Høgskulen på Vestlandet

Matematikk 3, emne 4 - Masteroppgave

MGUMA550-O-2023-VÅR2-FLOWassign

Predefinert informasjon

Startdato:	02-05-2023 09:00 CEST	**Termin:**	2023 VÅR2
Sluttdato:	15-05-2023 14:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Masteroppgave - Bergen		
Flowkode:	203 MGUMA550 1 O 2023 VÅR2		
Intern sensor:	(Anonymisert)		

Deltaker

Kandidatnr.:	220

Informasjon fra deltaker

Antall ord *:	25954

Egenerklæring *: Ja

Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *: Ja

Jeg godkjenner avtalen om publisering av masteroppgaven min *

Ja

Er masteroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

Nei

Er masteroppgaven skrevet ved bedrift/virksomhet i næringsliv eller offentlig sektor? *

Nei

Høgskulen på Vestlandet

MASTEROPPGAVE

Hva karakteriserer det matematiske innholdet i programmeringsoppgaver i algebra?

En kvalitativ analyse av programmeringsoppgaver i algebra i Matemagisk og Campus Inkrement

What characterizes the mathematical content in programming tasks for algebra?

A qualitative analysis of programming tasks in algebra in Matemagisk and Campus Inkrement

Robert Pepaj

Master i undervisningsvitenskap med fordypning i matematikkdidaktikk

Grunnskolelærerutdanning 5.–10. trinn
Fakultet for lærerutdanning, kultur og idrett. Bergen
Veileder: Nils Henry Williams Rasmussen

Innleveringsdato: 15.05.2023

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle

kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 12-1*.

Forord

Da jeg i 2018 begynte på grunnskolelærerstudiet i Bergen virket masteroppgaven som noe som var svært lenge til. Nå er den tiden kommet, og fem år som lærerstudent har gått og studielivet går mot slutten. Alle minnene, erfaringen og kunnskapen jeg har fått i løpet av disse fem årene tar jeg med meg videre i livet og ser frem til å jobbe i et spennende yrke som lærer.

Masteroppgaven har vært en stor del av studiet, og jeg ønsker først og fremst å takke min veileder Nils Henry Williams Rasmussen ved Høgskulen på Vestlandets fakultet for lærerutdanning, kultur og idrett. Engasjementet og veiledningene har vært til mye hjelp i arbeidet med masteroppgaven og rådene og tilbakemeldingene jeg har fått har vært viktige for sluttresultatet. Jeg ønsker også å takke andre ansatte ved Høgskulen på Vestlandet for kunnskapen de har gitt meg og innsatsen de har lagt i å undervise oss. Jeg vil også rette en takk til Aschehoug og Campus Inkrement som har gitt meg lov til å bruke deres læreverk i masteroppgaven.

De viktigste minnene jeg tar med meg fra studietiden er fellesskapet og samarbeidet jeg har hatt med medstudenter, så jeg vil rette en stor takk til mine medstudenter for alle diskusjoner, kollokviegrupper, sosiale sammenkomster og arbeidsøkter gjennom studietiden. Disse fem årene og minnene jeg har skapt er noe jeg alltid vil huske.

Til slutt vil jeg også takke familie, samboer og venner for motiverende ord under arbeidet med masteroppgaven.

Robert Pepaj

Bergen, 15. mai 2023

Sammendrag

Denne masteroppgaven i matematikkdidaktikk er en studie hvor programmeringsoppgaver i algebra i læreverkene Matemagisk 8-10 og Campus Inkrement analyseres. Formålet er å finne ut hva som karakteriserer det matematiske innholdet som finnes i programmeringsoppgavene i algebra i disse læreverkene. Programmering ble for alvor innført i matematikkfaget med LK20. Problemstillingen for studien er: *Hva karakteriserer det matematiske innholdet i programmeringsoppgaver innenfor algebra i læreverk for ungdomstrinnene?* For å bidra til å svare på problemstillingen ble tre forskningsspørsmål også besvart i studien.

For å svare på problemstillingen ble det i studien benyttet en kvalitativ analyse i form av en dokumentanalyse som metode for å samle inn oppgaver til datamaterialet. Deretter ble en tematisk analyse benyttet for å analysere og kategorisere datamaterialet. Valg av koder og kategoriseringsprosessen for programmeringstyper i oppgavene var også basert på teori av Bråting og Kilhamn (2021b), og bidro til å svare på forskningsspørsmålet om hvilke sammenhenger som finnes mellom det matematiske innholdet og programmeringstypen i programmeringsoppgavene i algebra. Når datamaterialet var samlet inn og kategorisert, og programmeringsoppgavene i algebra var identifisert, tok jeg deretter utgangspunkt i Smith og Steins (1998) rammeverk for kognitive krav for å besvare forskningsspørsmålet: Hvilke kognitive krav stiller algebraiske programmeringsoppgaver i matematikk i to læreverk for ungdomstrinnene til elever? Rammeverket gir mulighet for å plassere matematikkoppgaver i fire forskjellige nivåer av kognitive krav og sier noe om hva slags matematikk oppgavene legger til rette for og hvordan oppgavene er designet for å løses. For å analysere og kategorisere det algebraiske innholdet i programmeringsoppgavene, ble Kierans (2004a) GTG-modell brukt. Rammeverket ble også brukt til å besvare forskningsspørsmålet: Hvilke algebraiske aktiviteter finnes blant programmeringsoppgavene i algebra i disse to læreverkene for ungdomstrinnene? Rammeverket tar for seg de tre algebraiske aktivitetene genererende, transformerende og global/meta-nivå, som Kieran argumenterer for at er aktivitetene som finnes i skolealgebraen.

Funnene i studien gav meg innsikt i hva som karakteriserer det matematiske innholdet som finnes i programmeringsoppgaver i algebra i de to læreverkene som er undersøkt. Jeg fikk innsikt i hvordan programmeringsoppgavene er designet og hvilke kognitive krav og algebraiske aktiviteter som finnes blant oppgavene. Til slutt ble funnene mine diskutert og knyttet til tidligere forskning på programmeringsoppgaver, kognitive krav og algebra.

Abstract

This master's thesis in mathematics didactics is a study in which programming tasks in algebra in the textbook Matemagisk 8-10 and digital textbook Campus Inkrement are analyzed. The purpose is to find out what mathematical content is found in the algebra programming tasks in these textbooks. Programming was seriously introduced in the mathematics subject with LK20. The problem for the study is: *What characterizes the mathematical content in programming tasks within algebra in textbooks for junior high school?* To help answer the problem, three research questions were also answered in the study.

To answer the problem, a qualitative analysis in the form of a document analysis was used in the study as a method for collecting tasks for the data material. A thematic analysis was then used to analyze and categorize the data material. The selection and categorization process for programming types in the tasks was also based on the theory of Bråting and Kilhamn (2021b) and was also used to answer the research question that looks at the coherence between mathematical content and programming types found in the programming tasks in algebra. When the data material had been collected and categorized, and the programming tasks in algebra had been identified, I then used Smith and Steins (1998) framework for cognitive demands to answer the research question: What cognitive demands are found in algebraic programming tasks in mathematics in two textbooks for students? The framework makes it possible to place mathematics tasks in four different levels of cognitive demands and says something about the kind of mathematics the tasks facilitate and how the tasks are designed to be solved. To analyze and categorize the algebraic content of the programming tasks, Kieran's (2004a) GTG-model was used. The framework was also used to answer the research question: Which algebraic activities are found among the programming tasks in algebra in these two textbooks? The framework addresses the three algebraic activities generational, transformational, and global/meta-level, which Kieran argues are the core activities of school algebra.

The findings in the study gave me insight in what characterized the mathematical content in programming tasks in algebra in the two textbooks that have been examined. I gained insight into how the programming tasks are designed and which cognitive demands and algebraic activities are found among the tasks. Finally, my findings were discussed and linked to previous research on programming tasks, cognitive demands, and algebra.

Innholdsfortegnelse

Bildeoversikt

Figuroversikt

Tabelloversikt

1. Innledning

Samfunnet er i stadig endring, og med det, kreves det også endringer i skolen, slik at skolens innhold er relevant for fremtiden til elevene. NOU-utredningen fra 2013 pekte på at det trengs økt kompetanse i programmering og at programmering bør integreres i matematikk (NOU 2013: 2, s.102). Skolen skal bidra til at elever tilegner seg kunnskap, og skolen skal forberede elevene på et fremtidig yrkesliv og det å leve i et samfunn (Opplæringslova, 1998, § 1-1). Allerede i 2013 vokste det frem en internasjonal bevegelse som ønsket programmering i skolen, dette støttet store firmaer som blant annet Google (Sevik et al., 2016, s.6). Under grunnleggende ferdigheter i kunnskapsløftet 2020, finner vi digitale ferdigheter. I matematikk innebærer dette blant annet å kunne bruke programmering til å løse og utforske matematiske problemer (Utdanningsdirektoratet, 2019).

I utredningen «Fremtidens skole» fra 2015 (NOU 2015: 8), nevnes fire kompetanseområder, disse er fagspesifikk kompetanse, kompetanse i å lære, kompetanse i å kommunisere, samhandle og delta, og kompetanse i å utforske og skape. Sevik et al. (2016, s.12) mener at programmering kan knyttes opp mot alle disse fire kompetanseområdene. Sevik et al. definerer programmering som «mer enn å bare skrive programkode som kjøres på en datamaskin», programmering inkluderer også «prosessen med å komme frem til denne koden». (Sevik et al., 2016, s.9).

I 2020 ble den nye læreplanen LK20 innført i skolen (Kunnskapsdepartementet, 2019). Utdanningsdirektoratet (2019) beskriver kjerneelementene som det «viktigste faglige innholdet elevene skal arbeide med i opplæringen». Under kjerneelementet «utforsking og problemløsing» i matematikk, er *algoritmisk tenkning* synliggjort (Kunnskapsdepartementet, 2019). Utdanningsdirektoratet (2019) beskriver algoritmisk tenkning som en måte å systematisk tilnærme seg problemer, ved å vurdere hvilke steg som trengs for å løse problemet, for så og deretter anvende teknologisk kompetanse for å løse problemet. Utdanningsdirektoratet oppnevnte en ekstern arbeidsgruppe til å skrive en rapport om programmering i skolen, «Teknologi og programmering for alle». Rapporten ble skrevet av Sanne et al. (2016). I digital kompetanse er kjernekompetansen algoritmisk tenkning, som Sanne et al. (2016, s. 14) skriver at kjennetegnes ved at en kan bruke programmering og dataprogrammer, til å løse matematiske problemer. Algoritmisk tenkning er derfor tett knyttet til programmering, og en kan da si at programmering er knyttet til kjerneelementene i form av algoritmisk tenkning.

I læreplanen for matematikk for ungdomsskolen, kan en for alle trinnene på ungdomsskolen finne et kompetansemål knyttet til programmering. For 8. trinn finner vi kompetansemålet «utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering» (Kunnskapsdepartementet, 2019). For 9. trinn ser vi også at programmering eksplisitt nevnes i kompetansemålet «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering» (Kunnskapsdepartementet, 2019). På 10. trinn, er kompetansemålet i programmering «utforske matematiske egenskaper og sammenhenger ved å bruke programmering» (Kunnskapsdepartementet, 2019). Programmering er en stor endring i matematikkfaget, og vi ser at det på er en viktig del av matematikken ved at det eksplisitt er nevnt i kompetansemålene.

Programmering ble dermed for alvor innført i matematikkfaget med den nye læreplanen i 2020. Det har i de siste årene kommet en del forskning om hvordan programmering kan bidra til utvikling av matematisk kompetanse, men nasjonalt finnes det lite forskning. Med ny læreplan og en nye arbeidsmåter i matematikk, nemlig programmering, kommer det også nye oppdaterte lærebøker og læreverk. Forsström og Kaufmann (2018) har forsket på programmering og har blant andre etterspurt forskning på programmering innenfor andre matematiske temaer enn geometri. Flere forskningsartikler gir anbefalinger for hvordan programmeringsaktiviteter burde være designet for at de skal legge best mulig til rette for læring. Bråting & Kilhamn (2021b) har gjort en studie på barneskolen i Sverige der de har analysert programmeringsoppgaver i læreverk. Forskerne kom frem til at designet av programmeringsoppgaver er viktig for å knytte sammen programmering og matematikk og legge til rette for læring. Slik forskning finnes ikke nasjonalt, og det er ikke blitt gjort slike analyser av programmeringsoppgaver på ungdomstrinnene.

Ark&App utførte i 2016 et forskningsoppdrag for å innhente kunnskaper om læremidler i skolen (Gilje et al., 2016). Formålet var å innhente kunnskap om hvordan læremidler velges, og hvorfor, samt hvordan læremidler har en innvirkning på læreres tolkning og operasjonalisering av læreplanen (Gilje et al., 2016). Det kom frem av undersøkelsen at det i hovedsak er lærere selv som velger læremidler, og at disse er viktige for lærernes tolkninger av kompetansemålene (Gilje et al., 2016, s. xiii). Det kom frem at majoriteten av lærere mener at læremidlene de bruker dekker kompetansemålene, men at de gjerne supplerer med tilleggsressurser hvis det er nødvendig (Gilje et al., 2016, s. xiv). Videre opplyser flest lærere om at papirbaserte lærebøker brukes mest i grunnskolen (Gilje et al., 2016, s. xv). Denne undersøkelsen er gjort i 2016, og i 2023 finnes det også flere digitale læreverk.

Fan et al. (2013, s. 635) har gjort en studie på lærebøker i matematikk og kom frem til at forskere generelt er enige om at det er lærebøker som er hovedformidler av læreplanen og spiller en dominerende rolle i undervisning på tvers av ulike skolefag. Videre viser Fan et al. (2013, s. 636) til egen tidligere forskning der det kom frem at lærere som bruker ulike typer læreverk, også hadde forskjellige undervisningsstrategier. Studien konkluderte med at lærebøker kan spille en rolle i læreres pedagogikk, ved at lærebøkene formidler forskjellige pedagogiske budskap. Videre peker Fan (2013) på at lærebøker er laget for å oversette det abstrakte ved læreplaner, til aktiviteter som lærere og elever kan bruke. Lærebøkene er et slags mellomledd mellom de som lager læreplaner og pensum, og lærere (Fan et al., 2013, s. 636). Dette viser at lærebøker spiller en stor rolle i klasserommene, noe mine egne praksiserfaringer og arbeidserfaringer også bekrefter. Derfor er det også viktig at det gjøres forskning på læreverk og lærebøker som brukes i skolen, da det er oppgaver og aktiviteter i disse mye av elevers arbeid i matematikk baserer seg på.

Programmeringsoppgaver i matematikk finnes både i læreverk og i tilleggsressurser. Når en som matematikklærer skal planlegge undervisning og hva det skal undervises i, har jeg erfart at det ofte er læreverk og lærebøker en søker til, slik som Fan et al. (2013) og Gilje et al. (2016) også argumenterer for. Forlagene lager lærebøker ut ifra kompetansemålene, og skolene velger selv hvilke lærebøker de ønsker å bruke. Nye læreplaner og nytt kompetanseområde i matematikk, programmering, fører til nye oppdaterte læreverk som inneholder programmeringsoppgaver, slik at kompetansemålene dekkes. Nå som programmering er integrert i matematikkfaget, er det ikke tvil om at programmering vil ta plass og tid i matematikkfaget. At programmering har fått plass i matematikkfaget, har reist og kan reise nye spørsmål, som for eksempel hva elevene lærer av programmering i matematikk, og om det går på bekostning av andre ting i matematikkfaget. For å lære programmering, må det selvfølgelig vies tid til dette.

Blikstein (2018) har undersøkt programmering i skolen i flere land, og kom i sin undersøkelse frem til at nå som flere og flere land innfører programmering i skolen, etterspørres mye forskning. Mange spør seg om hvilket fag programmering burde implementeres i, og når det implementeres i matematikk, lurer mange på hva man lærer rent matematisk sett. Selv om programmering er tett knyttet til matematikk, slik som fysikk, er det ikke tradisjonelt sett på som en del av matematikken. Det er grunnen til at det ikke er åpenbart at det skal inkluderes i skolematematikken. Sverige har, slik som i Norge, innført programmering som en del av matematikkfaget, men i Sverige er programmering innført som et aspekt av algebra (Bråting

& Kilhamn, 2021b). Bråting & Kilhamn (2021a) spør seg om vi skal fortsette veien mot å bruke programmeringsspråk som for eksempel Scratch, bare fordi det kan føre til algoritmisk tenkning, eller om vi bør etterlyse programmeringsspråk og oppgaver som spesifikt er utviklet for matematisk utforskning. Det er altså flere som reiser spørsmål ved integrering av programmering i matematikkfaget.

Tidligere forskning viser at blant annet designet av programmeringsoppgaver i læreverk er viktig for hvorvidt elevene får et læringspotensial. Samtidig viser forskning som nevnt tidligere i innledningen at lærere støtter seg mye til læreverkene, og at det dermed er læreverkene elevene gjør oppgaver i og bruker i undervisningen. Det er stor etterspørsel etter mer forskning på programmering i matematikkfaget (Blikstein, 2018; Bråting & Kilhamn, 2021a; Forsström & Kaufmann, 2018). Som vi ser av forskningen til Fan et al. (2013) og undersøkelsen til Gilje et al., (2016), er lærere veldig tilknyttet læreverkene. Det er dermed nødvendig å ha læreverk som implementerer programmering på en god måte. Det er også nødvendig å finne ut hva slags matematikk elever møter på under arbeid med programmeringsoppgaver. På bakgrunn av at programmering nå er blitt en del av matematikkfaget i Norge, etterspørselen i forskningsfeltet og læreverkenes sentrale rolle i undervisningen, ønsker jeg derfor i denne studien å undersøke det matematiske innholdet i programmeringsoppgaver i norske læreverk for ungdomstrinnet.

1.1 Problemstilling og forskningsspørsmål

I denne masteroppgaven vil jeg undersøke og analysere algebraoppgaver innenfor programmering i matematikk, i de to utvalgte læreverkene for ungdomsskolen. Målet med studien er å kvalitativt analysere programmeringsoppgaver innenfor algebra i matematikk for ungdomstrinnene, for å finne ut hva som karakteriserer det matematiske innholdet blant programmeringsoppgavene, i form av hva slags nivå av kognitive krav disse algebraoppgavene i programmering stiller til elever, og hvilke algebraiske aktiviteter som finnes i oppgavene, og hvordan ulike typer programmering kan påvirke oppgavene eller matematikken. Målet er at dette til slutt skal gi et bilde av hva som karakteriserer matematikken algebraoppgaver innenfor programmering inneholder, og hvordan programmering, matematisk algebraisk innhold og kognitive krav henger sammen og kan påvirke innholdet i oppgaver. Det er interessant og nødvendig å finne ut hva elevene møter på, og hva slags matematikk programmeringsoppgavene legger til rette for. Samtidig bidrar studien også til et relativt nytt forskningsfelt og forskning som er etterspurt.

Da det ville blitt for omfattende å utføre en grundig analyse av alle oppgaver i læreverkene, ble programmeringsoppgaver innenfor et spesifikt matematisk tema valgt ut i studien. Valget falt på at programmeringsoppgaver i algebra skulle analyseres nærmere, dette valget ble tatt basert på tidligere forskning av blant annet Mason (2018) som argumenterer for at nyere programmeringsspråk har likhetstrekk med og stammer fra algebra. Bråting & Kilhamn (2021a) etterlyser mer forskning på programmering og algebra. I Sverige er programmering innført som et aspekt av algebra i matematikkpensumet (Bråting & Kilhamn, 2021b). Forsström og Kaufmann (2018) etterlyste også forskning på programmering innenfor andre temaer enn geometri. Dermed har jeg begrenset omfanget av oppgaven ved å la problemstillingen handle om programmeringsoppgaver i algebra.

Dermed er denne studiens overordnede problemstilling:

> «*Hva karakteriserer det matematiske innholdet i programmeringsoppgaver innenfor algebra i læreverk for ungdomstrinnene*»

Med «det matematiske innholdet» i denne studien menes de algebraiske aktivitetene og kognitive kravene som finnes blant algebraoppgaver innenfor programmering. Med algebraiske aktiviteter menes de tre forskjellige algebraiske aktivitetene «generational», «transformational» og «global/meta-level» som elevene møter på i skolealgebraen (Kieran 2004a). Hver aktivitet har forskjellige kjennetegn, og Kieran (2004a) skriver at de algebraiske aktivitetene er hovedaktivitetene i skolealgebraen, og at alle tre er nødvendige for læring. Med kognitive krav menes hva slags tenkning eller nivå av tenkning oppgavene krever av elever (Stein et al., 2000, s.11). De kognitive kravene i en oppgave kan kategoriseres ut ifra kjennetegn gitt av rammeverket til Smith og Stein (1998), og Smith og Stein argumenterer for at kognitive krav er viktige for om oppgavene legger godt til rette for læringsutbytte hos elever.

Jeg vil dermed bidra til å besvare problemstillingen ved å svare på disse tre forskningsspørsmålene:

- Hvilke kognitive krav stiller algebraiske programmeringsoppgaver i matematikk i to læreverk for ungdomstrinnene til elever?
- Hvilke algebraiske aktiviteter finnes blant programmeringsoppgavene i algebra i disse to læreverkene for ungdomstrinnene?
- Hvilken sammenheng er det mellom det matematiske innholdet og programmeringstypene i programmeringsoppgavene i algebra?

Resultatene fra denne studien vil kunne bidra til en større forståelse for hvordan programmeringsoppgaver i algebra i læreverk er utformet og designet. Resultatene vil og kunne si noe om programmeringstyper i oppgaver har sammenhenger med det matematiske innholdet. Jeg skal forsøke å svare på problemstillingen ved å foreta en analyse av programmeringsoppgavene i læreverkene Matemagisk 8-10 og Campus Inkrement. Det er det matematiske innholdet og programmeringstypene i oppgavene som skal analyseres, og formålet er at resultatene skal si noe om hva som karakteriserer det matematiske innholdet som finnes i programmeringsoppgavene i algebra i form av kognitive krav og algebraiske aktiviteter i disse læreverkene. Resultatene skal og si noe om det potensielt finnes noen sammenhenger mellom det matematiske innholdet og programmeringstypen i oppgavene.

1.2 Oppbygging av oppgaven

Første kapittel består av innledningen der jeg presenterer valg av tema, problemstilling og forskningsspørsmål. Kapittel 2 er kapittelet for tidligere forskning og teoretisk rammeverk, her presenteres tidligere forskning gjort på programmering i matematikk og relevant forskning i henhold til problemstillingen og tema, blant annet på algebra og kognitive krav. I dette kapittelet presenteres også det teoretiske rammeverket som videre brukes for å danne analyseverktøyet som brukes i analysen av oppgavene i læreverkene i studien. Kapittel 3 er metodekapittelet, her presenteres valg av analysemetoder, analysemetodene, forskningsdesign, læreverkene og oppgavene som er valgt ut i studien. Til slutt tar jeg for meg etiske betraktninger og kvaliteter ved studien. Kapittel 4 er analyse og resultatkapittelet, her viser jeg til eksempler på hvordan jeg har analysert, deretter presenteres funnene fra analysene samlet i samme kapittel. Kapittel 5 er diskusjonskapittelet, her knyttes funnene fra kapittel 4 opp mot teori og tidligere forskning som er presentert i kapittel 2, og funnene diskuteres. Til sist kommer avslutningen i kapittel 6, med blant annet konklusjon og forslag til fremtidig forskning på feltet.

