jeg å inkludere logikk, innenfor evaluering.

Automatisering (automation) er «å kunne implementere løsningen av problemer i programmeringsspråk eller å gjøre det menneskelige bidraget minimalt» (Gjøvik, 2019). En slik definisjon åpner for at algoritmisk tenkning både kan utføres av et digitalt middel som leser programmeringsspråk, eller et menneske som kan følge en algoritme. Shute et al. (2017) har i sin modell for algoritmisk tenkning plassert automatisering under algoritmebehandling, som én av fire underpunkter, ettersom automatiseringen er avhengig av en algoritme.

2.3.2 Arbeidsmåter og ferdigheter

I en del modeller av algoritmisk tenkning vektlegges ikke bare nøkkelbegrepene og kunnskapene elevene kan lære seg, men også viktige former for arbeidsmåter og ferdigheter. Utdanningsdirektoratet (2019) har valgt å kalle dette arbeidsmåter, mens internasjonal forskning bruker begreper som kan oversettes til holdninger, ferdigheter og egenskaper (Bocconi et al., 2016). Dersom en definerer summen av abstraksjon, algoritmebehandling, dekomposisjon, generalisering, evaluering og automatisering som kunnskaper, i en modell hvor det vektlegges ferdigheter, vil algoritmisk tenkning være en form for kompetanse (NOU

2018: 2). Algoritmisk tenkning er i læreplanen en problemløsningsmetode, en del av et kjerneelement og ikke et eget kompetansemål, og gir potensielt et feil bilde av begrepet.

Utdanningsdirektoratets modell for den algoritmiske tenkeren består av fem punkter (Figur 2). Tre av disse går ofte igjen i andre forskningsartikler, å holde ut, feilsøke og samarbeid. De tre felles punktene er viktig i mange forskjellige former for undervisning, men vektlegges spesielt ved algoritmisk tenkning (Bocconi et al., 2016; Fagerlund et al., 2019; Shute et al., 2016; Utdanningsdirektoratet, 2019). Utdanningsdirektoratets modell inneholder to begreper som er mindre vektlagt i tidligere forskning, å fikle og skape. To punkter som ikke er med i modellen, men som ofte nevnes i teoretiske rammeverk, er å jobbe med kompleksitet og å være åpen for flere løsninger (Bocconi et al., 2016; Shute et al., 2016).

Å kunne holde ut vektlegges i mange modeller, elever vil naturligvis gjøre flere feil når de jobber med problemløsningsmetoder, enn ved oppgaver som har en mer lineær progresjon. Elever må stadig gå frem og tilbake og gjøre endringer som kan være demotiverende, men de som holder ut vil lære seg og løse komplekse problemer. Å kunne feilsøke, eller på engelsk *debugging* er i noen modeller en kunnskap og i andre modeller en ferdighet. Utdanningsdirektoratet (2019) har valgt å definere feilsøking som en ferdighet, som vil gjenspeile en passende modell for grunnskolen, siden det er noe en gjør som en arbeidsmåte. Om en skulle en lage en modell tilpasset en informatiker, hvor begrepet algoritmisk tenkning stammer fra, ville det vært mer naturlig å plassere feilsøking (*debugging)* som noe som krever kunnskap og ikke bare en handling.

Den siste av de tre felles ferdighetene er samarbeid og kommunikasjon. En vanlig arbeidsmåte for programmerere er å fordele ulike oppgaver som skal løses, forså å slå dem sammen til et felles produkt. En viktig kunnskap er å kunne gjøre et stort problem opp i mindre deler, dekomponering, og krever god kommunikasjon mellom elevene og tydelig arbeidsfordeling.

To av punktene som nevnes i Utdanningsdirektoratets (Figur 2) modell er å kunne fikle med konkreter og programmer (*tinkering*) og å skape noe. Disse punktene brukes også i modellen til Woolard (2016 referert i, Bocconi et al., 2016). Inkluderingen av å skape noe kan tydeliggjøre tilpasningen av modellen til den norske grunnskolen for å vektlegge skaperglede som er et av opplæringens verdigrunnlag (Kunnskapsdepartementet, 2017). *Tinkering* beskriver et begrep som ikke har en god direkte oversettelse til norsk, men defineres som å

gjøre små endringer for å forbedre eller fikse noe. Blokkprogrammering er et godt eksempel på behovet for algoritmisk tenkning og muligheten med å fikle med de forskjellige blokkene, ved å enkelt utforske funksjonen til de forskjellige blokkene og gjøre løsningen på et problem bedre (Resnick, 2009).

To sentrale punkter som ikke brukes i Utdanningsdirektoratets (Figur 2) modell er å kunne arbeide med komplekse oppgaver og å være åpen for flere løsninger på et problem. Problemløsningsoppgaver er ofte komplekse oppgaver, og slike oppgaver har et stort behov for god applikasjon av abstraksjoner (Bocconi et al., 2016). Abstraksjon er den viktigste av kunnskapene og læres gjennom å løse komplekse oppgaver. Selv om modellen er egnet for grunnskolen kan det være viktig at elevene i tillegg har en forståelse av at å kunne tenke algoritmisk er egnet for å løse store komplekse oppgaver (Wing, 2006). Når elevene løser disse store komplekse oppgavene finner de ofte forskjellige løsninger på problemer, samt har forskjellige fremgangsmetoder. Elever må være åpne for at det finnes flere unike løsninger på et problem og lære av hverandre. I stedet for at elver bare regner for seg selv og alle kommer frem til samme svar på samme vis, kan matematikkundervisningen bli mere sosial (Resnick, 2009).

2.3.3 Algoritmisk tenking i undervisning

Algoritmisk tenkning består av både kunnskaper og ferdigheter, og det er viktig at lærere har en forståelse av hvorfor det er viktig å utvikle denne kompetansen hos elever. Nytten av å være en algoritmisk tenker vil være mest tydelig innenfor fagfeltet begrepet stammer fra, informatikken (Wing, 2006). Algoritmisk tenkning skiller seg fra programmering ved at det er en problemløsningsmetode som har overføringsverdi til andre fagfelt (Shute et al., 2017). Samtidig som programmering ikke har samme overføringsverdi til andre fagfelt, sammenlignet med algoritmisk tenkning, fikk studenter som gjennomførte programmeringskurs bedre kunnskaper som kjennetegner algoritmisk tenkning (Shute, 1991, referert i Shute et al. 2017). Kunnskapene som kreves for å være god i programmering, kjennetegnes av blant annet nøkkelbegrepene i modellene for algoritmisk tenkning (Bocconi et al., 2016; Shute et al., 2017; Utdanningsdirektoratet, 2019). Det er nødvendig å lage abstraksjoner av problemer forså å dekomponere disse. Algoritmebehandling gjør det mulig å automatisere løsningen, som så må evalueres og kan til slutt bli generalisert. Dette gjør programmering til en aktivitet for å lære seg å tenke algoritmisk (Bocconi et al., 2016). Algoritmisk tenkning innebærer kunnskaper som ikke er nødvendigvis bundet til å

programmere, og det er blitt mer vanlig å tenke algoritmisk uten bruk av digitale midler, ettersom begrepet har blitt relevant innenfor andre fagfelt (Shute et al., 2017).

Å kunne tenke algoritmisk er også nyttig for elevers digitale ferdigheter og livsmestring ved at det utvikler forståelse av teknologi (Kunnskapsdepartementet, 2017, 2020). I Norge er digital ferdighet et vanlig begrep for å beskrive én av fem grunnleggende ferdigheter, mens andre språk og land bruker ord som *literacy*, på norsk kyndighet, evnen til å kunne noe (Bocconi et al., 2016). Wing (2011) mener algoritmisk tenkning er 21. århundrets viktigste form for *literacy* og at *techonological literacy* er én av tolv *21st century skills* som viser til hvilke ferdigheter elever bør ha før de skal ut i arbeidslivet. Bocconi et al. (2016) poengterer at algoritmisk tenkning er det konseptet som danner en forståelse av hva som skjer i den digitale verden, som også er at av ferdighetsområdene under digitale ferdigheter (Kunnskapsdepartementet, 2017). De digitale ferdighetene skiller seg fra algoritmisk tenkning ved at det vektlegges å ta i bruk ulike digitale midler. Voogt, en av forskerne som ble intervjuet i studien til Bocconi et al. (2016) poengterer at dersom alle elever egnet å tenke algoritmisk ville elever ikke bare være brukere av disse digitale midlene, men også ha en god forståelse for hvordan de funker. Det er viktig at elever lærer seg å tenke algoritmisk, slik at de ikke vegrer seg for å utvikle sine digitale ferdigheter.

Yadav et al. (2014) gjorde et komparativt studie hvor de sammenlignet lærerstudenter som hadde fått og ikke fått, en innføring om algoritmisk tenkning og hvordan det kan brukes i undervisning. Gruppen som hadde fått innføringen viste endringer i hvordan de oppfattet begrepet, algoritmisk tenkning, sammenlignet med kontrollgruppen. Gruppen med innføringen gikk fra å se på algoritmisk tenkning noe som var i stor grad koblet til bruk av digitale midler for å løse oppgaver, til å se på det som en problemløsningsmetode ved bruk av algoritmer og kritisk tenkning. I likhet med Wing (2011) var ikke algoritmisk tenkning lenger begrenset til den digitale verden, men hadde også nytte i hvordan man strukturer tankene sine og løser problemer man møter i dagliglivet. McGinnis et al. (2020) gjennomførte en studie hvor de introduserte algoritmisk tenkning til naturfagslærerstudenter, med hensikt å måle om algoritmisk tenkning gjenspeilet deres egne tanker om undervisning. Studentene som oppfattet algoritmisk tenkning som «et tillegg» eller «nytt, men kompatibelt» var mest villig til å gjøre en endring i deres undervisning, og viste også best forståelse av begrepet. Felles for studiene gjennomført av Yadav et al. (2014) og McGinnis (2020) er viktigheten av tydelig redegjøring rundt hva algoritmisk tenkning innebærer og hvordan det kan implementeres i undervisning.

Det er gjort forskning på hvordan lærere i klasserommet oppfatter begrepet algoritmisk tenkning, som kan vise til forskjellige tolkninger av begrepet (Cabrera, 2019). En hindring for effektiv undervisning av algoritmisk tenkning er at begrepet har endret sin betydning i løpet av kort tid, som medfører til at det finnes mange forskjellige tolkninger av begrepet. Lærere som ønsker å implementere algoritmisk tenkning i sin undervisning kan møte på undervisningsopplegg eller etterutdanning som legger ulik verdi i begrepet (Cabrera, 2019). I likhet med studentene i Yadav et al. (2014) sin studie, fant Cabrera (2019) at forskere hadde like tolkninger av algoritmisk tenkning. En rekke artikler beskrev algoritmisk tenkning som en måte å eksklusivt bruke digitale midler, programmering, problemløsning eller å tenke som en datamaskin. Med disse manglende tolkningene av begrepet var det også en rekke artikler som viste til hindringer i å inkludere algoritmisk tenkning på grunnskolenivå (K-12). Ved tolkning av algoritmisk tenkning som noe som ikke hører hjemme i grunnskolen, var begrepet også beskrevet som vanskelig å lære, og bare egnet for de som var spesielt interessert. Lærere i studien var også bekymret for mangel i sin egen kompetanse til å undervise algoritmisk tenking, en konsekvens av at begrepet fremstilles som vanskeligere enn annet pensum i skolen (Carbera, 2019).

2.4 Lærernes holdninger til programmering

I de tidligere kapitlene har det blitt redegjort for hvordan endringene i læreplanen har medført et helt nytt emne som skal undervises i skolen. Programmering og algoritmisk tenkning er nytt i skolen og det kan være krevende for lærere å sette seg inn i nye undervisningsmetoder og fagstoff, samt hvordan de skal lære dette bort. I Sverige og Finland har programmering og algoritmisk tenkning vært innført som en del av deres læreplan. Forskning gjennomført i disse landene viser til flere elementer lærerne synes er vanskelig i forbindelse med innføringen av programmering i matematikk.

Forskningen gjennomført av Misfeldt et al. (2019) i Sverige beskrevet i kapittel 2.2.3, viser til lærere som er svært motiverte for å undervise programmering i matematikk, men som ikke føler seg forberedt. 55 av 130 (42%) svarte at de var entusiastiske og 23 av 130 (18%) var veldig entusiastisk, noe som er et godt utgangspunkt. Samtidig var det dessverre bare 43 av 131 (33%) som følte seg forberedt og 6 av 131 (0.5%) som følte seg veldig forberedt for å undervise. Det er viktig å merke at dette var det lærerne svarte på undersøkelsen etter de hadde deltatt på et seminar, så svarene er potensielt langt mer positive, enn hva de kunne ha vært før seminaret eller lenge etter. Lærerne ser også en sammenheng mellom matematikk og

programmering, og dette kan være årsaken til at lærerne var entusiastiske til å undervise programmering. På den andre siden var det også lærere som ikke så sammenhengen mellom programmering og matematikk, og var negative til innføringen. Lærere som har høy mestringstro på sin egen undervisning fører til elever som lærer mer, lærere som jobber lengre og blir mindre utbrent (Rich, 2020). Det er dermed viktig at lærere som er motivert til å tilegne seg den kompetansen de behøver. Denne kompetansen kan også gi motivasjon til de lærerne som svarer at de ikke er entusiastiske for å undervise programmering. Lemon & Garvis (2016) sine funn viser til at lærerstudenter i Australia heller ikke har nødvendig mestringstro rundt å undervise med teknologi i skolen. Dermed kan det være et behov for dagens og framtidens lærere å tilegne seg kompetanse for å kunne undervise programmering.

Misfeldt et al. (2019) poengterer at det kan være nødvendig å reflektere rundt hvordan programmering er implementert i matematikkfagplanene. Programmering vil kunne innføres på alle trinn, og dermed må alle matematikklærere kunne undervise programmering. Dersom alle matematikklærere har kompetansen som kreves for å undervise, vil det være færre lærere som har vanskeligheter med å koble programmering og matematikk opp mot hverandre. Dermed vil man unngå at lærere ser på programmering som kun et verktøy i undervisningen, men heller som en viktig del av matematikk som det er ment å være.

I Finland er det gjennomført lignende forskning av Korhonen et al. (2022) med en mer robust metode med 943 svar på et spørreskjema som ble sendt ut. Spørreskjemaene som ble sendt ut til lærerne inneholdt åpne spørsmål som ble kodet av forfatterne, som ga muligheten for en kvantitativ og kvalitativ analyse. Forskningen var basert på svar fra lærere som også deltok på et dagsseminar i etterkant av at programmering var innført som en del av den finske læreplanen i matematikk. Funnene fra forskningen viser til hovedtrekk ved innovasjon i skolen, interne og eksterne faktorer ved implementering av programmering i skolen, læreres holdninger og følelser i henhold til implementeringen av programmering i matematikk (Korhonen et al., 2022).

Rogers (2003, referert i Korhonen et al., 2022) sin teori forklarer fem hovedtrekk som avgjør adoptering av nye innovasjoner i skolen, som er *advantage, compatibility, complexity, trialability* og *observability.* Av disse fem ble de tre første nevnt i de finske lærernes tanker om adoptering av programmering i deres nye læreplan. Nesten halvparten av de finske lærerne nevner kompetanse og motivasjon i skolen som en fordel for lærerne og elevene. Programmering ble sett på som en viktig kompetanse elevene bør utvikle, ettersom det er

ettertraktet i arbeidslivet og utvikler elevenes logiske tenkning. Motivasjonen for å adoptere programmering ble begrunnet i lærernes muligheter for å undervise på nye måter og muligheter for å motivere elever som tidligere ikke har «passet inn» i skolesystemet, her var gutter spesielt nevnt (Korhonen et al., 2022). Lærerne var ivrig etter å utvikle sin egen kompetanse, men var samtidig bekymret for en potensiell mangel i deres kollegers kompetanse. Denne bekymringen var også påpekt av de lærerne i nabolandet deres (Misfeldt et al., 2019).

Ettersom programmering er nytt i læreplanen i Finland, er det ingen overraskelse at de finske lærerne synes adopteringen av programmering oppfattes som kompleks. I likhet med den norske skolemodellen har de finske lærerne stor frihet i hvordan de ønsker å legge opp undervisningen, dette medfører dermed noen vansker når innovasjoner skjer i skolen med utydelige rammer. Noen av de finske lærerne var bekymret over mangler på klarhet om hvordan de skal strukturere programmering i undervisningen. I hvilket eller hvilke fag programmering skal undervises, hvor mye skal programmering vektlegges innad fagene, hvilke læremidler bør brukes, var usikkerhetsmomenter hos de finske lærerne. Andre lærere så på dette fra en positiv vinkling, da de hadde stor frihet og kunne jobbe med det grunnleggende først. Bare 6.3% av de finske lærerne beskrev programmering som å ikke være nytt i deres undervisningspraksis. I øyne av McGinnis et al. (2020) bruk av teori om implementering av ny undervisningspraksis, kan en tenke seg at de finske lærerne var innstilt for å gjøre den nødvendige endringen av sin undervisningspraksis. Majoriteten av de finske lærerne var positivt innstilt til programmering i skolen, men et fåtall mente nytten ikke var så stor. Både egne verdier og refleksjoner over verdien av å tilegne seg kompetanse var lærerne skeptisk til, både for seg selv og elevene. I studien var det 943 av rundt 1500 som svarte på spørreskjemaet, og man kan ikke utelukke at andelen som ikke svarte på spørreskjemaet er mindre positivt innstilt til programmering i skolen (Korhonen et al., 2022).

Lærernes tanker rundt adoptering av programmering virker lovende for finsk skole, men det finnes en rekke faktorer som kan bremse opp adopteringsprosessen (Fullan, 2015 referert i Korhonen et al., 2022). Disse faktorene kan deles i intern og eksterne faktorer ved en skole, og det største bekymringene kommer ved den interne strukturen av skolene. Lærerne følte på en mangel i deres egen kompetanse, og 14.6% fryktet at det var lærere som ville nekte å undervise programmering. Dette ville også medføre at ansvaret falt på et fåtall av lærere å undervise programmering til elevene, og dette kunne medføre store forskjeller i hvilket skoletilbud elevene fikk. Det var også noen bekymringer som falt på skolens rutiner på

profesjonsutvikling og materiell som var tilgjengelig for lærerne. Svenske lærere velger å ikke ta i bruk lærebøker eller konstruere oppgaver, men er medlemmer av grupper på sosiale media hvor andre lærere deler undervisningsmateriell (Bråting et al., 2021). Det kan være bekymringsverdig at lærere heller tar i bruk sosiale media for undervisningsmateriell, heller enn å benytte det de får tilbydd gjennom skolen, både med tanke på variasjoner og kvaliteten i undervisningen. Pörn et al. (2021) har også funnet at finske lærere mener de ikke har godt nok undervisningsmateriell, men dette kan stamme fra en manglende kunnskap om koblingen mellom matematikk og programmering, som gjør at lærere ikke egner vurdere hva som er gode og dårlige oppgaver for deres elever.

De eksterne faktorene var mindre nevnt, men mere positive (Korhonen et al., 2022). Lærerne var positive til nytten av programmering og hvordan den var strukturert i læreplanene. Digitaliseringen av samfunnet, økt behov i arbeidsmarkedet og kompetanse for det 21. århundre trekkes fram som viktige eksterne behov for programmering i skolen.

De finske lærerne så dermed noen faktorer som kunne være en brems for adoptering av programmering i skolen, men de var stort sett positive. 27.7% av lærerne var delvis positive, 38.9% var positive og 10.0% endret sitt syn fra negativ til mer positiv etter de deltok dagsseminaret. Bare 4.2 % var negativ og 9.3% var delvis negativ (Korhonen et al., 2022). Kvantitativ analyse av dataen viste at kvinnene i større grad var de som endret sitt syn fra negativ til positiv etter dagsseminaret. Det var ingen signifikant forskjell i lærernes holdninger til programmering, avhengig av deres erfaring med programmering eller hvilken del av landet de underviste i. Korhonen et al. (2022) sammenlignet hvilke følelser de koblet mot innføringen av programmering, basert på deres holdninger. Ulike former for angst var en vanlig følelse, både blant de som var negativ og positiv, basert på bekymringsfaktorene som ble beskrevet i forrige avsnitt. Blant lærerne som hadde en positiv holdning, var følelsene *curiosity* (nysgjerrighet) og *elation* (glede/lettelse). Både de finske og svenske lærerne gledet seg til å undervise programmering og var nysgjerrige på hvordan de skulle få det til (Korhonen et al., 2022; Misfeldt et al., 2019). Nytten av programmering er stor, men innføringer og adoptering av nye emner i læreplaner er krevende, og poengteres i begge studiene som er gjennomført i våre naboland.

Lærernes holdninger til programmering og algoritmisk tenkning i matematikkfaget kan oppsummeres som generelt gode, men med en rekke bekymringer for kompetanse og komplekse utfordringer lærere må takle for å drive god undervisning. Det teoretiske

rammeverket for oppgaven tar for seg lærere som en homogen gruppe. Mitt ønske for dette prosjektet er å se på både lærere som en homogen gruppe, og om det finnes noen forskjeller blant lærere i ulike grupper som alder, undervisningserfaring i matematikk og deltakelse i etterutdanning eller kurs innenfor programmering. Videre i metodekapittelet reflekterer jeg rundt det teoretiske rammeverket og alle metodevalgene som har blitt gjort, som har påvirket funnene i prosjektet (Capraro et al., 2019).

3 Metode

Dette metodekapittelet presenterer den metodiske tilnærmingen som har blitt benyttet for å besvare problemstillingen. Den kvantitative tilnærmingen med bruk av en spørreundersøkelse krever refleksjoner rundt hvordan forskningen har blitt gjennomført. En kvantitativ tilnærming har som hensikt å gjøre en objektiv analyse av fenomener, med en vurdering av mulighetene for å generalisere funnene til populasjonen for prosjektet.

Metodekapittelet er i korte trekk strukturert slik: Forskningsdesignet plasserer prosjektet innenfor forskningsfeltet og drøfter hvordan metodikken skal hjelpe med å svare på problemstillingen og forskningsspørsmålene. Jeg har brukt en spørreundersøkelse som tilnærming for prosjektet og reflekterer rundt valg som har påvirket innsamlingen av data. Dataanalysen ser på hvordan datasettet fra spørreskjemaet anvendes for å gjennomføre tester som gir muligheten for å drøfte funnene og deres muligheter for generalisering. De forskningsetiske aspektene er viktig å ha reflektert rundt, spesielt det distanserte forholdet mellom meg som forsker og respondentene. Utviklingen av spørreskjema over nett har gitt få etiske utfordringer, og respondentens muligheter for å kunne velge svar som passer dem har stått sentralt. Reliabilitet og validitet er begreper som hjelper med å vurdere kvaliteten til prosjektet, hvor styrker og svakheter tydeliggjøres. Feilkildene vil oppsummere hva som kunne og burde vært gjort annerledes.