2. Tidligere forskning og teoretisk rammeverk

I dette kapittelet vil jeg redegjøre for tidligere forskning innenfor programmering i matematikk og analyse av programmeringsoppgaver i matematikk. Jeg vil også legge frem forskning på algebra i skolematematikken, algebra i programmering, algebraiske aktiviteter og tidligere studier gjort på kognitive krav i algebraoppgaver. Kapittelet tar først for seg programmering generelt, og deretter forskning gjort på programmering og programmering i skolen. Deretter presenteres tidligere forskning knyttet til algebra og kognitive krav i

oppgaver. Til slutt presenterer jeg det teoretiske rammeverket for programmeringstyper, algebraiske aktiviteter og kognitive krav som skal brukes for den videre analysen og det analytiske rammeverket i studien.

2.1 Tidligere forskning

2.1.1 Programmering i matematikk

Noen av de første internasjonale studiene på programmering i matematikk i skolen hevdet at programmering vil være til fordel for elevers læring i problemløsing og generalisering (Kieran & Yerushalmy, 2004, s.101). Programmering ble tidlig sett på som en algebraisk aktivitet, det var Papert sin LOGO-programmering som bidro til dette. Dette er fordi programmering innebar å uttrykke matematiske ideer og prosesser med et bestemt språk, på en generell måte, og med en tilhørende syntaks, slik som i algebra (Kieran & Yerushalmy, 2004, s.101). Papert mente at programmering bidro til at elevene utviklet matematisk forståelse gjennom testing og feilsøking, men ideene hans hadde ikke en stor innflytelse på 80- og 90-tallet, det kan være fordi teknologien ikke hadde kommet like langt som i dag (Bråting & Kilhamn, 2021a, s. 170). Nå ses programmering på som en ferdighet som er svært viktig i samfunnet.

Forsström og Kaufmann (2018) har gjort en studie i form av en litteraturgjennomgang for å undersøke hva slags potensiale programmering har i matematikkfaget. I og med at programmering med nyere tid blir sett på som en grunnleggende ferdighet for å kunne delta i den digitale verden, har det med tiden blitt en større interesse for å implementere programmering som en del av skolehverdagen (Grover & Pea, 2013). Prosessene som inngår i programmering, har blitt knyttet til matematisk tenkning, og flere europeiske land mener at siden programmering er tett knyttet til algoritmisk tenkning (Grover & Pea, 2013), så er det en viktig ferdighet også i forhold til problemløsning, logisk tenkning og kreativitet (Forsström & Kaufmann, 2018). Det har vært mye anbefalinger om hvor programmering kan passe inn i skolen, men liten konsensus på om det skal implementeres på tvers av fagene eller som et eget fag (Grover & Pea, 2013, s. 40).

Det vanligste vi har sett er at programmering har blitt implementert i andre fag, for det meste matematikk, og dette ved å bruke tverrfaglige tilnærminger (Balanskat & Engelhardt, 2015). Paperts LOGO-programmering var det første programmeringsspråket som ble introdusert i matematikkfaget, men forskningen på læringsutbytte av den var usikker. Noen studier viste at det var positivt læringsutbytte hos elever, mens andre studier viste at det ikke var noen forbedring i problemløsningsferdigheter eller læringsutbytte i matematikk hos elever etter å ha

vært gjennom LOGO-programmeringsprosjekter (Forsström & Kaufmann, 2018, s. 20). Senere studier på programmering i matematikk har også vist slike resultater (Forsström & Kaufmann, 2018).

Gjennomgangen til Forsström og Kaufmann (2018) viste at det finnes lite forskning på potensialet programmering har i matematikkundervisningen, mange studier viser til positive resultater som følge av programmering i matematikk, men generaliserbarheten til studiene er ikke tydelig. Forsström og Kaufmann (2018) etterlyser mer forskning på hvorfor programmering er inkludert i matematikk, og sammenhengen mellom programmering og matematikk. Dette gjør de på bakgrunn av at det er blitt så vanlig å inkludere programmering i matematikkfagene i Europa, slik som i Norge (Forsström & Kaufmann, 2018, s. 28-29). Et resultat de kom frem til var at det kun var geometri som klart kunne knyttes til programmering på bakgrunn av forskningen de hadde undersøkt, men hva elevene gjør når de arbeider med programmering er avgjørende. De ber samtidig om mer forskning på andre matematiske temaer enn geometri knyttet til programmering (Forsström & Kaufmann, 2018, s. 30), for eksempel algebra.

Ye et al. (2023) har gjort en systematisk litteraturgjennomgang der de har undersøkt karakteristikkene ved matematisk undervisning der algoritmisk tenkning ved programmering er i fokus. Dette har de gjort for å se hva elever kan lære av programmering, og se på sammenhengen mellom algoritmisk tenkning og det matematiske læringsutbytte. Av studiene Ye et al. (2023) har undersøkt, er det gjort flest studier på barnetrinnene, men også en del på ungdomsskolen. De to mest brukte programmeringsspråkene i studiene som ble undersøkt var tekstbasert programmering, for eksempel Pythonprogrammering som vi finner i Matemagisk, og blokkbasert programmering, som vi finner i Campus Inkrement. Ye et al. (2023, s. 21) kom frem til at det å jobbe med programmeringsoppgaver i matematikk ikke bare krever anvendelse av matematisk kunnskap, men at det også bidrar til fremveksten av ny matematisk kunnskap. Forskerne identifiserte tre måter programmering kan legge til rette for at nye matematiske kunnskaper oppstår hos elever under arbeid med programmeringsoppgaver (Ye et al., 2023, s. 21).

Ye et al. (2023, s. 21) argumenterer for at den ene måten elever kan danne nye matematiske ideer og sammenhenger på under arbeid med programmeringsoppgaver, er ved å reflektere over «outputs» i programmering, nemlig det programmer skriver ut som svar eller viser som resultat. Programmering gir muligheter for at elever effektivt kan lage mange utfall og eksempler, og hvis elevene deretter blir satt til å reflektere, generalisere og undersøke

matematiske egenskaper ved resultatene, kan det bidra til at ny matematisk kunnskap dannes (Ye et al., 2023, s. 22). En annen måte matematiske ideer konstrueres på under arbeid med programmeringsoppgaver, er ved refleksjon av programmeringsprosessen og egenskaper ved koder (Ye et al., 2023, s. 22). Elever kan ved å blant annet observere mønstre i programmeringskoder, se sammenhenger med matematikk og formulere matematiske regler ved hjelp av programmering, få et matematisk læringsutbytte (Ye et al., 2023, s. 22).

Ye et al. (2023, s. 22) skriver videre at programmeringsbasert undervisning også kan føre til en ny form for matematisk kunnskap som representeres av programmeringsspråk. Dette kom frem i Cui og Ng (2021) sin studie på elevers evne til problemløsning under arbeid med programmering. Cui og Ng (2021, s. 858) viser til en gruppe elever som ikke hadde noen forkunnskaper innenfor algebraiske uttrykk og variabler, men som klarte å bruke kommandoer og koder til å modellere matematiske sammenhenger ved programmering, men ikke ved matematisk notasjon. De mener dermed at programmering kan gjøre vanskelige ideer tilgjengelig for elever, og at programmering kan virke som en bro mellom aritmetikk og det å bygge forståelse for avansert algebra (Cui & Ng, 2021, s. 858-859).

Kort oppsummert viser gjennomgangen til Ye et al. (2023) at programmering og algoritmisk tenkning i matematikkundervisningen støtter abstraksjon av matematiske ideer, gir muligheter for å undersøke matematisk innhold ved å reflektere over programmeringskoder og resultater av programmering, samt at det gir en mulighet for å lære matematikk på en annerledes måte. Det kommer spesifikt fram av studien at integreringen av algoritmisk tenkning og programmering i matematikkundervisningen har vært støttende for algebra og geometri for alle klassetrinn (Ye et al., 2023, s. 23). Programmering og algoritmisk tenkning i undervisningen åpner også opp for at flere matematiske sammenhenger skal dannes, og bidrar til fremveksten av en ny form for matematisk kunnskap og resonneringsmetode, som legger grunnlaget for elevers fremtidige matematiske abstraksjon og utvikling (Ye et al., 2023, s. 23).

2.1.2 Analyse av programmeringsoppgaver i matematikk

Etter det vi kjenner til av forskning på programmering på ungdomstrinnet, finnes det ikke studier der det er blitt gjort analyser av programmeringsoppgaver i matematikk for ungdomstrinnene, og det er ikke gjort forskning på eller analyse av programmeringsoppgaver i matematikk i Norge. Dette har derimot blitt gjort i Sverige. Bråting og Kilhamn (2021b) har gjort en studie der de har analysert programmeringsinnholdet som ble lagt til i svenske lærebøker på barneskolen i 2020. Målet med studien var å finne ut hva som karakteriserer

programmeringsinnholdet i svenske lærebøker for barnetrinnene og deretter diskutere hvordan innholdet påvirker elevers muligheter til å lære matematikk og hva slags sammenhenger en finner mellom programmering og matematikk. Bråting og Kilhamn (2021b, s. 595) skriver at det å studere programmeringsinnholdet i lærebøkene kan hjelpe til med å finne ut hvilke læringsmuligheter elevene får. Bråting og Kilhamn (2021b) analyserte i studien programmeringsoppgaver kvalitativt ved å se på og sammenligne begreper innenfor programmering og innenfor matematikk som dukker opp i oppgavene, og samtidig knytte dette til programmeringstypene og prosedyrene oppgavene setter elevene til å bruke. De så altså på hvilken type programmering som finnes blant oppgavene, og hvilket matematisk innhold som finnes i oppgavene, samt hvordan oppgavene skal løses.

Bråting og Kilhamn (2021b, s.607) kom frem til at flertallet av oppgavetypene i lærebøkene bestod av like fremgangsmåter eller lik type programmering. Programmeringstypen som forekom oftest var av typen der elever skulle følge en prosedyre (følg), ofte går dette ut på at elevene skal programmere noe etter gitte instrukser, altså følge prosedyrer. Blant programmeringsoppgavene Bråting og Kilhamn undersøkte, var målet med oppgavetypen «følg» å oppnå et spesifikt mål. Det var få oppgaver som krevde fremgangsmåtene «forestill deg», der elevene for eksempel skal forutsi hva en programmeringskode gjør, «feilsøk», der elevene skal feilsøke programmeringskoder og «forklar», der elevene skal forklare programmeringskoder eller hva de har gjort, eksempelvis ved bruk av matematisk språk eller notasjon. Dette begrenset muligheten programmering har til å lære elever matematikk, blir det argumentert for (Bråting & Kilhamn, 2021b). Bråting og Kilhamn (2021b, s. 607) påpeker videre at feilsøking er noe av det fundamentale for å lære programmering, og mener at det gir gode muligheter til å lære matematikk. Bråting og Kilhamn (2021b) mener at det er stort potensiale til å få en større bredde i typen programmeringsoppgaver som finnes i lærebøkene, for å gjøre programmering rikere og danne en klarere sammenheng mellom programmering og matematikk. De påpeker spesielt dette med testing og feilsøking, i og med at det spesifikt er nevnt som en måte å lære matematikk på i tidligere forskning, og viser til blant annet Paperts LOGO-programmering (Bråting & Kilhamn, 2021b, s. 607), der feilsøking var sentralt for å lære matematikk.

Bråting og Kilhamn (2021a) har også gjort en studie på algebraisk tenkning knyttet til algoritmisk tenkning ved programmering. Det de ønsket og forsøkte å svare på var hvilke likheter og forskjeller en finner i algebra ved programmeringsspråk og ved algebraisk notasjon, og hvordan integrereringen av programmering i skolematematikken potensielt kan

fremme eller hemme elevers utvikling av algebraisk tenkning (Bråting & Kilhamn, 2021a, s. 171). Bråting og Kilhamn (2021a, s. 171-172) skriver at programmering er et nytt system med representasjoner brukt til å representere matematiske objekter og strukturer, men kom frem til at de representerte objektene ikke alltid er matematiske, og den involverte tenkningen ikke alltid er algebraisk. De argumenterer videre for at de semiotiske representasjonene i programmeringsspråk, altså matematiske tegn og symboler, ligner tegnene og symbolene ved matematisk notasjon, men at det samtidig finnes forskjeller mellom disse (Bråting & Kilhamn, 2021a). Forskerne mener at disse forskjellene må tas i betraktning i et læringsperspektiv, spesielt med tanke på at elever må konvertere mellom disse to registrene i matematikken når det arbeides med programmering (Bråting & Kilhamn, 2021a).

Bråting og Kilhamn (2021a) fant i studien sin ut at programmeringsoppgaver som kan fremme algebraisk bevissthet også kan føre til at elevers fokus tas bort fra algebra, og de ønsker mer forskning på hva som kan gå tapt hvis programmeringsspråk tas i bruk fremfor algebraisk notasjon. De fant også forskjeller i hvordan matematiske symboler brukes ved programmeringsspråk og ved matematisk notasjon. De peker også på det de kaller en ironisk vending, nemlig at programmeringsprogrammer og språk som er utviklet for andre formål enn å utforske matematikk, brukes som hovedkilde til å hente inspirasjon for programmeringsoppgaver i skolematematikken, og spør seg om vi skal bruke for eksempel blokkbasert programmering som Scratch bare fordi det er en måte å tenke algoritmisk på (Bråting & Kilhamn, 2021a, s. 182). Funnene til Bråting og Kilhamn i denne studien gjør det interessant å undersøke hva som karakteriserer det matematiske innholdet og hva slags algebraisk innhold som finnes i programmeringsoppgaver.

Wu og Yang (2022) har i en litteraturgjennomgang kommet frem til at det finnes lite forskning på forholdet mellom matematisk og algoritmisk tenkning. De kom også frem til at oppgavene de har undersøkt i forskningen ser på algoritmisk tenkning som en problemløsningsprosess ved bruk av programmering, og matematisk tenkning som det matematiske innholdet i programmeringsoppgavene (Wu & Yang, 2022, s. 16). Et annet funn Wu og Yang (2022, s. 16) kom frem til som samsvarer med funnet til Bråting og Kilhamn (2021b) når de undersøkte programmeringsoppgaver i lærebøker i Sverige, er at det var lite oppgaver der elevene skulle feilsøke koder.

2.1.3 Oppgaveanalyse i matematikk

Det finnes som nevnt lite forskning der det er blitt gjort analyse av programmeringsoppgaver i matematikk, men det er blitt gjort en god del forskning på analyse av oppgaver i matematikk

generelt. Stein et al. (1996, s. 456) argumenterer for at hvis elever skal kunne forstå matematikk på best mulig måte, burde de få oppgaver som er problembaserte, og ikke prosedyrepregede oppgaver som fører til gjenbruk av algoritmer elevene allerede kan. Det som kjennetegner problembaserte oppgaver er at elevene må ta valg basert på hva som må gjøres og hvordan det skal gjøres, de har flere løsningsmetoder, de har flere representasjonsformer og det kreves at elevene forklarer og rettferdiggjør deres prosedyrer muntlig eller skriftlig (Stein et al., 1996, s. 456). Videre skriver Stein et al. (1996, s. 456) at noe som hemmer matematisk tenkning er matematikkundervisning der elever får presentert et problem og en algoritme som kan brukes for å løse problemet, og deretter får like oppgaver som de individuelt skal jobbe seg gjennom ved å bruke denne algoritmen. Dette fører til at elever memorerer fakta, eller bruker formler, algoritmer og prosedyrer uten å vite hvorfor eller hvordan de er gunstige å bruke i fremtidige oppgaver (Stein et al., 1996, s. 457). Oppgaver elevene presenteres for, er med på å bestemme hva de lærer, og hvordan de tenker om, utvikler, bruker og forstår matematikk (Stein et al., 1996).

Smith og Stein (1998, s. 344) har i sin klasseromsstudie undersøkt faktorer som har innvirkning på elevers læring i og inntrykk av matematikk, de kom blant annet frem til at oppgaver elevene får, har betydning for hva elevene lærer, og at oppgaver som krever et høyt nivå av kognitiv tenkning, fører til et større læringsutbytte hos elevene. Oppgaver som ber elever utføre memorerte prosesser rutinemessig, fører til en type tenkning, og oppgaver som krever at elever tenker begrepsmessig og ser sammenhenger, fører til flere og helt andre tankeprosesser (Smith & Stein, 1998, s. 345). Stein et al. (1996, s. 472) skriver at samtidig som det er nødvendig med noen oppgaver der de kognitive kravene er lavere, slik at elever får øvd på prosedyrer og rutiner før de møter på komplekse oppgaver, er det ønskelig at flertallet av oppgavene stiller høye kognitive krav, fordi dette gir muligheter for et større læringsutbytte.

Ubuz et al. (2010) har gjort en studie på algebraoppgaver i lærebøker i Tyrkia. Studien undersøkte kognitive krav i algebraoppgaver ved å bruke rammeverket til Smith og Stein (1998). Totalt 72 oppgaver ble undersøkt, fra lærebøker for 6., 7. og 8. trinn. Studien kom frem til at rundt 60 prosent av algebraoppgavene stilte høye kognitive krav til elevene, og de resterende 40 prosent stilte lavere kognitive krav til elever (Ubuz et al., 2010, s. 489). Fordelingen mellom høye og lave kognitive krav var nokså lik mellom trinnene. Den største forskjellen som ble identifisert, var at det på 8. trinn, sammenlignet med 6. og 7. trinn, var cirka 20 prosent flere oppgaver innenfor kategorien «å gjøre matematikk» (Ubuz et al., 2010).

Dette anses som det høyeste nivået av kognitive krav, og disse oppgavene har potensiale til å føre til størst læringsutbytte hos elever (Stein et al., 1996). Ubuz et al. (2010) kom på 8. trinn frem til at 7 prosent av oppgavene de undersøkte lå under memorering, 33 prosent under prosedyrer uten forbindelser, 39 prosent under prosedyrer med forbindelser og 21 prosent under gjøre matematikk.

Yang og Sianturi (2022) har også gjort en tilsvarende analyse av kognitive krav i algebraoppgaver i lærebøker for barneskolen i Finland, Indonesia, Malaysia, Singapore og Taiwan. Forskerne har i studien også analysert de kognitive kravene oppgavene stiller til elever ved å også bruke Smith og Steins (1998) rammeverk. Jeg har fra denne studien valgt å vise til resultatene fra Finland, basert på at det føles mer generaliserbart til Norge og mer relevant enn resultater i andre verdensdeler. Yang og Sianturi (2022, s.86) kom frem til 23 prosent memoreringsoppgaver, 52 prosent under prosedyrer uten forbindelser, 24 prosent under prosedyrer med forbindelser og 1 prosent under gjøre matematikk.

I tillegg til forskning på kognitive krav, er det blitt gjort forskning på matematisk algebraisk innhold i oppgaver. Kieran har gjort mye forskning på algebra i skolematematikken. I sin modell for algebraiske aktiviteter, beskriver Kieran (2004a) tre kategorier av aktiviteter i algebraoppgaver. Disse aktivitetene er transformerende (transformational), genererende (generational) og globale (global/meta-level) algebraiske aktiviteter. Lærebøker har tradisjonelt lagt vekt på transformasjonsaspektene ved algebraiske aktiviteter hevder Kieran (2004a), og oppmerksomheten er ofte rettet mot reglene som må følges for å manipulere symbolske uttrykk og ligninger, fremfor den begrepsmessige forståelsen som støtter disse reglene (Kieran, 2004a, s. 24). De transformerende oppgavene kjennetegnes ved oppgaver der elever for eksempel skal løse ligninger for å bevare ekvivalens. Videre skriver Kieran (2004a, s. 31) at de globale aktivitetene som bevisføring og å se strukturer har blitt forsømt i skolealgebraen, selv om disse er viktige.

Kieran (2004b, s.148) skriver også at de globale aktivitetene er tett knyttet til algebraisk tenkning og bidrar til matematisk og algebraisk forståelse, de er også viktige for å utvikle måter å tenke på som er kritiske for å lykkes med algebra. Samtidig er de globale aktivitetene svært viktige for å kunne lykkes med de transformerende og genererende aktivitetene i algebra (Kieran, 2004b). Wilson, Ainley og Bills (2003) har i sin studie sammenlignet transformerende og genererende algebraiske aktiviteter. Studien kom frem til at de genererende aktivitetene ofte er vanskeligere for elever enn de transformerende aktivitetene, og mener at spesielt de genererende, men også de globale aktivitetene, burde få en større plass

i skolealgebraen, da det er disse som skaper mening ved og hjelper på forståelsen av de transformerende aktivitetene og algebra.

2.1.4 Algebra og digitale verktøy

Kieran (2017, s. 80) har sett på struktur og strukturelle aktiviteter i algebra i matematikken, og nevner at det å se på noe strukturelt er et oversett aspekt ved algebraisk tenkning. Struktur er en stor del av matematikkfaget, og algebraisk tenkning involverer elevers evne til å bygge, rettferdiggjøre og uttrykke formodninger om matematisk struktur og matematiske relasjoner (Kieran, 2017, s. 80). Essensen i tidlig algebra er å generalisere matematiske ideer, og samtidig rettferdiggjøre og representere generalisering på flere måter, og innenfor aritmetikk innebærer generalisering å identifisere det strukturelle (Kieran, 2017, s. 81). Radford (2015, s. 216-217) har i sin forskning kommet frem til at erfaring med strukturering er viktig i utviklingen av algebraisk tenkning, noe for eksempel oppgaver med figurtallmønster bidrar til, og argumenterer videre for at det er blitt gjort for lite forskning på måten dataverktøy kan utnyttes i utviklingen av strukturell tenkning på dette området. Dette viser Kieran (2017, s. 88) til, og skriver videre at dataprogrammer kan utnyttes i utviklingen av strukturell tenkning når en jobber med struktur innenfor tallregning. Dette viste en studie som ble utført på elever som brukte divisjonsalgoritmen i oppgaver med multiplikasjon og divisjon ved hjelp av et dataprogram (Kieran, 2017). Noe av det samme nevner Sevik (2016), nemlig at elever må lære å systematisk tilnærme seg problemer ved algoritmisk tenkning, og programmering skal bidra til å nettopp tenke algoritmisk. Strukturell tenkning kan etter det Kieran (2017) beskriver det som, ha likhetstrekk med algoritmisk tenkning.

Resultater av tidligere forskning som så på dataprogrammer som blant annet løser ligninger eller uttrykk ved å bruke deres struktur og bevare ekvivalenser, slik vi lærer å løse ligninger for hånd, kom frem til at interaksjoner mellom elever og teknologiske verktøy kan føre til begrepsmessig forståelse av strukturen til uttrykk og ligninger, og samtidig bidra til begrepsmessig forståelse av ekvivalens (Kieran & Yerushalmy, 2004, s.140). Mason (2018, s. 335) skriver at språk som er både lett manipulerbare og uttrykksfulle, stammer fra algebra og brukes i datateknologi, eksempler på dette er det eldre programmeringsspråket LOGO-programmering og CAS (computer algebra systems), men også nyere programmeringsspråk. Mason (2018) skriver videre at disse forskjellige programmeringsspråkene og nyere programmeringsspråk, kan stimulere til algebraisk tenkning og algebraisk bevissthet.

2.2 Teoretisk rammeverk

Rammeverket for studien presenteres i dette delkapittelet. Først presenterer jeg relevante begrepsavklaringer, og deretter det teoretiske rammeverket for analysen. Her legger jeg også frem Bråting og Kilhamns forskning på programmering i læreverk på barnetrinnet i Sverige, der de presenterer en kategorisering for typer programmering. Deretter presenterer jeg Smith og Steins rammeverk for identifisering av kognitive krav i matematikkoppgaver, samt Kierans GTG-modell for kategorisering og identifisering av algebraiske aktiviteter i algebra.

2.2.1 Begrepsavklaringer

2.2.1.1 Programmering

Programmering er noe de fleste forbinder med datamaskiner, konkrete eksempler er koding, utvikling og design ved bruk av data. I matematikken brukes programmering hovedsakelig som et verktøy for å løse matematiske oppgaver eller problemer. Programmeringsspråkene som vanligvis brukes i matematikk i skolen er Python og Scratch, der Python er tekstbasert programmering og Scratch er blokkbasert programmering. Dette ser vi av både oppgavetypene vi finner i læreverkene i Norge, og av forskning på programmering i matematikk (Chan et al., 2023, s. 2051). Ved blokkbasert programmering drar man inn visuelle blokker med koder, det kan sammenlignes med å bygge Lego. Når blokkene er bygd sammen riktig, danner disse en programkode som utfører en oppgave. Ved tekstbasert programmering bruker man et programmeringsspråk i tekstform og skriver selv inn koder, når alle kodene er skrevet inn riktig, danner denne teksten en programkode som utfører en oppgave.

Sevik ser på programmering som å «identifisere et problem og tenke ut mulige løsninger på problemet, til å skrive kode som kan forstås av en datamaskin, og å feilsøke og kontinuerlig forbedre denne koden» (Sevik, 2016, s. 9). Sevik (2016, s. 13) skriver også at det innebærer mye prøving og feiling å programmere, og at programmering gir en tilnærming til problemløsing og er en prosess der en finner best mulig løsning på et problem ved hjelp av færrest mulig steg. Bråting og Kilhamn (2021a, s. 173) skriver at programmering ofte blir omtalt som et pedagogisk verktøy som brukes til å utvikle elevers algoritmiske tenkning. Det å utvikle algoritmisk tenkning ved programmering, er også noe som vektlegges i læreplanen for matematikk for 1.-10. trinn, MAT01-05 (Kunnskapsdepartementet, 2019) og Sevik (2016) sitt fagnotat om programmering i skolen.

2.2.1.2 Algoritmisk tenkning

Studien min tar for seg programmering og ikke algoritmisk tenkning, men å utvikle algoritmisk tenkning er en del av hvorfor programmering er integrert i matematikkfaget i Norge (Sevik, 2016; Kunnskapsdepartementet, 2019; Bråting & Kilhamn, 2021a). Derfor velger jeg å definere algoritmisk tenkning i og med at det er naturlig at begrepet vil dukke opp flere steder i studien.

Aho (2012, s. 832) definerer *algoritmisk tenkning* som tankeprosessen som inngår i å formulere problemer slik at løsningene kan representeres som programmeringstrinn og algoritmer. Denning (2017, s. 33) viser også til Ahos (2012) definisjon og mener at en algoritme i programmering består av serier av trinn som kontrollerer en maskin, og at algoritmisk tenkning inkluderer designet av modellen, og ikke bare trinnene som kontrollerer den. Wing (2008, s. 3717) skriver at algoritmisk tenkning er en type analytisk tenkning, og er innenfor matematikk en måte å tilnærme seg det å løse problemer på, videre skriver Wing (2008) at essensen av algoritmisk tenkning er abstrahering. Bråting og Kilhamn (2021a, s. 172) viser til Wing (2008) som mener at algoritmisk tenkning er en grunnleggende ferdighet for alle, og essensielt en menneskelig måte å tenke på som gjør oss i stand til å bruke datamaskiner til å løse problemer. Det finnes altså flere definisjoner på algoritmisk tenkning, men hovedpoengene er at det er en måte å tenke på som gjør det mulig å løse problemer systematisk og trinnvis, ved hjelp av teknologi og datamaskiner.

2.2.1.3 Algebraisk tenkning

Studien tar og for seg algebra, og dermed er det naturlig å definere algebraisk tenkning. Algebraisk tenkning er et bredt begrep som tar for seg ulike typer resonnement og måter å representere ting på når en gjør algebraiske aktiviteter (Bråting & Kilhamn, 2021a, s. 172). Radford (2017, s. 8, direkte oversatt til norsk av meg) skriver at algebraisk tenkning tyr til:

a) Ubestemte mengder
b) Idiosynkratiske (særegne) eller spesifikke kulturelt og historisk utviklete måter å representere eller symbolisere disse ubestemte mengdene og deres operasjoner på.
c) Algebraisk tenkning omhandler også ubestemte mengder på en analytisk måte.

Videre skriver Radford (2017, s. 8) at ubestemte mengder omhandler mer enn bare tall, det kan være ukjente, variabler, parametere og så videre. Selv om ubestemte mengder kan uttrykkes med alfanumeriske symboler, kan de også uttrykkes gjennom andre semiotiske representasjonssystemer (Radford, 2017). Det å håndtere ubestemte mengder og deres

operasjoner på en analytisk måte, vil si at selv om mengdene er ukjent, så brukes de i utregninger som om de var kjent (Radford, 2017, s. 8). Radford (2017) gir et eksempel på algebraisk tenkning ved løsningen av ligningen «2x+2=10+x», hvis elever prøver og feiler, kvalifiserer ikke dette for algebraisk tenkning, men hvis elevene subtraherer 2 fra hver side av likhetstegnet, og får «2x=8+x», kan vi si at elevene tenker algebraisk, for da tenker elevene at 2x+2 er det samme som 10+x, og forstår også likhetstegnet og ekvivalens på riktig måte.

2.2.1.4 Type programmeringsoppgave

Programmeringstype eller type programmering/programmeringsoppgave betyr i denne studien hva slags programmeringsfremgangsmåte eller prosedyre det skal brukes for å løse programmeringsoppgaver, dette kan være å for eksempel tolke programmer, lage programmer eller kopiere og feilsøke programkoder.

2.2.1.5 Begrepsmessig forståelse og prosedyrekunnskap

Conceptual knowledge, oversatt til *begrepsmessig forståelse*, kan ifølge Hiebert og Lefevre (1986, s. 3) ses på som et nettverk av kunnskap, der sammenhengen mellom kunnskap, sammenhenger i matematikk og det å koble ny kunnskap til eksisterende kunnskap er viktig. Begrepsmessig forståelse kan oppsummert ses på som kunnskap om hvorfor noe skal gjøres, *know why*, og går i matematikk ut på at en vet hvordan en oppgave skal løses, og hvorfor det er slik (McCormick, 1997, s. 143). *Procedural knowledge*, oversatt til *prosedyrekunnskap*, innebærer det å ha kunnskap om språk eller symbolske representasjonssystemer i matematikk, og kunnskap om algoritmer og regler som verktøy for å løse matematikkoppgaver (Hiebert & Lefevre, 1986, s. 6). Prosedyrekunnskap kan oppsummert ses på som kunnskap om hvordan noe skal gjøres, altså *know how* (McCormick, 1997, s. 143). Dette er relevante begreper som i studien dukker opp knyttet til det matematiske innholdet i oppgavene i form av algebraiske aktiviteter og kognitive krav.

2.2.2 Programmeringsoppgaver i matematikk

Det finnes flere teoretiske rammeverk for analyse av oppgaver i matematikk. Et spesifikt rammeverk for å se på hva slags programmeringsoppgaver som finnes, og hvordan disse kan klassifiseres, er rammeverket brukt i studien til Bråting og Kilhamn (2021b). Bråting og Kilhamn (2021b) har i sin forskning hentet inspirasjon fra CT-rammeverket av Brennan og Resnick og «The 5E's» av Benton et al. Disse rammeverkene har blitt brukt til å lage et analytisk rammeverk for å analysere programmeringsoppgaver i lærebøker. I utgangspunktet ville forskerne analysere programmeringsseksjonene av matematikkbøkene ved å bruke rammeverkene separat, men Bråting og Kilhamn (2021b) fant ut at ingen av rammeverkene

kunne beskrive innholdet på en tilstrekkelig måte. Det kan være fordi begge rammeverkene beskriver en ideell situasjon, og forskerne kombinerte derfor begge rammeverkene for å danne et analytisk verktøy, og i dette verktøyet har de blant annet laget en klassifisering for hva slags typer programmering en finner i programmeringsoppgaver.

Det analytiske rammeverket til Bråting og Kilhamn (2021b) inneholder seks forskjellige beskrivelser for prosedyrer i oppgaver, altså hvilken fremgangsmåte eller type programmering en skal bruke for å løse oppgaven. Rammeverket brukes til å se etter hvilke typer programmering som finnes blant oppgavene som er analysert. Det nye rammeverket til Bråting og Kilhamn (2021b, s. 599, oversatt til norsk av meg) består av prosedyrene eller programmeringstypene:

A) Følge – følg instruksjonene trinn for trinn, gjenta eller fortsett et mønster
B) Finne regel – finn ut prosedyren, regelen eller mønsteret som genererer et resultat, for eksempel en nummerserie
C) Feilsøk – Feilsøk en kode
D) Forme og skape – gi instruksjoner, lag et mønster, skriv en kode, representer med symboler
E) Forklare – Forklar ved å bruke naturlig språk, bruk ord til å beskrive en prosedyre, en regel, et mønster eller et konsept
F) Forestille seg – Forutsi hva som vil skje, reflekter over mulige resultater når betingelser eller verdier endres

I sin studie har Bråting og Kilhamn (2021b) analysert programmeringsoppgaver med det analytiske rammeverket basert på hvilke fremgangsmåter oppgavene setter elevene til å bruke og sammenhengen mellom programmeringsbegreper og matematiske begreper. De har deretter sett på sammenhengen av fremgangsmåter og begreper for å kvalitativt beskrive hvordan programmeringsoppgavene danner en sammenheng mellom matematikk og programmering. Begrepene identifiseres og klassifiseres, og et begrep klassifiseres som matematisk hvis det finnes i tradisjonell skolematematikk og formidler en matematisk ide. En matematisk ide kan være vanskelig å definere, men kommer til uttrykk i de matematiske begrepene og representasjonene som brukes (Bråting & Kilhamn, 2021b, s. 600-601). Et eksempel på at en matematisk ide kommer til uttrykk, er en oppgave der elevene bes om å tegne en firkant. Elevene får oppgitt at de må «snu 90 grader og gjenta 4 ganger». Her brukes altså matematiske termer til å beskrive og konstruere et geometrisk objekt, og dermed klassifiseres det som matematiske eller geometriske begreper (Bråting & Kilhamn, 2021b, s.

600-601). Et begrep klassifiseres som algoritmisk hvis begrepet har en spesifikk mening innenfor programmeringskonteksten i oppgaven, eksempel på slike begreper er kode, løkke og bug (Bråting & Kilhamn, 2021b, s. 601).

Til sist ser rammeverket på kombinasjonen av handlinger og begreper for å se på hvilken måte potensielle forbindelser mellom programmering og matematikk lages, og deretter diskuteres det hvordan disse forbindelsene kan føre til matematisk læring (Bråting & Kilhamn, 2021b, s. 601). Forskerne ser spesielt på hvordan elever blir spurt om å reformulere algoritmiske ideer ved å bruke matematisk terminologi eller matematisk notasjon, og om oppgavene lar elevene utforske matematiske ideer som er nye for dem (Bråting & Kilhamn, 2021b, s. 601). Hver oppgave som ble analysert ble karakterisert ut ifra typen programmering som finnes i oppgaven, og alle algoritmiske og matematiske begreper som eksplisitt ble nevnt i oppgaven, ble identifisert (Bråting & Kilhamn, 2021b, s. 8).

Slik jeg nevnte tidligere, fant Bråting og Kilhamn (2021b) i sin forskning svært lite variasjon i typer programmering blant oppgavene. Det var svært mange oppgaver der elever skulle følge en prosedyre eller instruksjoner, og lite oppgaver der elevene skulle forestille seg, feilsøke og forklare ved å bruke flere representasjonsformer. Lite feilsøkingsoppgaver blir spesielt sett på som problematisk, da tidligere forskning på programmering (Papert, 1993) viser at elever kan lære matematikk ved å teste og feilsøke. Bråting & Kilhamn (2021b) mener at det er et stort potensiale i å ha en større bredde i typen programmeringsoppgaver, noe som vil styrke programmeringsinnholdet og sammenhengen mellom programmering og matematikk. Rammeverket er laget for, og ble i studien brukt på barnetrinnene, på ungdomstrinnene er det andre mål for utviklingen av algebrakompetanse, dermed trenger jeg et teoretisk rammeverk til å analysere det matematiske innholdet i form av kognitive krav og algebraiske aktiviteter ved oppgavene også.

2.2.3 Rammeverk for analyse av algebraoppgaver

Kieran (2019, s. 271) skriver at rammeverk for oppgaveanalyse kan variere ut ifra hvordan undervisningen spiller inn på oppgavedesignet, men også etter hva slags typer oppgaver rammeverkene er tilpasset. Eksempler på dette kan være (1) om rammeverkene ser på matematisk forståelse som begreper, prosedyrer og representasjoner, (2) om rammeverkene ser på utviklingen av matematiske resonnementer (generalisering, kritisk tenkning og bevis), eller om (3) rammeverkene ser på utvikling av modellering og problemløsning, og om (4) rammeverkene evaluerer matematisk forståelse, prosesser og problemløsning (Kieran, 2019,

s.271). Noen rammeverk kan også være bedre tilpasset spesifikke oppgaver og verktøy som brukes som hjelp til å løse oppgavene.

Kieran (2019, s.272) beskriver flere rammeverk, og beskriver *intermediate* rammeverk som rammeverk som karakteriseres av eksplisitte verktøy som kan brukes på oppgaver, og disse rammeverkene brukes gjerne på spesifikke læreplansområder. *Domain* rammeverk er rammeverk som i motsetning til intermediate rammeverk, er spesifikke for matematisk innhold som algebra og geometri, eller for verktøy som datamaskiner og kalkulatorer (Kieran, 2019, s. 273). Forskning på oppgaver som bruker disse rammeverkene ser ofte på tidligere forskning innenfor et spesifikt område, i tillegg til å bruke visse nivåer av intermediate rammeverk og mer generelle rammeverk (Kieran, 2019, s. 273).

Kieran (2019, s. 274) viser videre til hvordan hun i sin studie har brukt flere teoretiske rammeverk til å danne et domain rammeverk for sin studie på bruk av CAS i algebraundervisning. Rammeverket består av intermediate rammeverk og domain rammeverk, der alle rammeverkene brukes i forskjellig mengde og på forskjellig måte (Kieran, 2019, s. 274). Målet med studien rammeverket ble brukt i var å se på samspillet mellom det tekniske og det begrepsmessige, ved å se på teknikkene og teoriene som elevene utviklet under bruken av digitale verktøy (Kieran, 2019, s. 275). Hun kom i studien frem til at det avgjørende for tanken om at begrepsmessig forståelse kan oppstå sammen med teknikk ved bruk av digitale verktøy, var at de transformerende aktivitetene i algebra (for eksempel faktorisering og løsning av ligninger) måtte knyttes til de globale aktivitetene ved algebra (for eksempel det å se strukturer, generalisere, forutsi og bevise) (Kieran, 2019, s. 275).

Et av disse domain rammeverkene, er Kierans (2004a) *GTG-modell* for algebraiske aktiviteter i skolealgebraen. Kieran (2004a, s. 22) skriver at de tre aktivitetene i modellen er hovedaktivitetene i skolealgebraen. Det er dette rammeverket jeg skal bruke til å undersøke algebrainnholdet i programmeringsoppgavene. Jeg skal bruke modellen for å identifisere hvilke algebraiske aktiviteter som fremkommer blant programmeringsoppgavene, ettersom aktivitetene henger sammen med utviklingen av algebraisk og matematisk kompetanse, og sier noe om det matematiske og algebraiske innholdet i oppgavene (Kieran, 2004a).

Generational activity, oversatt til *genererende aktiviteter*, er aktiviteter i algebra som innebærer det å formulere uttrykk og ligninger som er algebraiske objekter, og fokuset ligger i representasjonen av situasjoner, egenskaper, mønstre og sammenhenger (Kieran, 2004a, s. 22-23). Eksempler på dette er ligninger med en ukjent som representerer kvantitative

problemsituasjoner, generaliseringsuttrykk fra geometriske mønstre eller numeriske sekvenser og uttrykk som representerer numeriske sammenhenger (Kieran, 2004a, s. 23). Typisk for disse oppgavene er at elevene først må tolke noe, for eksempel en situasjon, og deretter representere det algebraisk (Kieran, 2014a, s. 28). Kieran (2014a, s. 28) hevder også at det er ved de genererende aktivitetene mening i algebra oppstår, blant annet rundt ekvivalens og variabler.

Transformational activity, oversatt til *transformerende aktiviteter* er algebraiske aktiviteter som inkluderer blant annet faktorisering, multiplisering av polynomuttrykk og løsning av ligninger (Kieran, 2004a, s. 24). En stor del av denne typen aktivitet går ut på å endre uttrykk og ligninger for å bevare ekvivalens (Kieran, 2004a, s. 24). Disse aktivitetene er gjerne regnetekniske eller prosedyrerettede, og har fått stor oppmerksomhet i læreverk og i skolen (Kieran, 2004a). Konkrete eksempler på slike aktiviteter er regnetekniske oppgaver der elever skal bruke regler for å løse for eksempel ligninger.

Global/meta-level activity, oversatt til globale aktiviteter, er matematiske aktiviteter der algebra bukes som et verktøy, algebra kan, men trengs altså ikke å brukes (Kieran, 2004a, s. 24). Eksempler på slike aktiviteter er oppgaver innenfor problemløsning, modellering, generalisering og bevisføring, noe som også lar seg gjøre uten bruk av algebra (Kieran, 2004a, s. 24). De globale aktivitetene inneholder mer generelle matematiske prosesser, og gir en kontekst for bruk av algebra, og kan samtidig være med på å motivere til bruken av genererende eller transformerende aktiviteter (Kieran, 2004a, s. 29). Disse aktivitetene kan ikke utelukkes fra algebra, for da mister en hensikten med å kunne algebra (Kieran, 2004a, s. 24).

Technique, oversatt til teknikk, ses på som en blanding av rutine og refleksjon, og er viktig når en skal løse oppgaver, samtidig som den bidrar til en forståelse av objektene som håndteres og er viktig ved refleksjon over begreper (Kieran, 2004a, s. 28). Videre skriver Kieran (2004a, s. 31) at manipulasjonsteknikker ved bruk av teknologiske verktøy fører til begrepsmessig forståelse i algebra, spesielt i oppgaver som søker å skape forståelse rundt ekvivalens. Samtidig må en være obs på at noen oppgaver og verktøy kan gjøre det enkelt å unngå algebraiske representasjoner og transformasjoner (Kieran, 2004a, s. 31). Forskningen viser også at teknikk kan føre til begrepsmessig forståelse ved algebraiske transformasjoner (Kieran, 2004a, s. 32). Til slutt hevder Kieran (2004a, s. 31) at begrepsmessig forståelse kan dannes hos elever ved bruk av teknikk under arbeid med algebra ved hjelp av digitale verktøy.

Kieran (2013) har i sin forskning kritisert skillet mellom begrepsmessig forståelse og prosedyrekunnskap. Hun kaller skillet en falsk todeling i matematikkundervisningen (2013, s. 153), og argumenterer for at en ikke kan se på begrepsmessig forståelse og prosedyrekunnskap som separate enheter. Kieran (2013) skriver også at det å utarbeide og sette sammen prosedyrer er konseptuelt av natur og at dette bidrar til forståelse av matematikk. Hun påpeker i artikkelen at todelingen har vært spesielt skadelig i algebra, fordi algebra blir sett på som et prosedyredominert domene for symbolske manipulasjoner i skolematematikken, der det begrepsmessige er fraværende (Kieran, 2013, s. 154). Kieran (2013) skriver at målet med artikkelen er å fremme en nytenkning av algebra, og viser til at det finnes både begrepsmessige og prosedyremessige komponenter innenfor algebra, samt et samspill mellom dem. Kieran (Kieran, 2013, s. 167) kommer i sin forskning frem til at det begrepsmessige i symbolsk algebra er en integrert del av det prosedyremessige, enten i endringen av prosedyrer, eller sammensetningen av nye prosedyrer. Kieran (2013, s. 169) fremhever til slutt at læring av algebra rent prosedyremessig er negativt, og viser til viktigheten av reelle problemløsningssituasjoner og flere representasjonsformer for algebraiske problemer.

Kieran viser blant annet til viktigheten av reelle problemløsningssituasjoner, noe som kan knyttes til globale og genererende aktiviteter i algebra. De transformerende aktivitetene kan ses på som mer prosedyrerettede enn de genererende og globale aktivitetene, men samspillet mellom aktivitetene er viktig, slik som samspillet mellom prosedyrekunnskap og begrepsmessig forståelse i algebra. Kierans (2004a) GTG-modell tar for seg det algebraiske ved oppgavene, videre skal jeg presentere et rammeverk som tar for seg kognitive krav i matematikkoppgaver.

Når en programmeringskomponent kommer inn i bildet, trengs det et verktøy som også kan brukes på oppgaveanalyse av matematikkoppgaver generelt, og ikke bare på algebrainnholdet i oppgaver. Dermed skal jeg også ta for meg et rammeverk av Smith og Stein (1998) som kan brukes på alle typer matematikkoppgaver.

2.2.4 Nivåer av kognitive krav i matematikkoppgaver

Cognitive demand, oversatt til kognitive krav, defineres av Stein et al. (2000, s. 11) som hvilken type og hvilket nivå av tenkning som kreves av elever for å klare å løse oppgaver. Videre argumenterer Stein et al. (2000, s. 11) for at kognitive krav i oppgaver er svært viktige, fordi elevers læringsmuligheter ikke kommer av å sette elever sammen i grupper eller gi dem et verktøy som en kalkulator, men det kommer av hva slags tenkning og hvilken tenkning

elever engasjerer seg i, og det er dette som avgjør hva elevene lærer. Oppgaver som krever at elevene skal utføre memorerte prosedyrer fører til en type tenkning, og oppgaver som krever at elever skal bruke begreper og se sammenhenger mellom prosedyrer og relevante matematiske ideer, fører til en annen type tenkning (Stein et al., 2000, s. 12). Dermed er det matematiske innholdet i programmeringsoppgavene i form av hvilke kognitive krav oppgavene stiller til elevene, viktig for det potensielle læringsutbyttet i oppgavene i henhold til det Stein et al. (2000) har argumentert for i sin forskning.

Smith og Stein (1998) har laget et rammeverk for oppgaveanalyse av kognitive krav som inneholder fire forskjellige nivåer, der to av nivåene representerer lavt kognitivt krevende oppgaver, og to av nivåene representerer høyt kognitivt krevende oppgaver (Smith & Stein, 1998). Rammeverket kan brukes som en vurdering og gir en slags rangering av oppgavene etter kognitive krav, og kan brukes på alle typer matematiske oppgaver (Smith & Stein, 1998, s. 345). Rammeverket inneholder kjennetegn på og beskrivelser av hvilke oppgaver som er høyt eller lavt kognitivt krevende, og det gis også eksempler på hvordan oppgaver skal plasseres. Rammeverket kan brukes til å vurdere kvaliteten av oppgaver, ettersom forskningen viser at oppgaver med høyt kognitivt krevende nivå gir potensiale for et større læringsutbytte hos elever (Smith & Stein, 1998). Samtidig kan rammeverket brukes på oppgaver uten at en observerer eller samler inn data av elever som løser oppgavene, da det brukes for å analysere oppgaver og ikke løsningen av dem.