3.1 Forskningsdesign

Valget av metode er nært knyttet opp mot problemstillingen og forskningsspørsmålene til studien. Bakgrunnen for valg av problemstilling er en blanding av min indre motivasjon for forskningsfeltet, samt en interesse for å bidra med å tette en del av kunnskapshullet i forskningen i dag (Gleiss & Sæther, 2021). Mitt masterprosjekt omhandler et kunnskapsområde som er lite forsket på i Norge. Lignende forskning er blitt gjennomført, men i andre land, på høyere alderstrinn og ved bruk av kvalitativ metode. For å kunne spisse problemstillingen, har forskningslitteraturen vært gunstig for å finne teori som er mulig å operasjonalisere i prosjektet[8]. Problemstillingen og forskningsdesginet har vært i kontinuerlig endring i løpet av prosjektets gjennomføring. Problemstillingen og hensikten med prosjektet må tilpasses de metodiske mulighetene og begrensingene masteroppgaven har, samt de

[8] Hentet fra prosjektskisse

teoretiske ramme som forskningslitteratur bidrar med (Pentagon og Gleiss & Sæther, s. 26). Eksempelvis har studenter som skriver master begrenset tid og ressurser sammenlignet med større forskningsprosjekt. Dette vil påvirke hvilke konklusjoner og generaliseringer av populasjon som er mulig å gjøre i mitt prosjekt, men mer kunnskap på dette feltet kan bidra med å danne et grunnlag for fremtidig forskning.
Mitt ønske for oppgaven er at den kan belyse et kunnskapshull hvor det er behov for forskning nå og i fremtiden (Capraro et al., 2019). Kunnskapshullet som jeg ønsker å fylle, eller i det minste legge til rette for økt kunnskap, er mangelen på dokumentasjon av norske læreres meninger og tanker rundt implementeringen av programmering og algoritmisk tenkning i læreplanen. I Sverige og Finland ble programmering og algoritmisk tenkning en del av læreplanen tidligere enn i Norge (Korhonen et al., 2022; Misfeldt et al., 2019). I disse nordiske landene er lærerne motivert for å undervise programmering og algoritmisk tenkning, men lærerne vekker bekymringer rundt behovet for kompetanse og struktur. I Norge har det blant annet blitt forsket på programmering som del av matematikkfaget (Kaufmann & Stenseth, 2023) og kvantitativ forskning rundt studenters syn på verdien av programmering i matematikk (Kaufmann et al., 2019). Etter mitt syn mangler det dermed forskning på lærernes tanker og holdninger til hvordan implementeringen av programmering og algoritmisk tenkning har blitt implementert i matematikkundervisningen etter LK20.

I mitt masterprosjekt har jeg valgt å gjennomføre en spørreundersøkelse i en kvantitativ tilnærming gjennom Nettskjema. Problemstillingen for prosjektet vil også være mulig å forske på kvalitativt, men jeg er interessert i funn som så godt som mulig vil gjenspeile en populasjon. Dette krever et større utvalg av forskningsdeltakere, enn hva som er mulig med kvalitativ forskning i et masterprosjekt (Gleiss & Sæther, 2021).[9] I lys av problemstillingen er jeg interessert i å finne ut hvilke meninger og synspunkter lærere har. Dette gir prosjektet to mulige vinklinger, spørreundersøkelse eller intervju, den ene oftest kvantitativ, den andre kvalitativ. Det hadde vært muligheter for å ta i bruk andre metoder med likheter i teorigrunnlaget, å gjennomføre observasjoner i klasserom vil kreve en annen problemstilling som gjør det mulig å forske på undervisning og atferd i klasserommet. Endringer i metoden kan ha stor innvirkning i teorigrunnlaget som kreves og konklusjoner som trekkes i oppgaven (Gleiss & Sæther, 2021). Jeg har hatt et behov for å samle inn data fra mange respondenter,

[9] Delvis hentet fra prosjektskisse

siden jeg ønsker å sammenligne hvilket syn lærere med ulik programmeringskompetanse har innen inkludering av programmering i matematikkundervisningen. Kvantitativ metode krever et mer strukturert arbeid før datainnsamlingen og en tydelig operasjonalisering av begrep til målbare variabler i spørreundersøkelsen[10]. Opprinnelig hadde jeg et ønske om å kombinere spørreundersøkelse og observasjon, men spørreundersøkelsen ble for omfattende til å utføre observasjon som supplerende metode. Jeg ønsket å spisse prosjektet mitt mot kvantitativ tilnærming for å få resultat med best mulig kvalitet. Dersom metoden henger sammen med problemstilling, teori og funn er det uproblematisk å la personlige interesser bestemme hvilken metode som tas i bruk for innsamling og analyse av data (Gleiss & Sæther, 2021).

Det er viktig at datainnsamler reflekterer rundt sin posisjon og maktforhold i prosjekter. Innsamling, analyse og konklusjonene som trekkes bør i så liten grad som mulig preges av min posisjon som gjennomfører prosjektet. Dette vil reflekteres i oppgaven og feilkilder for oppgaven skal heller reflekteres rundt, for å gi funnene en reell verdi (Gleiss & Sæther, 2021)[11].

3.1.1 Refleksivitet

Gjennom metodekapittelet ønsker jeg å kunne begrunne valgene som har blitt gjort i forskningsprosessen og hvordan min posisjon har betydning og implikasjoner på forskningen. Forskning som kunnskapsinnhenting har store styrker, men også sine svakheter. Dette prosjektet gjennomføres med en helt egen posisjonalitet og kunnskapen som dannes vil aldri bli helt lik dersom den gjennomføres av noen andre (Gleiss & Sæther, 2021). Min alder, kjønn, kultur, interesser, forventninger, kunnskap, relasjoner og så videre er alle faktorer som påvirker forskningsprosessen og er viktig å tenke over for meg som forfatter av oppgaven, samtidig er det viktig for leser for å kunne danne egne tolkninger og meninger.

Gjennom hele forskningsprosessen er det viktig for meg å være kritisk til min egen betydning for bidraget til forskningsfeltet. En viktig del av prosjektet er å være bevist på forholdet mellom datainnsamler og forskningsdeltakere (Gleiss & Sæther, 2021)(Cohen et al., 2018). I dette prosjektet er jeg en «outsider» som på mange måter peker på en avstand mellom forsker og forskningsdeltakerne. Jeg står også i en posisjon til å definere kategorier av type deltakere

[10] Hentet fra prosjektskisse

[11] Hentet fra prosjektskisse

og svarene de gir. Spørreundersøkelser kan på en måte ses på som et skjevt maktforhold, hvor forfatter kan velge hva som er viktig med datamaterialet og gjøre tolkninger av forskningsdeltakernes svar, samtidig som en kan velge vekk eller være kritisk til andre svar. Denne delen av maktforholdet og min posisjon kan ses på som en ubalanse. Samtidig er jeg helt avhengig av forskningsdeltakerne og det hadde vært uetisk å utnytte denne skjevheten. Dette er mitt ansvar som forsker å gjøre så riktig jeg klarer ved å begrunne og reflektere rundt mine valg i forskningsprosessen (Gleiss & Sæther, 2021). Gleiss og Sæther (2021) poengterer i nyere tid at både forsker og forskningsdeltakerne ses på som subjekter i forskningen, hvor vi alle ønsker å utvikle forståelsen av tema som undersøkes, som i dette prosjektet er programmering og algoritmisk tenkning i skolen. Kunnskapen dannes i et samarbeid mellom forsker og forskningsdeltaker, og det er viktig at forfatter belyser alles bidrag for å kunne trekke konklusjoner som representerer alle subjektene.

3.1.2 Oppgavens teorigrunnlag

Oppgavens teorigrunnlag er en viktig del av enhver oppgave, da den skaper både rammene og retningen for prosjektet (Spangler & Williams, 2019). En viktig del av forskning er hvordan det kollektive samarbeidet mot å videreutvikle oppfatningen av fenomener i hverdagen (Gleiss & Sæther, 2021). Ordet teori brukes i mange sammenhenger, fra hverdagssamtaler, «jeg har en teori om…», helt til svært komplekse og ulike forståelser av abstrakte begreper og ideologier. Dermed er det viktig å reflektere rundt hvilken type teori som er anvendt for prosjektet (Spangler & Williams, 2019).

Det teoretiske rammeverket for oppgaven har som hensikt å gi forskeren og leseren en lik innsikt og forståelse av fenomener innenfor feltet, samtidig som det setter en ramme for prosjektet. Spangler & Williams (2019) sammenligner det teoretiske rammeverket med en vinduskarm, ikke bare retter karmen fokuset for hvor man skal se, men også hvor en ikke skal se. Teorikapittelet viser ikke bare hvilken teori og tidligere forskning som er innenfor karmen, men også hva som er utelatt. Disse teoretiske rammeverkene utvikles i forskningsfeltene gjennom «samtalene» som skjer mellom forskerne. Forskere vil enten finne eksisterende teorier som passer deres forskning av fenomener, eller tilpasse sine teoretiske rammeverk. Det kan hende en kombinasjon av lignende teoretiske rammeverk gir et bedre resultat (Spangler & Williams, 2019). En kan også finne et godt rammeverk fra andre forskningsfelt, og ønske å ta i bruk disse i sin egen forskning. Eksempelvis forsket McGinnis et al. (2020) på algoritmisk tenkning hos naturfagsstudenter, noe som kan være overførbart og nyttig for dette prosjektet.

Gleiss & Sæther (2021) peker på to ulikheter som er viktig å reflektere over i dette prosjektet, som er teori og empiri, og abstrakt teori og tidligere forskning. «Teorier blir ofte forklart som en idé eller et system av ideer om et fenomen, mens empiri er konkrete studier av fenomener» (Gleiss & Sæther, 2021). Samtidig er mye av teori basert på empiriske undersøkelser av fenomener. Dette kan gjøre overgangen mellom teori og empiri utydelig eller vanskelig å konkretisere, når vil et begrep gå fra å være empirisk undersøkt til å bli et teoretisk begrep? Abstrakt teori og tidligere forskning går litt videre inn på hvordan en kan plassere og diskutere sine egne resultater og funn mot forskningsfeltet. Abstrakte teorier er anvendbare innenfor flere forskningsfelt, for eksempel begrepet algoritmisk tenkning. Algoritmisk tenkning kan anvendes som et teoretisk begrep, fordi det finnes mange bidrag i forskningsfeltet for å definere og forme et slikt abstrakt begrep. Avhengig av en rekke faktorer som språk, fagfelt slik som matematikk, programmering eller problemløsning, i hvilken tid forskningen ble gjennomført og så videre, vil en finne ulikheter i tidligere forskning på hvordan begrepet anvendes for å kunne trekke konklusjoner. Teori som er benyttet gjennom prosjektet varierer fra abstrakte teorier om begrep som algoritmisk tenking og programmering, til tidligere empirisk forskning av feltet på lærernes syn på programmering. En viktig del av diskusjonen i oppgaven vil dermed være å reflektere rundt mine funn i lys av det teoretiske rammeverket og hvilken forskjell det utgjør å diskutere funnene mot teori og empiri.

Forskningsfeltet dette prosjektet tilhører er lite utforsket og samtidig er det en økende interesse for feltet sammenlignet med andre felt i matematikken. Derfor er det viktig å innhente informasjon fra forskning som er så nært feltet som mulig (Cohen et al., 2018). Hadde noen forsket på eksakt samme problemstilling og teori som denne oppgaven, ville hensikten bak prosjektet være å se om forskningen som ble gjennomført er pålitelig. Målet med dette prosjektet er å utforske et fagfelt som er forsket på i andre nordiske land, men ikke i Norge (Korhonen et al., 2022; Misfeldt et al., 2019). Dette kan ses på som å «sette i gang en samtale» innenfor et forskningsfelt hvor det er et behov for grundig forskning (Cohen et al., 2018; Gleiss & Sæther, 2021).

3.1.3 Litteratursøk

Søket etter litteratur og forskning var i starten av dette prosjektet en kompleks prosess. I starten hadde ikke oppgaven en tydelig definert retning, og litteratursøk med generelle søkeord var utfordrende (Siebert, 2019). Søkeordene «algoritmisk tenkning» og oversettelsen

«computational thinking» ble viktige søkeord for å finne informasjon. «Programmering» ble et altfor bredt begrep, men å spesifisere «programmering i skolen» ga en rekke gode funn i Oria og Google Scholar. Veileder for dette prosjektet har publisert forskning og var dermed også godt orientert innenfor forskningsfeltet og bidro med nylig skrevede forskningsartikler (Kaufmann et al., 2022; Kaufmann & Stenseth, 2023). Disse tok for seg forskning på feltet som var gjort i nordiske land, hvor både Sverige og Finland har gjort godt forskningsarbeid, skrevet på engelsk. Etter å ha funnet en rekke gode forskningsartikler, var det enkelt å bla gjennom litteraturlistene for å finne forskning som diskuterte teorier, begreper og annen tidligere forskning, både på samme og ulike felt (Cohen et al., 2018; Gleiss & Sæther, 2021). En ulempe med å ta i bruk denne metoden er at dersom man ikke får lest en variasjon av forskning, men heller leser forskning hvor forskerne stadig referer til hverandres teori og empiri kan gi et feil bilde av feltet. Begrepet algoritmisk tenkning hadde en rekke ulikheter blant forskere fra Amerika, Europa og Norden. En viktig del av oppgaven var derfor å reflektere hvilke teorier som ville skape en operasjonalisering av begrepet algoritmisk tenkning og programmering som stemmer med forskningsfeltet og er forståelig for forskningsdeltakerne (Capraro et al., 2019).

Nye synspunkter og/eller endringer i tolkninger av teorigrunnlaget kan gjøre store forskjeller i konklusjonene som trekkes i oppgaven. Jeg fant stadig ny forskning i løpet av prosjektet som ga en mer presis eller tilføyende teori (Siebert, 2019). I prosjektet kan dette vises hvor forskjellig begrepet algoritmisk tenkning defineres og forklares i forskjellige forskningsartikler (Gleiss og Sæther, 2021).

3.2 Valg av tilnærming, spørreundersøkelse

Etter at det var bestemt at spørreundersøkelse skulle brukes som tilnærming for innhenting av data til prosjektet, og reflektert rundt hvilke muligheter og begrensninger dette medfører, fulgte et behov for et spørreskjema å sende ut. Spørreskjemaet er i stor grad preget av forhåndstruktur fra den som lager det, og det var dermed viktig å lage et spørreskjema som er tydelig for respondentene og som gir et datasett som er forskbart. Svakheter eller unødvendige feilkilder i spørreskjemaet vil minske kvaliteten på datasettet som senere analyseres. Siden dette ikke kan endres etter spørreskjemaet er sendt ut, har det vært viktig for å reflektere rundt hva som må inkluderes. Videre redegjøres det for utforming og struktur av spørreskjemaet, utvalget av respondenter som blir invitert til å svare og hvordan datasettet kan analyseres på en hensiktsmessig måte.

3.2.1 Utforming og målenivåer

I utformingen av spørreskjemaet vil det være et stort behov for å operasjonalisere begrepene programmering og algoritmisk tenkning, slik at de blir forskbare. Hensikten med å operasjonalisere de teoretiske begrepene er å dele begrepene i mindre deler og gjøre dem til målbare variabler, slik at de senere kan analyseres og knyttes opp mot begrepene i teorien (Cohen et al., 2018; Gleiss & Sæther, 2021)[12]. Gleiss & Sæther (2021) definerer en variabel som «en egenskap eller et kjennetegn som varierer mellom respondentene i undersøkelsen», en variabel vil også ha tilhørende verdi med ulike målenivå.

Et eksempel på operasjonalisering som er relevant i dette prosjektet er å dele det teoretiske begrepet, algoritmisk tenkning opp i nøkkelbegrepene og ferdighetene til konkrete spørsmål eller påstander. Nøkkelbegrepene under algoritmisk tenkning består av syv underbegreper som også videre må operasjonaliseres, noen av underbegrepene krever flere påstander for at jeg kunne betrakte det som en riktig operasjonalisert. Operasjonaliseringen gir muligheten til å spørre i hvilken grad lærere mener at disse er til nytte i undervisning på mellomtrinnet, og man ender opp med en rekke verdier for de forskjellige variablene[13]. Dette arbeidet gjør at algoritmisk tenkning kalles et konstrukt, og dette konstruktet består av totalt 17 påstander (Gleiss & Sæther, 2021). To av disse 17 påstandene måler variabelen abstrahering, som har de mulige verdiene «uenig, «delvis uenig», «nøytral», «delvis enig» og «enig» på en Likert-skala fra 1-5, samt muligheten for å svare «vet ikke». Som forsker har jeg muligheten til å definere hvilke verdier som er mulig å svare for respondentene. Variabler og verdier deles inn i forskjellige målenivåer, avhengig av hvilke verdier som kan høre til variabelen (Cohen et al., 2018; Gleiss & Sæther, 2021).

Det er vanlig å skille mellom fire målenivåer: nominalnivå, ordinalnivå, intervallnivå og forholdstallsnivå (Johannessen et al., 2016 referert i Gleiss & Sæther, 2021). Variabler på nominalnivå har ingen naturlig orden og enheten har like eller ulike verdier (Grønmo, 2020). Kjønn er en variabel på nominalnivå og har tradisjonelt sett hatt verdiene «mann» eller «kvinne» (Gleiss & Sæther, 2021). Variabler på nominalnivå kan ha flere enn bare to verdier, for eksempel har variabelen, kommuner i Norge, 357 mulige verdier, og det finnes ingen naturlig rekkefølge å sortere disse kommunene. Variabler på ordinalnivå kjennetegnes av at

[12] Hentet fra prosjektskisse

[13] Delvis hentet fra prosjektskisse

det finnes en naturlig rekkefølge for verdiene. Grader av enighet til en påstand er en svært sentral variabel på ordinalnivå i kvantitativ forskning, det finnes en tydelig rekkefølge i hvilken verdi variabelen har. Det er derimot vanlig praksis i kvantitativ forskning å gjøre disse gradene av enighet om til en variabel på intervallnivå.

I mitt prosjekt består store deler av spørreskjemaet som ble sendt ut av påstander hvor respondenten bestemmer sin grad av enighet, «uenig» (1), delvis uenig» (2), «nøytral» (3), «delvis enig» (4) og «enig» (5). Variabler på intervallnivå gjør det mulig å ta i bruk kvantitative analysemetoder som t-test, korrelasjonsanalyse og regresjonsanalyse (Cohen et al., 2018; Gleiss & Sæther, 2021; Grønmo, 2020). Variablene på intervallnivå har en fast avstand mellom hverandre, fra «uenig» (1) til «delvis enig» (4) er differansen tre (3), denne faste avstanden mellom grader av enighet kalles en Likert-skala (Cohen et al., 2018; Gleiss & Sæther, 2021). Variabler på forholdstallsnivå skiller seg fra variablene på intervallnivå i form av at variabler på forholdstallsnivå har et naturlig nullpunkt. Variabelen alder har målenivået forholdstall, fordi det er naturlig å si at en person som er 60 år er dobbelt så gammel som en på 30 år (Grønmo, 2020; Jacobsen, 2015 referert i Gleiss & Sæther, 2021). Jeg vil ikke gjøre noen analyser i min oppgave som behandler intervallnivå og forholdstallsnivå ulikt, begge disse målenivåene er metriske og kan brukes til t-test og korrelasjonsanalyse (Cohen et al., 2018). Operasjonaliseringen av teoretiske begrep krever at variabler deles inn i en av disse målenivåene, og et slikt arbeid er min forenkling av et fenomen som kan være ganske komplekst (Gleiss & Sæther, 2021). Mitt ansvar som forsker er å gjøre denne operasjonaliseringen på en måte som ivaretar teorien på riktig måte, og samtidig gjør det mulig for respondenten å kunne utrykke sine meninger (Cohen et al., 2018).

Som tidligere nevnt er strukturen for et spørreskjema bestemt på forhånd, og er ikke mulig å tilpasse etter det er sendt ut (Cohen et al., 2018; Gleiss & Sæther, 2021). Noen kvalitative tilnærminger kan dra nytte av å endres dersom forskeren oppdager mangler eller forbedringer som kan gi et rikere datasett, denne muligheten fantes ikke i mitt prosjekt. En stor andel av tiden i mitt prosjekt gikk til å skape et spørreskjema vil kunne gi et rikt datasett (vedlegg 1).

Gleiss & Sæther (2021) skiller mellom tre ulike typer spørsmål, faktaspørsmål, holdningsspørsmål og atferdsspørsmål. Faktaspørsmål har som hensikt å samle inn informasjon om respondenten, f.eks. alder, kjønn, undervisningserfaring i matematikk og deltakelse i etterutdanning og kurs innenfor programmering. Jeg valgte å samle inn denne informasjon på første side i spørreskjemaet, dermed er respondenten klar over hvilke

faktainformasjon og personopplysninger som innhentes om dem (Cohen et al., 2018). Det hadde vært skuffende for respondenten å besvare et spørreskjema i ti minutter, forså å finne ut at det er nødvendig at det samles inn personopplysninger som hen ikke ønsker å oppgi. Resten av spørreskjemaet består av holdningsspørsmål, både i form av spørsmål og påstander hvor respondenten tar stilling til sin grad av enighet (Gleiss & Sæther, 2021). Holdningsspørsmålene er ofte vanskeligere å besvare, enn faktaspørsmålene, som også er en av grunnene til å ha faktaspørsmål på første side. Holdningsspørsmål inkluderes i min undersøkelse i form av respondentenes holdninger til implementeringen av programmering og algoritmisk tenkning i matematikkfaget, meninger om hvor nyttig dette er, hvordan det kan undervises, samt vurderinger av det teoretiske begrepet algoritmisk tenkning. Atferdsspørsmål undersøker respondentenes handlinger eller vaner, men dette vil ikke være nødvendig i mitt spørreskjema (Gleiss & Sæther, 2021). Spørsmål om atferd er ikke interessant for problemstillingen og forskningsspørsmålene som er satt, men dette er en del av feltet som kan være nyttig å forske på.