De fire nivåene (oversatt til norsk av meg) i rammeverket til Smith og Stein (1998, s. 348) er:

<u>Memorering</u> (lavere kognitivt krevende nivå)

- Oppgaver som krever at man gjengir eller bruker fakta, regler, formler eller definisjoner man har lært tidligere.
- Oppgavene kan ikke løses med prosedyrer fordi prosedyren ikke finnes eller fordi tidsrammen oppgaven løses i er for kort.
- Oppgaver som ikke er tvetydige, oppgavene krever reproduksjon og er spesifikke, og fremgangsmåten for hva man skal gjøre eller reprodusere, er klar.
- Det er ingen tilknytning til begrepene eller meningen som ligger bak faktaene, reglene eller definisjonene som læres eller reproduseres.

Eksempler på slike oppgaver er oppgaver der elever får presentert flere like oppgaver etter hverandre, eller er blitt gitt en svært lik eksempeloppgave før de selv skal løse en oppgave på samme måte.

Prosedyrer uten forbindelser (lavere kognitivt krevende nivå)

- Disse oppgavene er algoritmiske, fremgangsmåten eller prosedyren er spesifikt gitt eller kommer tydelig frem fra tidligere instruksjoner, erfaringer eller plasseringen av oppgaven.
- Begrenset kognitiv etterspørsel for å løse oppgaven, liten tvetydighet rundt hva som må gjøres og hvordan det skal gjøres.
- Det er ingen tilknytning til begrepene eller meningen som ligger bak faktaene, reglene eller definisjonene som læres eller reproduseres.
- Oppgavene er resultatfokusert, riktig svar er fokuset, ikke å utvikle matematisk forståelse.
- Krever ingen forklaring som fokuserer på å beskrive prosedyren som blir brukt til å løse oppgaven.

Eksempler på slike oppgaver er oppgaver som løses ved bruk av innøvde prosedyrer eller fremgangsmåter, og hvordan oppgavene skal løses er gitt av oppgaveteksten.

Prosedyrer med forbindelser (høyere kognitivt krevende nivå)

- Oppgavene retter elevers oppmerksomhet mot bruken av prosedyrer med hensikt om å utvikle dypere nivåer av forståelse for matematiske begreper og ideer. Forståelsen av prosedyren i fokus, fremfor fokus på å finne løsningen.
- Foreslår eksplisitte eller implisitte fremgangsmåter som er brede generelle prosedyrer tett knyttet til underliggende begreper, i motsetning til smale algoritmer.
- Representeres ofte på flere måter, for eksempel visuelle diagrammer, symboler og problemsituasjoner. Viktig å danne forbindelser mellom flere representasjonsformer.
- Krevet et visst nivå av kognitiv innsats, generelle prosedyrer kan følges, men ikke tankeløst. Elevene må engasjere seg i begreper som ligger til grunn for prosedyrene for å fullføre oppgavene og utvikler dermed matematisk forståelse.

Eksempler på slike oppgaver kan være utforskningsoppgaver eller oppgaver der flere matematiske prosedyrer eller fremgangsmåter må brukes sammen. Det kreves også tenkning for å finne ut hvordan oppgavene skal løses og hvordan prosedyrer skal brukes, og oppgavene ligner ikke helt på tidligere gitte oppgaver.

Å gjøre matematikk (høyere kognitivt krevende nivå)

- Oppgavene krever kompleks og ikke-algoritmisk tenkning, en forutsigbar, innøvd fremgangsmåte er ikke eksplisitt foreslått av oppgaven.
- Krever at elevene utforsker og forstår matematiske begreper, prosesser eller forbindelser.
- Krever egenkontroll eller selvregulering av egne kognitive prosesser.
- Krever at elever bruker relevant kunnskap og erfaring på en riktig måte når elevene jobber gjennom oppgaven.
- Krever at elevene analyserer oppgaven og aktivt undersøker oppgavebegrensninger som kan begrense mulige løsningsmetoder og løsninger.
- Oppgaven krever betydelig kognitiv innsats og kan gi en viss grad av angst hos elever på grunn av uforutsigbare metoder som kreves i løsningsprosessen.

Smith & Steins (1998) eksempel på en slik oppgave er: «Lag en reell situasjon for problemet $\frac{2}{3} * \frac{3}{4}$, løs problemet du lager uten å bruke regler og forklar løsningen».

Ved disse oppgavene kan ikke elevene bruke en algoritmisk tilnærming, og oppgavene er gjerne komplekse og fører til at elevene må utforske og analysere oppgavene for å komme frem til løsningen. Elevene kan ikke bruke tidligere innlærte fremgangsmåter eller regler. Eksempler på slike oppgave er problemløsningsoppgaver.

Bråting og Kilhamns rammeverk for programmeringstyper, Kierans GTG-modell for algebraisk aktivitet og Smith og Steins rammeverk for kognitive krav, skal i studien brukes sammen for å analysere det matematiske innholdet i oppgavene i form av algebraiske aktiviteter og kognitive krav og programmeringstypene i oppgavene. Ved å bruke disse rammeverkene sammen, vil jeg kunne se hva som karakteriserer det matematiske innholdet som finnes i programmeringsoppgavene, i form av hvilke kognitive krav oppgavene stiller til elever, hvilke algebraiske aktiviteter som finnes og hvilke programmeringstyper elevene skal bruke blant oppgavene. Deretter kan funnene ses i sammenheng, for å se hvordan kognitive krav og algebraiske aktiviteter kan ha en sammenheng med programmeringstype, og potensielt hvordan disse tre har sammenheng med hverandre.

3. Metode

Dette kapittelet skal gjøre rede for forskningsdesignet og fremgangsmåten for studien. Kapittelet inkluderer analysemetoden og rammeverk knyttet til denne, dokumentanalyse, tematisk analyse, valg av læreverk og oppgaver med begrunnelser for valg av disse. Kapittelet

beskriver også de analytiske rammeverkene som er brukt i analysen av oppgavene i læreverkene. Kapittelet skal også si noe om reliabilitet, validitet og etiske refleksjoner. Dette kapittelet gir også beskrivelser for hvordan rammeverkene har blitt brukt og tilpasset i analysen. Kapittelet skal beskrive mine intensjoner med forskningen og gi en beskrivelse av forskningens prosess og design, for å gi et bilde av troverdigheten studien har.

3.1 Forskningsdesign og metode

Formålet med studien er å svare på problemstillingen «*Hva karakteriserer det matematiske innholdet i programmeringsoppgaver innenfor algebra i læreverk for ungdomstrinnene*» ved å først foreta en kvalitativ dokumentanalyse for å samle inn og identifisere relevante oppgaver. For å kunne svare på problemstillingen og identifisere algebraoppgavene jeg skal analysere, har jeg utført en tematisk analyse av alle programmeringsoppgavene for ungdomsskoletrinnene i Matemagisk 8-10 og Campus Inkrement. Den tematiske analysen ble brukt til kategorisering og identifisering av programmeringsoppgavene i algebra og identifisering av programmeringstype blant disse. For å identifisere programmeringstype ble Bråting og Kilhamns (2021b) rammeverk også brukt. Tematisk analyse ble deretter brukt sammen med rammeverket til Smith og Stein (1998) for å analysere og identifisere de kognitive kravene i oppgavene, og deretter ble de algebraiske aktivitetene analysert og identifisert ved bruk av tematisk analyse og Kierans (2004a) GTG-modell. Formålet er at resultatene av analysene skal kunne besvare forskningsspørsmålene og problemstillingen og si noe om hva som karakteriserer det matematiske innholdet i programmeringsoppgavene i algebra, og om dette har noe sammenheng med programmeringstype. Problemstillingen skal besvares ved å svare på disse følgende forskningsspørsmålene:

- Hvilke kognitive krav stiller algebraiske programmeringsoppgaver i matematikk i to læreverk for ungdomstrinnene til elever?
- Hvilke algebraiske aktiviteter finnes blant programmeringsoppgavene i algebra i disse to læreverkene for ungdomstrinnene?
- Hvilken sammenheng er det mellom det matematiske innholdet og programmeringstypene i programmeringsoppgavene i algebra?

Studien jeg har utført har undersøkt programmeringsoppgaver i læreverkene Matemagisk 8-10 og Campus Inkrement. Bowen (2009, s. 27) definerer dokumentanalyse som en systematisk prosedyre for å vurdere og evaluere dokumenter, og det er en dokumentanalyse jeg har utført når jeg har brukt oppgaver fra læreverk som datamateriale til analysen. Alle oppgavene i

læreverkene ble analysert for å få med alle algebraoppgavene, da det å utelukke noen oppgaver ville gitt et feil bilde i henhold til problemstillingen resultatmessig. Funnene i analysen av oppgavene blir i studien fremstilt statistisk ved bruk av tabeller og diagrammer, det gis beskrivelser, og det gis også eksempler på analyseprosessen og oppgaver med bilder og tekst for å sikre mest mulig gjennomsiktighet i studien.

3.2 Valg av analysemateriale

3.2.1 Valg av læreverk og oppgaver

Ved bruk av dokumentanalyse er det viktig at en som forsker avgjør dokumentenes relevans for forskningens problemstilling og formål, det er også viktig å finne ut om innholdet i dokumentene passer til studiens rammeverk (Bowen, 2009, s. 33). For å foreta et utvalg av læreverk som er relevante, undersøkte jeg hvilke læreverk som inneholder programmeringsoppgaver for ungdomsskoleelever, samt hvilke av disse læreverkene som brukes i skolen. Ettersom programmering ble innført med LK20, finner en ikke programmeringsoppgaver i bøker gitt ut før 2020. Jeg brukte blant annet sosiale medier til å finne grupper der matematikklærere i skolen diskuterer blant annet læreverk som brukes i skolen. Jeg la ut en spørreundersøkelse i gruppen «Matematikkdidaktikk» på Facebook der mange matematikklærere daglig diskuterer matematikkfaget. Jeg fikk 153 respondenter der prosentvis fordeling per læreverk fremstilles i tabellen under:

Læreverk	Matemagisk	Kidsakoder	Kikora	Campus Inkrement	Skolestudio	Skolen.cdu.no	Aunivers	Salaby	Annet
Prosentvis stemmer	12 %	23 %	13 %	11 %	1 %	13 %	2 %	8 %	11 %

Tabell 1 - Prosentvis oversikt over bruk av læreverk

Vi ser av svarene at det er ganske jevnt mellom flere læreverk, men at flest bruker Kidsakoder. Kidsakoder.no beskriver seg selv som en frivillig bevegelse som jobber for at unge skal lære å kode i skolen, det er egentlig ikke et læreverk utgitt av et forlag, men en ressurs som kan brukes. Jeg har derfor sett bort i fra kidsakoder.no, fordi jeg ikke har noe annet grunnlag for å si at lærere har lagt vekt på, eller brukt innholdet fra kidsakoder.no i sin undervisning. Jeg har i innledningen presentert forskning som sier at læreverk er sentrale i undervisning, og det har derfor vært nødvendig at jeg begrenser meg til å studere læreverk.

Ved innhenting av informasjon rundt bruk av læreverk i skolen, har jeg samarbeidet med en medstudent som også analyserer programmeringsoppgaver i sin masteroppgave. Vi har skrevet samtykkeskjema der vi samtykker til å dele informasjonen rundt bruken av læreverkene og eventuelle eposter fra forlagene (Vedlegg 8). Medstudenten jeg har samarbeidet med rundt innhenting av informasjon, har sendt ut 140 individuelle eposter til

skoler og lærere i Vestland og Rogaland. 130 av epostene gikk gjennom, og av totalt 40 svar på de forskjellige læreverkene, ser den prosentvise fordelingen slik ut:

Læreverk	Matemagisk	Kidsakoder	Kikora	Campus Inkrement	Skolestudio	Skolen.cdu.no	Maximum	Annet
Prosentvis stemmer	20,0 %	5,0 %	10,0 %	22,5 %	7,5 %	15,0 %	7,5 %	7,5 %

Tabell 2 - Prosentvis oversikt over bruk av læreverk

Av disse svarene ser vi at Matemagisk og Campus Inkrement brukes mest, og av mine resultater så vi også at disse var en god del brukt. Jeg valgte dermed å analysere Matemagisk 8-10 og Campus Inkrement utfra hvor mye de ble brukt. Jeg så også på andre fordeler ved læreverkene, blant annet så jeg på hvilke læreverk som har implementert programmering i selve læreverket, og ikke bare som en ekstra ressurs. Valgene ble også foretatt basert på egen erfaring med læreverkene etter gjennomgang av læreverkene. Jeg hadde også tilgang til begge disse læreverkene, og begge læreverkene inneholder mange programmeringsoppgaver som vil gi et godt grunnlag for analysen. Dermed var det disse to læreverkene som ble brukt i denne studien.

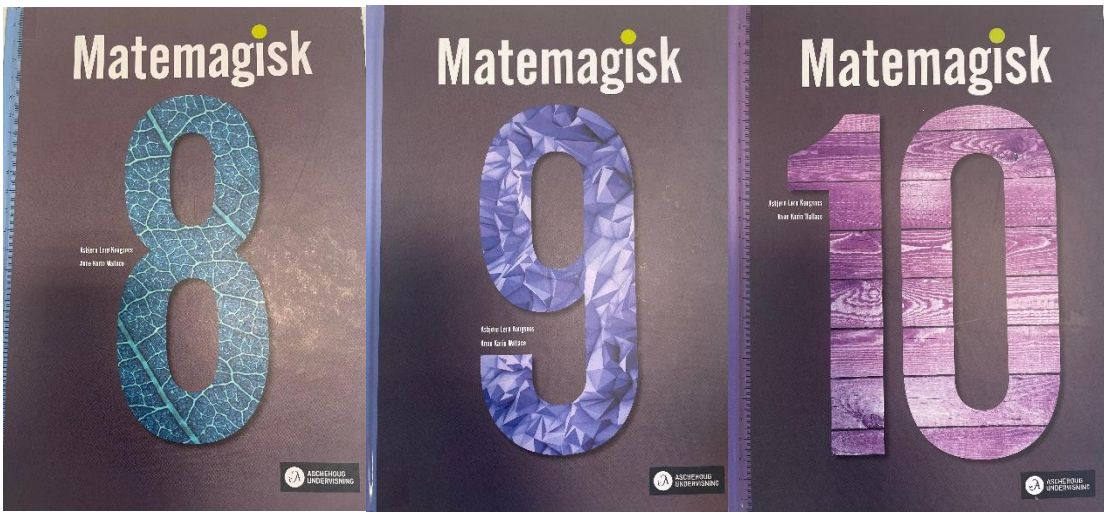
I tillegg til spørreundersøkelsen min og til min medstudent, sendte jeg en epost til Aschehoug angående Matemagisk. Aschehoug svarer på epost at de ikke oppgir markedsandeler, men bekrefter at Matemagisk 8-10 brukes mye i skolen og har en betydelig markedsandel (Vedlegg 7). Medstudenten sendte epost til Campus Inkrement, og Campus Inkrement svarer også at de ikke oppgir salgstall, men at det er totalt over 1000 skoler i Norge som bruker Campus Inkrement, som indikerer at det er mye brukt (Vedlegg 9). Jeg har også erfart i praksis og i arbeid som lærervikar at Campus Inkrement og Matemagisk er mye brukt.

3.2.2 Læreverkene

Læreverket Matemagisk har valgt å fokusere på tekst-basert programmering, altså programmering ved bruk av programmeringsspråket Python. Her skriver en egne koder ved bruk av tekst. På Campus Inkrement, har de fokusert på blokk-basert programmering, slik som for eksempel Scratch. Det er også Scratch-oppsettet som brukes på Campus Inkrement. Disse to programmeringsspråkene ser ut til å være mest brukt i skolen basert på forskning og bruk i læreverk (Chan et al. 2023). Ved bruk av blokk-basert programmering drar man inn blokker med koder, eller illustrerte koder, og lager en sammensatt kode ved hjelp av blokker. Dette kan være positivt for analysen og resultatene, da oppgavene vil inkludere to av programmeringsspråkene som er mest brukt i skolen.

3.2.3 Læreverket Matemagisk

Matemagisk 8, 9 og 10 er lærebøker som er produsert av forlaget Aschehoug og er skrevet av Asbjørn Lerø Kongsnes og Anne Karin Wallace. Læreverkene er fornyet etter LK20 og inneholder flere programmeringsoppgaver. På Aschehoug sine nettsider skriver de at elever med Matemagisk kan «utforske og diskutere fra første stund» (Aunivers, 2021). I Matemagisk bøkene er programmering ikke lagt inn som et eget kapittel, men lagt inn i andre delkapitler, så det faller inn som en naturlig del av matematikken. Mange læreverk velger å ha programmering i et eget kapittel. Matemagisk-bøkene har også et stort programmeringsinnhold, spesielt Matemagisk 8, noe som gir et større datamateriale.



Bilde 1 - Illustrasjon av Matemagisk lærebøkene for 8., 9., og 10. trinn, Kongsnes & Wallace (2020 & 2021).

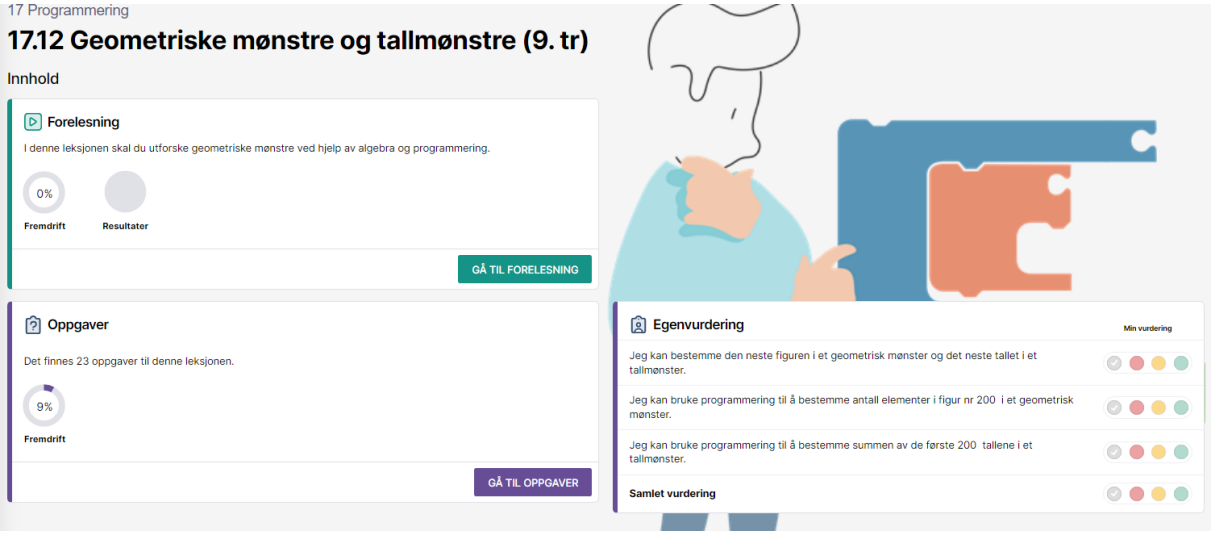
På siden «Velkommen til Matemagisk» får en samme introduksjon i alle bøkene. Der står det at Matemagisk skal «legge til rette for at dere som elever skal få være aktive, utforske og oppdage matematiske sammenhenger» (Kongsnes & Wallace, 2020). Det står videre at bøkene legger opp til samarbeid, problemløsning og inneholder virkelighetsnære og varierte oppgaver, slik at matematikken skal bli relevant og meningsfull (Kongsnes & Wallace, 2020). Det finnes også en elevhåndbok, men bøkene kan også brukes alene. I Matemagisk har Aschehoug valgt å implementere programmering som en del av de andre kapitlene. På denne måten mener jeg at Aschehoug allerede har klart å fjerne en del av skillet mellom programmering og matematikk. Programmeringsoppgavene og programmering føles som en naturlig del av boken og matematikkinnholdet, fordi at programmeringsoppgavene legges sammen med de andre oppgavene i boken.

I denne oppgaven har jeg også blitt tillatt av Aschehoug å bruke skjermbilder av enkeltoppgaver til eksempler i studien etter å ha snakket med Aschehoug per epost (Vedlegg 5).

3.2.4 Læreverket Campus Inkrement

Campus Inkrement er et digitalt læreverk som har fokus på omvendt undervisning. På hjemmesiden til Campus Inkrement kan vi lese at læreverket brukes av over 1000 skoler, og drives av Inkrement AS som består av pedagoger og utviklere, som alle har lang erfaring med digitale læremidler. Campus Inkrement har valgt å ha programmering som et eget kapittel under Matematikk for 8.-10. trinn. Programmeringen på Campus Inkrement er blokkbasert som Scratch, og skiller seg fra programmeringen i Matemagisk som er tekstbasert ved bruk av Python. Det er totalt syv delkapitler under programmering på Campus for ungdomstrinnene, og hvert delkapittel inneholder forelesninger, regler, eksempler, et sett med oppgaver og en egenvurdering. Campus Inkrement har sammenlignet med Matemagisk implementert programmering som et eget kapittel, og ikke som en del av andre kapitler. Bildet nedenfor viser noen delkapitler fra kapittel 17 som er programmeringskapittelet. Bilde 3 viser forsiden for et delkapittel.

Bilde 2 - Delkapitler under programmeringskapittelet, hentet fra Campus Inkrement.



Bilde 3 - Illustrasjon av et delkapittel under programmering, hentet fra Campus Inkrement.

I denne oppgaven har jeg også blitt tillatt av Campus Inkrement å bruke skjermbilder av enkeltoppgaver til eksempler i studien etter å ha kommunisert med Campus Inkrement per epost, så fremt at det er enkeltoppgaver som brukes og ikke illustrasjoner av hele læreverket eller alle oppgavene (Vedlegg 6).

3.3 Analysemetode

Denne studien er en kvalitativ dokumentanalyse der det er brukt en tematisk analyse (Braun & Clarke, 2006) som fremgangsmåte for å kode og kategorisere programmeringsoppgavene i læreverkene. Oppgavene ble først analysert og kodet ved bruk av en tematisk analyse (Braun & Clarke, 2006) i programvaren Nvivo. Dette ble gjort for å identifisere matematisk tema og deretter programmeringstype, slik at det som kan klassifiseres som algebraoppgaver ble identifisert og hver oppgaves programmeringstype ble identifisert. Deretter ble programmeringsoppgavene i algebra analysert og kodet ved bruk av Smith og Steins (1998) rammeverk for å si noe om hvilket nivå av kognitive krav oppgavene stiller. Videre ble algebrainnholdet i oppgavene analysert ved bruk av Kierans (2004a) GTG-modell for algebraiske aktiviteter.

Analysemetoden som ble brukt i alle fire analysene, var tematisk analyse (Braun & Clarke, 2006). Det ble også brukt en klassifisering utfra de analytiske rammeverkene som er presentert i rammeverkkapittelet. Denne seksjonen vil gi en mer detaljert forklaring på hvordan dette har blitt gjennomført, og det vil i kapittel 4 bli gitt eksempler med beskrivelser og bilder, og resultater fra analysen.