Ikke bare er det viktig å formulere gode spørsmål og påstander, men svaralternativene i et spørreskjema er ofte lukkede og bestemmer hva respondenten har muligheten til å svare (Cohen et al., 2018). I et spørreskjema kan forskeren velge å ha lukkede eller åpne svaralternativ. Lukkede svaralternativ har en stor fordel ved at det er enklere å sammenligne svarene i datasettet, dette er en forutsetning for kvantitativ dataanalyse (Gleiss & Sæther, 2021). Åpne svaralternativ kan også kodes slik at de kan plasseres i en grad av enighet eller kategori, men dette er svært tidkrevende og åpner for feilkilder hos forskeren. En langt mer vanlig bruk av åpne svaralternativ er at forskeren gir mulighet for respondenten til å utdype et tidligere spørsmål eller påstand som hadde et lukket svaralternativ (Cohen et al., 2018; Gleiss & Sæther, 2021). Jeg valgte å ta i bruk et åpent svaralternativ én gang i spørreskjemaet, men ikke med hensikt om å kunne inkludere kvalitativ analyse som en del av funnene i oppgaven. I samråd med min veileder valgte jeg å ha et spørsmål med åpent svaralternativ, «Hva mener du kjennetegner begrepet algoritmisk tenkning?». Årsaken bak dette spørsmålet var å avdekke om noen av respondentene hadde store misoppfatninger om hva begrepet algoritmisk tenkning betyr, ettersom det er et nylig innført begrep i læreplanen (Kunnskapsdepartementet, 2020).

Det er også andre viktige deler av strukturen i spørreskjemaene jeg som forsker bør ha reflektert rundt for å skape et datasett med kvalitet. Spørsmålsbatteri er nyttig for å kunne få respondenten til å svare på ulike aspekter av samme fenomen (Cohen et al., 2018; Gleiss &

Sæther, 2021). Når variabelen abstrahering måles som en del av variabelen algoritmisk tenkning må respondenten ta stilling til påstandene «det er viktig å kunne trekke ut de sentrale delene av et problem» og «det er viktig å kunne legge bort unødvendige deler av et problem». Jeg som forsker har da muligheten å se i datasettet om det er likhet i grad av enighet til påstandene, og hvis det ikke er en likhet, drøfte hva den kan skyldes (Cohen et al., 2018). Kanskje spørsmålet er formulert dårlig eller at respondenten mener at «å trekke ut de sentrale av et problem» er viktigere eller mindre viktig, som «å kunne legge bort deler av et problem». En viktig kvalitet med godt utviklede spørreskjema er at svarene ikke overlapper hverandre og det er mulig for respondentene å velge mellom ulike svar som kan være relevante (Gleiss & Sæther, 2021).

Ved flere spørsmål og påstander i spørreskjemaet har respondenten muligheten å svare «annet» eller «vet ikke». «Annet» viser at jeg som forsker anerkjenner at det kan være flere relevante svar, men jeg forventer dette til å være en liten andel. Skulle det være mange som har besvart «annet» er det et viktig alternativ som burde ha vært mulig å velge, men som jeg har oversett. «Vet ikke» har også en lignende effekt og det er en vanlig misoppfatning at dette er det samme som å være «nøytral» (eller hva en velger å kalle midtpunktet på en Likert-skala). Ved en høy andel av respondenter som har valgt «vet ikke» tyder til enten en dårlig formulert påstand, eller at respondenten ikke har forutsetninger til å velge en grad av enighet (Cohen et al., 2018). I mitt spørreskjema svarte 14 av 45 «vet ikke» knyttet til spørsmålet «Slik programmering er i læreplanen nå, bør det gjøres endringer?». Spørsmålet er ikke utydelig formulert, men det er nok mange av respondentene som enten ikke vet nok om programmering i læreplanen eller har en formening om hvordan læreplanene bør endres.

Et viktig valg er om Likert-skalaen skal ha odde- eller partall antall grader av enighet. Oddetall gir en middelkategori, som jeg kalte «nøytral», som jeg synes passet for mitt spørreskjema. Dersom jeg hadde valgt å ikke inkludere en middelkategori, ville jeg ha tvunget respondenten til å velge en grad av enighet som heller mot en side. Jeg tror uten denne middelkategorien ville langt flere respondenter valgt «vet ikke», dersom de ikke har sterke meninger om et fenomen (Cohen et al., 2018; Gleiss & Sæther, 2021). En annen variasjon i spørreskjemaet er å veksle mellom påstander som er positivt eller negativt formulert, for å sikre seg at respondentene er oppmerksom når de besvarer spørreskjemaet.

For å finne ut hvilke variabler som skulle operasjonaliseres til spørsmål og påstander brukte jeg like formuleringer fra tidligere kvantitativ forskning av Rich (2020), Misfeldt et al. (2019)

og Korhonen et al. (2022). Bruken av lignende spørsmål er både tidssparende og viktigst for meg som forsker i et masterprosjekt, en trygghet at jeg operasjonaliserer noen variabler på en lignende måte som tidligere forskning. Jeg laget en logg over hvilke teorier og variabler som skulle måles i spørreskjemaet, og uten inkluderingen av spørsmål og svaralternativ fra tidligere forskning hadde det vært vanskelig å luke ut de dårlige spørsmålene som bare skaper rot i datasettet.

Det er altså en rekke faktorer som må tas hensyn til i oppbygningen av spørreskjemaet og noe av det viktigste er at det forstås likt av respondentene og meg som forsker, som skal analysere verdiene til variablene som er operasjonalisert (Cohen et al., 2018; Gleiss & Sæther, 2021).[14] Jeg hadde tilgang på noen som kunne gjennomføre en pilotundersøkelse av spørreskjemaet, men fikk nyttige tilbakemeldinger fra en dyktig veileder.

3.2.2 Utvalg

Utvalget av respondenter vil være hovedsakelig en blanding av sannsynlighetsutvalg og et ikke-sannsynlighetsutvalg (Gleiss & Sæther, 2021)[15]. Et relevant kriterium for deltakerne at de jobber som matematikklærere på mellomtrinnet, og jeg har et ønske om å kunne generalisere denne gruppen. Andre relevante opplysninger er å dele respondenter inn i grupper på alder, kjønn og erfaring med programmering i undervisning som kjennetegner et ikke-sannsynlighetsutvalg. Dette gjør det mulig å sammenligne hvilke avhengige variabler som kan korrelere i ulik grad med andre uavhengige variabler. Danielsen (2013) forklarer slike utvalg i masterprosjekter bekvemmelighetsutvalg, hvor forskeren ideelt sett ønsker et utvalg som representeres er tilfeldig valgt, men dette var i mitt prosjekt vanskelig å oppnå (Capraro et al., 2019).

Når spørreskjemaet var ferdig utformet, ble det i første runde lagt ut på Facebook i gruppene «Status lærer» med 42.000 medlemmer og «Matematikkdidaktikk» med 20.200 medlemmer. Det ble også sendt ut en mail til alle skoler i Troms og Finnmark som var delt i mellomtrinn basert på Utdanningsdirektoratets nasjonale skoleregister.

[14] Delvis hentet fra prosjektskisse

[15] Hentet fra prosjektskisse

Rektor eller andre mottakere av mail ved skolene vil fungere som portvakter, og man var derfor avhengig av at disse portvaktene sender ut spørreskjemaet til lærerne (Gleiss & Sæther, 2021). Forholdet mellom portvakten (rektor) og forskningsdeltakerne (lærerne) kan påvirke resultatene i prosjektet hvis maktforholdet oppleves som skjevt, bekymringer for deling av kritisk informasjon eller at det legges press på forskningsdeltakerne til å svare. Viten at dette er faktorer som kan påvirke forskningen var det viktig at mailen som ble sendt ut til skolen gjorde det tydelig for portvakt og forskningsdeltakerne at spørreundersøkelsen var anonym og svarene ikke kan spores i ettertid (Gleiss & Sæther, 2021). Både portvaktene og forskningsdeltakerne ble oppfordret til å dele spørreundersøkelsen videre, hvor det var ønsket en snøballeffekt for å rekruttere flere portvakter og forskningsdeltakere. En mulig ulempe med denne effekten er et økt antall svar hvor forskningsdeltakerne deler samme mening. Dette er fordi det er sannsynlig at forskningsdeltakerne deler spørreundersøkelsen til noen de deler sin mening med. Et mulig utfall er at de som er begeistret for programmering i undervisningen blir representert i større grad, enn hva som kan gjenspeiles i populasjonen (Gleiss & Sæther, 2021). Dette er også en situasjon hvor jeg som forsker ikke har mulighet til å miste slike potensielle forskningsdeltakere, men det er en viktig vurdering å gjøre.

Etter første runde ble det betraktet at det hadde vært gunstig å utvide utvalget og inkludere skoler i Nordland med samme kriterier. Tanken var at det var et behov for å få respondenter som hadde stor mulighet for å ha tilhørighet til UiT Norges arktiske universitet, som kunne øke antall respondenter. Ideelt sett kunne mailene har vært sendt ut til større deler av landet, men det ville resultert i en stor mengde manuelt arbeid. Svært få svarte på mailene, hovedsakelig autosvar, men majoriteten, 28 av respondenter mottok spørreskjemaet over mail fra skoleleder. Grupper fra sosiale media ga 14 respondenter, og 3 mottok spørreskjemaet på andre måter, som det antas er personer som har delt skjemaet privat. I respekt av læreres tid ble det purret én gang både på mailene og i gruppene på Facebook, men det er vanskelig å dokumentere hvor effektivt dette var. Cohen et al. (2018) anbefaler å purre to ganger, mens grupper på sosiale media ofte har strenge regler på spam av samme innlegg, derfor ble det gjennomført kun én purring. Totalt var det 45 respondenter.

Utvalget av respondenter er altså ikke et perfekt sannsynlighetsutvalg, som svekker generaliseringsmulighetene i prosjektet. Når spørreskjemaet blir sendt over mail vil det kanskje være mer sannsynlig at skoler jeg har relasjoner til fra tidligere vil ha en høyere besvarelsesandel. Dersom utsender har relasjoner til skoler som vil være skeivfordelt i henhold til f.eks. sosioøkonomisk bakgrunn eller andre variabler, som kan påvirke resultat og

funn i prosjektet. I likhet, når spørreskjema legges ut på sosiale medier vil det kanskje tiltrekke seg lærere som er spesielt interessert i jobben sin eller fagområdet for prosjektet. Dette er også en grunn til å inkludere variablene kjønn, alder og erfaring med programmering i undervisning. Disse faktorene gir muligheten for å generalisere utvalget svakere, men slike faktorer må tas høyde for i et masterprosjekt for å gi et rikt nok datasett ved kvantitativ metode.

Ved kvantitativ tilnærming er det færre forskningsetiske faktorer å ta hensyn til, ettersom det er anonymt og vanskelig å spore enkeltpersoners svar[16]. Etter ønske om å få et høyere antall respondenter, kunne de som ønsket å lese masteroppgaven sende meg en mail, slik at deres svar ikke var koblet opp mot en mail. Ved hjelp av veileder ble det også bekreftet at dette ikke var i strid med noen forskningsreglementer av studieadministrasjonen ved Høgskolen i Østfold.

3.3 Statistisk analyse

Kvantitativ datanalyse skiller seg fra kvalitativ dataanalyse ved at analyseprosessen er rimelig standardisert og baserer seg på statistiske analysemetoder (Gleiss & Sæther, 2021). Tilgangen på dataprogrammene nettskjema.no, Excel og SPSS (Statistical Package for the Social Sciences) har gjort kvantitativ forskning «mindre komplisert», enn før. Disse programmene gjør mye av det manuelle arbeidet i form av strukturering og matematiske beregninger. Litt av arbeidet med statistisk analyse er å behandle Excel-arket slik at det kan bli importert i SPSS. Gradene av enighet på endres fra tekst til sin tilsvarende tallverdi, «uenig» endres til «1», «litt uenig» endres til «2» og så videre. Verdien «vet ikke» ble fjernet fra Excel arket som måtte eksporteres til SPSS

For å analysere data ble SPSS tatt i bruk som dataanalyseverktøy. Noe av dataanalysen vil ha et behov for å presenteres univariat, men ettersom hensikten med prosjektet var å forske på forskjeller og sammenhenger mellom ulike grupper vil univariat analyse heller brukes for å skape oversikt i datamaterialet. (Gleiss & Sæther, 2021). Ved bruk av univariat analyse var det enkelt å få en tabelloversikt over gjennomsnittsverdi og standardavvik til et sett med variabler av ett fenomen. Ettersom problemstillingen og forskningsspørsmålene så på sammenhengen mellom ulike variabler var det hensiktsmessig og nødvendig å gjennomføre

[16] Hentet fra prosjektskisse

flere bivariat analyser i form av t-test og korrelasjonsanalyse (Cohen et al., 2018; Gleiss & Sæther, 2021). Korrelasjonsanalyse vil ga verdi i Pearsons r (korrelasjonsnivå) og signifikansnivå (p-verdi). Når slike analyser gjøres kan man si noe om hvor godt for eksempel en lærer sin alder korrelerer med ønske om videre utdanning innen programmering og algoritmisk tenkning, altså med de teoretiske begrepene som har blitt operasjonalisert[17]. Utvalget for prosjektet lå et sted mellom sannsynlighets- og ikke-sannsynlighetsutvalg, altså et bekvemmelighetsutvalg (Cohen et al., 2018; Danielsen, 2013; Gleiss & Sæther, 2021). En viktig forutsetning for å kunne generalisere utvalget var at utvalget er et sannsynlighetsutvalg, men det var fortsatt spennende å analysere om funnene er så signifikant at de kunne generaliseres i en «hva om utvalget var generaliserbart»-situasjon. Her er også størrelsen på utvalget en svakhet (Gleiss & Sæther, 2021).

3.3.1 Univariat analyse

Univariat analyse er en form for statistisk analyse som bare undersøker hvordan enheten fordeler seg på én variabel (Gleiss & Sæther, 2021). Siden univariat analyse bare viser en variabel er dette en egnet type analyse for å skaffe seg en oversikt over datasettet. En automatisk generert rapport fra spørreundersøkelsen viser en univariat analyse av alle variablene, men den har ikke regnet ut gjennomsnitt eller standardavvik (vedlegg 1). Variabler som var viktig for prosjektet eller som var overraskende sammenlignet med det teoretiske rammeverket ble analysert univariat (Gleiss & Sæther, 2021). Gjennomsnittet og standardavviket ga en tallverdi på konsentrasjonen av verdiene og hvor mye de avviker fra gjennomsnittet. Det ga muligheter for å regne differansen i gjennomsnitt og standardavvik mellom variabler.

3.3.2 Bivariat analyse

Bivariat analyse skiller seg fra univariat analyse, ved at den ser på forskjeller eller sammenhenger mellom to eller flere variabler (Gleiss & Sæther, 2021). De to formene for bivariat analyse som jeg har brukt i mitt prosjekt er t-test og korrelasjonsanalyse.

T-test er en form for bivariat analyse som kan brukes til å sammenligne gjennomsnitt for to grupper og få oppgitt et signifikansnivå (Cohen et al., 2018). Forskjellen mellom gruppene

[17] Delvis hentet fra prosjektskisse

kan vise seg å være signifikant, dersom denne p-verdien er lavere enn 0,1. I forskning analyser man med et fastsatt konfidensintervall på 90%, 95% eller 99%, som gir analyser med 0,1, 0,05 eller 0,01 i signifikansnivå. Dette avgjøres på forhånd og i samfunnsvitenskap er det ikke behov for like strenge konfidensintervall som i forskning på virkninger av medisin hvor konfidensintervallet ofte settes til 99% (Cohen et al., 2018). Jeg har satt konfidensintervallet til 90%, dette medfører at jeg vil få flere analyser som gir funn som er signifikante, enn ved høyere konfidensintervall. I samfunnsvitenskap er man ikke like streng på konfidensintervallet og signifikansnivå på under 0,1 er generaliserbare, bare hvis utvalget er representativt for populasjonen. Det er vanlig å ha satt en nullhypotese for variabelen man ønsker å gjøre en t-test på, og hvis p-verdien er høyere enn 0,1 beholdes nullhypotesen.

Dette prosjektet vil i større grad være en utforskning av flere deler av kunnskapshullet som eksisterer. Av denne grunn valgte jeg å ta i bruk forskningsspørsmål, istedenfor å måtte sette veldig mange nullhypoteser som skulle testes (Cohen et al., 2018). P-verdien vil ga dermed en indikasjon på om forskjellen mellom de to gruppene er statistisk signifikant eller at det muligens skyldes tilfeldigheter. En t-test som har et lavt signifikansnivå vil ha så lav sannsynlighet for å gi et ekstremutfall, som vil tyde på at forskjellen mellom de to gruppene ikke skyldes en tilfeldighet, men må ha en årsak (Cohen et al., 2018). Denne p-verdien kan enten være 1-sidig eller 2-sidig. De forskjellige p-verdiene brukes dersom jeg har en antakelse eller teori som tilsier at forskjellen mellom de to gruppene går i en retning (Cohen et al., 2018). En t-test i SPSS gir to forskjellige verdier for «equal variances assumed» og «eqaul variances not assumed». Jeg har gjennom all analyse tatt i bruk at jeg ikke kan anta en varians, ettersom jeg har ingen teori eller antagelser om at det vil være ulike variasjoner mellom gruppene.

Korrelasjonsanalyse skiller seg fra t-test ved at den ikke ser på forskjellen på en variabel innenfor to grupper, men ser på samvariasjonen mellom to variabler (Gleiss & Sæther, 2021). Tallet som gir en verdi for korrelasjonen mellom variablene kalles Pearsons r og kan ha verdier mellom -1 og 1. Desto nærmere Pearsons r er 0, desto mindre er korrelasjonen. Når verdien for Pearsons r er større enn 0, er samvariasjonen positiv. Eksempelvis vil en person som er lav sannsynligvis ha små føtter, og en person som er høy vil sannsynligvis ha store føtter. Altså en høy verdi av variabel X, vil sannsynligvis medføre en høy verdi for variabel Y. Derimot når Pearsons r er negativ vil en høy verdi av variabel X, sannsynligvis medføre en lav verdi for variabel Y. En person som har røyket et høyt antall sigaretter, variabel X, vil det medføre en lav forventet levealder, variabel Y (Cohen et al., 2018). Når Pearsons r nærmer

seg ±1, vil punktene plottet i et grafvindu avvike mindre fra en lineær graf. Dersom Pearsons r = 1, vil man dermed kunne gi en eksakt Y-verdi bare ved å vite X-verdien, og omvendt. I likhet med t-tester gir korrelasjonsanalyse en p-verdi, målet på hvor signifikant korrelasjonsanalysen er. I korrelasjonsanalysen for dette prosjektet er den også satt til 0,1. En tommelfingerregel bruk av Cohen et al. (2018) ved korrelasjonskoeffisienten er slik:

svak korrelasjonskoeffisient, $r \in [0.1, 0{,}3>$

medium korrelasjonskoeffisient, $r \in [0.3, 0{,}5>$

sterk korrelasjonskoeffisient, $r \in [0.5, 1]$.

3.4 Studiens kvalitet

En del av ansvaret for prosjektet er å reflektere rundt kvaliteten på forskningsarbeidet (Gleiss & Sæther, 2021). Begrepene reliabilitet og validitet brukes både i kvantitativ og kvalitativ forskning, med noen forskjeller i hvordan validiteten vurderes i de to tilnærmingene (Cohen et al., 2018). Det er vanskelig å måle reliabilitet og validitet, men det viktigste er at det gjøres tydelig hvilke valg som er gjort for å styrke kvaliteten, og i et masterprosjekt må man ofte gjøre valg som vil svekke kvaliteten på forskningen (Capraro et al., 2019). Dersom denne forskningen skal være til nytte for å fylle en del av kunnskapshullet, er det essensielt at andre forskere kan lese og vurdere prosjektets kvalitet. Positivisme og konstruktivisme er to former for vitenskapsteoretiske tradisjoner som har ulikt syn på hvordan kvaliteten av forskning skal vurderes, og mye av forskning i samfunnsvitenskap ligger et sted på spekteret mellom disse ytterpunktene. Dette prosjektet er gjennomført med en kvantitativ tilnærming som ofte ses i lys av positivisme, ved bruk av instrumenter som kan gi en tallverdi på kvaliteten av funn (Gleiss & Sæther, 2021).

3.4.1 Reliabilitet

Reliabilitet viser til kvaliteten på forskningsprosessen, ordet pålitelighet er et hverdagsord som ofte brukes om reliabilitet (Gleiss & Sæther, 2021). Gjennom et positivistisk forskningssyn etterstreber man at datamaterialet skal være så lite påvirket at måten det er samlet inn på og hvem som har samlet det inn. En mulig svekkelse av reliabilitet i spørreundersøkelser med kvantitativ tilnærming er utformingen av spørsmålene og svaralternativ. Dette er diskutert tidligere i oppgaven i kapittel 3.2.1, men det står sentralt at spørsmålene formuleres slik at de forstås på samme måte av respondentene og at de føler at de

lukkede svaralternativene passer med hva de hadde svart ved et åpent svaralternativ. For å sikre god relabilitet redegjøres det gjennom metodekapittelet hvordan min posisjon påvirker innsamlingen av data, og hvordan jeg prøver å minske disse faktorene. I motsetning vil et prosjekt med konstruktivistisk forskningssyn verdsette hvordan forskerens posisjon påvirker forskning. I konstruktivismen er posisjonalitet en forutsetning for bias, og forskning vil bli forskjellig, dersom to forskere gjennomfører samme forskningsprosjekt (Cohen et al., 2018; Gleiss & Sæther, 2021).

3.4.2 Validitet

Validiteten sier noe om kvaliteten på datasettet og gyldigheten av funnene og konklusjonene som gjøres (Capraro et al., 2019; Gleiss & Sæther, 2021). I positivismen er målet med forskning at den gjenspeiler verden slik den er, som vil medføre høy validitet. Det ligger til grunn at metoden er en egnet måte å besvare problemstillingen. Begrepsvaliditet handler om forskeren faktisk måler det hen ønsker å måle, dette vil innebære en operasjonalisering av teoretiske begrep. Eksempelvis om spørsmålene og svaralternativene i spørreskjemaet for prosjektet måler ett teoretiske begrepet, eller ved lav begrepsvaliditet er det kanskje flere teoretiske begrep som måles i ett spørsmål (Cohen et al., 2018). «Begrepsvaliditet handler som å vurdere hvor godt overlapp det er mellom fenomenet man undersøker, og måten det har blitt operasjonalisert» (Gleiss & Sæther, 2021). En måte man kan styrke kvaliteten på forskningen er ved å gjenbruke spørsmål fra tidligere forskning som har høy begrepsvaliditet i seg selv. Jeg har brukt spørsmål fra tre forskningsartikler av Rich (2020), Misfeldt et al. (2019) og Korhonen et al. (2022) hvor jeg har vurdert begrepsvaliditeten i deres forskning som egnet og gjort tilpasninger til dette prosjektet.