En kvalitativ dokumentanalyse består av skumlesing (overfladisk undersøkelse), lesing (grundig undersøkelse) og tolkning, dette kombinerer elementer fra både innholdsanalyse og tematisk analyse (Bowen, 2009, s. 32). For å finne programmeringsoppgaver til de videre analysene av oppgavene i læreverkene, foretok jeg først en kvalitativ dokumentanalyse. Etter å ha analysert læreverkene og gjort meg kjent med oppgavene, valgte jeg ut alle programmeringsoppgavene i læreverkene til videre analyse. Jeg startet med å identifisere oppgaver som kunne klassifiseres som algebraoppgaver innenfor programmering i de to læreverkene. Den tematiske analysen krever en nøye og mer fokusert lesing og gjennomgang av datamaterialet, dataene kodes og det konstrueres kategorier basert på egenskapene ved dataene (Bowen, 2009, s. 32). Når algebraoppgavene var identifisert, kodet og kategorisert, startet prosessen med å analysere algebraoppgavene utfra programmeringstype, kognitive krav og algebraisk aktivitet.

Ved tematisk analyse ønsker en å finne mønster i dataene, samtidig er det viktig at den teoretiske posisjonen en inntar er tydelig og transparent, og rammeverket og metodene må ha sammenheng med hva en som forsker ønsker å finne ut av (Braun & Clarke, 2006). Min tematiske analyse faller innenfor typen «teoretisk tematisk analyse», som går ut på at jeg som forsker har en spesifikk teoretisk interesse innenfor temaet. Dette vil gi en mer detaljert analyse av noen deler av dataene, nemlig algebraoppgavene innenfor programmering i læreverkene, og analysen vil gi en mindre detaljert beskrivelse av dataene i helhet, altså alle programmeringsoppgavene i læreverkene (Braun & Clarke, 2006). Kodeprosessen og analyseprosessen har foregått ved hjelp av Braun & Clarkes (2006) trinn for trinn beskrivelse av tematisk analyse, og det gis i dette kapittelet også beskrivelser av denne. Programmeringstypene er navngitt og identifisert ved inspirasjon fra Bråting og Kilhamns (2021b) rammeverk for programmeringsoppgaver.

3.3.1 Analysemetode for å identifisere matematisk tema i programmeringsoppgaver

Braun og Clarke (2006, s. 87) har i sin artikkel laget en trinn for trinn oppskrift på hvordan en går frem for å gjøre en tematisk analyse, og det er disse prinsippene jeg har brukt for å analysere programmeringsoppgavene. Det vil også bli vist eksempler til kodeprosessen under de seks fasene i kapittel 4. Eksemplene vil inneholde kategorier, underkategorier og koder fra kodeprosessen. Jeg startet kodeprosessen ved å ta for meg en og en oppgave, og kode oppgaven etter matematisk tema og etter hvilken type programmering elevene blir satt til å gjøre. I Nvivo vil jeg til slutt få overordnede temaer der jeg kan se antall koder som finnes

under hvilket tema, samtidig som jeg kan se hva slags type programmeringsoppgaver som er blitt kodet under algebra.

I fase 1 er det viktig at en er godt kjent med datamaterialet sitt, og har lest gjennom det flere ganger (Braun & Clarke, 2006, s. 87). Analysen startet med å først og fremst samle bilder av alle programmeringsoppgaver fra Campus Inkrement i et dokument, og oppgavene fra Matemagisk 8-10 i et annet dokument. For å kunne beskrive oppgavene og kode oppgavene på en så ryddig og oversiktlig måte som mulig, har jeg først sett gjennom alle programmeringsoppgavene i læreverkene, altså analysert dokumentene. Jeg gikk nøye gjennom oppgavene for å bli kjent med innholdet, det gjorde meg godt kjent med datamaterialet mitt, slik som Braun og Clarke (2006) sier er viktig i den første fasen av en tematisk analyse. Oppgavene har blitt kodet to ganger, en kode for programmeringstype, og en annen for matematisk tema.

I fase 2 starter kodeprosessen og analyseprosessen, kodene organiserer dataene i meningsfulle grupperinger, men kodene er forskjellige fra teamene som ofte er bredere eller dekker større områder av datamaterialet (Braun & Clarke, 2006, s. 88). De viktigste rådene Braun og Clarke (2006) gir, er å kode for så mange potensielle temaer som mulig. I fase to har jeg benyttet meg av programmet Nvivo til å kode oppgavene, dette er fordi Nvivo gir meg mulighet til å importere dokumentene med skjermbildene av oppgavene, og kode på en oversiktlig måte, samtidig som kodene enkelt kan endres og fjernes. Det er også enkelt å legge til flere koder etter hvert. Nvivo gir også muligheten til å sammenligne dataene og kodene på en oversiktlig måte. Fordelen med Nvivo er at det er oversiktlig ved et stort antall koder, og det er lett å samle kodene til større temaer og kategorier. Dermed mente jeg at det mest gunstige ville være å kode i Nvivo da jeg kom til å få mange koder basert på oppgaveantall og to koder per oppgave, en for matematisk tema, og en for programmeringstype.

Fase 3 begynner når alt datamaterialet har blitt kodet og samlet og en har en lang liste med koder, i denne fasen er målet for analysen at kodene skal sorteres sammen til potensielle temaer (Braun & Clarke, 2006, s. 89). En ender nå opp med temakandidater og underkategorier, men det er usikkert om temaene er ferdigstilte, må kombineres, endres på, separeres eller fjernes (Braun & Clarke, 2006, s. 90-91). Under fase 3 i min analyse startet arbeidet med å lage temaer eller kategorier. Når alle oppgavene var kodet, samlet jeg koder som hensiktsmessig kunne plasseres sammen i større temaer. Jeg begynte med å gå gjennom kodene og datamaterialet og se hvilke matematiske temaer som finnes i oppgavene og hvilke typer programmeringsoppgaver som finnes, og samlet sammen koder under større kategorier.

I fase 4 forbedres temaene, noen temaer er kanskje ikke temaer likevel, flere temaer kan gå sammen til et tema og noen må kanskje deles opp til flere temaer. Hvis det tematiske kartet virker og temaene gir mening, kan en gå videre til neste fase (Braun & Clarke, 2006, s. 91-92). I analysens fase 4 gikk jeg gjennom temaene og kategoriene på nytt, for å sjekke om alle kodene som var plassert under de midlertidige temaene passet inn. Når alt var gjennomgått, gikk jeg i gang med fase 5.

I fase 5 forbedres og defineres temaene en vil presentere i analysen. Det viktigste i slutten av denne fasen er å klart kunne definere hva temaene er og ikke er. Det kan en sjekke ved å se om en kan beskrive hva hvert tema inneholder med noen få setninger (Braun & Clarke, 2006, s. 92). Et eksempel på dette vil være å prøve å beskrive hva slags oppgaver som ligger under temaet «algebra». I denne delen av analysen ble altså temaene dobbeltsjekket, og det ble gjort endringer dersom det var nødvendig.

Den 6. og siste fasen begynner når alle temaene er ferdig, målet er å fortelle historien til datamaterialet på en måte som overbeviser leseren om verdien og validiteten av analysen (Braun & Clarke, 2006, s. 93). Det påpekes at det er viktig å vise til eksempler som inneholder det essensielle en ønsker å vise uten unødvendig kompleksitet (Braun & Clarke, 2006, s. 93). Det er denne fasen jeg har brukt når jeg har vist til dataene i studien, og fremstilt resultater basert på kodingen av datamaterialet. Jeg har også vist til eksempler i studien som viser det essensielle som er identifisert under analysen, og gitt eksempler for å illustrere valgene som er tatt ved kategoriseringen og kodingen i analyseprosessen. Det er fra fase 6 temaene og kategoriene som skal brukes videre hentes. Jeg bruker disse resultatene videre i studiens analysekapittel, resultatkapittel og diskusjonskapittel. Resultatene fra denne fasen er grunnlaget for å danne argumenter i henhold til problemstillingen (Braun & Clarke, 2006).

3.3.2 Analysemetode for å identifisere programmeringstype i programmeringsoppgaver

Jeg har på samme måte som analysemetoden for å identifisere matematisk tema, brukt tematisk analyse for å identifisere programmeringstyper blant programmeringsoppgavene. Oppgavene ble igjen analysert og kodet, nå etter programmeringstype, ved bruk av Braun og Clarkes (2006) seks faser for tematisk analyse. Jeg kommer derfor ikke til å gi en gjennomgang av fasene igjen under dette delkapittelet da det er gjort i delkapittel 3.4.1. Resultater og eksempler fra denne analyseprosessen presenteres også i kapittel 4.

For å identifisere og kode programmeringstypene i oppgavene, har jeg hentet inspirasjon fra Bråting og Kilhamns (2021b) rammeverk som er presentert i kapittel 2. Her gir forskerne

eksempler på ulike typer programmeringsoppgaver vi finner. Jeg analyserte type programmeringsoppgaver med Bråting og Kilhamns (2021b) kategorier som utgangspunkt. Rammeverket viser at det finnes ulike aspekter ved programmeringsoppgaver, og det er disse ideene jeg har tatt med meg videre når jeg har kodet programmeringsoppgavene for ungdomstrinnene. I og med at Bråting og Kilhamn har undersøkt programmeringsoppgaver på barnetrinnet, og ikke spesifikt innenfor algebra, har jeg ikke brukt nøyaktig de samme kategoriene, men tilpasset dem til mitt datamateriale. På denne måten kan vi få informasjon om hvordan ulike typer programmering kan ha sammenhenger med det matematiske innholdet i form av kognitive krav og algebraiske aktiviteter. Kategoriene jeg endte opp med å bruke i kategoriseringen av programmeringstype i analysen er:

Lag et program (følg)	Oppgaver der elevene skal lage et program etter gitte instrukser eller gitte problemer
Tolke og forklare	Oppgaver der elevene skal forklare og tolke allerede gitte programkoder
Kopier og utvid/endre	Oppgaver der en kode er gitt, og elevene skal utvide eller endre denne koden for å utvide et program, eller få et program til å fungere bedre
Feilsøk	Oppgaver der elevene skal feilsøke gitte koder som inneholder feil, og finne ut hva det er som gjør at programmer ikke fungerer

«Lag et program» kan sammenlignes med kategorien «følg» og «forme og skape» i Bråting og Kilhamns (2021b) rammeverk. Elever skal følge instruksjoner og komme frem til et resultat ved å bruke koding. «Tolke og forklare» er tett knyttet til forklare, men det er ikke gitt at elevene skal bruke naturlig språk i forklaringen, det kan også være oppgaver der elever skal tolke en gitt programkode og finne ut hva programmet løser og avgi et svar. «Feilsøk» finner vi også i Bråting og Kilhamns rammeverk, her skal elevene finne og/eller rette opp feil i koder.

Hver oppgave telles i analysen opp som en enkelt oppgave til tross for at noen oppgaver inneholder flere deloppgaver. Enkelte oppgaver kan bli kategorisert innenfor to kategorier i analysen, ettersom deloppgavene kan være forskjellige og en todelt kategorisering vil være

hensiktsmessig. Et eksempel kan være at to forskjellige deloppgaver inneholder to forskjellige algebraiske aktiviteter. Den tematiske analysens resultater for matematisk tema og programmeringstype gir en oversikt over matematisk tema i oppgavene, og samler alle algebraoppgavene til videre analyser på en oversiktlig måte. Resultatene identifiserer samtidig programmeringstypene blant oppgavene, og kan videre brukes til å si noe om sammenhengen mellom matematisk innhold og programmeringstype.

3.3.3 Analysemetode for analyse av kognitive krav i oppgaver

For å identifisere de kognitive kravene i algebraoppgavene, ble det benyttet en tematisk analyse. Rammeverket som ble brukt for å kategorisere de kognitive kravene, er Smith og Steins (1998) rammeverk for kognitive krav. Rammeverket kan brukes mer generelt på oppgaver i matematikk uten å observere elever som faktisk løser oppgaver. Rammeverket nivå-kategoriserer oppgaver ut ifra hvilke kognitive krav oppgavene stiller til elevene, og det inneholder karakteristikker for alle nivåene. Rammeverket gir en indikator på hvor kognitivt krevende oppgaver er. Jeg skal her presentere min tolkning av rammeverket og hvordan rammeverket har blitt brukt til å analysere og kategorisere algebraoppgaver innenfor programmering i denne studien.

Alle de fire kategoriene fikk tildelt forskjellige navn i form av forkortelser som ble brukt i kode- og analyseprosessen (Lm, Lp, Hp, Hm), slik at oppgavene kunne beskrives og kodes mer effektivt. Oppgavene ble kvalitativt beskrevet og analysert for så å bli plassert i nivåer av kognitive krav, og oppgavenes plassering i læreverket og læreverkenes eksempeloppgaver ble også tatt i betraktning. Dette skyldes for eksempel kategorien memorering, da en eksempeloppgave som ligner svært på oppgaven som er plassert etter eksempelet kan påvirke de kognitive kravene oppgaven stiller. De fire ulike nivåene av kognitive krav samt beskrivelser av hva som kjennetegner de ulike nivåene (Smith & Stein, 1998) er gitt under:

Memorering (Lm)	Ved analyse av memoreringsoppgaver må en være obs på at tidligere gitte oppgaver eller oppgavers plassering kan være en faktor. Hvis elevene får presentert en eksempeloppgave som er ferdig programmert, og deretter en oppgave der de skal følge samme prosedyren, men sifrene i matematikkstykket er endret, vil denne oppgaven plasseres under kategorien memorering. Dette er fordi elevene da bare kan bruke de samme programmeringsblokkene eller skrive nøyaktig de samme kodene, og kun endre på noen siffer og få riktig svar.

Prosedyrer uten forbindelser (Lp)	Oppgaver der elevene ikke skal reprodusere noe, men likevel blir gitt en tydelig fremgangsmåte eller prosedyre, havner under kategorien prosedyrer uten forbindelser. Min tolkning av dette er oppgaver der elever for eksempel skal bruke tidligere lærte prosedyrer eller algoritmer for å løse blant annet ligninger, og oppgavene ligner tidligere gitte oppgaver.
Prosedyrer med forbindelser (Hp)	Fremgangsmåtene som blir gitt ved oppgaver under prosedyrer med forbindelser består ikke utelukkende av algoritmer, og det brukes gjerne flere representasjonsformer, og elevene kan ikke følge generelle prosedyrer de har lært tidligere uten å tenke over om det de gjør er rett eller galt. Disse oppgavene ligger nærmere problemløsningsoppgaver enn prosedyrer uten forbindelser, og elevene må forstå prosedyrene for å kunne anvende dem riktig. Det er mer fokus på prosedyre enn resultat.
Gjøre matematikk (Hm)	Den siste kategorien å gjøre matematikk, omfavner oppgaver der elevene ikke blir gitt noen fremgangsmåter, og det kreves at elevene skal utforske, analysere og bruke relevant kunnskap og erfaring for å selv klare å jobbe seg gjennom oppgaven og komme til en løsning. Denne kategorien kan for eksempel inneholde problemløsningsoppgaver og utforskningsoppgaver.

Ved å bruke Smith og Steins rammeverk, har jeg analysert og kodet algebraoppgavene i datamaterialet og plassert oppgavene innenfor de fire nivåene memorering (Lm), prosedyrer uten forbindelser (Lp), prosedyrer med forbindelser (Hp) og gjøre matematikk (Hm). Oppgavene har blitt kategorisert etter beskrivelsene Smith og Stein (1998) gir for oppgaver innenfor de fire forskjellige nivåene. Oppgavene som kodes som Lm og Lp er oppgaver som stiller lave kognitive krav til elevene. Oppgaver som ble kodet som Hp og Hm stiller høye kognitive krav til elevene. Oppgavene innenfor Hp og Hm vil ligge nærmere problemløsningsoppgaver, og kreve forståelse for prosedyrer og bruken av dem, og er derfor høyere kognitivt krevende.

Det som skiller prosedyrer uten forbindelser fra prosedyrer med forbindelser, er at oppgavene har lite tvetydighet rundt hvordan de løses, det kreves ikke forklaringer eller beskrivelser av hva som er gjort og hvorfor svaret er rett, og oppgavene er resultatfokusert. Dersom en oppgave skal være høyere kognitivt krevende, og plasseres under prosedyrer med forbindelser, må elevers oppmerksomhet være rettet mot selve prosessen og prosedyren med å komme frem til svaret. Oppgaven må gjerne også kreve bruk av tidligere lærte prosedyrer på en annen måte, og ikke bære preg av mengdetrening der samme prosedyre brukes likt gang på

gang. Både oppgaver under prosedyrer uten/med forbindelser vil kreve bruk av prosedyrer, men oppgavene under prosedyrer med forbindelser krever at elevene forstår prosedyrene de bruker, og ikke bare bruker de på «autopilot» for å komme frem til en løsning, slik som under prosedyrer uten forbindelser. Det er mer fokus på prosedyren og forståelsen av den, fremfor det å komme frem til riktig svar.

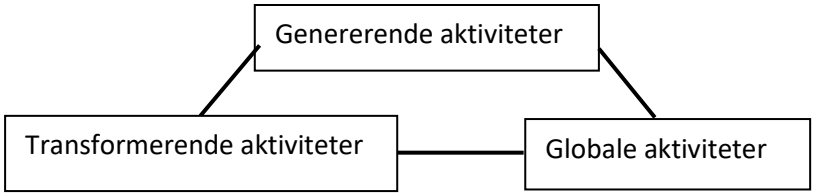
De kognitive kravene har blitt analysert utfra matematikken i oppgavene. Dette er gjort fordi problemstillingen tar for seg og fokuserer på det matematiske innholdet i oppgavene. Analysen av programmeringstype vil ta for seg hvilke potensielle sammenhenger vi finner mellom det matematiske innholdet i oppgavene og programmeringstype.

Eksempler med beskrivelser og forklaringer for hvordan oppgavene er kodet og kategorisert kommer frem i kapittel 4, men rammeverket har altså blitt brukt som en oppskrift for å identifisere hva slags kognitive krav oppgavene stiller til elever, og som hjelp til å plassere oppgaver i de fire nivåene. Rammeverket brukes dermed til å identifisere det matematiske innholdet i oppgavene i form av kognitive krav, og bidrar videre i studien til resultater og diskusjon rundt det matematiske innholdet.

3.3.4 Analysemetode for analyse av algebraisk aktivitet

For å analysere algebraisk aktivitet i algebraoppgavene, brukte jeg en tematisk analyse og Kierans (2004a) kategorier og beskrivelser for algebraisk aktivitet. Når programmering blandes med algebra, er det vanskeligere å bruke rammeverket for algebraiske aktiviteter til Kieran alene, derfor er rammeverket til Smith og Stein også brukt i studien. Rammeverket til Kieran kan brukes til å se på elevers arbeid og interaksjoner når de jobber med de forskjellige algebraaktivitetene, men også til å analysere oppgaver. I denne studien er det kun innholdet i oppgavene i læreverkene som undersøkes. Rammeverket gir innsyn i hvordan lærebøker legger opp til at elever skal engasjere seg i de forskjellige algebraiske aktivitetene, og hva slags algebra som kommer til uttrykk i programmeringsoppgaver.

Algebraoppgavene har blitt plassert i de forskjellige algebraiske aktivitetskategoriene ved å ta utgangspunkt i Kierans (2004a) egne beskrivelser av de forskjellige aktivitetene. De tre forskjellige algebraiske aktivitetene er som nevnt i kapittel 2 genererende aktiviteter, transformerende aktiviteter og globale aktiviteter, og fremstilles i GTG-modellen under.



Med bakgrunn i Kierans (2004a) beskrivelser av genererende algebraiske aktiviteter, har programmeringsoppgaver der elevene for eksempel skal tolke problemsituasjoner gitt i form av tekst, og uttrykke dette matematisk, blitt kategorisert som genererende oppgaver. Også oppgaver der elevene skal tolke geometriske mønstre har blitt plassert under denne kategorien.

De transformerende aktivitetene er aktiviteter de fleste forbinder med «vanlige» regnetekniske oppgaver i algebra. Med bakgrunn i Kierans (2004a) beskrivelser av transformerende aktiviteter, er oppgaver som har blitt plassert under denne aktiviteten blant annet oppgaver der elever skal løse ligninger, regne ut funksjonsverdier og endre form på uttrykk for å bevare ekvivalens.

Det som kjennetegner de globale aktivitetene er at de inneholder generelle matematiske prosesser, og gir samtidig en kontekst for bruk av algebra, noe som kan motivere til bruken av de andre aktivitetene. Oppgavene krever heller ikke algebra eller algebraiske formler for å løses, men de kan benyttes. Disse aktivitetene krever ofte resonnering, og eksempler på slike oppgaver er modelleringsoppgaver og problemløsningsoppgaver der elevene hensiktsmessig kan benytte algebra, men ikke er nødt til å bruke det. Med bakgrunn i Kierans (2004a) beskrivelser er oppgaver som har blitt plassert under globale aktiviteter oppgaver der elever hensiktsmessig kan bruke algebra som et verktøy for å bevise, generalisere eller løse matematiske problemer der algebra nødvendigvis ikke trengs å brukes.

Oppgaver der elevene skal representere for eksempel funksjoner grafisk eller i tabeller, eller modellere, ville ved bruk av for eksempel graftegner havnet under genererende eller globale aktiviteter, avhengig av oppgaveformulering og prosedyrer som skal brukes. Når programmering benyttes, skal elevene ofte finne funksjonsverdier eller ekstremalverdier. Dette gjør at oppgavene fremdeles blir regnetekniske, og formålet er å bevare ekvivalens og løse ligningen til funksjonen. Dermed kan slike oppgaver bli plassert under transformerende aktiviteter fremfor genererende eller globale aktiviteter.

3.4 Sammenligning av kategorier

Ved å bruke disse rammeverkene som er presentert i analysemetoden sammen, vil jeg kunne si noe om det matematiske innholdet i oppgavene i form av kognitive krav og algebraisk aktivitet, samt hvordan programmeringstype kan ha en potensiell sammenheng med de kognitive kravene og/eller de algebraiske aktivitetene. Det gir også mulighet til å se hvordan de kognitive kravene, programmeringstypene og algebraiske aktivitetene potensielt kan ha sammenheng med og påvirke hverandre. Kategoriene har altså blitt sammenlignet med hverandre for å se etter strukturer og sammenhenger, og det gis eksempler på dette i kapittel 4. For eksempel har kognitive krav blitt sammenlignet med algebraisk aktivitet for å se om det er noen sammenhenger mellom nivåer av kognitive krav og type algebraisk aktivitet.

3.5 Kvalitet og etiske betraktninger

3.5.1 Reliabilitet

Reliabilitet sier noe om hvor pålitelig data er. Ting som kan påvirke påliteligheten til data er hvordan dataene er samlet inn, hvordan dataene brukes i undersøkelsen, hvor nøyaktig dataene er og hvordan dataene analyseres eller bearbeides. Noe som kan styrke reliabiliteten, er hvis flere forskere undersøker noe samtidig, det kan motvirke for eksempel partiskhet (Christoffersen & Johannessen, 2012). Fordi jeg har utført studien alene, er det er svært viktig at studiens fremgangsmåter og analyser er transparente, for å styrke reliabiliteten. Det skal også være mulig å etterprøve analyser som er gjort og de skal kunne avgi svar. Dette kan gjøres ved at jeg selv som forsker reflekterer over egen påvirkning, og at forskningsprosessen gjøres så synlig og transparent som mulig slik at andre også kan reflektere over den (Postholm & Jacobsen, 2018, s. 224). For å øke studiens reliabilitet, har jeg forsøkt å gi en så detaljert og transparent beskrivelse av analysemetoden som mulig. Jeg har også gitt begrepsavklaringer og innsikt i hvordan jeg har valgt ut læreverk, samlet inn oppgaver, og hvorfor oppgaver som er kategorisert på ulike måter, er kategorisert slik som de er. Jeg har også forklart hvorfor og hvordan spesifikke oppgaver er valgt ut til analyse, analysert og kategorisert. Alt dette øker muligheten for at analysen kan etterprøves, og er med på å styrke reliabilitet.

Eksemplene i studiens analysekapittel, vedleggene og beskrivelsene i metodekapittelet, viser og beskriver analyseprosessen og hvordan oppgaver har blitt kategorisert og analysert. Dette mener jeg kan være med på å styrke reliabiliteten, da det gir mulighet for å tydelig se hvordan analysen har foregått, og hva jeg kom frem til. I oppgaven viser jeg også til eksempler og gir beskrivelser for hvordan analyseprosessen har foregått, både med tekst og bilder. Noe en må ta i betraktning, er at måten oppgaver kategoriseres på, eller analyseres, kan variere ut ifra

hvem som analyserer dataene. Hvis oppgavene hadde blitt kategorisert ut ifra andre kriterier eller på andre måter, ville det kunne gitt forskjellig resultater. Jeg har gjort det jeg kan for å forsøke å gi studien og analysen så mye reliabilitet som mulig, ved å være objektiv i analysene og kategorisere innholdet etter beskrivelser i læreverkene og beskrivelser gitt av rammeverkene. Jeg har samtidig vært transparent, ved å vise til spesifikke eksempler og gitt begrunnelser for hvorfor oppgaver er kategorisert som de er gjort. Jeg har også tydelig begrenset hva som skal analyseres, og vært åpen om valg som er foretatt, og analysen har foregått i flere trinn for at det ikke skal oppstå usikkerhet rundt hva som skal analyseres og hvordan det har blitt analysert.

I forkant av studien har jeg blitt gjort oppmerksom på hva slags funn jeg kan forvente meg rundt blant annet mengden oppgaver innenfor et matematisk tema eller mengden oppgaver innenfor et nivå av kognitive krav eller en type algebraisk aktivitet. Dette kommer av tidligere forskning på for eksempel kognitive krav blant algebraoppgaver eller algebraiske aktiviteter i skolealgebraen. Jeg har dermed vært obs på dette og vært så objektiv som mulig for å begrense personlig partiskhet.

3.5.2 Validitet

Ved en kvalitativ studie er det svært viktig at det tas hensyn til studiens validitet. Validitet finnes i forskjellige former, disse er blant annet intern og ytre validitet, og begrepsvaliditet (Christoffersen & Johannessen, 2012, s. 24). Christoffer og Johannessen (2012) beskriver begrepsvaliditet som relasjonen mellom de konkrete dataene og fenomenet som undersøkes, og skriver at det er viktig at dataene er gode representasjoner av fenomenet. Begreper og definisjoner jeg bruker i studien må derfor være i samsvar med den teoretiske rammen, og jeg har forsøkt å få dette til ved å gjøre rede for og vise til definisjoner i kapittelet for teoretisk rammeverk, og valgt ut alle programmeringsoppgavene for ungdomstrinnene i læreverkene som data slik at det blir mest mulig sammenheng mellom problemstillingen og datamaterialet. Ved intern validitet er det viktig å tenke på om resultatene i studien er gyldige i forhold til det som er undersøkt (Christoffersen & Johannessen, 2012, s. 229). Ytre validitet sier noe om studiens overførbarhet, altså om resultatene en kommer frem til i studien er generaliserbare (Postholm & Jacobsen, 2018, s. 238).

Den ytre validiteten i denne studien er ikke høy, da det kun er to læreverk som er blitt analysert, og resultatene vil derfor ikke være generaliserbare til andre læreverk. Den interne validiteten vil etter min mening være høy, da jeg har tatt i bruk analysemetoder basert på godt

innarbeidete rammeverk som er blitt designet for oppgaveanalyse, noe som vil gi resultater som er gyldige i forhold til det som undersøkes.

3.5.3 Etiske betraktninger

I denne studien er det kun deler av læreverkene, begrenset av problemstillingen, som kommer frem. Studiens formål er å finne ut karakteristikkene ved det matematiske innholdet som finnes i programmeringsoppgaver innenfor algebra, og oppgaven er derfor vinklet inn mot det matematiske innholdet i programmeringsoppgavene innenfor algebra, og ikke alt innholdet i læreverkene. Dette er med på å forhindre å sette spesifikke læreverk i positivt eller negativt lys. Jeg er verken ute etter å reklamere for eller promotere læreverk, eller snakke ned læreverk. Studien er kun ute etter å nøytralt analysere programmeringsoppgaver i læreverkene, og det er viktig å tenke på etikken rundt analysen. Ved å være ute etter å snakke ned læreverk, som er et marked, kan en være med på å skade bruken av, salg av, og inntekten til forlagene som lager læreverkene. Jeg ønsker å fremstille læreverkene på en rettferdig måte, samtidig som jeg igjen påpeker at det kun er deler av læreverkene som kommer frem i studien, så jeg har ikke muligheten til å komme frem til konklusjoner for hele læreverket. Læreverkene i studien har heller ikke blitt satt opp mot hverandre eller sammenlignet for å eksempelvis forhindre at ett læreverk stilles i bedre lys enn et annet.

3.6 Mulige svakheter ved analysemetoden

I denne studien har de to utvalgte læreverkene Matemagisk og Campus Inkrement ikke blitt sammenlignet. For å få et større antall oppgaver å analysere i studien, har dermed alle oppgavene fra begge læreverk og begge programmeringsspråkene blitt analysert sammen. Dermed har ikke programmeringsspråkene Python og Scratch blitt sammenlignet, da læreverkene bruker forskjellig programmeringsspråk i oppgavene. Programmeringsspråkene har dermed ikke blitt tatt hensyn til i analysen, og det kan være at det er forhold ved språkene som ikke kommer frem i resultatene. En kan for eksempel risikere at noen av resultatene som har kommet frem i studien, bare gjelder for ett av programmeringsspråkene. Det kan også være sammenhenger som kun gjelder for ett av språkene. Dette er noe som kan være interessant å undersøke nærmere i videre studier.

4. Analyse og resultater

I dette kapittelet skal jeg redegjøre for analysen, eksempler fra analyseprosessen og resultatene fra analysen ved å gi beskrivelser og bildeeksempler. Først skal jeg redegjøre for den tematiske analysen under identifiseringen av algebraoppgaver og programmeringstype,

samt vise til eksempler på hvordan programmeringsoppgavene har blitt kodet og analysert ved bruk av tematisk analyse i Nvivo. Dette ble gjort for å identifisere algebraoppgavene og hva slags programmeringstyper som finnes blant oppgavene. Deretter redegjør jeg for hvordan jeg har analysert de kognitive kravene i oppgavene med eksempler. Til slutt legger jeg frem hvilke algebraiske aktiviteter som ble identifisert blant oppgavene under analysen av algebraisk aktivitet og eksempler fra denne analysen. Alle analysene har fulgt tematisk analyses fremgangsmåter, og det blir gitt beskrivelser og bildeeksempler for alle analysene. Funnene fra alle analysene legges frem under funn.

Programmet Nvivo ble brukt i den første og andre analysen. Kommentarfunksjonen i Word ble benyttet for den tredje og fjerde analysen. Dette er fordi det ved identifiseringen av algebraoppgaver og programmeringstype ville bli mange koder og Nvivo gir da muligheter for en oversiktlig kodeprosess. Ved analysen av kognitive krav og algebraiske aktiviteter var algebraoppgavene identifisert, og det var færre oppgaver å kode. Det ble også til disse analysene brukt rammeverk som er godt innarbeidet, og dermed var det like hensiktsmessig å bruke Word.

Analysen består av fire trinn, som fordeler seg slik:

1. Tematisk analyse for å identifisere algebraoppgavene
2. Tematisk analyse for identifisering av programmeringstype
3. Tematisk analyse av kognitive krav ved bruk av rammeverket til Smith og Stein for kognitive krav.
4. Tematisk analyse av algebraiske aktiviteter ved bruk av Kierans GTG-modell for algebraiske aktiviteter.

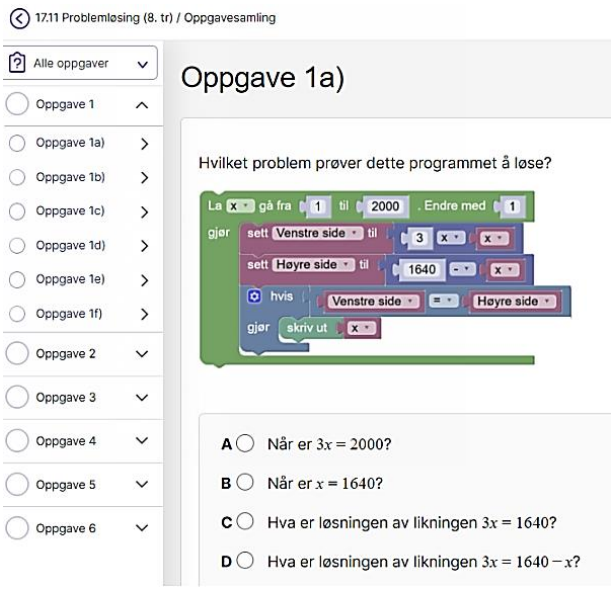
4.1 Identifisering av algebraoppgaver og programmeringstyper

For å analysere og identifisere algebraoppgavene og programmeringstypene blant disse i læreverkene Matemagisk og Campus Inkrement, har jeg brukt analysemetoden tematisk analyse og de seks fasene for tematisk analyse av Braun og Clarke (2006) som jeg beskrev i kapittel 3.

4.1.1 Identifisering av algebraoppgaver

Fase 1 i analyseprosessen var starten av analysen og arbeidet med å innhente programmeringsoppgavene som skulle analyseres videre. I fase 1 jobbet jeg også med å bli kjent med datamaterialet. Ved analysens fase 2 startet kodeprosessen, og oppgave 1a (bilde 4) er et eksempel på hvordan algebraoppgavene har blitt identifisert og kodet. Denne oppgaven

består av en programkode som løser en ligning, og elevene skal tolke programmet og finne ligningen. Oppgaven er dermed kodet til «Tolke en programkode som løser en ligning, gjenkjenne ligningen programmet løser». Det matematiske temaet er da algebra, og slik ble en algebraoppgave identifisert, og kan senere i analysen grupperes sammen med andre algebraoppgaver.



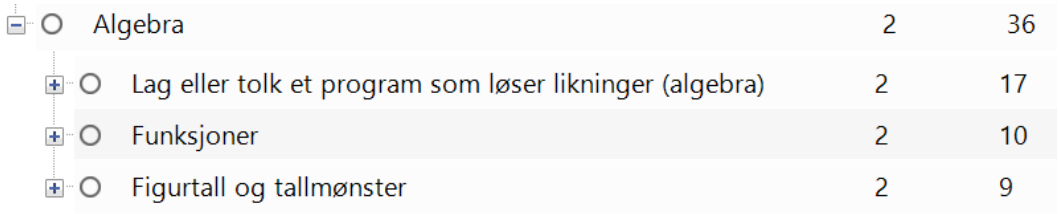
Bilde 4 - Eksempeloppgave fra Campus Inkrement.

Ved analysens fase 3 samles kodene og en får flere større kategorier. Et eksempel på to større kategorier er «Figurtall og tallmønster» og «Funksjoner». Begge kategoriene består av algebraoppgaver, men står under fase 3 hver for seg. Alle oppgaver som inneholdt figurtall eller tallmønster og ble kodet deretter, ble plassert sammen, og oppgaver der elevene skal jobbe med funksjoner, ble plassert sammen. Når alle kodene var samlet sammen i større kategorier, gikk jeg i gang med fase 4.

Bilde 5 – Eksempel fra fase 3 i Nvivo.

Under fase 4 og 5 startet arbeidet med å forbedre kategorier og samle kategorier til overordnede temaer. Her ble for eksempel «Figurtall og tallmønster» og «Funksjoner» samlet sammen under temaet «Algebra». Dermed ble etter hvert alle algebraoppgavene samlet inn

under et overordnet tema som gjør den videre analysen mer oversiktlig, og en kan enkelt se hvor mange og hva slags algebraoppgaver som har blitt identifisert.



Algebra	2	36
Lag eller tolk et program som løser likninger (algebra)	2	17
Funksjoner	2	10
Figurtall og tallmønster	2	9

Bilde 6 - Eksempel fra det overordnede temaet algebra i Nvivo.

Den tematiske analysens siste fase, 6, starter når alle temaene er ferdige og forbedret i fase 5. Bilde 7 viser en oversikt over alle de overordnede temaene etter at alle oppgavene var identifisert, kodet og samlet sammen, og temaene var forbedret og ferdigstilt.

Codes

Name	Files	References
Lag et program (følg)	2	120
Sannsynlighet	2	46
Algebra	2	41
Tallregning og tallforståelse	2	32
Tolke og forklare	2	30
Kopier og utvid og eller endre	1	23
Geometri	2	22
Brøk, prosent og desimaltall	2	18
Programmering med løkker og tall	1	10
Statistikk	1	6
Feilsøk	1	2

Bilde 7 - Overordnede temaer etter fase 6 i Nvivo.

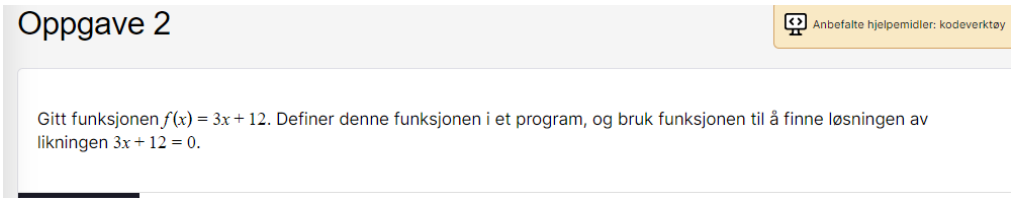
De matematiske temaene som inneholdt flest oppgaver, var «Algebra» og «Sannsynlighet». I og med at jeg gjorde meg godt kjent med datamateriale, slik som Braun og Clarke (2006) argumenterer for at det skal gjøres ved tematisk analyse, var jeg allerede før jeg begynte kodeprosessen ganske godt kjent med hvilke oppgaver som skal plasseres under hvilken kategori. Oppgaver som havnet under kategorien algebra, omhandler alt fra algebraiske ligninger til figurtall og algebraiske funksjoner. Det er også blant annet tekstoppgaver der elever skal finne ukjente mengder innenfor algebraoppgavene, her må elevene for eksempel selv sette opp algebraiske uttrykk. Det var altså varierte oppgaver innenfor algebra, og det var både oppgaver innenfor tekstbasert programmering og blokkbasert programmering.

4.1.2 Identifisering av programmeringstype

For å identifisere programmeringstype, har jeg fulgt de samme prinsippene og fasene som for identifiseringen av matematisk tema, og kommer derfor ikke til å gjengi beskrivelsene av fasene for kodeprosessen like omfattende.

Oppgave 1a (bilde 4), brukt som eksempel i delkapittel 4.1.1, ble kodet til «Tolke en programkode som løser en ligning, gjenkjenne ligningen programmet løser». *Tolke en programkode* blir da typen programmering elevene skal bruke. I Nvivo vil da den samme koden kunne legges inn under temaene «Tolke en programkode» og «Algebra», på denne måten vil en da kunne se både hvilket matematisk tema oppgaven faller under, samt hvilken type programmeringsoppgave det er.

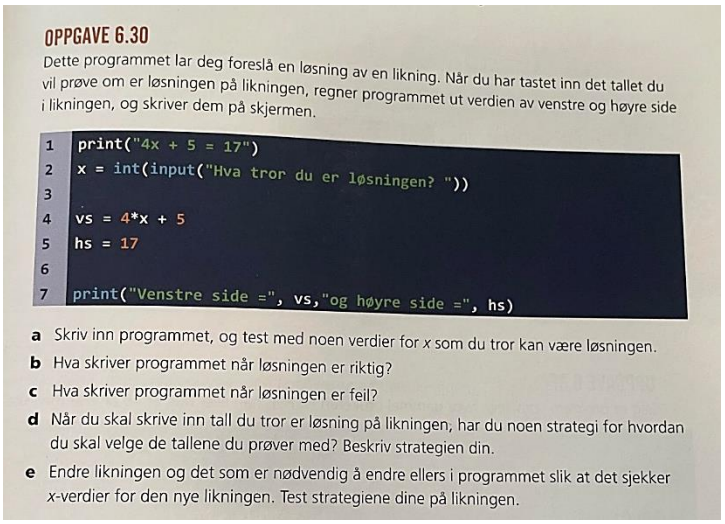
Andre eksempler på programmeringstyper er oppgave 2 (bilde 8), som er blitt kodet til programmeringstypen «Lag et program», dette er fordi oppgaven ber elevene om å definere en funksjon i et program og lage en programkode som finner løsningen av ligningen.



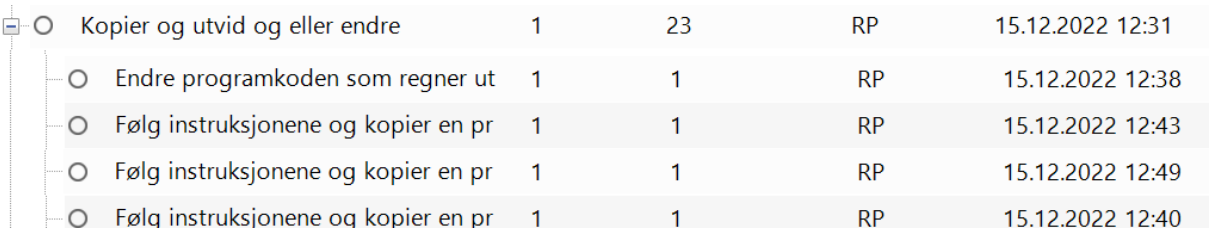
Bilde 8 - Oppgave fra Campus Inkrement

Oppgave 6.30 (bilde 9) er kodet til «Kopier og utvid og/eller endre», dette er fordi elevene i oppgaven skal kopiere en gitt kode, og deretter finne ut hva koden gjør, for så å endre programkoden.

Dette er eksempler på de tre programmeringstypene som er identifisert blant algebraoppgavene. Under fase 3 startet også arbeidet med å lage større kategorier eller temaer. Da ble oppgaver med samme programmeringstyper gruppert. Bilde 10 viser eksempel på en av programmeringstypene og kodene under kategorien.



Bilde 9 - Oppgave fra Matemagisk



Kopier og utvid og eller endre	1	23	RP	15.12.2022 12:31
Endre programkoden som regner ut	1	1	RP	15.12.2022 12:38
Følg instruksjonene og kopier en pr	1	1	RP	15.12.2022 12:43
Følg instruksjonene og kopier en pr	1	1	RP	15.12.2022 12:49
Følg instruksjonene og kopier en pr	1	1	RP	15.12.2022 12:40

Bilde 10 - Eksempel fra koding av programmeringstype

Under fase 4 og 5 jobbet jeg med å gå gjennom de overordnede temaene og kategoriene på nytt. Når jeg var ferdig med den tematiske analysen ved identifisering av programmeringstype, satt jeg igjen med fire ferdigstilte temaer som var navngitt etter programmeringstype. Disse temaene svarer til programmeringstypene som ble identifisert for programmeringsoppgavene.

Algebra	2	41	RP
Brøk, prosent og desimaltall	2	18	RP
Feilsøk	1	2	RP
Geometri	2	22	RP
Kopier og utvid og eller endre	1	23	RP
Lag et program (følg)	2	120	RP
Programmering med løkker og tall	1	10	RP
Sannsynlighet	2	46	RP
Statistikk	1	6	RP
Tallregning og tallforståelse	2	32	RP
Tolke og forklare	2	30	RP

Bilde 11 - Overordnede temaer i Nvivo.

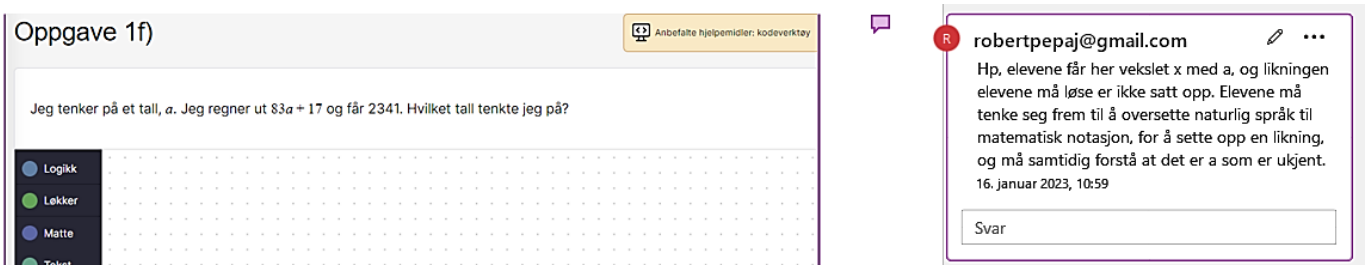
En ser av temaene at det ble identifisert flest oppgaver av programmeringstypen «Lag et program», og færrest av typen «feilsøk». Oppgavene av typen «feilsøk» var ikke algebraoppgaver. I og med at dette har vært en analyse der oppgavene har blitt kodet etter matematisk tema og programmeringstype, har hver kode dermed blitt plassert under to overordnede temaer. Et av temaene etter hvilket matematisk tema oppgaven faller inn under, og et for programmeringstype. Eksempelvis kan en oppgave være plassert under både «Algebra» og programmeringstypen «Lag et program».

De overordnede temaene eller kategoriene i det kodete datamaterialet sier noe om hvor mange algebraoppgaver det finnes i læreverkene, og hvor stor del av programmeringsoppgavene som er algebraoppgaver. De sier også noe om programmeringstypen i hver av oppgavene, og gjør at jeg enkelt kan orientere meg frem til spesifikke oppgaver. Jeg får et oversiktlig datamateriale, der jeg kan se oppgaver etter matematiske temaer og type programmering.

Disse resultatene brukte jeg til å videre analysere de kognitive kravene og algebraiske aktivitetene blant algebraoppgavene, for nå var alle oppgavene identifisert og oversiktlig samlet.