Begrepet validitet kan også deles i indre og ytre validitet (Cohen et al., 2018). Indre validitet vurderes ved å reflektere rundt andre mulige årsaker på funn i dataanalysen. Den indre validiteten er høy dersom det er godt reflektert om sammenhengen mellom uavhengige variabler og avhengige variabler kan skyldes påvirkning av andre variabler eller andre forklaringer (Cohen et al., 2018).

Muligheten for generalisering kalles ytre validitet. Høy ytre validitet vil medføre at funnene og konklusjonene som blir gjort av utvalget, kan generaliseres, altså at det gjelder en hel populasjon. I dette prosjektet er det drøftet tidligere hvordan utvalget er et bekvemmelighetsutvalg i kapittel 3.2.2 (Danielsen, 2013). Basert på hvordan prosjektet ikke hadde et sannsynlighetsutvalg var det vanskelig å kunne gjøre en generalisering, selv om

funnene viste seg å være signifikant. Det kan være interessant for kommende forskning å se på en lik problemstilling, metode og teori med et utvalg som gir muligheter for generalisering. Hvis dette skulle ha blitt gjort, kan det hende at enten forskeren får samme resultat, eller noe helt annet. Denne svakheten i mitt prosjekt gjør det ugunstig å generalisere for populasjonen jeg forsker på.

3.5 Forskningsetiske aspekter

Grunnleggende forskningsetiske prinsipp som informert samtykke, konfidensialitet og å unngå negative konsekvenser gjelder all forskning, også spørreundersøkelser (Gleiss & Sæther, 2021). Informert samtykke var viktig ettersom spørreskjemaene ble sendt ut digitalt, det var dermed viktig å informere om dette i mailene som ble sendt ut og gruppene det ble lagt ut på[18]. Spørreskjemaet ble sendt ut gjennom nettskjema.no, hvor en enkelt kunne informere respondenter om at de samtykket til at deres svar ble brukt i forskning. Spørreskjemaet hentet ikke inn noen sensitive personopplysninger. Innhenting av sensitive personopplysninger kan medføre færre respondenter, både før og i løpet av besvarelsen av spørreskjemaet (Cohen et al., 2018). Eksempelvis kan spørsmål eller påstander som måler variabler som navn, adresse, helseforhold, religion, arbeidsplass og så videre føre til at forskningsdeltakeren ikke føler seg trygg på forskeren. Dette er også reflektert rundt i underkapittelet om utvalg, 3.2.2.

Variablene som ble valgt til å måles i spørreskjemaet medfører at dette prosjektet var konfidensialitet nesten ikke et problem, fordi det vil være vanskelig for andre å identifisere personer basert på de få opplysningene som kreves[19]. Jeg valgte derfor å ha alle variablene om personopplysninger som alder, kjønn, hvilket trinn lærere underviser på, hvor mange år en har undervist matematikk og deltakelse i kurs om programmering, på første side. Dermed vet forskningsdeltakeren hvilke personopplysninger som brukes, og resten av spørreskjemaet målte respondentens meninger og tanker. Personopplysningene som ble hentet inn i spørreskjemaet medførte at prosjektet ikke måtte meldes inn til et personvernombud. Disse personopplysningene var en viktig del av prosjektet og en del av hensikten med forskningen.

[18] Hentet fra prosjektskisse

[19] Hentet fra prosjektskisse

Det var viktig at verdier for et spørsmål eller påstand var mulig å svare, slik at respondenten fikk muligheten til å svare det hen mente. Siden en av utfordringene med spørreundersøkelser er at det er lite rom for å rette opp i feil når spørreskjemaet er sendt ut (Gleiss & Sæther, 2021)[20].

3.6 Feilkilder

Kapittelet om feilkilder vil fungere som en oppsummering av metoden og hvilke mangler og utfordringer som kunne vært løst på bedre måter, når jeg som student har skrevet min første masteroppgave. Utvalget for oppgaven var nok den største feilkilden for prosjektet. Et utvalg som ikke representerer populasjonen, er i kvantitativ forskning uheldig for mulighetene til å generalisere interessant funn og presiseres gjennom oppgaven.

Jeg ville gjort noen få endringer i spørreskjemaet som ble sendt ut, etter jeg ser hvor store mengder data som ble samlet inn. Spørreundersøkelsen gikk ikke mye i dybden på spesifikke tema, samtidig kan dette være vanskelig å gjøre på et felt som ikke er forsket mye på. Spørreskjema har også en median svartid på 13 minutter, og jeg antar at jeg kunne hentet inn nok informasjon fra respondenter med en svartid på rundt 5 minutter. Jeg ville endret svaralternativene på «Hvor gammel er du?» og «Hvor mange år har du undervist i skolen?» fra muligheten til å svare hvilket intervall som passet respondenten, til et felt hvor de kunne fylle inn tall. Jeg ville dermed hatt variabelen på forholdstallsnivå, som i seg selv ikke er så viktig, men verdien for deres alder ville gitt en korrelasjonsanalyse som var litt mer presis. Dette ville også gitt muligheten til å dele gruppene i to, dermed ville t-testen hatt like mange respondenter i gruppene. T-testene måtte deles inn i over og under 40 år, fordi dette var nærmest en halvert fordeling.

I analysen har jeg drøftet funn opp mot det teoretiske rammeverket som er satt. Dersom analysen bærer preg av mangel på kvalitet i form av reliabilitet eller validitet, drøftes effekten av en slik feilkilde.

[20] Hentet fra prosjektskisse

4 Analyse og diskusjon

Analysekapittelet vil starte med beskrivelse av de testene som har blitt gjort, så vil jeg drøfte hvor gyldig funnene er, hva som kan være mulige årsaker og hva som kan være mulige begrensninger. Analysen vil se på funn som er enten i samsvar eller konflikt med tidligere forskning og det teoretiske rammeverket. Majoriteten av bivariat analyse er gjort ved t-tester, hvor gruppene er delt i alder, 19 respondenter under 40 år og 25 respondenter over 40 år. Den andre gruppeinndelingen er om respondentene har deltatt på kurs eller tatt etterutdanning som er relevant for programmering, hvor 23 svarte ja, og 22 svarte nei. T-testene som er gjort basert på alder kunne vært gjennomført som korrelasjonsanalyser dersom jeg hadde hatt respondentenes alder som en variabel på forholdstallsnivå. Analysekapittelet vil i grove trekk følge samme rekkefølge som teorikapittelet og underkapitlene dekker et forskningsspørsmål hver.

For ordens skyld viser jeg til problemstillingen som er satt for prosjektet:

> **Hva synes matematikklærere på mellomtrinnet om å inkludere programmering i matematikkfaget etter fagfornyelsen?**

Tabellene i analysen er skyggelagt avhengig av hvilket konstrukt påstandene måler. Dersom antall respondenter er fulltallig, er tallet for N=… markert i fet skrift.

Påstander som ikke er inkludert i analysen er enten formulert på en slik måte de har svekket reliabilitet eller at jeg ikke fant noen nytte med å analysere dem i t-test eller korrelasjonsanalyse.

4.1 Programmering

Dette underkapittelet vil ta for seg, programmering i skolen, blokkprogrammering og sammenhengen mellom programmering og matematikk. Jeg har delt underkapitlene opp etter hvilket forskningsspørsmål drøftingen av analysen har som hensikt å besvare.
Forskningsspørsmål 1 var:

> Hvilket syn har lærerne på undervisning av programmering i matematikk?

4.1.1 Programmering i skolen

Tabell 2 viser til resultater av gjennomsnitt og standardavvik i læreres grad av enighet i henhold til påstandene om programmering i skolen. Påstand 1 til 4 og 9 til 11 er besvart av alle respondenter og har høye gjennomsnitt (4.53-4.80) med lave standardavvik (0.495-0.726). Påstand 5 og 6 tar opp samme konstrukt som påstand 7, men med store forskjeller i gjennomsnitt (3.24-4.48) og standardavvik (0.698-1.179). Hvor jeg oppdaget standardavvik sjekket jeg alle variablene med en t-test for gruppene alder og etterutdanning, dette ga ofte svar på hvilken uavhengig variabel som forårsaket avvikene.

Hva som kan skyldes forskjellene i gjennomsnitt og standardavvik presenteres i tabell 3. Tabellen viser en oversikt over hvordan svarene til respondentene er fordelt i aldersgruppene over 40 år og under 40 år. Respondentene svarte på en Likert-skala fra 1-5 hvor enig de er i påstanden «Programmering åpner for at elever ukritisk kopierer hverandres verk». T-testen viser at respondenter over 40 år var nært nøytral til denne påstanden (2.96) med et stort avvik (1.160), men respondenter under 40 år var i større grad enig i påstanden (3.78) med mindre spredning i svarene (0.732). På forhånd av analysen hadde jeg ingen teori eller forventning av analysen skulle ha en retning, dermed brukte jeg tosidig p-verdi for å bedømme i hvilken grad analysen kan sies å være signifikant. P-verdien for denne analysen er 0.012 som vil si at det er veldig liten sannsynlighet for at funnet skyldes tilfeldigheter og funnet er signifikant med over 98,8% sikkerhet. Påstand 5, «programmering er egnet for å lære elever om kritisk kildebruk» og påstand 7, «programmering åpner for at elever kan kopiere og lære av hverandres verk» har mindre forskjeller i gjennomsnittene (0.21 og 0.19) med p-verdier på 0.570 og 0.357, og er dermed ikke signifikant. Altså kan det ikke sies med sikkerhet at forskjellen mellom de over 40 år og under 40 år ikke skyldes tilfeldigheter, og hadde jeg satt en nullhypotese for disse variablene hadde den blitt bevart.

Jeg vil ikke lenger vise til funn som ikke er signifikante i analysen, med mindre det burde drøftes hvorfor en mangel på signifikans kan være et interessant funn. Funn som er signifikante er interessant å drøfte fordi den statistiske analysen viser at det er høyst sannsynlig at forskjellen eller korrelasjonen ikke skyldes tilfeldigheter, fordi det har liten hensikt å diskutere tilfeldigheter.

Tabell 2: Konstrukter, programmering i skolen

Programmering i skolen	N	Gjennomsnitt	Standardavvik
1. Programmering kan gjøre skolearbeid mer interessant for elevene	**45**	4.53	.726
2. Programmering er egnet for at elever får utvikle kreativiteten sin	**45**	4.73	.495
3. Programmering er egnet for at elever får oppleve skaperglede	**45**	4.80	.505
4. Programmering er egnet for at elever får utvikle problemløsningsegenskapene sine	**45**	4.73	.618
5. Programmering er egnet for å lære elever om kritisk kildebruk	41	3.24	1.179
6. Programmering åpner for at elever ukritisk kopierer hverandres verk	42	3.31	1.070
7. Programmering åpner for at elever kan kopiere og lære av hverandres arbeid	44	4.48	.698
9. Innenfor programmering er det viktig at læreren kan utvide oppgaver for elever som er ferdig	**45**	4.78	.420
10. Programmering er egnet for alle elever å lære	**45**	4.78	.670
11. De viktigste konseptene innenfor programmering kan forstås av elever på mellomtrinnet	**45**	4.76	.529

Tabell 3: T-test, konstrukt kopiering, gruppe alder

Kopiering/alder	Alder	N	Gjennom snitt	Standard avvik	Tosidig P-verdi
5. Programmering er egnet for å lære elever om kritisk kildebruk	Over 40	24	3.33	1.167	.570
	Under 40	17	3.12	1.219	
6. Programmering åpner for at elever ukritisk kopierer hverandres verk	Over 40	24	**2.96**	**1.160**	**.012**
	Under 40	18	**3.78**	**.732**	
7. Programmering åpner for at elever kan kopiere og lære av hverandres arbeid	Over 40	**25**	4.56	.768	.357
	Under 40	**19**	4.37	.597	

På tross av at programmering nylig er innført i læreplanene for matematikk, er lærerne enig i påstandene 1-4 som ofte trekkes fram som viktige årsaker til å lære programmering (NOU, 2013: 2; Resnick et al., 2019). Programmering kommer høyst sannsynlig til å bli en viktig del av deres jobb og dannelse, ikke bare i form av å skape programmer selv, men også å ha en forståelse av den digitale verden vi lever i (Cheng, 2019). Fordelene som fremmes for å drive programmering vil også drøftes videre i hvilke nøkkelbegreper og ferdigheter elever utvikler ved å trene på algoritmisk tenkning. Den nære sammenhengen mellom programmering og algoritmisk tenkning kan tyde på et behov for kompetanse i undervisning om både programmering og algoritmisk tenkning, slik at elevene blir interessert, utvikler kreativitet, opplever skaperglede og blir dyktige problemløsere (Bocconi et al., 2016; Laurent et al., 2022).

T-testen i tabell 3 hjelper med å forklare forskjellene og avvikene i tabell 2. Lærerne over og under 40 år er enig i ulik grad om programmering åpner for at elever ukritisk kopierer hverandres verk. Denne forskjellen kan muligens skyldes at yngre lærere er mere erfaren i bruk av digitale midler, og ser hvilke muligheter elever har for å kopiere hverandres verk. En sentral del av kulturen i programmering er å ta i bruk og fikle med eksisterende programmer, men dette kan være krevende for elever på mellomtrinnet (Kaufmann & Stenseth, 2021). Elevene har muligheter for å kopiere blokkene som brukes av medelever eller et problem som har oppstått i nyere tid, få et KI-program til å skape en løsning. Lærerne er også nøytral til programmering for å lære om kritisk kildebruk, sannsynlig grunnet hvor enkelt det er å kopiere hverandres løsninger. Noen program er også lagt opp slik at de bare har én løsning, som kan skape flere utfordringer. Lærerne er enig i at programmering er egnet for å lære av hverandres arbeid derimot. En viktig del av lærerens rolle når elevene jobber med programmering er å gi elevene gode fremgangsmåter for å utvikle sin egen programmeringskompetanse. Dette kan de gjøre ved å søke på nett, eller diskutere med medelever (Kaufmann & Stenseth, 2021; Resnick, 2009). Programmering kan dermed være en arena hvor elevene får diskutert hverandres løsninger, og læreren får som rolle å gi elevene videre utfordringer som passer dem.

Lærerne er enig i påstandene 10 og 11 og mener at programmering er egnet for alle elever å lære og at de viktigste konseptene kan forstås av alle elever. Carbera (2019) peker på denne vanlige misoppfatningen at programmering ikke er egnet for alle elever å lære. Fokuset på hvilke egenskaper elevene utvikler og hvor viktig det er at eleven har forståelse av digitale midler de tar i bruk, fremmer at programmering skal være en del av læreplanen i matematikk

på mellomtrinnet (NOU, 2018: 2; Resnick, 2019). Lærerne mener altså at programmering er viktig for alle elever. Programmering kan også ha muligheten til å fange opp elever som ikke viser så stor interesse for matematikk, men bruken av programmering og digitale midler gir et konkret eksempel på hvor viktig den grunnleggende matematikken vil være for elevene.

4.1.2 Blokkprogrammering

Tabell 4 viser til resultater av univariat analyse av konstrukter innenfor blokkprogrammering. Påstand 1 og 2 er besvart av nesten alle respondenter (N=44 og N=42) med høye gjennomsnitt (4.80 og 4.79) og lave standardavvik (0.509 og 0.470). Påstand 6 til 9 måler samme konstrukt med varierende gjennomsnitt (4.07-4.50) og standardavvik (0.665-0.985), og jeg fant analyser som ga signifikante verdier for t-tester og korrelasjonsanalyse som kan vise til disse forskjellene.

En t-test med den uavhengige variabelen alder viser forskjeller i påstand 6, «Tilbakemeldinger fra programmer som elevene jobber med gjør blokkprogrammering mer egnet, enn tekstprogrammering». Respondenter over 40 år har et gjennomsnittlig høyere grad av enighet til påstanden (4.29) sammenlignet med de under 40 år (3.76). Analysen er gjort med en tosidig p-verdi på 0.092, det vil si at funnet akkurat er innenfor grensen for å være signifikant med et konfidensintervall på 90%. Innenfor samme tema viste en korrelasjonsanalyse sammenhenger mellom lærernes tilgang på ressurser og tre påstander om blokkprogrammering. Det er store forskjeller mellom påstand 3, «Blokkprogrammering er egnet fordi oppgaver kan løses med lav inngangsterskel» og påstand 4, «Blokkprogrammering er egnet fordi oppgaver kan løses med stor takhøyde».

Korrelasjonsanalysen gir en Pearsons r verdi på 0.063 og et signifikansnivå på 0.686 for påstand 3, den har dermed, både en mindre enn svak korrelasjonskoeffisient og en tosidig p-verdi som gjør den ikke signifikant. Påstand 4 som måler det motsatte av påstand 3, gir en medium korrelasjonskoeffisient på 0.409 og en tosidig p-verdi på 0.008, dermed er funnet er signifikant. Lærernes tilgang på ressurser korrelerte også med påstand 5, «Blokkprogrammering er mer sosialt, fordi løsningen på oppgaver kan enkelt deles med andre over nett» og påstand 7, «Blokkprogrammering gjør det mindre sannsynlig at elevene blir sittende fast med en oppgave», hvor begge hadde en medium korrelasjonskoeffisient på 0.441 og 0.436 og tosidige p-verdier på 0.006 og 0.004, disse korrelasjonene er dermed med over 99% sikkerhet ikke på grunn av tilfeldigheter.

Tabell 4: Konstrukter, blokkprogrammering

Blokkprogrammering	N	Gjennom snitt	Standard avvik
1. Blokkprogrammering gjør det mulig å undervise programmering på mellomtrinnet	44	4.80	.509
2. Blokkprogrammering gjør det enklere for elever som tidligere ikke har vist interesse for tekstprogrammering å lære seg programmering	42	4.79	.470
6. Tilbakemeldinger fra programmet som elevene jobber med gjør blokkprogrammering mer egnet enn tekstprogrammering	41	4.07	.985
7. Blokkprogrammering gjør det mindre sannsynlig at elevene blir sittende fast med en oppgave	43	4.37	.725
8. Blokkene gjør det enklere for å finne en mulig løsning på en oppgave	44	4.50	.665
9. Elevene bør lære seg blokkprogrammering fordi det vil senere være enklere å lære seg tekstprogrammering	40	4.45	.876

Tabell 5: T-test, konstrukt fremgangsmåter, gruppe alder

Blokkprogrammering /alder	Alder	N	Gjennom snitt	Standar davvik	Tosidig P-verdi
6. Tilbakemeldinger fra programmet som elevene jobber med gjør blokkprogrammering mer egnet, enn tekstprogrammering	Over 40	24	**4.29**	**.806**	**.092**
	Under 40	17	**3.76**	**1.147**	

Tabell 6: Korrelasjonsanalyse, tilgang på ressurser, blokkprogrammering

Korrelasjonsanalyse		3. Lav inngangsterskel	4. Stor takhøyde	5. BP Sosialt	7. Ikke stuck
9. Tilgang på ressurser	Pearsons r	.063	**.409**	**.441**	**.436**
	Tosidig P-verdi	.686	**.008**	**.006**	**.004**
	N	43	41	38	42

Tekstprogrammering har før vært et stort hinder for å lære elever programmering i grunnskolen (Benton, 2017; Cheng, 2019; Laurent; 2022; Resnick, 2019). Vanskene med syntaksfeil og tiden det tar å forstå hvordan et program må bygges opp, tar så mye tid å lære at det har vært utilgjengelig for elever i grunnskolen å lære seg programmering. Blokkprogrammering er et visuelt programmeringsspråk som unngår disse vanskene, og gir elever muligheter til å programmere. Lærerne i spørreundersøkelsen er enig i at blokkprogrammering gjør det mulig for elever å lære seg programmering, og elevene som ikke har vist interesse for tekstprogrammering, kan lære seg å programmere. Variabler, vilkår, løkker og funksjoner er alle begrep som er brukt i læreplanen LK20 som elevene skal lære om (Kunnskapsdepartementet, 2020). Blokkene som brukes i blokkprogrammering gir konkrete eksempler på hvordan disse blokke kan brukes til å lage variabler, sette vilkår og bruke løkker som gjentar funksjoner.

Andre fordeler som gjør blokkprogrammering egnet for mellomtrinnet er at elevene ofte kan få en illustrasjon av som skjer når de endrer på blokkene ved å kjøre programmet, og elevene vil få en visuell feedback. Lærerne over 40 år er i større grad enig i at tilbakemeldinger fra programmet gjør det mer egnet en tekstprogrammering. En mulig forklaring på dette kan være at lærerne over 40 år har erfaring av hvor vanskelig tekstprogrammering kan være. Lærerne over 40 år kan ha erfaring med bruk av programmet Logo som ble brukt i undervisning før slutten av 1900 tallet (Cheng, 2019), men dette er bare en mulig forklaring.

Muligheten til å skape og være kreativ er en stor fordel med blokkprogrammering. Det er en rekke programmer som i dag er designet for at det skal brukes av barn i undervisningssammenheng. Programmene gjør at elever kan samarbeide om felles prosjekter, samt at programmene er tilpasset elevenes interesser, som tidligere var en utfordring med programmet Logo (Cheng, 2019; Resnick, 2019).

Det vises å være en sammenheng mellom lærerne som har tilgang på ressurser, og en rekke andre avhengige variabler om blokkprogrammering fra spørreskjemaet. LIST står for lav inngangsterskel og stor takhøyde og er ofte ikke koblet opp mot programmering, men mange lærere er kjent med dette begrepet, og den lave inngangsterskelen nevnes som en fordel for å drive med blokkprogrammering i undervisning for barn (Resnick, 2019). Lærerne var i større grad enig i at programmering var egnet grunnet lav inngangsterskel, enn for stor takhøyde (vedlegg 1). Korrelasjonsanalysen viser til medium korrelasjonskoeffisient for stor takhøyde, og ingen signifikante funn for lav inngangsterskel. Dette kan bety at lærerne som har større tilgang til ressurser for programmering, vil se den store takhøyden som programmering åpner for, som også kan skyldes mere avansert utstyr.