4.2 Analyse av kognitive krav i programmeringsoppgavene

De kognitive kravene programmeringsoppgavene innenfor algebra stiller til elevene, ble kodet ved bruk av Smith og Steins (1998) rammeverk som jeg har forklart og beskrevet i kapittel 2 og 3. Oppgavene ble kodet ved at de ble lagt inn i et Word-dokument, og tilegnet koder i margen ved hjelp av kommentarfunksjonen. Word ble som begrunnet brukt fremfor Nvivo ved denne analysen, fordi det ved kodingen av kognitive krav er et færre antall oppgaver og det kreves et færre antall koder, samtidig som et godt innarbeidet rammeverk er benyttet. Dermed oppnår jeg en god oversikt også ved å kode gjennom kommentarfunksjonen i Word. Oppgavene ble kodet etter hvilket nivå eller hvilken kategori de faller inn under i rammeverket, gitt av rammeverkets eksempler og forklaringer på hver kategori. Hver oppgave ble da tilegnet en kode og en forklarende tekst. Slik ble alle oppgavene kodet i rekkefølge, og deretter ble antall oppgaver innenfor de forskjellige nivåene telt opp og fremstilt statistisk ved hjelp av diagrammer. Analysen har fulgt samme fremgangsmåte som ved en tematisk analyse. Det har blitt laget ulike koder fra rammeverket til Smith & Stein, som har blitt brukt til å kategorisere oppgavene. Oppgavene har blitt kodet ved å kategoriseres ut ifra rammeverkets beskrivelser av hvilke oppgaver som hører til hvilke kategorier, samtidig som oppgavenes plasseringer i læreverkene er tatt i betraktning. Bildet under viser et eksempel på kodingen av kognitive krav for en oppgave hentet fra Campus Inkrement.



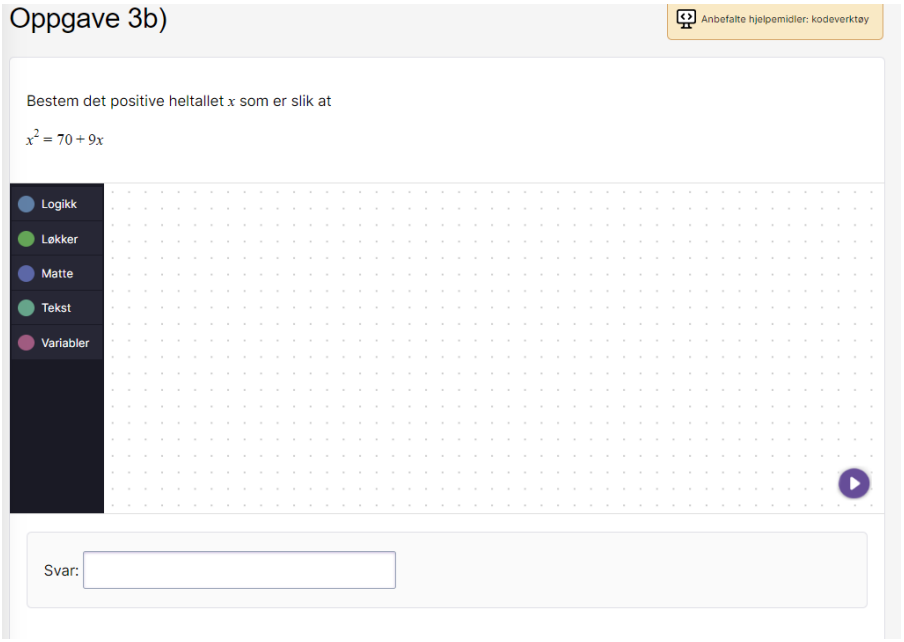
Bilde 12 - Eksempel på koding av kognitive krav for en algebraoppgave

Det har vært ulike krav til hvordan oppgavene er blitt plassert. For at en oppgave skal kategoriseres som *memorering* (Lm), der L står for lavere kognitive krav, har forelesningsvideoer og eksempler elevene blir presentert for i læreverket blitt tatt i betraktning. Plasseringen av oppgaven er dermed også vesentlig. Hvis en oppgave ligner veldig på en oppgave elevene er blitt presentert tidligere, blir den kategorisert som en

memoreringsoppgave dersom elevene kan gjøre det samme om igjen. Hvis en oppgave har en tydelig fremgangsmåte som er gitt av oppgaven, og oppgaven er resultatfokusert og oppgavene er algoritmiske, er oppgavene kategorisert som *prosedyrer uten forbindelser* (Lp). Oppgaver der prosedyrer skal brukes, men forståelsen av prosedyrene som brukes er i fokus, og elevene må ha en forståelse for prosedyrene, er kodet til *prosedyrer med forbindelser* (Hp), der H står for høyere kognitive krav. Det ble ikke funnet noen oppgaver innenfor kategorien *å gjøre matematikk*. Jeg vil i dette delkapittelet presentere eksempler fra analysen, og viser til eksempler for oppgaver som har blitt kodet til hver kategori av kognitive krav, for å gi en bedre oversikt av hvordan og hvorfor oppgaver har blitt kategorisert slik de har gjort.

4.2.1 Oppgaver i kategorien memorering

Oppgave 3b er hentet fra Campus Inkrement og er kodet som Lm, som svarer til memorering i rammeverket til Smith & Stein (1998). Elevene skal ved hjelp av blokkprogrammering lage en programkode som finner det positive heltallet x som gir riktig svar i ligningen som er gitt av oppgaveteksten. Oppgaven er kodet til memorering på bakgrunn av eksempeloppgaven gitt i delkapittelet i læreverket.

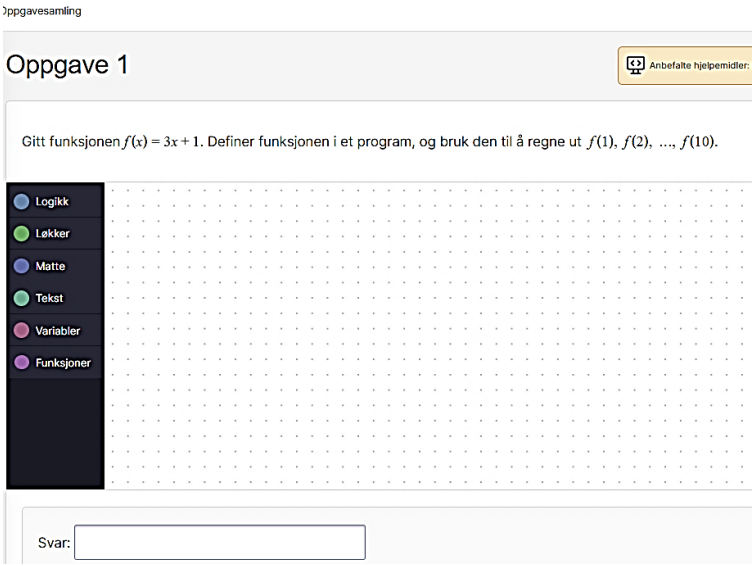


Bilde 13 - Oppgave fra Campus Inkrement.

Bilde 14 - Eksempeloppgave fra Campus Inkrement.

Oppgaven er kodet til memorering fordi eksempeloppgaven over er lik oppgaven som gis. Det som skiller memorering fra prosedyrer uten forbindelser, er at oppgavene innenfor prosedyrer uten forbindelser ikke nødvendigvis er like oppgaver elever har møtt på før, men prosedyrene og fremgangsmåtene som skal brukes, er like og kan brukes om igjen på samme måte. Elevene skal lage en programkode for å finne verdien for x i andregradsligningen, noe som eksempelet også går ut ifra. Elevene kan dermed kopiere programblokkene fra eksempelet og kun endre verdiene for venstre side og høyre side av ligningen som er gitt av oppgaven, og dermed er det ikke en kognitivt krevende oppgave og en oppgave der elevene bare kan bruke fremgangsmåten de har sett tidligere ved å memorere denne eller se på denne for å løse oppgaven. Oppgaven krever altså at elevene gjengir eller bruker regler og formler, den er ikke tvetydig, men spesifikk og krever reproduksjon, og det er tydelig for elevene hva som skal gjøres eller reproduseres. Det matematiske innholdet i oppgaven er andregradsligninger innenfor algebra, men oppgaven løses ved å bruke «gjett og sjekk» ved programmering. Dette gjør at elevene ikke selv må prøve å sette ulike verdier for x i høyre og venstre side av likhetstegnet og regne seg frem for å komme frem til en løsning, utregningen er noe programmet gjør for elevene.

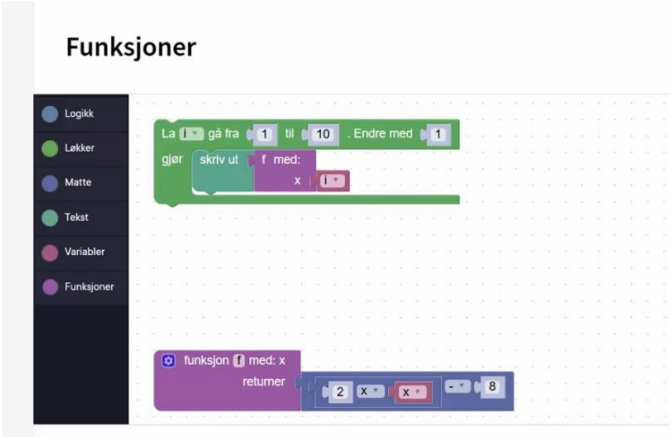
Oppgave 1 innenfor delkapittelet funksjoner ble også kodet til memorering. I oppgaven får elevene oppgitt en funksjon, og de skal deretter definere funksjonen i et program og bruke den til å regne ut funksjonsverdier for f(1) til og med f(10). Oppgaven ble kodet til memorering fordi elevene i hvert delkapittel har tilgang til forelesningsvideoer på Campus Inkrement der de får en gjennomgang av temaet, eksempeloppgaver og skal løse oppgaver. I forelesningsvideoen går de gjennom en oppgave som ser slik ut:



Bilde 15 - Oppgave fra Campus Inkrement.

Bilde 16 - Eksempeloppgave fra forelesningsvideo på Campus Inkrement.

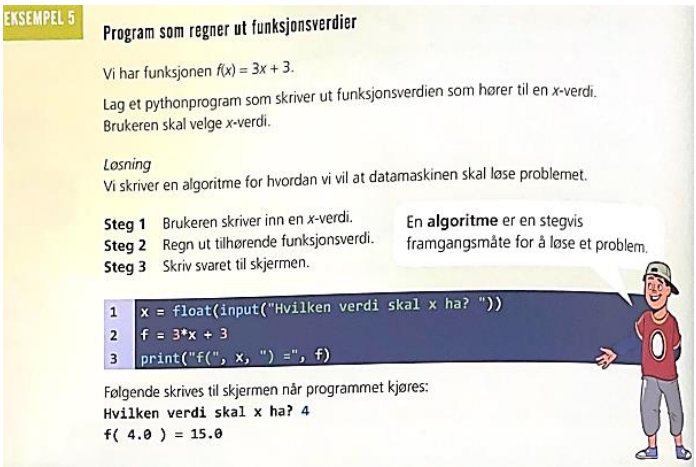
I forelesningsvideoen viser de også til en løsning av oppgaven (bilde 15). Oppgaven som gjennomgås i forelesningsvideoen er svært lik oppgave 1 som elevene skal løse i oppgavesettet for funksjoner. Det er samme funksjonsverdier som skal finnes, det eneste som er endret er selve funksjonen.



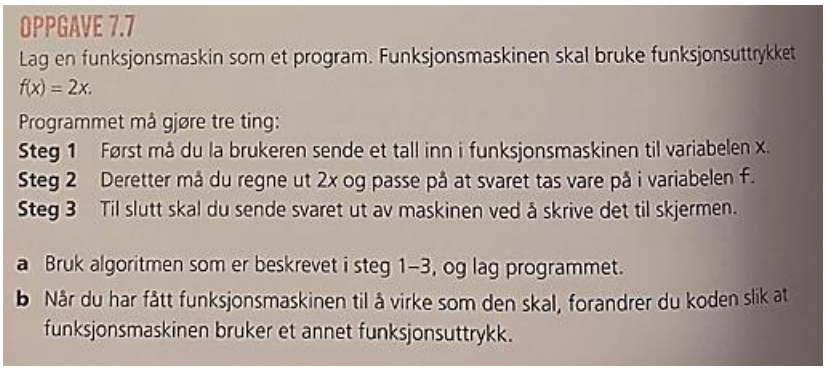
Bilde 17 - Eksempel på løsning fra forelesning på Campus Inkrement.

Dette gjør at elevene har en mulighet til å kopiere programmeringsblokkene og koden, og kun endre funksjonen som er gitt ved å endre verdien i noen av blokkene, dette kan de gjøre ved å memorere forelesningsvideoen eller ta den opp igjen når de skal løse oppgavene. Oppgaven har dermed lavt kognitivt krevende nivå, og krever at elevene gjengir eller bruker regler og formler, den er ikke tvetydig, men spesifikk, og det er klart hva som skal gjøres og reproduseres.

Vi ser slike eksempler også i læreverket Matemagisk. Eksempel 5 til høyre (bilde 16) er i læreverket plassert rett før oppgave 7.7 (bilde 17) presenteres. Dette fører til at oppgave 7.7 blir kodet som en memoreringsoppgave, da elevene kan bruke koden i eksempelet og kun endre funksjonsuttrykket i koden.



Bilde 18 - Eksempeloppgave fra Matemagisk.



OPPGAVE 7.7

Lag en funksjonsmaskin som et program. Funksjonsmaskinen skal bruke funksjonsuttrykket $f(x) = 2x$.

Programmet må gjøre tre ting:

Steg 1 Først må du la brukeren sende et tall inn i funksjonsmaskinen til variabelen x.
Steg 2 Deretter må du regne ut $2x$ og passe på at svaret tas vare på i variabelen f.
Steg 3 Til slutt skal du sende svaret ut av maskinen ved å skrive det til skjermen.

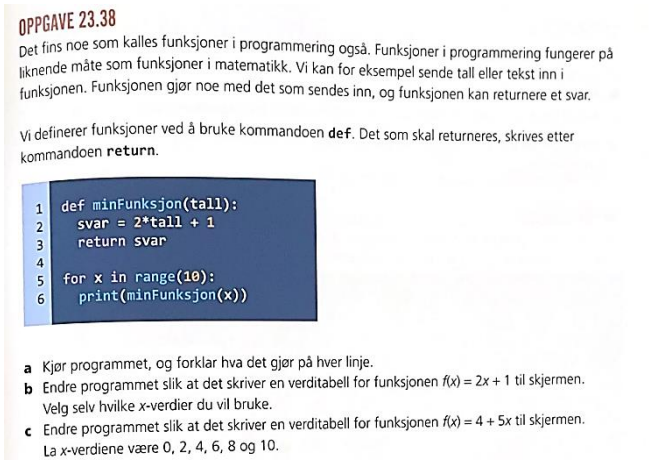
a Bruk algoritmen som er beskrevet i steg 1–3, og lag programmet.

b Når du har fått funksjonsmaskinen til å virke som den skal, forandrer du koden slik at funksjonsmaskinen bruker et annet funksjonsuttrykk.

Bilde 19 - Oppgave fra Matemagisk.

4.2.2 Oppgaver i kategorien prosedyrer uten forbindelser

I matemagisk finner vi også programmeringsoppgaver innenfor funksjoner. I denne oppgaven får elevene oppgit en programkode i Python. Elevene skal først kopiere og kjøre programkoden og forklare hva den gjør på hver linje. Deretter skal elevene i neste deloppgave endre programmet slik at det skriver ut en verditabell for samme funksjon, de skal selv velge x verdier. I siste deloppgave skal elevene bytte ut funksjonen i programkoden med en gitt funksjon og få programmet til å skrive ut en verditabell med gitte x-verdier. Oppgaven har en veldig konkret og lite tvetydig fremgangsmåte som er gitt. Både funksjonsuttrykkene og fremgangsmåtene er gitt, samtidig får elevene oppgitt en kode som er svært lik den de skal bruke videre. Det er liten tvetydighet rundt hva som skal gjøres, og det gir dermed begrenset kognitiv etterspørsel og oppgaven stiller derfor lave kognitive krav. Oppgaven er resultatfokusert og det kreves ingen forklaring på hva som er gjort etter at programkoden endres. Oppgaven har jeg dermed kodet til Lp, altså posedyrer uten forbindelser.



OPPGAVE 23.38

Det fins noe som kalles funksjoner i programmering også. Funksjoner i programmering fungerer på liknende måte som funksjoner i matematikk. Vi kan for eksempel sende tall eller tekst inn i funksjonen. Funksjonen gjør noe med det som sendes inn, og funksjonen kan returnere et svar.

Vi definerer funksjoner ved å bruke kommandoen `def`. Det som skal returneres, skrives etter kommandoen `return`.

```
1  def minFunksjon(tall):
2    svar = 2*tall + 1
3    return svar
4
5  for x in range(10):
6    print(minFunksjon(x))
```

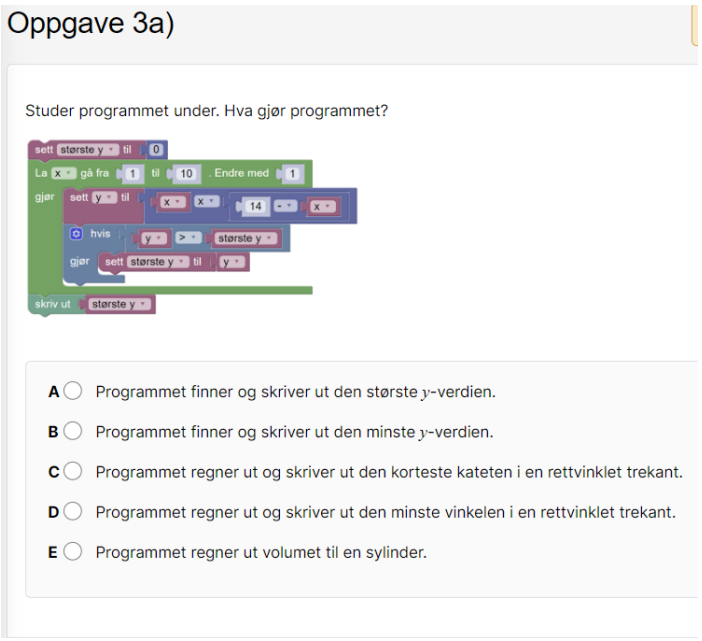
a Kjør programmet, og forklar hva det gjør på hver linje.

b Endre programmet slik at det skriver en verditabell for funksjonen $f(x) = 2x + 1$ til skjermen. Velg selv hvilke x-verdier du vil bruke.

c Endre programmet slik at det skriver en verditabell for funksjonen $f(x) = 4 + 5x$ til skjermen. La x-verdiene være 0, 2, 4, 6, 8 og 10.

Bilde 20 - Oppgave fra Matemagisk.

Oppgave 3a fra Campus Inkrement er også kodet til Lp, prosedyrer uten forbindelser. Elevene skal her finne ut hva utfallet av programmeringskoden er, nemlig at programmet skriver ut den største y-verdien. Elevene har vært borti slike oppgaver tidligere, det kommer av oppgavene som er gitt før. Siste programmeringsblokken i programmet er «skriv ut største y». Elevene kan bare ved å se på den siste blokken se hva utfallet av programmeringsblokkene er, og dermed kreves det ikke en stor kognitiv innsats for å finne ut hva programmet gjør. Svarene elevene kan velge mellom i oppgaven er også svært forskjellige, der kun to av de svarer til at programmet skriver ut en y-verdi. Dette gir mulighet for at andre svar fort kan utelukkes. Oppgaven krever begrenset kognitiv etterspørsel, og den er resultatfokusert, og elevene må ikke avgi noen forklaring på hva de velger å svare.



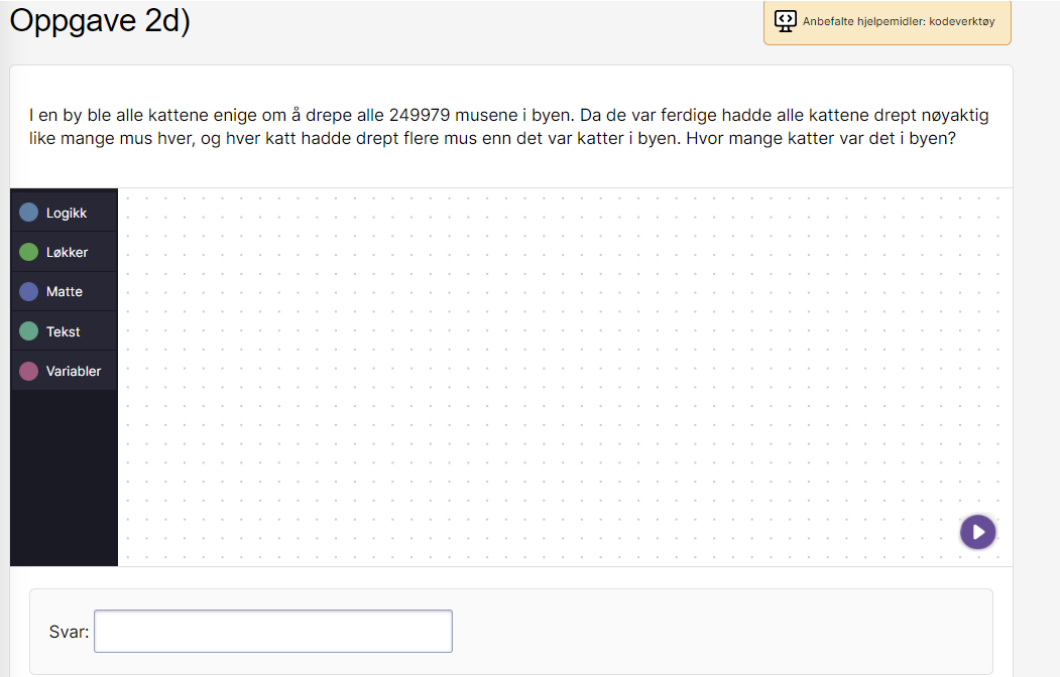
Bilde 21 - Oppgave fra Campus Inkrement.

Oppgave 1c nedenfor er også kodet til prosedyrer uten forbindelser. Dette er en tekstoppgave der elevene skal lage et program som løser en ligning, der ligningen gir svar på hvor mange griser Iben Johanne i oppgaven har totalt. Denne oppgaven kunne vært kodet til prosedyrer med forbindelser, altså en oppgave som stiller høyere kognitive krav, dersom ligningen ikke hadde vært gitt etter tekstinformasjonen i oppgaven. Elevene får allerede av teksten nok informasjon til å kunne sette opp en ligning, dette ville gitt en mer kognitivt krevende oppgave der elevene først må tolke informasjonen i tekstoppgaven, hente ut den matematiske informasjonen, sette opp en matematisk ligning, og deretter løse denne ligningen ved å lage en programkode. Fordi ligningen er gitt, blir teksten egentlig irrelevant, for målet er å finne ut hva x er. Elevene kunne like gjerne bare blitt bedt om å løse ligningen og kun blitt presentert ligningen. Fordi ligningen er gitt, fremgangsmåten er gitt, og elevene har løst ligninger ved bruk av programmering før denne oppgaven presenteres, er også denne oppgaven kodet til prosedyrer uten forbindelser. Fremgangsmåten som skal brukes for å løse ligninger er i læreverket gitt av både forelesningsvideoer, eksempeloppgaver og tidligere oppgaver elevene har løst selv, de vet dermed hvordan de skal gå frem for å løse også denne ligningen ved programmering.

Bilde 22 - Oppgave fra Campus Inkrement.

4.2.3 Oppgaver i kategorien prosedyrer med forbindelser

Oppgave 2d nedenfor er kodet til prosedyrer med forbindelser, den stiller høyere kognitive krav til elevene. Oppgaven foreslår ingen konkrete fremgangsmåter annet enn at oppgaven skal løses ved bruk av programmering. Det foreligger ikke direkte informasjon her eller tidligere i læreverket om hvorvidt man skal stille opp en ligning eller hvilken matematisk fremgangsmåte en skal bruke i programmeringen. Oppgaven ligner heller ikke oppgaver elevene tidligere blir presentert for i læreverket. Oppgaven fremstilles som en problemsituasjon, og tidligere løste oppgaver eller eksempler vil ikke være til hjelp for å direkte kunne hente ut hvilken fremgangsmåte eller prosedyre elevene skal bruke. Elevene kan bruke generelle prosedyrer, men må selv komme frem til hva slags prosedyrer som skal brukes og hvordan de skal brukes.