Det finnes mye gratis ressurser for programmering og blokkprogrammering, men det kan hende at disse i større grad vektlegger den lave inngangsterskelen som Resnick (2019) nevner. Tilgangen på ressurser fører sammenhenger også med lærernes oppfatning av at blokkprogrammering er mere sosialt og at elevene ikke setter seg fast i oppgaver. Produkter innenfor blokkprogrammering som gir muligheter for delinger av løsninger og tilbakemeldinger fra programmet kan forklare denne sammenhengen. Hvorfor lærerne ikke har tilgang på ressurser kan variere, Pörn (2021) og Bråting et al (2021) peker på vansker svenske og finske lærere har utrykt om ressurser de har tilgang til, og det er et annet spørsmål om de er kompetent nok til å gjenkjenne gode ressurser. Det er stor spredning i respondenter fra min spørreundersøkelse henter sine ressurser fra, forskning (42.2%), læreverk (57.8%), kolleger (53.3%), søk på nett (48.9%), grupper på sosiale medier (35.6%), ved å lage det selv (44.4%) eller annet (8.9%). Denne spredningen er positiv i form av at lærerne har stor frihet i å velge hvor de får tak i sine ressurser, men også negativ i form av store variasjoner i kvalitet og undervisningsmåter for elevene. (Bråting et al., 2021; Pörn, 2021).

Blokkprogrammering tillater en tidlig innsats i elevenes programmeringskompetanse, og lærerne er også enig i at det er viktig å la elevene lære seg blokkprogrammering, for og senere kunne lære seg tekstprogrammering, og dette stemmer overens med forskningen til Shute et al. (2017).

4.1.3 Programmering og matematikk

Tabell 7 viser til resultater av univariat analyse av konstrukter innenfor lærernes syn på sammenhengen mellom programmering og matematikk. Påstandene 1 til 6 er besvart av nesten alle (min. N=42) og det er variasjoner innenfor samme konstrukt, både i gjennomsnitt (4.02-4.58) og standardavvik (.625-9.22). Påstand 12 og 13 måler variabler innenfor samme konstrukt, men det er stor forskjell i gjennomsnitt (4.36 og 2.71) og standardavvik (.727 og 1.218). Jeg fant ingen t-tester som kunne vise til en signifikant forskjell i gruppene alder og etterutdanning, men korrelasjonsanalyse ga interesse funn.

Tabell 8 viser en korrelasjonsanalyse som er gjort for å utforske sammenhengen mellom den uavhengige variabelen med påstanden, «Sammenhengen mellom programmering og matematikk er tydelig for meg» og en rekke avhengige variabler som måler påstander om lærernes oppfattede sammenheng mellom programmering og matematikk (vedlegg 1):

1. Programmering er et nyttig verktøy i matematikk
3. Programmering gjør at elever oppfatter matematikk mer nyttig utenfor matematikkfaget
4. Programmering gjør at elever utvikler algoritmisk tenkning
5. Programmering er et verktøy for å lære seg matematikk
7. Programmering er en egnet måte å uttrykke matematikk på
8. Man drar nytte av matematisk kompetanse når man skal lære seg å programmere
10. Man kan forstå sentrale konsepter i matematikken gjennom programmering
11. Man kan få en bedre forståelse av prosedyrer og algoritmer gjennom programmering
12. Elever vil utvikle kompetanse på tvers av programmering og matematikk, altså vil kompetansen vil være overførbar

Påstand 1 har en sterk korrelasjonskoeffisient på 0.674 og en tosidig p-verdi på <0.001, ellers har påstandene medium korrelasjonskoeffisient med Pearsons r verdier fra 0.311, opp til 0.460 og tosidige p-verdier fra 0.045 til 0.002 som gjør alle disse funnene signifikant.

Tabell 7: Konstrukter, programmering og matematikk

Programmering og matematikk	N	Gjennomsnitt	Standardavvik
1. Programmering er et nyttig verktøy i matematikk	43	4.58	.626
2. Programmering gjør at elever engasjerer seg mer for matematikk	42	4.05	.909
3. Programmering gjør at elever oppfatter matematikk mer nyttig utenfor matematikkfaget	44	4.02	.902
4. Programmering gjør at elever utvikler algoritmisk tenkning	44	4.43	.625
5. Programmering er et verktøy for å lære seg matematikk	44	4.18	.896
6. Programmering i seg selv, er matematikk	43	4.23	.922
12. Elever vil utvikle kompetanse på tvers av programmering og matematikk, altså vil kompetansen vil være overførbar	42	4.36	.727
13. Elever vil kunne se likhetene i programmering og matematikk	**45**	2.71	1.218

Tabell 8: Korrelasjonsanalyse, sammenheng programmering og matematikk er tydelig, konstrukter i programmering og matematikk

Korrelasjonsanalyse		1.	3.	4.	5.	7.	8.	10.	11.	12.
9. Sammenhengen mellom matte og programmering er tydelig	Pearson s r	**.674**	**.349**	**.458**	**.460**	**.346**	**.424**	**.311**	**.383**	**.311**
	Tosidig P-verdi	**<.001**	**.020**	**.002**	**.002**	**.025**	**.004**	**.043**	**.010**	**.045**
	N	43	44	44	44	42	44	43	44	42

Sammenhengen mellom matematikk og programmering er for øyeblikket litt utydelig, er det ment som er verktøy for å jobbe med matematikk eller er programmering i seg selv matematikk? Laurent et al. (2022) mener det er lite som tyder på en kobling mellom matematikk, mens Kaufmann & Stenseth (2021) konkluderer med at det kan være en kobling mellom matematikk og programmering, men at det krever lærere som evner å lære elevene denne sammenhengen.

Fra spørreundersøkelsen er lærerne i størst grad enig i bruken av programmering som et verktøy i matematikken og hvordan programmering kan hjelpe med å tenke algoritmisk. Påstandene er basert på Kilhamn et al. (2021) sine kategorier for sammenhenger mellom programmering og matematikk. Kategoriene er ikke motstridende mot hverandre, og respondentene plasseres gjennomsnittlig mellom «delvis enig» og «enig». Lærerne ser nytten av programmering som et verktøy for å kunne gjøre matematikk og tydeliggjøre matematiske konsepter. Dyktige programmerere kan bruke digitale midler for å lære matematiske konsepter på andre måter, enn hva som har vært mulig i tidligere læreplaner. Dette samstemmer med det Kilhamn et al. (2021) beskriver i kategori 1 og 4. Ettersom programmering åpner for nye måter å jobbe med matematikk, medfører dette at elevene utvikler andre kunnskaper og ferdigheter. Dekomponering, algoritmebehandling, å holde ut og *debugging* er egenskaper elevene ikke fikk lært seg før programmering ble en del av matematikk. Lærerne mener at programmering gjør at elever blir bedre algoritmiske tenkere, som er kategori 3 i Kilhamn et al. (2021) sin forskning. Til slutt viser kategori 2 til at elevene blir mere motivert og engasjert av programmering og matematikk. Dette kan skyldes av matematikk har tradisjonelt sett vært undervist på en ensidig måte. Elevene kan se nytten av matematikken de lærer utenfor selve matematikkfaget, og respondentene er også delvis enig i dette.

Respondentene tror elever vil utvikle kompetanse på tvers av matematikk og programmering, men at de vil ha vansker med å se likhetene selv. Laurent et al. (2021) peker på vanskene som respondentene kanskje også har opplevd. For å se likhetene mellom programmering og matematikk er det en rekke komplekse generaliseringer som må gjøres, som elever på mellomtrinnet kanskje er for unge for. Det er mulig at dyktige lærere som har en god forståelse for sammenhengen mellom programmering og matematikk kan hjelpe elevene å oppfatte disse likhetene mellom fagfeltene (Kaufmann & Stenseth, 2021).

Korrelasjonsanalysen i tabell 8 tyder på at lærere som selv synes sammenhengen mellom matematikk og programmering er tydelig, i større grad synes at de evner å lære det bort til elevene sine. Respondentene fra min spørreundersøkelse er mere enig i påstandene, enn de svenske lærerne fra forskningen til Misfeldt et al. (2019), men et stort problem med å trekke denne konklusjonen er utvalget for spørreundersøkelsen, og tydeliggjøres kanskje av korrelasjonsanalysen. Kaufmann & Stenseth (2021) peker på at koblingen mellom programmering og matematikk kan være sterk, men det krever dyktige lærere som har en faglig kompetanse i programmering og matematikk. Det hadde vært interessant om det hadde blitt gjort forskning med en kvalitativ tilnærming, hvordan dyktige lærere underviser programmering i matematikkfaget.

4.2 Algoritmisk tenkning

Siden innføringen av begrepet algoritmisk tenkning av Wing (2006), har det vært en rekke endringer i hvilke nøkkelbegrep/kunnskaper og arbeidsmåter/ferdigheter som vektlegges i forskjellige modeller, og hvilke fagfelt begrepet anvendes innenfor. Dette underkapittelet vil se på hvordan norske lærere på mellomtrinnet forstår og vektlegger begrepet algoritmisk tenkning i både programmering og matematikk. Forskningsspørsmålet for dette underkapitlet er dermed:

Hva er læreres forståelse av algoritmisk tenkning?

4.2.1 Nøkkelbegrep og kunnskaper

Tabell 9 viser univariat analyse av nøkkelbegrep og kunnskaper i algoritmisk tenkning. Lærerne svarte sin grad av enighet hvor viktig disse påstandene er å kunne for både programmering (P) og matematikk (M). Påstandene måler forskjellige variabler og er skyggelagt for å gjøre det tydelig hvilke samlevariabler som består av to og enestående variabler. Punkt 1 og 2 måler abstrahering, 3 og 4 måler algoritmebehandling, 5 måler

automatisering, 6 måler dekomponering, 7 og 8 måler generalisering, 9 måler evaluering og 10 måler analysering. Abstrahering er delt i to variabler og påstand 1 mener lærerne er viktigere, enn påstand 2, både i programmering og matematikk (samlet differanse på 0.42 gjennomsnitt). Algoritmebehandling er også delt i to variabler, og begge disse viser til et høyere gjennomsnitt innenfor programmering (samlet differanse på 1.65) og standardavviket er høyere for matematikk (samlet standardavvik på 0.975). Automatisering er påstanden som er minst besvart (N=38 og N=37) med lavest gjennomsnitt (3.05 og 2.49), men med store standardavvik (1.355 og 1.367). Dekomponering som variabel har eksakt samme verdier for matematikk og programmering, både i gjennomsnitt (4.70) og standardavvik (.734). Påstandene for generalisering og analysering er besvart av alle respondenter og har høye gjennomsnitt for samlevariabelen generalisering (4.80-4.93) og variabelen analysering (4.71 og 4.58). Evaluering har høyest gjennomsnitt av alle variablene for både programmering og matematikk (4.95) og lavest standardavvik (0,211), delvis grunnet at det statistisk sett er vanskelig å avvike fra et så høyt gjennomsnitt.

Tabell 10 og 11 viser to t-tester som kan hjelpe med å forklare forskjellene i påstand 3, 4 og 6 fra tabell 9. T-testene ga resultat som viser til signifikante forskjeller for de avhengige variablene for begge uavhengige variabler, alder og etterutdanning. Tabell 10 viser påstand 4 og 6 med fordeling av lærere som har deltatt/ikke deltatt i etterutdanning eller kurs om programmering, mens tabell 11 viser påstand 3, 4 og 6 inndelt i gruppene for alder.

I tabell 10 viser påstand 4 som måler én del av samlevariabelen algoritmebehandling, en forskjell i nytten innenfor matematikk (0.71) som er signifikant med en p-verdi på 0.066. Det er ikke en signifikant forskjell innenfor programmering. Påstand 6 måler variabelen dekomponering og er lik innenfor programmering og matematikk, ettersom lærerne som har tatt etterutdanning synes det er viktigere, enn lærerne som ikke har etterutdanning (2(4.95 og 4.45)) og funnene er signifikant med en p-verdi på 0.026.

Forskjeller i påstand 3 ga signifikante funn for gruppene over 40 år og under 40 år. Påstand 3 måler også algoritmebehandling. Innenfor programmering synes lærerne over 40 at dette er svært viktig (4.96), mens lærerne under 40 ikke verdsetter det like mye (4.32). Denne differansen (0.64) er signifikant med en tosidig p-verdi på 0.006 for programmering. For matematikk er differansen (0.79) også signifikant med en tosidig p-verdi på 0.036. For påstand 4 er det ingen signifikante funn for programmering i likhet med tabell 10. Det er stor forskjell i grad av enighet mellom lærerne over og under 40 år (0.93) og funnet er signifikant

med en p-verdi på 0.018. Påstand 6 ga lignende signifikante funn i tabell 10 og 11 hvor gruppene over 40 år de som har etterutdanning verdsetter dekomponering.

Tabell 9: Konstruktet nøkkelbegrep algoritmisk tenkning, delt i programmering og matematikk for hver påstand

Nøkkelbegrep og kunnskaper		N	Gjennomsnitt	Standardavvik
1. Det er viktig å kunne trekke ut de sentrale delene av et problem	P	44	4.91	.291
	M	**45**	4.93	.252
2. Det er viktig å kunne legge bort unødvendige deler av et problem	P	43	4.51	.798
	M	**45**	4.49	.843
3. Det er viktig å kunne definere og følge instruksjoner steg for steg, mot en løsning	P	**45**	4.69	.793
	M	**45**	4.24	1.190
4. Det er viktig å kunne lage en løsning som kan forstås av et digitalt middel	P	44	4.57	.695
	M	43	3.37	1.273
5. Det er viktig å kunne få et digitalt middel til å minimere det menneskelige bidraget i løsningen av et problem	P	38	3.05	1.355
	M	37	2.49	1.367
6. Det er viktig å kunne dele et hovedproblem opp i mindre deler og løse dem hver for seg	P	44	4.70	.734
	M	44	4.70	.734
7. Det er viktig å kunne gjenkjenne mønster og koblinger mellom forskjellige problemer	P	**45**	4.89	.318
	M	**45**	4.89	.318
8. Det er viktig å kunne bruke kunnskap fra et tidligere løst problem, slik at det enklere kan brukes ved lignende problem	P	**45**	4.80	.588
	M	**45**	4.93	.252
9. Det er viktig å kunne gjøre vurderinger av sitt eget arbeid	P	44	4.95	.211
	M	44	4.95	.211
10. Det er viktig å kunne sjekke hvordan en løsning kunne vært mer effektiv og hva som er begrensninger ved løsningen	P	**45**	4.71	.549
	M	**45**	4.58	.621

Tabell 10: T-test, algoritmebehandling og dekomponering. gruppe etterutdanning

Algoritmisk tenkning nøkkelbegrep/kurs		Kurs	N	Gjennomsnitt	Tosidig P-verdi
4. Det er viktig å kunne lage en løsning som kan forstås av et digitalt middel	P	Ja	**23**	4.65	.412
		Nei	21	4.48	
	M	Ja	**23**	**3.04**	**.066**
		Nei	20	**3.75**	
6. Det er viktig å kunne dele et hovedproblem opp i mindre deler og løse dem hver for seg	P	Ja	22	**4.95**	**.026**
		Nei	**22**	**4.45**	
	M	Ja	22	**4.95**	**.026**
		Nei	**22**	**4.45**	

Tabell 11, T-test, algoritmebehandling og dekomponering, gruppe alder

Algoritmisk tenkning nøkkelbegrep/alder		Alder	N	Gjennomsnitt	Tosidig P-verdi
3. Det er viktig å kunne definere og følge instruksjoner steg for steg, mot en løsning	P	Over 40	**26**	**4.96**	**.006**
		Under 40	**19**	**4.32**	
	M	Over 40	**26**	**4.58**	**.036**
		Under 40	**19**	**3.79**	
4. Det er viktig å kunne lage en løsning som kan forstås av et digitalt middel	P	Over 40	25	4.64	.438
		Under 40	**19**	4.47	
	M	Over 40	25	**3.76**	**.018**
		Under 40	18	**2.83**	
6. Det er viktig å kunne dele et hovedproblem opp i mindre deler og løse dem hver for seg	P	Over 40	25	**4.88**	**.010**
		Under 40	**19**	**4.47**	
	M	Over 40	25	**4.88**	**.010**
		Under 40	**19**	**4.47**	

I form av kunnskap innenfor algoritmisk tenkning er det bare ett av de seks begrepene som markerer seg som betraktelig mindre viktig hos respondentene, automatisering. Samtidig er de neste lavere gjennomsnittene innenfor algoritmebehandling i matematikk, ellers er respondentene nærmere «enig» i gjennomsnitt for alle påstander om hvor viktig disse

nøkkelbegrepene er i programmering og matematikk. Gjennomsnittene tyder mot at respondentene ikke synes det er så viktig å lage løsninger som skal forstås av digitale midler. Både automatisering og algoritmebehandling i matematikk har lave gjennomsnitt, og det er nok sannsynlig at lærerne på mellomtrinnet synes det er viktig at elevene får en forståelse av programmering, matematikk og algoritmisk tenkning før de overlater det manuelle arbeidet til digitale midler. Påstand 5 om automasjon er besvart av N=38 og N=37, noe som kan tyde på at formuleringen av påstanden er noe utydelig, eller at respondentene har vansker for å velge en grad av enighet. Jeg tror påstanden kunne vært formulert på en måte hvor målet med automatisering er gjøre det mulig for et digitalt middel for å bidra, kontra å minimere det menneskelige bidraget, slik det er formulert i spørreskjemaet.

Nøkkelbegrepet abstraksjon er målt i to påstander, og respondentene vektlegger å større grad å trekke ut sentrale deler av et problem, enn å legge bort unødvendige deler. Jeg tror dette kan komme av hvor komplekse oppgaver elever på mellomtrinnet egner å jobbe med. Elevene jobber ofte med isolerte problemer hvor det ikke er et behov for å se bort fra mye informasjon, som kanskje en informatiker har bruk for. Abstraksjon er kjernen i begrepet algoritmisk tenkning, men det kan tenkes at en modell laget for barn i større grad vektlegger påstand 1, enn påstand 2.

Dekomposisjon og generalisering er en sentral del av den originale definisjonen av algoritmisk tenkning og respondentene fra spørreundersøkelsen synes også dette er viktig innenfor algoritmisk tenkning, både i programmering og matematikk (Bocconi et al., 2016; Shute et al., 2017; Wing, 2006). Evaluering et nøkkelbegrep som har mange forskjellige navn i forskning, blant annet, *debugging*, *iteration*, *efficiency and performance constraints* og Udir (2019) bruker ordet logikk. Kjernen i begrepet evaluering er å gjøre kontinuerlig gjøre vurderinger av eget og andres arbeid, på en hensiktsmessig måte som også er effektiv. Respondentene fra spørreundersøkelsen synes også evaluering er en viktig kunnskap i programmering og matematikk.

Analysen viser resultat fra t-testene i tabell 10 og 11, og jeg har vansker med å se mulige årsaker for de signifikante forskjellene mellom gruppene, til den grad at det nesten blir gjetting. I tabell 10 kan jeg tenke meg at etterutdanningen eller kursene som lærerne har deltatt på har presentert de ulike begrepene fra Udir (2019) sin modell for den algoritmiske tenkeren, ettersom dekomponering for de med etterutdanning har et gjennomsnitt på 4.95. I tabell 11 kan jeg tenke meg at lærerne som er over 40 år er vant med en matematikk som i

større grad er preget av å løse problemer steg for steg. Lærerne under 40 år er kanskje mere vant med matematikk som i større grad vektlegger relasjonell forståelse, som den nye læreplanen også gjør, og dette kan skylde forskjellene i påstand 3.

4.2.2 Arbeidsmåter og ferdigheter

Den univariate analysen av arbeidsmåter og ferdigheter skiller seg fra nøkkelbegrepene og kunnskapene, ved at det er lite forskjeller i gjennomsnitt og standardavvik. Tabell 12 viser lærernes syn på hvor viktig syv forskjellige arbeidsmåter og ferdigheter er innenfor programmering og matematikk. Variabel 1 og 5 har marginalt lavere gjennomsnitt (4.59-4.68) enn andre variabler. Det var heller ingen variabler som ga store forskjeller i hvilken grad de er viktig i programmering og matematikk.

Tabell 12: Konstruktet arbeidsmåter algoritmisk tenkning, delt i programmering og matematikk for hver påstand

Ferdigheter		N	Gjennomsnitt	Standardavvik
1. Det er viktig å kunne jobbe med problemer som virker komplekse	P	44	4.59	.622
	M	42	4.69	.517
2. Det er viktig å kunne være utholden når man jobber med krevende oppgaver	P	**45**	4.91	.358
	M	**45**	4.93	.252
3. Det er viktig å kunne jobbe med oppgaver som kan ha flere løsningsmetoder og åpne svar	P	**45**	4.89	.318
	M	44	4.98	.151
4. Det er viktig å kunne kommunisere og samarbeide mot et felles mål	P	**45**	4.78	.599
	M	**45**	4.82	.535
5. Det er viktig å kunne fikle med både konkreter og digitale midler for å fikse noe som ikke funker	P	44	4.68	.740
	M	44	4.66	.645
6. Det er viktig å kunne både designe og skape løsninger på problemer	P	44	4.80	.462
	M	44	4.68	.601
7. Det er viktig å forstå feil og begrensninger av sin egen løsning på et problem, for deretter å rette opp i den	P	44	4.89	.321
	M	44	4.84	.370

Tabell 12 viser at alle påstandene om arbeidsmåter og ferdigheter er verdsatt blant respondentene. Udir (2019) sin modell inneholder arbeidsmåtene, fikle, skape, feilsøke, holde ut og samarbeide. Fra min spørreundersøkelse valgte jeg å inkludere to andre arbeidsmåter/ferdigheter som er viktig for den algoritmiske tenkeren som forskning vektlegger (Bocconi et al., 2019; Resnick, 2009). Problemløsningsoppgaver er ofte kompleks og åpner for at elevene kan komme fram til ulike svar, med forskjellige fremgangsmåter. En viktig del av samarbeidet blant elevene er at de kan diskutere hvordan de har kommet fram til sin løsning. Komplekse problemer har ofte ulike løsningsmetoder, og respondentene mener også at det er viktig at elevene takler å jobbe med slike oppgaver. For elevene kan det også være mere gjenspeilende av deres fremtidige arbeid, hvor de trenger gode problemløsningsevner og vil finne forskjellige løsninger (NOU, 2013:2).

4.2.3 Algoritmisk tenkning i undervisning

I lys av i hvilken grad lærerne mente ulike nøkkelbegrep og arbeidsmåter som er viktig for den algoritmiske tenkeren i programmering og matematikk, presenterer denne univariate analysen hvordan algoritmisk tenkning kan være nyttig både i og utenfor programmering og hvilken virkning forståelse av algoritmisk tenkning kan ha. Påstand 4 og 5 har høyt gjennomsnitt (4.67 og 4.73), men standardavvikene er noe høy for påstand 4 (.905). Påstand 7 og 8 er besvart av alle med en høyt gjennomsnittlig enighet i disse påstandene, og standardavvikene viste seg å skyldes interessante funn. Begge påstandene er besvart av alle respondenter.

Tabell 14 viser en t-test gjennomført for å se på forskjellene i påstand 8, «Det er viktig at elever ikke frykter å ta i bruk digitale midler», delt i gruppene over og under 40 år. Respondenter over 40 år er svært enig i påstanden (4.92) med et naturlig lite avvik (0.272), ettersom det er vanskelig å få et avvik fra et svært høyt gjennomsnitt, mens lærerne under 40 år, også er enig i påstanden (4.32), er det større forskjell i hva de svarer (1.204). T-testen er signifikant med en tosidig p-verdi på 0.042, tosidig p-verdi er brukt siden det ikke er mulig å forvente forskjellen mellom gruppenes retning.

Tabell 15 viser påstand 7, «det er viktig at elever ikke bare er brukere av digitale midler, men har en forståelse av hvordan de fungerer», måler samme konstrukt som påstand 8, men her er det signifikante forskjeller i gruppene av de som har/ikke har etterutdanning eller kurs. De som ikke har tatt etterutdanning er i høyest grad enig i påstanden (4.91) med lite avvik (0.294), og de som har tatt etterutdanning i mindre grad er enig (4.39) med større avvik

(1.118). Denne forskjellen mellom gruppene er signifikant med en tosidig p-verdi på 0.042, og forskjellen høyst sannsynlig skyldes noe annet, enn tilfeldigheter.

Tabell 13: Konstrukter, algoritmisk tenkning i undervisning

Algoritmisk tenkning i undervisning	N	Gjennomsnitt	Standardavvik
2. Programmering som verktøy kan gjøre algoritmisk tenkning mer konkret og tydelig for elevene	**45**	4.53	.786
4. Å tenke algoritmisk er ikke bare nyttig når man jobber med programmering	**45**	4.67	.905
5. Algoritmisk tenkning kan gjøre det enklere for både et menneske og en datamaskin å løse et problem	44	4.73	.544
7. Det er viktig at elever ikke bare er brukere av digitale midler, men har en forståelse for hvordan de fungerer	**45**	4.64	.857
8. Det er viktig at elever ikke frykter å ta i bruk digitale midler	**45**	4.67	.853

Tabell 14: T-test, konstruktet algoritmisk tenkning i undervisning, gruppe alder

Algoritmisk tenkning i undervisning/alder	Alder	N	Gjennomsnitt	Standard avvik	Tosidig P-verdi
8. Det er viktig at elever ikke frykter å ta i bruk digitale midler	Over 40	**26**	**4.92**	**.272**	**.044**
	Under 40	**19**	**4.32**	**1.204**	

Tabell 15: T-test, konstruktet algoritmisk tenkning i undervisning, gruppe etterutdanning

Algoritmisk tenkning /etterutdanning	Kurs	N	Gjennomsnitt	Standar davvik	Tosidig P-verdi
7. Det er viktig at elever ikke bare er brukere av digitale midler, men har en forståelse for hvordan de fungerer	Ja	**23**	**4.39**	**1.118**	**.042**
	Nei	**22**	**4.91**	**.294**	

Algoritmisk tenkning skiller seg fra programmering ved at det er en problemløsningsmetode. Samtidig er det en rekke sammenhenger i form av hva som kjennetegner den algoritmiske tenkeren og gode programmerere (Shute, 1991, referert i Shute et al. 2017; Utdanningsdirektoratet, 2019). Respondentene er enig i at bruken av programmering kan

gjøre kunnskaper og ferdigheter for den algoritmiske tenkeren tydeligere. Tross denne nære koblingen mellom programmering og algoritmisk tenking, medfører dette ikke at å tenke algoritmisk tenking bare er egnet i den digitale verden (Bocconi et al., 2016). Lærerne ser verdien av å kunne tenke algoritmisk både med og uten digitale midler, og dette tyder på at lærerne har en forståelse av algoritmisk tenkning lik beskrivelsene til Bocconi et al. (2016) og Cabrera (2019). Basert på utvalget for dette prosjektet er jeg derimot kritisk til å generalisere at dette gjelder norske lærere. Yadav et al. (2014) og Cabrera (2019) finner derimot at studenter og annen forskning har definisjoner og beskrivelser av algoritmisk tenkning som begrenset til digitale midler og programmering. En stor ulempe med hvordan algoritmisk tenkning fremstilles i forskning som komplekst, eller ikke egnet for alle elever å lære kan være en reell oppfatning av populasjonen for dette prosjektet, som utvalget ikke viser til, og dette er gjennomgående for hele prosjektet (Cabrera, 2019).

Bocconi et al. (2016) har gjort forskning som retter søkelys mot hvor viktig det er å kunne tenke algoritmisk i hverdagen. Elevene vil i hovedsak være brukere av digitale midler og å ha en forståelse for digitale midler er en viktig digital ferdighet (Kunnskapsdepartementet, 2020). Elever som utvikler god forståelse for digitale midler, vil ikke frykte å ta dem i bruk, noe respondentene er enig i. Fordelingen i t-testene har blitt analysert med en tosidig p-verdi fordi jeg hadde ingen forventning om hvordan retning fordelingen ville ha, ei har jeg heller en god forklaring på hvorfor respondentene over 40 år og de som har tatt etterutdanning, i større grad er enig i påstand 8 og 7 (tabell 14 og 15).

Lærerne ble spurt hvordan algoritmisk tenkning passet inn i deres undervisningspraksis. 37.8% svarte, «Nytt, men kompatibelt med min undervisningspraksis», 53.3% svarte, «Allerede implementert i min undervisningspraksis» og 6.7% svarte, «Et tillegg, noe nytt i min undervisningspraksis» (McGinnis et al., 2020). Jeg er skeptisk til hvilken grad McGinnis et al. (2020) sin forskning kan overføres til å brukes på utvalget og populasjonen for dette prosjektet, men det er viktig differensiere mellom hva lærerne påstår om deres undervisningspraksis, og hvordan den faktisk er. Stemmer det at lærerne som mener noe er nytt i deres undervisningspraksis, ikke er endringsvillig? Eller var dette allerede en del av deres undervisningspraksis? Dette er et annet kunnskapshull som krever en annen metodisk tilnærming, og hadde vært spennende å forske på.

4.3 Lærernes holdninger

Det siste analysekapitlet vil analysere og drøfte lærernes holdninger og meninger til implementeringen av programmering i matematikk. Det er viktig å poengtere at dette er lærernes holdninger og meninger, og det har en helt annen hensikt å forske på lærernes holdninger, enn handlinger. Mange av påstandene måler det samme som forskningen til Misfeldt et al. (2019) og Korhonen et al. (2022) og det kan være interessant å drøfte resultater fra min spørreundersøkelse og forskningen som er gjort i Sverige og Finland, som har hatt programmering som en del av læreplanen noen år lengre enn Norge. Forskningsspørsmålet for dette underkapitlet er dermed:

Hvilke holdninger har lærere til programmering?

4.3.1 Lærernes holdninger til programmering

Den siste univariate analysen handler om lærernes meninger om implementeringen av programmering i matematikk, og deres egne oppfattede evner, se tabell 16. Disse spørsmålene ga de mest varierte svarene av alle konstruktene i spørreundersøkelsen. Påstand 2 viser til at lærerne tror de kan ha innflytelse på elevenes læring (4.69) og med et lavere standardavvik, enn noen av de andre påstandene (0.514). Påstand 1 viser at lærerne er litt over delvis entusiastiske om å undervise programmering (4.13) med et høyt standardavvik (1.014). Påstand 5 viser til en lav (2.60) og varierende (1.405) oppfatning av at deres egen kompetanse er tilstrekkelig. Påstand 6, 7 og 9 ser på tilpasset opplæring, læreplanmålene og tilgang på ressurser, med litt over nøytral grad av enighet i påstandene (3.22-3.52) og igjen store avvik (1.254-1.460). Sist ser påstand 12 på om lærerne ønsker mere etterutdanning eller kursing for å undervise programmering i matematikk, hvor lærerne som en homogen gruppe ønsker mere etterutdanning (4.39), men avviket er også stort (1.146).

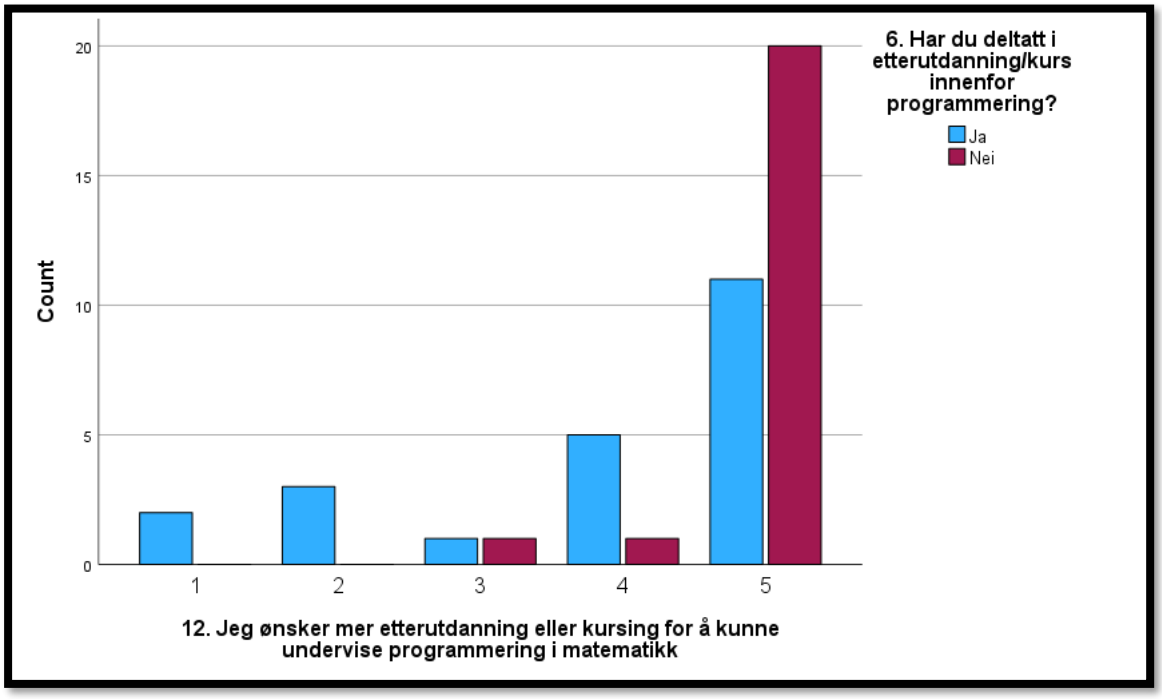
Tabell 17 viser en oversikt over en t-test som ble gjennomført om lærernes holdninger til programmering i matematikk, delt i gruppene som har/ikke har deltatt i etterutdanning eller kurs. Jeg hadde en forventning om at lærerne som har deltatt i etterutdanning eller tatt kurs innenfor programmering, i større grad ville være enig i påstandene som måler lærernes holdninger til programmering, derfor er alle t-testene målt med ensidig p-verdi. Påstand 1, «Jeg er entusiastisk om å undervise programmering i matematikk», har en gjennomsnittsdifferanse på 0.53 og begge gruppene har like avvik fra gjennomsnittet (0.941 og 1.037). Forskjellen mellom disse gruppene er signifikant med en ensidig p-verdi på 0.041. T-testen på påstand 5, «Min egen kompetanse i programmering i matematikk er tilstrekkelig»,

viser at lærerne som har tatt etterutdanning er litt mer enn nøytralt enig i påstanden (3.43), mens lærerne som ikke har deltatt i etterutdanning er litt uenig (1.73) og avviket er mindre i denne gruppen (0.609 differanse i standardavvik). T-testen er signifikant med en p-verdi på <0.001. T-testene for påstand 6, 7 og 9 viser signifikante forskjeller mellom gruppene med differanser i gjennomsnitt på 1.24, 0.52 og 0.91, med ubetydelige forskjeller i standardavvik, selv om det er verdt å nevne at standardavvikene i seg selv er høye. Den siste t-testen på påstand 12, «Jeg ønsker mer etterutdanning eller kursing for å kunne undervise programmering i matematikk», viser at lærerne som ikke har deltatt i etterutdanning eller kurs, ønsker dette i høy grad (4.86). Lærerne som har deltatt i etterutdanning, ønsker fortsatt mere av dette (3.91), men avviket er større i denne gruppen (0.943 differanse i standardavvik). Forskjellen mellom gruppene er signifikant med en ensidig p-verdi på 0.003. Figur 3 viser i hvilken grad disse to gruppene er enig i påstanden. Også verdt å nevne er at jeg fant ingen signifikante forskjeller for gruppene over og under 40 år, og alder dermed ikke er en uavhengig variabel som påvirker lærernes holdninger til programmering i matematikk.

Tabell 18 viser en korrelasjonsanalyse som er gjort for å se hvordan lærernes entusiasme kan ha en sammenheng med andre variabler. Lærernes entusiasme har en medium korrelasjonskoeffisient med påstand 1, «Programmering kan gjøre skolearbeid mer interessant for elevene» (Pearsons R = 0.308, Tosidig p-verdi = 0.014) og påstand 3, «Programmering er egnet for at elever får oppleve skaperglede» (Pearsons R = 0.364, Tosidig p-verdi = 0.014). Det er også en sterk sammenheng mellom lærernes entusiasme og enighet i påstand 4, «Programmering er egnet for at elever får utvikle problemløsningsegenskapene sine» (Pearsons R = 0.639, Tosidig p-verdi = <0.001). Påstandene 2, 5 og 9 om algoritmisk tenkning i undervisning gir svært like verdier for Pearsons R (0.422, 0.421 og 0.424) og den tosidige p-verdien (0.004, 0.004 og 0.005). Se tabell 13 for påstand 2 og 5, påstand 9, «Algoritmisk tenkning er en viktig digital ferdighet (én av fem grunnleggende ferdigheter)». Det er en medium korrelasjonskoeffisient, og analysen viser at funnene er signifikant med stor sikkerhet.

Tabell 16: Konstrukter lærernes holdninger til programmering

Lærernes holdninger til programmering	N	Gjennom snitt	Standard avvik
1. Jeg er entusiastisk om å undervise programmering i matematikk	**45**	4.13	1.014
2. Jeg kan ha innflytelse på mine elevers læring i programmering i matematikk	**45**	4.69	.514
5. Min egen kompetanse i programmering i matematikk er tilstrekkelig	**45**	2.60	1.405
6. Jeg kan planlegge variert undervisning avhengig av nivået på elevene i klassen i programmering i matematikk	**45**	3.22	1.460
7. Jeg synes det er tydelig hva elevene skal lære ut ifra læreplanmålene i programmering i matematikk	42	3.52	1.254
9. Jeg har tilgang på de ressursene som kreves for at jeg skal undervise programmering i matematikk	44	3.36	1.480
12. Jeg ønsker mer etterutdanning eller kursing for å kunne undervise programmering i matematikk	44	4.39	1.146



Figur 3: Histogram over fordelingen av lærere som har deltatt på etterutdanning som ønsker videre utdanning (skala 1-5)

Tabell 17: T-test, konstrukter lærernes holdninger til programmering, gruppe etterutdanning

Lærernes holdninger /etterutdanning	Kurs	N	Gjennomsnitt	Standard avvik	Ensidig P-verdi
1. Jeg er entusiastisk om å undervise programmering i matematikk	Ja	**23**	**4.39**	**.941**	**.041**
	Nei	**22**	**3.86**	**1.037**	
5. Min egen kompetanse i programmering i matematikk er tilstrekkelig	Ja	**23**	**3.43**	**1.376**	**<.001**
	Nei	**22**	**1.73**	**.767**	
6. Jeg kan planlegge variert undervisning avhengig av nivået på elevene i klassen i programmering i matematikk	Ja	**23**	**3.83**	**1.230**	**.002**
	Nei	**22**	**2.59**	**1.436**	
7. Jeg synes det er tydelig hva elevene skal lære ut ifra læreplanmålene i programmering i matematikk	Ja	22	**3.77**	**1.232**	**.090**
	Nei	20	**3.25**	**1.251**	
9. Jeg har tilgang på de ressursene som kreves for at jeg skal undervise programmering i matematikk	Ja	22	**3.82**	**1.532**	**.020**
	Nei	**22**	**2.91**	**1.306**	
12. Jeg ønsker mer etterutdanning eller kursing for å kunne undervise programmering i matematikk	Ja	22	**3.91**	**1.411**	**.003**
	Nei	**22**	**4.86**	**.468**	

Tabell 18: Korrelasjonsanalyse, lærernes entusiasme, konstrukter programmering og algoritmisk tenkning i undervisning

Korrelasjons analyse		1. P elev interesse	3. P elev SG	4. P elev PB	2. P AT tydelig	5. P AT anvendelig	9. AT DF
1. Entusiastisk	Pearsons r	**.364**	**.364**	**.639**	**.422**	**.421**	**.424**
	Tosidig P-verdi	**.014**	**.014**	**<.001**	**.004**	**.004**	**.005**
	N	**45**	**45**	**45**	**45**	**45**	42

Respondentene oppgir i spørreundersøkelsen at de er entusiastiske om å undervise programmering i matematikk, også i større grad enn de svenske lærerne i forskningen til Misfeldt et al. (2019). 21 og 18 av 45 er «delvis enig» eller «enig» i påstand 1, «jeg er entusiastisk om å undervise programmering i matematikk». Dermed er 27% flere entusiastisk om å undervise programmering i matematikk, sammenlignet med de svenske lærerne. Lærerne er i mindre grad enig i at deres kompetanse er tilstrekkelig, og de svarer ganske likt de svenske lærerne. 33.3 av lærerne i min spørreundersøkelse føler at deres kompetanse er tilstrekkelig, mens i Sverige fant Misfeldt et al. (2019) at 33.5% av lærerne følte seg forberedt med tanke på kompetanse. Felles for begge landene er dermed at lærerne er entusiastiske om å undervise, men føler seg ikke kompetent nok. Rich (2020) har funnet at lærerne som har mestringstro fører til elever som lærer mer, lærere som jobber lengre og blir mindre utbrent.

Forskningen til Korhonen et al (2022) har også sett på implementeringen av programmering, men i Finland. De finske lærerne trekker fram fordeler, kompatibilitet og kompleksitet som de tre viktigste faktorene for programmering i matematikk. Fordelene ved programmering er drøftet i en rekke av de tidligere kapitlene, å kunne løse problemer, forstå digitale midler, være kreativ, motiverte elever og så videre, og disse tankene deler respondentene av spørreundersøkelsen. De finske lærerne vekket bekymring rundt hvordan programmering er strukturert i læreplanene med vide rammer for hvordan og hvilket fag programmering skal undervises. Respondentene synes heller ikke læreplanmålene i matematikk er tydelig, og det kan være bekymringsverdig at elever kan oppleve forskjellige undervisningstilbud, grunnet utydelige læreplaner, noe som ble nevnt av mange finske lærere (Korhonen et al., 2022). Det er stor kontrast mellom norske og finske lærere hvordan de opplever programmering i deres undervisningspraksis. 6.3% av finske lærere mener at programmering «ikke er noe nytt» i deres undervisningspraksis, mens blant respondentene i min spørreundersøkelse, svarer 6.7% at det er «nytt» i deres undervisningspraksis, og 37.8% svarte at det var «nytt, men kompatibelt». I lys av McGinnis et al. (2020) sin forskning kan det være forskjeller i hvilken grad lærerne er villig til å endre deres undervisningspraksis, men det er vanskelig å trekke konklusjoner på dette området med dataen jeg har, og videre forskning på dette hadde vært interessant.

T-testen viser til interessant funn av hvilken virkning etterutdanning har på lærernes holdninger til programmering. Korhonen et al. (2022) fant ingen sammenhenger på dette området. Respondentene som har deltatt på etterutdanning er i større grad enig i påstandene som er positivt ladet, dermed kan etterutdanningen og kursene vise til å ha en positiv

virkning, spesielt deres egen oppfatning av kompetanse og evne til å planlegge tilpasset undervisning for elevene. Etterutdanningen viser til å ha en positiv effekt, og lærerne som ikke har deltatt på etterutdanning, svarer at de ønsker dette (Figur 3). Korrelasjonsanalysen i tabell 18 viser også at lærerne som er entusiastisk, mener at nytten av å undervise programmering er større innenfor elevenes interesse, skaperglede, problemløsning og virkningen av algoritmisk tenkning. De finske lærerne ytret også at en av årsakene for deres entusiasme til å undervise programmering, var at de kunne gjøre undervisningen interessant for elevene som tradisjonelt ikke har «passet inn».

Jeg hadde forventet noen signifikante funn om lærernes alder og deres holdninger til programmering i matematikk, men det var ingen konkrete funn verdt å diskutere. Heller er mangelen på funn interessant å diskutere. Med en forventning om at yngre lærere som er antatt mer kompetent i bruk av digitale midler, ville jeg tro denne gruppen skulle i større grad være enig i påstander om holdninger til programmering.

De vises en rekke fellestrekk mellom lærerne som svarte på spørreundersøkelsen, svenske lærere (Misfeldt et al., 2019) og finske lærere (Korhonen et al., 2022) om implementeringen av programmering i matematikkfaget. Lærerne er motiverte for å undervise, men føler de mangler kompetansen de føler de trenger. Spørreundersøkelsen i dette prosjektet og forskningen til Korhonen et al. (2022) viser til at etterutdanning og kurs av lærere har en effekt på holdninger til programmering i matematikk. Økt entusiasme, kompetanse og bedre bruk av ressurser kan være potensielle positive virkninger. Igjen, så gjør utvalget for min spørreundersøkelse det ikke mulig å generalisere funnene for populasjonen, men det viser til signifikante funn innad et lite bekvemmelighetsutvalg.

5 Avslutning

Gjennom dette masterprosjektet har det vært ønsket å bidra til forskningsfeltet og selv utvikle min egen kompetanse som forsker og lærer. Det teoretiske rammeverket og den metodiske tilnærmingen har bidratt med å utforske problemstillingen, og besvare følgende forskningsspørsmål:

I. Hvilket syn har lærerne på undervisning av programmering i matematikk?
II. Hva er læreres forståelse av algoritmisk tenkning?
III. Hvilke holdninger har lærere til programmering?

Konklusjon på forskningsspørsmål 1 er at lærerne synes programmering er en nyttig implementering i den nye læreplanen. Programmering i matematikk kan være en god måte å utvikle interesse, kreativitet, skaperglede og problemløsningsevner. Programmering kan fungere som en fin arena for å diskutere løsninger og lære om bruk av hverandres ideer. Lærerne synes at programmering er en læringsmåte som er egnet for alle elever, og det er en viktig del av deres digitale dannelse. Blokkprogrammering gjør det mulig å undervise programmering på mellomtrinnet med enkle fremgangsmåter og gir muligheter for å være både kreativ og sosial. Lærerne som har deltatt i etterutdanning kan være de som evner best å ta i bruk ressurser innenfor programmering, og ressurser viser seg å være en utfordring for de nordiske landene. Det er mulig at sammenhengen mellom programmering og matematikk er sterk, men mangler i kompetanse blant lærerne kan være en grunn til elever ikke lærer matematikk gjennom programmering i likt tempo som annen matematikkundervisning. Lærerne ser forskjellig bruk av programmering i matematikk, som verktøy, motivasjon eller å trene algoritmisk tenkning.

Konklusjon på forskningsspørsmål 2 er at lærerne som utgjør utvalget i spørreundersøkelsen, har en god forståelse av begrepet algoritmisk tenkning. Deres forståelse av begrepet er ikke helt likt anvendelsen av begrepet innen informatikken, men heller tilpasset behovet for matematikk på mellomtrinnet. Automasjon og algoritmebehandling er to nøkkelbegrep lærerne ikke synes er like viktig for elevene, både innenfor programmering og matematikk. Ferdighetene til den algoritmiske tenkeren viser store likheter mellom forskning og hvordan lærerne verdsetter dem i undervisningen. Lærerne ser nytten av at elevene kan tenke algoritmisk, i hovedsak som en problemløsningsmetode for elevene. Ifølge lærerne trenger

ikke elevene å ta i bruk digitale midler for å bli dyktige algoritmiske tenkere, og dette kan være viktig for deres forståelse av digitale midler i fremtiden.

Konklusjon på forskningsspørsmål 3 er at de norske lærerne i denne undersøkelse deler en rekke av de samme erfaringene og holdningene med de svenske og finske lærerne. Lærerne påstår de er entusiastisk for å undervise programmering i matematikk, men føler de mangler kompetansen som er nødvendig. Lærernes entusiasme og etterutdanning er to viktige faktorer for andre holdninger om programmering. Respondentene deler ikke samme erfaring med de finske lærerne, når det kommer til implementeringen av programmering i deres undervisningspraksis. De norske lærerne svarer at algoritmisk tenkning passer godt inn i deres tidligere undervisningspraksis, mens de finske lærerne ser på dette som noe nytt. Dette kan tyde til forskjeller i villigheten til å endre sin undervisningspraksis, slik at det passer endringen i læreplanen, men dette er en konklusjon jeg skal være forsiktig med å trekke.

Problemstillingen for oppgaver var: Hva synes matematikklærere på mellomtrinnet om å inkludere programmering i matematikkfaget etter fagfornyelsen? Denne gruppen lærere ser nytten av programmering og har god forståelse av begrepet algoritmisk tenkning, samt anvendelsen av denne problemløsningsmetoden i undervisning. Lærerne er entusiastisk for å undervise programmering i matematikk, men peker på en rekke usikkerhetsmomenter i deres kompetanse. Disse funnene peker på en rekke av de samme positive tendensene og utfordringene til andre nordiske land.

Lærernes kompetanse i form av etterutdanning eller kurs virker til å være en viktig faktor som påvirker lærernes syn på programmering i undervisning, forståelse av sammenhenger mellom programmering, matematikk og algoritmisk tenkning og holdninger til programmering i matematikkfaget. Alder som uavhengig variabel er i mitt utvalg av respondenter, ikke en viktig faktor når noen av forskningsspørsmålene skal besvares.

Begrensninger i den metodiske tilnærmingen, spesielt utvalget for prosjektet er en svakhet. Funnene som har gitt signifikante verdier kan ikke generaliseres grunnet usikkerheter i hvor representativt og lite utvalget er. Signifikante funn i prosjektet kan heller brukes som muligheter for utforskning i annen fremtidig forskning.

Det finnes en rekke interessant funn i analyseprosessen som ikke har fått plass i oppgaven, men funnene som er presentert var mest relevant i lys av problemstillingen. Funnene fra oppgaven viser til en rekke positive tendenser blant norske lærere og gode muligheter for

programmering i matematikk. Derimot peker lærerne på ulike usikkerhetsmomenter i deres egen kompetanse, hvor mangelen på etterutdanning eller kurs er en viktig faktor. Det vil være spennende å følge utviklingen av dette fagfeltet i årene fremover.

5.1 Veien videre

Gjennom oppgaven har jeg drøftet en rekke interessant retninger oppgaven har tatt. Det har ikke vært kapasitet til å gå mye i dybden på tema, i tillegg til at det er begrensninger i den metodiske tilnærmingen. Innsamlet data, analyse og drøfting fra dette prosjektet kan forhåpentligvis være til inspirasjon for videre forskning på feltet. Videre arbeid kan gjennomføres enten ved en annen metodisk tilnærming, et annet teoretisk rammeverk eller lignende forskning med et utvalg som representerer en populasjon.

Referanseliste

Battista, M., Smith, M. S., Boerst, T., Sutton, J., Confrey, J., White, D., & Quander, J. (2009). Research in Mathematics Education: Multiple Methods for Multiple Uses. *Journal for Research in Mathematics Education*, *40*(3), 216-240.

Benton, L., Saunders, P., Kalas, I., Hoyles, C., & Noss, R. (2018). Designing for learning mathematics through programming: A case study of pupils engaging with place value. *International Journal of Child-Computer Interaction*, 16, 68–76. https://doi.org/10.1016/j.ijcci.2017.12.004

Bocconi, S., Chioccariello, A. & Earp, J. (2018). The Nordic approach to introducing Computational Thinking and programming in compulsory education. *Report prepared for the Nordic@BETT2018 Steering Group*. https://doi.org/10.17471/54007

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). Developing computational thinking in compulsory education. *European commission, JRC science for policy report*. https://komenskypost.nl/wp-content /uploads/2017/01/jrc104188_computhinkreport.pdf.

Bråting, K., & Kilhamn, C. (2022) The Integration of Programming in Swedish School Mathematics: Investigating Elementary Mathematics Textbooks, *Scandinavian Journal of Educational Research*, 66:4, 594-609. https://doi.org/10.1080/00313831.2021.1897879

Bråting, K., Kilhamn, C., & Rolandsson, L. (2021). "Integrating programming in Swedish school mathematics: description of a research project. I *Proceedings of MADIF 12 The twelfth research seminar of the Swedish Society for Research in Mathematics Education Växjö, January 14–15, 2020: Sustainable mathematics education in a digitalized world* (s. 101-110). SMDF Svensk Förening för Matematikdidaktisk Forskning. http://matematikdidaktik.org/wp-content/uploads/2021/03/MADIF12_dokumentation.pdf

Cabrera, L. (2019). Teacher Preconceptions of Computational Thinking: A Systematic Literature Review. *Journal of Technology and Teacher Education*. 27., 305-333.

Capraro, R. M., Bicer, A., Lee, Y., & Vela, K. (2019). Putting the quantitative pieces together to maximize the possibilities for a successful project. In *Designing, Conducting, and Publishing Quality Research in Mathematics Education* (s. 97-110). Springer, Cham.

Cheng, G. (2019). Exploring factors influencing the acceptance of visual programming environment among boys and girls in primary schools. *Computers in Human Behavior*. 92, 361-372. https://doi.org/10.1016/j.chb.2018.11.043

Cobb, P. (2007). Putting philosophy to work. In F. K. J. Lester (Ed.), *Second handbook of research on mathematics teaching and learning* (Vol. 1, s. 3-38). Charlotte.

Cohen, L., Manion, L., & Morrison, K. (2018). Research methods in education. Routledge.

Danielsen, A. (2013). Kunnskapsbygging i skolen via kvantitative verktøy – statistikk og spørreskjema. I M. Brekke. & T. Tiller (Red.). *Læreren som forsker : innføring i forskningsarbeid i skolen* (s. 138-156). Universitetsforlaget.

Eikemo, T. A., & Clausen, T. H. (2012). Kvantitativ analyse med SPSS. *En praktisk innføring i kvantitative analyseteknikker* (2. utg., s. 356). Tapir akademisk forlag.

Fagerlund, J, Häkkinen, P, Vesisenaho, M, Viiri, J. Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Application in Engineering Education*. 2021; 29: 12– 28. https://doi.org/10.1002/cae.22255

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

Gleiss, M. & Sæther, E. (2021): Forskningsmetode for lærerstudenter. *Å utvikle ny kunnskap i forskning og praksis.* Cappelen Damm akademisk.

Grønmo, Sigmund. (2020). *Målenivå* i *Store norske leksikon* på snl.no. https://snl.no/m%C3%A5leniv%C3%A5

Kaufmann, O. T., & Stenseth, B. (2020). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology.* 52(7), 1029-1048. https://doi.org/10.1080/0020739X.2020.1736349

Kilhamn, C., Bråting, K., & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. Bringing Nordic Mathematics Education into the Future: Proceedings of Norma 20. *The Ninth Nordic Conference on Mathematics Education*, 169–176. http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-439334

Korhonen, T., Salo, L., Laakso, N., Seitamaa, A., Sormunen, K., Kukkonen, M., & Forsström, H. (2022): Finnish teachers as adopters of educational innovation: perceptions of programming as a new part of the curriculum. *Computer Science Education*. https://doi.org/10.1080/08993408.2022.2095595

Kunnskapsdepartementet (2017). Overordnet del – verdier og prinsipper for grunnopplæringen. Fastsatt som forskrift ved kongelig resolusjon. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/overordnet-del/?lang=nob

Kunnskapsdepartementet (2017). Rammeverk for grunnleggende ferdigheter. Fastsatt som forskrift som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/laring-og-trivsel/rammeverk/rammeverk-for-grunnleggende-ferdigheter/

Kunnskapsdepartementet. (2020). *Læreplan i matematikk 1.-10. trinn (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05

Kunnskapsdepartementet. (2020). *Læreplan i naturfag (NAT01-04)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/nat01-04

Kunnskapsdepartementet. (2020). *Læreplan i valgfaget programmering (PRG01-02)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/prg01-02

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/10.1016/j.learninstruc.2022.101667

Lemon, N., & Garvis, S. (2016) Pre-service teacher self-efficacy in digital technology. Teachers and Teaching, 22:3, 387-408. https://doi.org/10.1080/13540602.2015.1058594

Lester Jr, F. K. (2010). On the theoretical, conceptual, and philosophical foundations for research in mathematics education. In *Theories of mathematics education* (s. 67-85). Springer Berlin Heidelberg.

Mæland, K. (2021). *Desentralisert kompetanseutvikling – Et steg i riktig retning for å oppfylle Fagfornyelsens verdigrunnlag? En kvalitativ studie om innføringen av programmering og skapende aktivitet i klasserommet*. [Masteroppgave, NTNU]. NTNU Open. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2785129

McGinnis, J. R., Hestness, E., Mills, K., Ketelhut, D. J., Cabrera, L., & Jeong H. (2020). Preservice science teachers' beliefs about computational thinking following a curricular module within an elementary science methods course. Contemporary Issues in Technology and Teacher Education, 20(1), 85-107.

Misfeldt, M., Szabo, A., & Helenius, O. (2019). Surveying teachers' conception of programming as a mathematics topic following the implementation of a new mathematics curriculum. https://hal.archives-ouvertes.fr/hal-02417074/document

Neverdahl, A. (2019). *Dokumentanalyse av diskursen om innføring av programmering i skolen*. [Masteroppgave, OsloMet].ODA. https://oda.oslomet.no/oda-xmlui/handle/10642/8739

NOU 2013: 2. (2013). *Hindre for digital verdiskaping*. Kommunal- og distriktsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2013-2/id711002/

NOU 2015: 8. (2015). *Fremtidens skole — Fornyelse av fag og kompetanser*. Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/

NOU 2018: 2. (2018). *Fremtidige kompetansebehov I — Kunnskapsgrunnlaget*. Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2018-2/id2588070/

Pörn, R., Hemmi, K., & Kallio-Kujala, P. (2021). Inspiring or confusing – a study of Finnish 1–6 teachers' relation to teaching programming. *LUMAT: International Journal on Math, Science and Technology Education*, *9*(1), 366–396. https://doi.org/10.31129/LUMAT.9.1.1355

Pörn, R., Löfwall Hemmi, K., & Kallio-Kujala, P. (2021). "Programming is a new way of thinking" – teacher views on programming as a part of the new mathematics curriculum in Finland. I *Proceedings of MADIF 12 The twelfth research seminar of the Swedish Society for Research in Mathematics Education Växjö, January 14–15, 2020: Sustainable mathematics education in a digitalized world* (s. 91-100). SMDF Svensk Förening för Matematikdidaktisk Forskning. http://matematikdidaktik.org/wp-content/uploads/2021/03/MADIF12_dokumentation.pdf

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch:

Programming for all. *Communications of the ACM*, 52, 60–67. https://doi:10.1145/1592761.1592779

Rich, P. J., Larsen, R. A., & Mason, S. L. (2021) Measuring teacher beliefs about coding and computational thinking, Journal of Research on Technology in Education, 53:3, 296-316. https://doi.org/10.1080/15391523.2020.1771232

Schoenfeld, A. H. (2007). Method. In F. K. Lester (Ed.), *Second handbook of research on mathematics teaching and learning* (s. 69-107): NCTM.

Shute, V., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*. 22, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Siebert, D. K. (2019). Conducting a Timely Literature Search. In *Designing, Conducting, and Publishing Quality Research in Mathematics Education* (s. 17-30). Springer, Cham.

Silver, E. and Herbst, P. (2007). *The role of theory in mathematics education scholarship*. In F. Lester (Ed.), Second Handbook of Research in Mathematics Teaching and Learning (s. 39-67). Information Age.

Spangler, D. A., & Williams, S. R. (2019). The role of theoretical frameworks in mathematics education research. In *Designing, Conducting, and Publishing Quality Research in Mathematics Education* (s. 3-16). Springer, Cham.

Statlig spesialpedagogisk tjeneste. (2021). Programmering. Statped. https://www.statped.no/laringsressurser/teknologitema/programmering-for-barn-med-saerskilte-behov

Stenlund, E. (2021). *Programmering og Fagfornyelsen.* [Masteroppgave, UiO]. DUO Vitenarkiv. https://www.duo.uio.no/handle/10852/87187

Thorsnes, J. (2020). Teachers' attitudes and self-efficacy towards teaching programming and the impact of continuing education. [Masteroppgave, NTNU]. NTNU Open. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2777523

Utdanningsdirektoratet. (2019). Algoritmisk tenkning. Udir. https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019). *Hva er kjerneelementer?* Udir. https://www.udir.no/laring-og-trivsel/lareplanverket/stotte/hva-er-kjerneelementer/

Utdanningsdirektoratet. (2020). *Hva er nytt i matematikk?* Udir. https://www.udir.no/laring-og-trivsel/lareplanverket/fagspesifikk-stotte/nytt-i-fagene/hva-er-nytt-i-matematikk/

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35. https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

Wing, J. M. (2011). Research notebook: Computational thinking—what and why? *The link magazine*. Carnegie Mellon University. https://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf

Yadav, A., Mayfield, C. & Zhou, N., Hambrusch, S., & Korb, J. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education*. http://dx.doi.org/10.1145/2576872

Vedlegg 1: Spørreskjema – resultater

Nettskjema

Spørreskjema programmering og algoritmisk tenkning - Håvard Salamonsen

Oppdatert: 30. juni 2023 kl. 8:45

Hei, mitt navn er Håvard Salamonsen og jeg går 5. året GLU 5-10 ved UiT Norges arktiske universitet. I forbindelse med min masteroppgave skal jeg forske på læreres syn på programmering og algoritmisk tenkning i matematikkfaget. Oppgaven baserer seg på et digitalt spørreskjema med en kvantitativ tilnærming som tar ca. 15 minutter å besvare. Spørreskjemaet kan besvares av alle lærere som underviser i matematikk på 5.-7. trinn. Som en belønning for å svare på spørreundersøkelsen vil det være mulig å sende en mail 1. juni 2023 til hsa076@uit.no for å motta masteroppgaven når den er ferdigstilt. Spørreskjemaet inneholder ikke sensitive opplysninger og det vil være tilnærmet umulig å spore hva enkeltpersoner har svart. Jeg har dermed en mailliste på de som ønsker å lese oppgaven, siden funnene kan være nyttig lesning for deg som lærer. Takk på forhånd!

1. Hvor gammel er du?

Antall svar: 45

Svar	Antall	% av svar	
≥ 60 (60 år eller eldre)	2	4.4%	4.4%
[50-60) (Fra og med 50 år til 60 år)	16	35.6%	35.6%
[40-50) (Fra og med 40 år til 50 år)	8	17.8%	17.8%
[30-40) (Fra og med 30 år til 40 år)	13	28.9%	28.9%
< 30 (Under 30 år)	6	13.3%	13.3%

2. Hvilket kjønn passer deg best?

Antall svar: 45

Svar	Antall	% av svar	
Annet	0	0%	0%
Kvinne	31	68.9%	68.9%
Mann	14	31.1%	31.1%

3. Hvordan oppdaget du dette spørreskjemaet?

Antall svar: 45

Svar	Antall	% av svar	
Annet	3	6.7%	6.7%
Anbefaling fra andre	0	0%	0%
Gruppe på sosial media	14	31.1%	31.1%
Via mail/fra skoleleder	28	62.2%	62.2%

Side: 1/28

4. Hvilket trinn underviser du på?

Antall svar: 43

Svar	Antall	% av svar	
7. trinn	23	53.5%	53.5%
6. trinn	23	53.5%	53.5%
5. trinn	17	39.5%	39.5%

5. Hvor mange år har du undervist matematikk i skolen?

Antall svar: 45

Svar	Antall	% av svar	
≥ 20 (20 eller mer)	9	20%	20%
[15-20) (Fra og med 15 til 20)	4	8.9%	8.9%
[10-15) (Fra og med 10 til 15)	6	13.3%	13.3%
[5-10) (Fra og med 5 til 10)	10	22.2%	22.2%
< 5 (Mindre enn 5)	16	35.6%	35.6%

6. Har du deltatt i etterutdanning/kurs innenfor programmering?

Antall svar: 45

Svar	Antall	% av svar	
Nei	22	48.9%	48.9%
Ja	23	51.1%	51.1%

1. Programmering kan gjøre skolearbeid mer interessant for elevene

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	28	62.2%	62.2%
Delvis enig	15	33.3%	33.3%
Nøytral	0	0%	0%
Delvis uenig	2	4.4%	4.4%
Uenig	0	0%	0%

2. Programmering er egnet for at elever får utvikle kreativiteten sin

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	34	75.6%	75.6%
Delvis enig	10	22.2%	22.2%
Nøytral	1	2.2%	2.2%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

3. Programmering er egnet for at elever får oppleve skaperglede

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	38	84.4%	84.4%
Delvis enig	5	11.1%	11.1%
Nøytral	2	4.4%	4.4%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

4. Programmering er egnet for at elever får utvikle problemløsningsegenskapene sine

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	36	80%	80%
Delvis enig	7	15.6%	15.6%
Nøytral	1	2.2%	2.2%
Delvis uenig	1	2.2%	2.2%
Uenig	0	0%	0%

5. Programmering er egnet for å lære elever om kritisk kildebruk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	4	8.9%	8.9%
Enig	7	15.6%	15.6%
Delvis enig	10	22.2%	22.2%
Nøytral	13	28.9%	28.9%
Delvis uenig	8	17.8%	17.8%
Uenig	3	6.7%	6.7%

6. Programmering åpner for at elever ukritisk kopierer hverandres verk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	3	6.7%	6.7%
Delvis enig	19	42.2%	42.2%
Nøytral	12	26.7%	26.7%
Delvis uenig	4	8.9%	8.9%
Uenig	4	8.9%	8.9%

7. Programmering åpner for at elever kan kopiere og lære av hverandres arbeid

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	25	55.6%	55.6%
Delvis enig	16	35.6%	35.6%
Nøytral	2	4.4%	4.4%
Delvis uenig	1	2.2%	2.2%
Uenig	0	0%	0%

8. Innenfor programmering er det viktig at læreren kan hjelpe elever som sitter fast

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	23	51.1%	51.1%
Delvis enig	15	33.3%	33.3%
Nøytral	2	4.4%	4.4%
Delvis uenig	5	11.1%	11.1%
Uenig	0	0%	0%

9. Innenfor programmering er det viktig at læreren kan utvide oppgaver for elever som er ferdig

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	35	77.8%	77.8%
Delvis enig	10	22.2%	22.2%
Nøytral	0	0%	0%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

10. Programmering er så vanskelig at ikke alle elever burde lære det

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	1	2.2%	2.2%
Delvis enig	0	0%	0%
Nøytral	0	0%	0%
Delvis uenig	6	13.3%	13.3%
Uenig	38	84.4%	84.4%

11. De viktigste konseptene innenfor programmering kan forstås av elever på mellomtrinnet

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	36	80%	80%
Delvis enig	7	15.6%	15.6%
Nøytral	2	4.4%	4.4%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

12. På hvilket trinn bør elevene lære seg programmering?

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Ikke på grunnskolen	1	2.2%	2.2%
8.-10. trinn	2	4.4%	4.4%
5.-7. trinn	21	46.7%	46.7%
1.-4. trinn	20	44.4%	44.4%

1. Blokkprogrammering gjør det mulig å undervise programmering på mellomtrinnet

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	37	82.2%	82.2%
Delvis enig	5	11.1%	11.1%
Nøytral	2	4.4%	4.4%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

2. Blokkprogrammering gjør det enklere for elever som tidligere ikke har vist interesse for tekstprogrammering å lære seg programmering

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	34	75.6%	75.6%
Delvis enig	7	15.6%	15.6%
Nøytral	1	2.2%	2.2%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

3. Blokkprogrammering er egnet fordi oppgaver kan løses ved en lav inngangsterskel

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	37	82.2%	82.2%
Delvis enig	6	13.3%	13.3%
Nøytral	1	2.2%	2.2%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

4. Blokkprogrammering er egnet fordi oppgaver kan løses ved en stor takhøyde

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	28	62.2%	62.2%
Delvis enig	7	15.6%	15.6%
Nøytral	5	11.1%	11.1%
Delvis uenig	2	4.4%	4.4%
Uenig	0	0%	0%

5. Blokkprogrammering er mer sosialt, fordi løsningen på oppgaver kan enkelt deles med andre over nett

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	6	13.3%	13.3%
Enig	18	40%	40%
Delvis enig	11	24.4%	24.4%
Nøytral	8	17.8%	17.8%
Delvis uenig	2	4.4%	4.4%
Uenig	0	0%	0%

6. Tilbakemeldinger fra programmet som elevene jobber med gjør blokkprogrammering mer egnet enn tekstprogrammering

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	4	8.9%	8.9%
Enig	17	37.8%	37.8%
Delvis enig	13	28.9%	28.9%
Nøytral	9	20%	20%
Delvis uenig	1	2.2%	2.2%
Uenig	1	2.2%	2.2%

7. Blokkprogrammering gjør det mindre sannsynlig at elevene blir sittende fast med en oppgave

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	2	4.4%	4.4%
Enig	21	46.7%	46.7%
Delvis enig	18	40%	40%
Nøytral	3	6.7%	6.7%
Delvis uenig	1	2.2%	2.2%
Uenig	0	0%	0%

8. Blokkene gjør det enklere for å finne en mulig løsning på en oppgave

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	25	55.6%	55.6%
Delvis enig	17	37.8%	37.8%
Nøytral	1	2.2%	2.2%
Delvis uenig	1	2.2%	2.2%
Uenig	0	0%	0%

9. Elevene bør lære seg blokkprogrammering fordi det vil senere være enklere å lære seg tekstprogrammering

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	5	11.1%	11.1%
Enig	26	57.8%	57.8%
Delvis enig	8	17.8%	17.8%
Nøytral	4	8.9%	8.9%
Delvis uenig	2	4.4%	4.4%
Uenig	0	0%	0%

1. Programmering er et nyttig verktøy i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	2	4.4%	4.4%
Enig	27	60%	60%
Delvis enig	15	33.3%	33.3%
Nøytral	0	0%	0%
Delvis uenig	1	2.2%	2.2%
Uenig	0	0%	0%

2. Programmering gjør at elever engasjerer seg mer for matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	14	31.1%	31.1%
Delvis enig	19	42.2%	42.2%
Nøytral	7	15.6%	15.6%
Delvis uenig	1	2.2%	2.2%
Uenig	1	2.2%	2.2%

3. Programmering gjør at elever oppfatter matematikk mer nyttig utenfor matematikkfaget

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	14	31.1%	31.1%
Delvis enig	20	44.4%	44.4%
Nøytral	8	17.8%	17.8%
Delvis uenig	1	2.2%	2.2%
Uenig	1	2.2%	2.2%

4. Programmering gjør at elever utvikler algoritmisk tenkning

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	22	48.9%	48.9%
Delvis enig	19	42.2%	42.2%
Nøytral	3	6.7%	6.7%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

5. Programmering er et verktøy for å lære seg matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	18	40%	40%
Delvis enig	19	42.2%	42.2%
Nøytral	5	11.1%	11.1%
Delvis uenig	1	2.2%	2.2%
Uenig	1	2.2%	2.2%

6. Programmering i seg selv, er matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	2	4.4%	4.4%
Enig	19	42.2%	42.2%
Delvis enig	19	42.2%	42.2%
Nøytral	2	4.4%	4.4%
Delvis uenig	2	4.4%	4.4%
Uenig	1	2.2%	2.2%

7. Programmering er en egnet måte å uttrykke matematikk på

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	14	31.1%	31.1%
Delvis enig	23	51.1%	51.1%
Nøytral	4	8.9%	8.9%
Delvis uenig	0	0%	0%
Uenig	1	2.2%	2.2%

8. Man drar nytte av matematisk kompetanse når man skal lære seg å programmere

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	26	57.8%	57.8%
Delvis enig	16	35.6%	35.6%
Nøytral	1	2.2%	2.2%
Delvis uenig	0	0%	0%
Uenig	1	2.2%	2.2%

9. Man kan gjøre mer matematikk grunnet implementeringen av programmering

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	7	15.6%	15.6%
Enig	8	17.8%	17.8%
Delvis enig	19	42.2%	42.2%
Nøytral	6	13.3%	13.3%
Delvis uenig	3	6.7%	6.7%
Uenig	2	4.4%	4.4%

10. Man kan forstå sentrale konsepter i matematikken gjennom programmering

Antall svar: 44

Svar	Antall	% av svar	
Vet ikke	1	2.3%	2.3%
Enig	17	38.6%	38.6%
Delvis enig	22	50%	50%
Nøytral	3	6.8%	6.8%
Delvis uenig	0	0%	0%
Uenig	1	2.3%	2.3%

11. Man kan få en bedre forståelse av prosedyrer og algoritmer gjennom programmering

Antall svar: 44

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	30	68.2%	68.2%
Delvis enig	12	27.3%	27.3%
Nøytral	1	2.3%	2.3%
Delvis uenig	1	2.3%	2.3%
Uenig	0	0%	0%

12. Elever vil utvikle kompetanse på tvers av programmering og matematikk, altså vil kompetansen vil være overførbar

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	20	44.4%	44.4%
Delvis enig	18	40%	40%
Nøytral	3	6.7%	6.7%
Delvis uenig	1	2.2%	2.2%
Uenig	0	0%	0%

13. Elever vil ha vansker med å se likhetene i programmering og matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	4	8.9%	8.9%
Delvis enig	23	51.1%	51.1%
Nøytral	6	13.3%	13.3%
Delvis uenig	6	13.3%	13.3%
Uenig	6	13.3%	13.3%

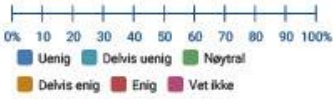
Hva mener du kjennetegner begrepet algoritmisk tenkning

- Dele større oppgaver i mindre/enklere oppgave som så er lettere å løse. Å lære av sine feil og prøve på nytt igjen er også viktig.
- - angripe systematisk. - steg for steg til problemet er løst - problemløsning
- Bryte ned til overkommelige mål.
- Systematisk Oppskrift Konsekvens
- Løse oppgaver systematisk Lære seg til å se ulike løsninger for å komme videre, jobbe steg for steg
- Gripe an problemer på en systematisk måte Finne mønstre i problemløsning Teknologisk kompetanse
- Prøve Holde ut Feilsøke
- se sammenhenger løse problemer med systematisk tenking Finne systematiske løsning
- Å kunne sette sammen deloperasjoner til en større helhet. Knytte matematiske operasjoner til noe fysisk, man får noe til å skje. Dele vanskelige regneoperasjoner inn i mindre, mer forståelige operasjoner, samtidig som man forstår helheten.
- Bryte ned et problem i mindre deler. Utforske. Logikk
- Bryte ned/dele opp, forstå og løse/komme med forslag.
- Få mere komplekse problemer ned til flere mindre problem, som lar seg løse - Tilnærmingen er systematisk - lærer seg å organisere og analysere
- Forstå hvilken regnemåte som egner seg i et regnestykke. Kunne anvende riktig regnemåte når man står ovenfor et matematisk problem. Mestre de fire regneartene.
- Dele problemer opp i delproblemer.
- - Se sammenheng mellom nye problemer og egen kunnskap. - "Så gjør vi så når vi vasker vårt tøy". - Kan (uheldigvis) bytte ut matematisk resonnering med gjentatt bruk av kjente løsningsmetoder.
- løse ett problem tankegang problemløsning
- 1. Jobbe systematisk/trinnvis. 2. Fokus på problemløsning. 3. Lage verktøy som gjør problemløsning enklere.
- Bryte ned oppgaven, slik at det blir mer oversiktlig. Lettere å løse oppgaven steg for steg.
- Bryte ned problemer Legge en plan for en løsning Sentral del av problemløsning
- Følge en oppskrift/kode/metode for å løse et problem Steg for steg, systematisk tenking, testing Mulig å gjennomføre tilfredsstillende uten noe særlig grad av refleksjon over hvorfor metoden virker
- - Det å kunne se sammenhenger og mønster - Kreativitet og problemløsning
- Gjøre et problem lettere. Lage algoritmer (oppskrifter) for å løse et problem.
- Algoritmisk tenking er fremgangsmåten en bruker for å løse et problem hvor svaret ikke er gitt. I algoritmisk tenkning er veien viktigere enn målet. Algoritmisk tenking hjelper elevene å samarbeide, reflektere og tenke kritisk.
- Lære om hvordan man gjør en operasjon, trenger ikke nødvendigvis å forstå hvorfor algoritmenvirker.
- Å tenke i steg og prosedyrer. Sortere operasjoner i en praktiskrekkefølge. Å dele store problem i mindre mer låsbare problem.
- Dele opp store problemer i mindre mer håndterlige delproblemer. Hjelper elevene å se ting i kontekst og bruke lært kunnskap i nye sammenhenger. Åpner opp for samarbeidslæring.

- Løse ulike problem, og tilnærme seg dem på en systematisk måte. Organisere og analysere informasjon på en logisk måte. Bryte ned komplekse problem til mindre mer håndterbare delproblemer som lettere lar seg løse.
- Å dele opp et komplekst problem i mindre delproblemer. Å lete etter løsninger på de mindre delproblemer. Å sette løsningene på delproblemer sammen for å løse det mer komplekse problemet.
- Logikk Systemer Matematikk
- Analysere, løse og formulere matematiske problemstilling. Generalisere eller finne mønstre i abstrakte problemer eller konsepter. Tilrettelegger for utforskning og utfordringer i undervisningen.
- steg for steg Presist Bryte ned helheten (problemet)
- -Å løse en oppgave eller et problem ved å tenke steg for steg -Å komme fram til overførbare oppskrifter for å løse en oppgave/et problem - å bryte en fremgangsmåte ned i flere små steg.
- Evnen til å se problemer som komponenter med flere ledd og som deler av en kjede Evnen til å sette sammen disse delene og flytte rundt på rekkefølgen i forsøk på å løse et problem Evnen til å analysere sammenhenger i en årsak-virkning- kontekst og utforske denne
- Problemløsningsmetode Feilsøking Steg for steg
- - problemløsningsmetode - systematisk - skapende
- Elevene kan tenke logisk og løse problemer stegvis. De kan eliminere unødvendig informasjon og finne møster. De kan vurdere eget arbeid.

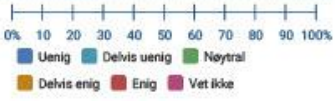
1. Det er viktig å kunne trekke ut de sentrale delene av et problem

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	0	4	40	0	
Matematikk	0	0	0	3	42	0	



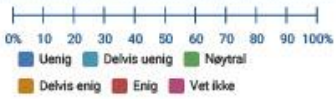
2. Det er viktig å kunne legge bort unødvendige deler av et problem

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	2	2	11	28	2	
Matematikk	0	3	1	12	29	0	



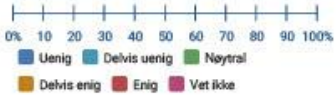
3. Det er viktig å kunne definere og følge instruksjoner steg for steg, mot en løsning

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	3	0	5	37	0	
Matematikk	2	5	0	11	27	0	



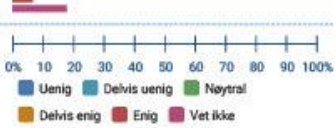
4. Det er viktig å kunne lage en løsning som kan forstås av et digitalt middel

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	5	9	30	1	
Matematikk	4	8	8	14	9	2	



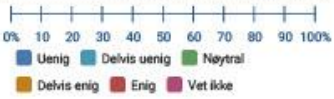
5. Det er viktig å kunne få et digitalt middel til å minimere det menneskelige bidraget i løsningen av et problem

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	6	8	9	8	7	7	
Matematikk	13	6	8	7	3	8	



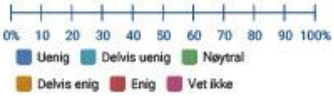
6. Det er viktig å kunne dele et hovedproblem opp i mindre deler og løse dem hver for seg

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	2	1	5	36	0	
Matematikk	0	2	1	5	36	0	



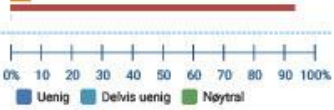
7. Det er viktig å kunne gjenkjenne mønster og koblinger mellom forskjellige problemer

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	0	5	40	0	
Matematikk	0	0	0	5	40	0	



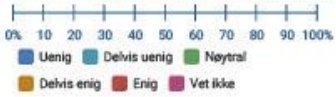
8. Det er viktig å kunne bruke kunnskap fra et tidligere løst problem, slik at det enklere kan brukes ved lignende problemer

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	1	1	4	39	0	
Matematikk	0	0	0	3	42	0	



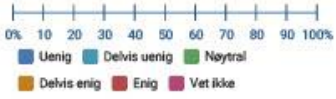
9. Det er viktig å kunne gjøre vurderinger av sitt eget arbeid

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	0	2	42	0	
Matematikk	0	0	0	2	42	0	



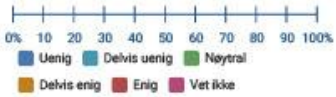
10. Det er viktig å kunne sjekke hvordan en løsning kunne vært mer effektiv og hva som er begrensninger ved løsningen

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	2	9	34	0	
Matematikk	0	0	3	13	29	0	



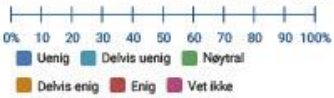
1. Det er viktig å kunne jobbe med problemer som virker komplekse

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	3	12	29	1	
Matematikk	0	0	1	11	31	1	



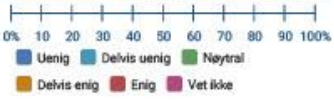
2. Det er viktig å kunne være utholden når man jobber med krevende oppgaver

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	1	2	42	0	
Matematikk	0	0	0	3	42	0	



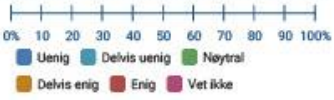
3. Det er viktig å kunne jobbe med oppgaver som kan ha flere løsningsmetoder og åpne svar

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	0	5	40	0	
Matematikk	0	0	0	1	43	0	



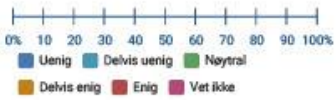
4. Det er viktig å kunne kommunisere og samarbeide mot et felles mål

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	1	1	5	38	0	
Matematikk	0	1	0	5	39	0	



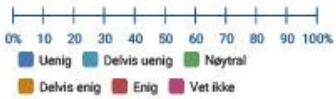
5. Det er viktig å kunne fikle med både konkreter og digitale midler for å fikse noe som ikke funker

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	2	1	6	35	1	
Matematikk	0	1	1	10	32	1	



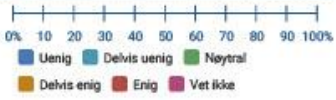
6. Det er viktig å kunne både designe og skape løsninger på problemer

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	1	7	36	1	
Matematikk	0	0	3	8	33	1	



7. Det er viktig å forstå feil og begrensninger av sin egen løsning på et problem, for deretter å rette opp i den

Svar	Uenig	Delvis uenig	Nøytral	Delvis enig	Enig	Vet ikke	Diagram
Programmering	0	0	0	5	39	1	
Matematikk	0	0	0	7	37	1	



1. I hvilket fag bør algoritmisk tenkning undervises?

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Ingen	0	0%	0%
Tverrfaglig	26	57.8%	57.8%
Et eget programmeringsfag	7	15.6%	15.6%
Matematikk	12	26.7%	26.7%

2. Programmering som verktøy kan gjøre algoritmisk tenkning mer konkret og tydelig for elevene

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	30	66.7%	66.7%
Delvis enig	11	24.4%	24.4%
Nøytral	2	4.4%	4.4%
Delvis uenig	2	4.4%	4.4%
Uenig	0	0%	0%

3. Algoritmisk tenkning bør kun læres ved bruk av programmering

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	8	17.8%	17.8%
Delvis enig	3	6.7%	6.7%
Nøytral	1	2.2%	2.2%
Delvis uenig	9	20%	20%
Uenig	24	53.3%	53.3%

4. Å tenke algoritmisk er kun nyttig når man jobber med programmering

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	1	2.2%	2.2%
Delvis enig	2	4.4%	4.4%
Nøytral	1	2.2%	2.2%
Delvis uenig	3	6.7%	6.7%

Uenig	38	84.4%	84.4%

5. Algoritmisk tenkning kan gjøre det enklere for både et menneske og en datamaskin å løse et problem

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	34	75.6%	75.6%
Delvis enig	8	17.8%	17.8%
Nøytral	2	4.4%	4.4%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

6. Algoritmisk tenkning gjør det enklere å kommunisere med en datamaskin, slik at den kan gjøre det du ønsker

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	32	71.1%	71.1%
Delvis enig	8	17.8%	17.8%
Nøytral	3	6.7%	6.7%
Delvis uenig	0	0%	0%
Uenig	1	2.2%	2.2%

7. Det er viktig at elever ikke bare er brukere av digitale midler, men har en forståelse for hvordan de fungerer

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	36	80%	80%
Delvis enig	5	11.1%	11.1%
Nøytral	2	4.4%	4.4%

Delvis uenig	1	2.2%	2.2%
Uenig	1	2.2%	2.2%

8. Det er viktig at elever ikke frykter å ta i bruk digitale midler

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	37	82.2%	82.2%
Delvis enig	4	8.9%	8.9%
Nøytral	2	4.4%	4.4%
Delvis uenig	1	2.2%	2.2%
Uenig	1	2.2%	2.2%

9. Algoritmisk tenkning er en viktig digital ferdighet (én av fem grunnleggende ferdigheter)

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	27	60%	60%
Delvis enig	8	17.8%	17.8%
Nøytral	5	11.1%	11.1%
Delvis uenig	1	2.2%	2.2%
Uenig	1	2.2%	2.2%

10. Algoritmisk tenkning ser jeg på som

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Allerede implementert i min undervisningspraksis	24	53.3%	53.3%
Nytt, men kompatibelt med min undervisningspraksis	17	37.8%	37.8%
Et tillegg, noe nytt i min undervisningspraksis	3	6.7%	6.7%

1. Jeg er entusiastisk om å undervise programmering i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	18	40%	40%
Delvis enig	21	46.7%	46.7%
Nøytral	2	4.4%	4.4%
Delvis uenig	2	4.4%	4.4%
Uenig	2	4.4%	4.4%

2. Jeg kan ha innflytelse på mine elevers læring i programmering i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	32	71.1%	71.1%
Delvis enig	12	26.7%	26.7%
Nøytral	1	2.2%	2.2%
Delvis uenig	0	0%	0%
Uenig	0	0%	0%

3. Jeg kan forklare viktige konsepter til mine elever innen programmering i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	23	51.1%	51.1%
Delvis enig	14	31.1%	31.1%
Nøytral	2	4.4%	4.4%
Delvis uenig	4	8.9%	8.9%

Uenig	1	2.2%	2.2%

4. Jeg kan hjelpe elevene mine å se nytten av programmering i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	25	55.6%	55.6%
Delvis enig	16	35.6%	35.6%
Nøytral	2	4.4%	4.4%
Delvis uenig	0	0%	0%
Uenig	1	2.2%	2.2%

5. Min egen kompetanse i programmering i matematikk er tilstrekkelig

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	5	11.1%	11.1%
Delvis enig	10	22.2%	22.2%
Nøytral	5	11.1%	11.1%
Delvis uenig	12	26.7%	26.7%
Uenig	13	28.9%	28.9%

6. Jeg kan planlegge variert undervisning avhengig av nivået på elevene i klassen i programmering i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	10	22.2%	22.2%
Delvis enig	15	33.3%	33.3%
Nøytral	3	6.7%	6.7%
Delvis uenig	9	20%	20%

Uenig	8	17.8%	17.8%

7. Jeg synes det er tydelig hva elevene skal lære ut ifra læreplanmålene i programmering i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	3	6.7%	6.7%
Enig	10	22.2%	22.2%
Delvis enig	16	35.6%	35.6%
Nøytral	5	11.1%	11.1%
Delvis uenig	8	17.8%	17.8%
Uenig	3	6.7%	6.7%

8. Slik programmering er i læreplanen nå, bør det gjøres endringer?

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	14	31.1%	31.1%
En større del av læreplanen	1	2.2%	2.2%
En litt større del av læreplanen	11	24.4%	24.4%
Slik det er i læreplanen er bra	16	35.6%	35.6%
En litt mindre del av læreplanen	1	2.2%	2.2%
En mindre del av læreplanen	2	4.4%	4.4%

9. Jeg har tilgang på de ressursene som kreves for at jeg skal undervise programmering i matematikk

Antall svar: 44

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	13	29.5%	29.5%
Delvis enig	13	29.5%	29.5%
Nøytral	1	2.3%	2.3%
Delvis uenig	11	25%	25%

Uenig	6	13.6%	13.6%

10. Jeg foretrekker å bruke lærestoff for programmering i matematikk...

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Annet	4	8.9%	8.9%
ved å lage det selv	20	44.4%	44.4%
gjennom sosiale grupper på nett	16	35.6%	35.6%
gjennom søk på nett	22	48.9%	48.9%
fra kolleger	24	53.3%	53.3%
fra betalte læreverk (både digitalt og i bøker)	26	57.8%	57.8%
fra faglitteratur eller forskning	19	42.2%	42.2%

11. Sammenhengen mellom programmering og matematikk er tydelig for meg

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	0	0%	0%
Enig	23	51.1%	51.1%
Delvis enig	17	37.8%	37.8%
Nøytral	2	4.4%	4.4%
Delvis uenig	2	4.4%	4.4%
Uenig	1	2.2%	2.2%

12. Jeg ønsker mer etterutdanning eller kursing for å kunne undervise programmering i matematikk

Antall svar: 45

Svar	Antall	% av svar	
Vet ikke	1	2.2%	2.2%
Enig	31	68.9%	68.9%
Delvis enig	6	13.3%	13.3%
Nøytral	2	4.4%	4.4%
Delvis uenig	3	6.7%	6.7%
Uenig	2	4.4%	4.4%