Bilde 23 - Oppgave fra Campus Inkrement.

Oppgaven nedenfor er også kodet til prosedyrer med forbindelser, den stiller altså høyere kognitive krav til elevene. Elevene skal her undersøke delelighet ved å bruke programmet, og deretter skal de overføre denne kompetansen til matematikk ved å representere denne matematiske regelen ved bruk av algebraiske uttrykk. Elevene må her skifte mellom representasjonsformer og beskrive hva som skjer. Oppgaven har en klar fremgangsmåte, men en konkret fremgangsmåte eller prosedyre elevene skal bruke er ikke gitt, annet enn at algebraiske uttrykk skal brukes. Oppgavens oppmerksomhet er rettet mot selve fremgangsmåten, altså å vise hvorfor denne regelen er gyldig. Oppgaven representeres også på flere måter, først ved programmering, og deretter skal elevene representere det de finner ut ved bruk av matematisk algebraisk notasjon. Oppgaven krever en prosedyre, men det er forståelsen rundt prosedyren som er i fokus, og elevene må ved algebraiske uttrykk vise hvorfor regelen stemmer, og samtidig overføre det de undersøker ved hjelp av programmeringsspråk til algebraisk notasjon.

Her ser du en forbedret versjon av pythonprogrammet.

```
1  starttall = int(input("Starttall = "))
2  antall = int(input("Antall påfølgende tall = "))
3  sum = 0
4  for tall in range(antall):
5    sum = sum + starttall + tall
6  if sum % antall == 0:
7    print("Summen av disse", antall, "naturlige tallene")
8    print("er delelig med", antall)
9  else:
10   print("Summen av disse", antall, "naturlige tallene")
11   print("er IKKE delelig med", antall)
```

g Undersøk, ved å bruke programmet, for hvilke verdier av n, summen av n påfølgende naturlige tall alltid er delelig med n.

h Vis ved regning med algebraiske uttrykk at summen av n påfølgende naturlige tall er delelig med n hvis n er et oddetall.

Bilde 24 - Oppgave fra Matemagisk.

4.3 Analyse av algebraiske aktiviteter

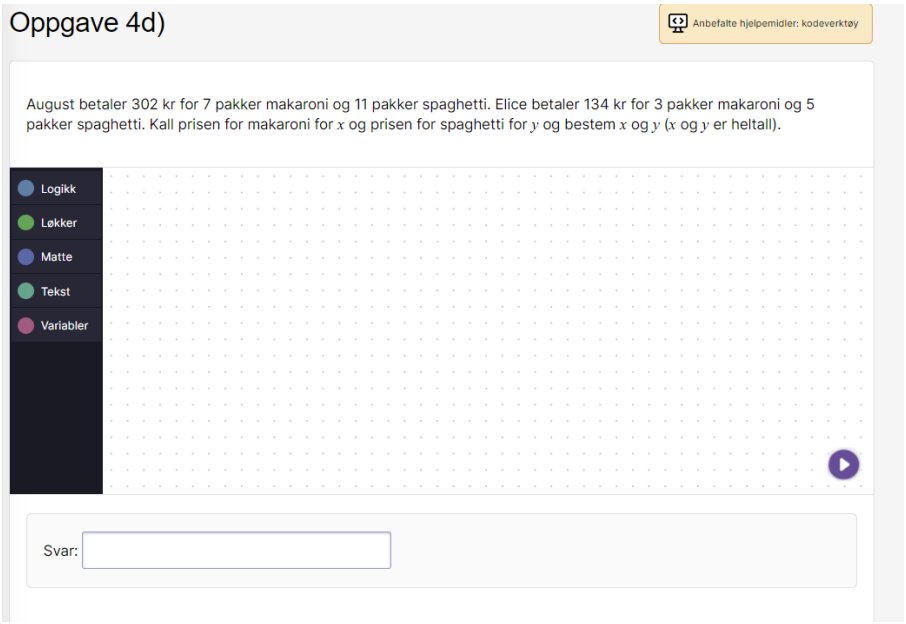
Analysen av de algebraiske aktivitetene blant programmeringsoppgavene i algebra foregikk på samme måte som analysen av de kognitive kravene, og har fulgt fremgangsmåten som er gitt i beskrivelsen av tematisk analyse tidligere. Oppgavene ble plassert i et Word-dokument og kodet etter hvilken algebraisk aktivitet som ble identifisert i oppgaven, kodene ble dannet med utgangspunkt i GTG-modellen til Kieran (2004a). Kodene ble deretter knyttet til rammeverkets beskrivelser. Også her har oppgavene blitt kodet ved bruk av Word, av samme grunnlag som analysen av kognitive krav. I margen ble det kommentert hva slags algebraisk aktivitet oppgaven inneholder eller legger opp til. Oppgaver plassert innenfor *genererende aktiviteter* er oppgaver der elevene skal representere situasjoner og egenskaper algebraisk, for eksempel tolke tekstoppgaver og lage en ligning som løser problemet teksten gir. Det som skiller disse oppgavene fra *transformerende aktiviteter*, er at de transformerende aktivitetene blant annet går ut på å løse gitte ligninger, og ikke selv lage algebraiske uttrykk ved å tolke oppgavetekst eller problemet gitt av oppgaven. De *globale aktivitetene* skiller seg fra de andre aktivitetene ved at disse oppgavene bruker algebra som et verktøy, de kan løses uten bruk av algebra, men algebra kan hensiktsmessig brukes til modellering, generalisering, problemløsning og bevisføring.

Jeg har tidligere presentert GTG-modellen med forklaringer på de ulike algebraiske aktivitetene, og jeg skal i dette delkapittelet vise til eksempler fra analysen av hver av de algebraiske aktivitetene, med en forklaring på oppgavene og hvorfor oppgavene er kategorisert slik som de er. Deretter presenteres funnene fra hele analysen i helhet.

4.3.1 Genererende algebraiske aktiviteter

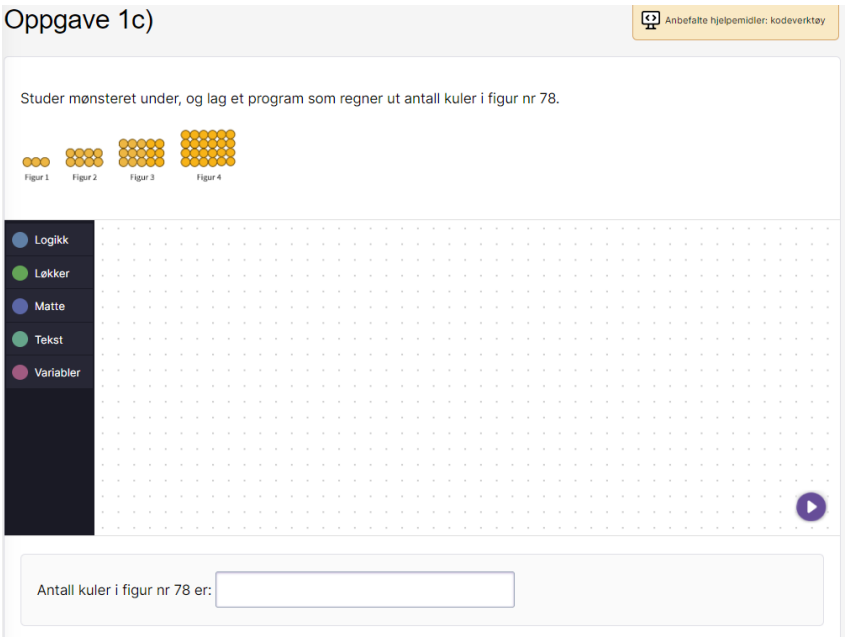
Oppgave 4d er en genererende algebraisk aktivitet, elevene må tolke en situasjon der en person handler i butikken, og får oppgitt mengden varer og prisen på varene som handles. Dette må tolkes, og deretter fremstilles algebraisk i et program, for å løse et ligningssett som vil gi svar på prisen per vare.

Elevene må først tolke språk eller tekst, for så å deretter representere dette matematisk ved å bruke algebraisk notasjon og igjen representere den algebraiske løsningen ved bruk av programmeringsspråk. Oppgaven er en genererende aktivitet på bakgrunn av forklaringene og Kierans beskrivelser av genererende aktiviteter i hennes GTG-modell.



Bilde 25 - Eksempeloppgave fra Campus Inkrement.

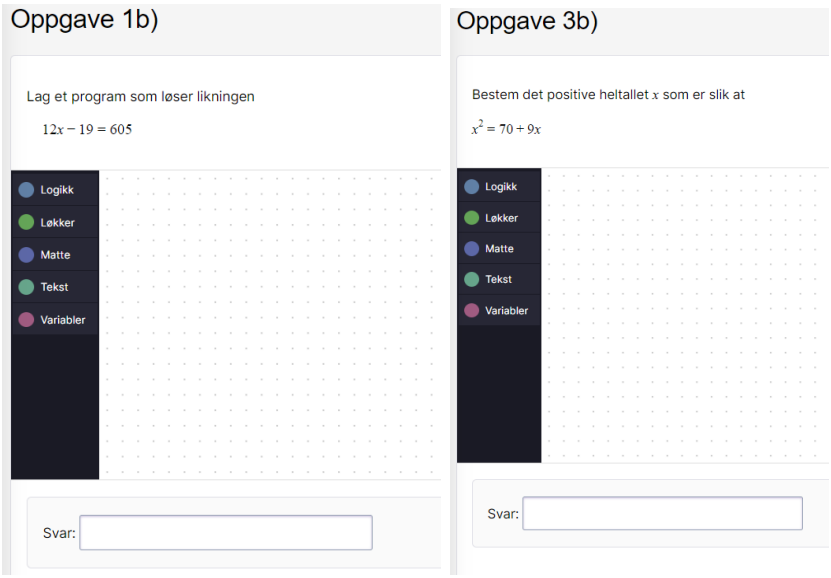
Oppgave 1c er også en genererende aktivitet. Her skal elevene først tolke og studere et geometrisk mønster, der figur 1-4 er oppgitt. Elevene skal deretter lage et program som regner ut antall kuler i figurnummer 78. Elevene må programmere et algebraisk uttrykk som gir antall figurer i figur 78, ved å tolke og forstå det geometriske mønsteret, og lage et algebraisk uttrykk som kan løse problemet oppgaven spør etter. Fokuset i oppgaven er mønstre og sammenhenger, og dette er også noe som karakteriserer genererende aktiviteter i henhold til GTG-modellens beskrivelser.



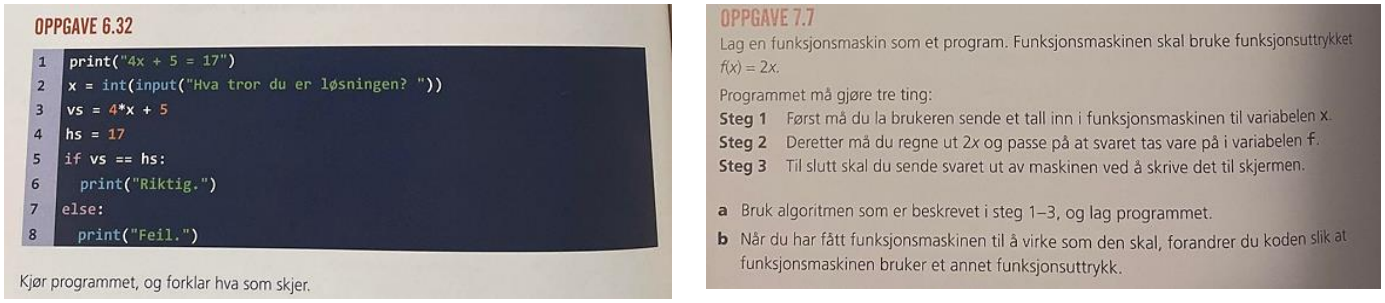
Bilde 26 - Eksempeloppgave fra Campus Inkrement.

4.3.2 Transformerende algebraiske aktiviteter

Transformerende algebraiske aktiviteter er oppgaver som mange forbinder med «vanlige» eller tradisjonelle oppgaver i skolealgebraen, altså regnetekniske oppgaver som ofte innebærer å løse blant annet ligninger ved å bruke prosedyrer eller algoritmer. Oppgave 1b, 3b, 6.32 og 7.7 er alle eksempler på transformerende aktiviteter. Oppgave 1b, 3b og 6.32 er alle oppgaver der ligninger skal løses, ved å finne riktig x-verdi slik at ekvivalensen bevares og vi finner ut verdien til den ukjente x. Oppgave 7.7 er en oppgave der elevene skal lage en funksjonsmaskin ved bruk av programmering, men funksjonen skal ikke fremstilles som en graf, funksjonens ligning skal løses, så også her ser vi at en ligning skal løses. Disse fire oppgavene (bilde 27 og 28), er alle eksempler på transformerende oppgaver som er blitt identifisert.



Bilde 27 - Eksempeloppgaver fra Campus Inkrement.



Bilde 28 - Eksempeloppgaver fra Matemagisk.

Det som kjennetegner disse fire oppgavene, og de andre identifiserte transformerende aktivitetene, er at det er algebraiske uttrykk og ligninger som skal løses ved bruk av regnetekniske ferdigheter. I og med at dette er programmeringsoppgaver, er det programmering som brukes til å løse ligningene, men det matematiske innholdet i oppgaven er fremdeles det å løse ligninger. Disse oppgavene er ikke globale fordi algebraen ikke brukes som et verktøy, og oppgavens mål er ikke blant annet bevisføring, modellering eller generalisering. Oppgavene er ikke generende, fordi elevene ikke skal tolke noe for så å

formulere algebraiske uttrykk med fokus på å representere sammenhenger, mønstre eller egenskaper. Ligningene elevene skal løse, er allerede gitt av oppgavene.

4.3.3 Globale algebraiske aktiviteter

Globale algebraiske aktiviteter er oppgaver der algebra ikke er nødt til å benyttes, men kan benyttes som et verktøy for å løse oppgavene. Eksempler på globale aktiviteter er oppgaver innenfor problemløsning og oppgaver der elever må generalisere. Disse oppgavene stimulerer ofte til resonnering, og gir algebraen en kontekst i matematikken.

Oppgave 1e er et eksempel på en global aktivitet i algebra. Elever kan velge å benytte algebra for å løse oppgaven, men det er altså ikke nødt til å brukes. Elever kan for eksempel bruke «gjett-og-sjekk» metoden, eller bruke formell algebra. Noen elever vil nok også klare å resonnere seg frem til svaret uten bruk av algebra, men algebra er hensiktsmessig å bruke, og vil også fungere som en form for bevis for at svaret elevene kommer frem til stemmer.

Bilde 29 - Eksempeloppgave fra Campus Inkrement.

Deler av denne oppgaven, deloppgave h (bilde 30) er kategorisert som en global algebraisk aktivitet, men tar en hele oppgaven i betraktning kan den sies å være både en genererende og global aktivitet. I oppgaven skal elevene undersøke numeriske sammenhenger og deretter bruke algebraiske uttrykk til å bevise at summen av *n* påfølgende naturlige tall, er delelig med *n* hvis *n* er et oddetall. Det som gjør aktiviteten genererende, er at elevene først må undersøke og tolke noe ved bruk av programmet, og det de skal tolke og uttrykke algebraisk er en numerisk sammenheng. Elevene må også bytte mellom flere representasjonsformer. Det er deloppgave h som er en global algebraisk aktivitet i denne oppgaven. Selv om algebra her må brukes gitt av oppgaveteksten, så brukes fortsatt algebra som et verktøy for å bevise noe som kan bevises også uten bruk av algebra, og


Her ser du en forbedret versjon av pythonprogrammet.

```
1  starttall = int(input("Starttall = "))
2  antall = int(input("Antall påfølgende tall = "))
3  sum = 0
4  for tall in range(antall):
5    sum = sum + starttall + tall
6  if sum % antall == 0:
7    print("Summen av disse", antall, "naturlige tallene")
8    print("er delelig med", antall)
9  else:
10   print("Summen av disse", antall, "naturlige tallene")
11   print("er IKKE delelig med", antall)
```

g Undersøk, ved å bruke programmet, for hvilke verdier av *n*, summen av *n* påfølgende naturlige tall alltid er delelig med *n*.

h Vis ved regning med algebraiske uttrykk at summen av *n* påfølgende naturlige tall er delelig med *n* hvis *n* er et oddetall.

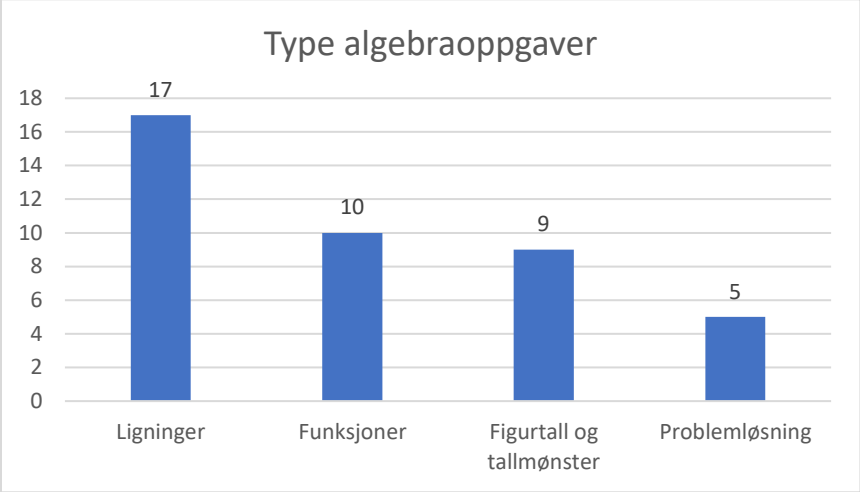
Bilde 30 - Eksempeloppgave fra Matemagisk.

bevisføring er noe som kjennetegner globale aktiviteter. Her må elevene bruke algebra som et verktøy til å bevise det de finner ut ved hjelp av programmet i deloppgave g. Elevene kunne ha kommet frem til et mønster eller hvilke tall denne regelen gjelder for, uten bruk av algebra og kun ved bruk av programmet og programmeringsspråk. Fordi elevene her må bruke algebraiske uttrykk, gir også oppgaven en kontekst for algebraen, noe globale aktiviteter gjør.

4.4 Resultater

4.4.1 Funn av algebraoppgaver blant programmeringsoppgavene i læreverkene

«Algebra»-kategorien der jeg plasserte algebraoppgavene innenfor programmering inneholder varierte oppgaver. Her finner vi oppgaver som baserer seg på å tolke programmer som løser ligninger, tekstoppgaver som baserer seg på problemløsning der elever skal lage programmer som løser problemer som inneholder ukjente mengder, og oppgaver med algebraiske funksjonsuttrykk. Det finnes også figurtallmønsteroppgaver. Jeg identifiserte 41 algebraoppgaver med algebraisk matematisk innhold blant alle programmeringsoppgavene i læreverkene, og det er disse jeg har brukt i den videre analysen for å analysere karakteristikkene ved det matematiske innholdet i programmeringsoppgaver i algebra i læreverkene. Algebraoppgavene som ble identifisert og plassert under hovedkategorien eller temaet algebra, består av 4 overordnede kategorier, som så slik ut:

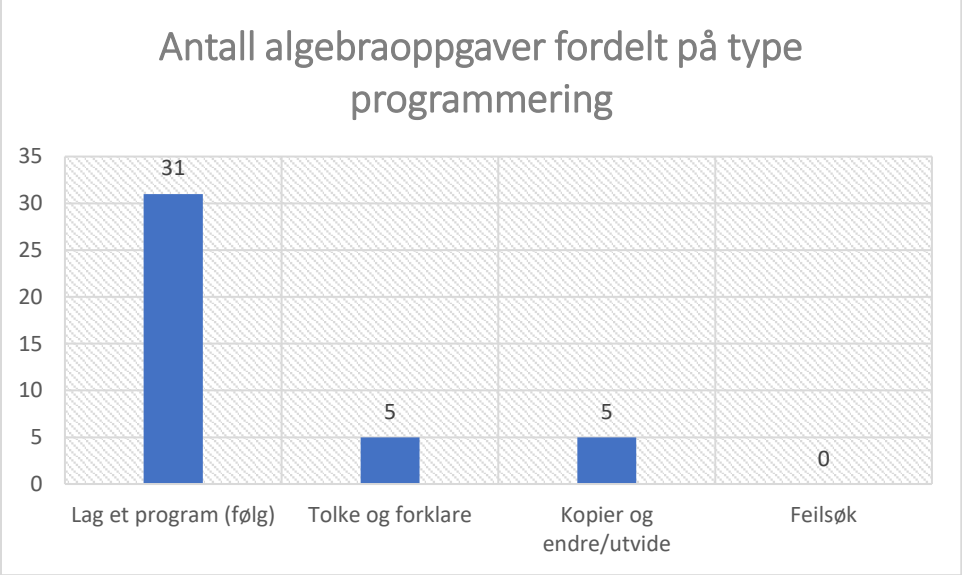


Figur 1 - Stolpediagram som viser oversikt over hva slags typer algebraoppgaver som er funnet.

Flertallet av algebraoppgavene som ble identifisert i læreverkene gikk ut på at elever skulle løse, tolke eller endre ligningsuttrykk, hovedsakelig for å bevare ekvivalens og løse ligningene eller finne ut hvordan ligninger har blitt løst. Det var 10 oppgaver som omhandlet algebraiske funksjonsuttrykk, der elevene skulle løse ligningen i et algebraisk funksjonsuttrykk, definere algebraiske funksjoner ved bruk av programmering for å regne ut

funksjonsverdier, eller finne ekstremalverdier. 9 oppgaver omhandlet figurtall og tallmønster, og 5 oppgaver var problemløsningsoppgaver.

4.4.2 Funn av type programmeringsoppgaver i læreverkene



Figur 2 - Stolpediagram som viser antall algebraoppgaver kategorisert etter type programmering.

Blant algebraoppgavene ble det funnet tre ulike programmeringstyper, disse var *lag et program (følg)*, *kopier og endre/utvide* og *tolke og forklare*. Vi ser at det er samme tendenser her, det er flest oppgaver som går ut på at elevene skal lage et program, færre oppgaver blant de andre typene programmering, og ingen oppgaver der elevene skal feilsøke.

Det som kjennetegner oppgaver innenfor «Lag et program» er at elevene får gitt en matematisk problemstilling som de skal bruke programmering til å løse. Det er variasjon innenfor oppgavene, noen oppgaver har en klar fremgangsmåte og oppgavenes plasseringer gjør at elevene kan løse oppgavene enkelt, mens andre oppgaver er mer tvetydige og krever at elevene har en større forståelse av både programmering og matematikk for å løses.

Oppgavene innenfor «Tolke og forklare» er oppgaver der en løsning på et matematisk problem allerede er gitt med en programmeringskode, som elevene skal tolke for å finne ut hva det er programmet har funnet løsningen på. Her kan det også være oppgaver der elevene skal forklare hva deres egne koder eller gitte koder gjør.

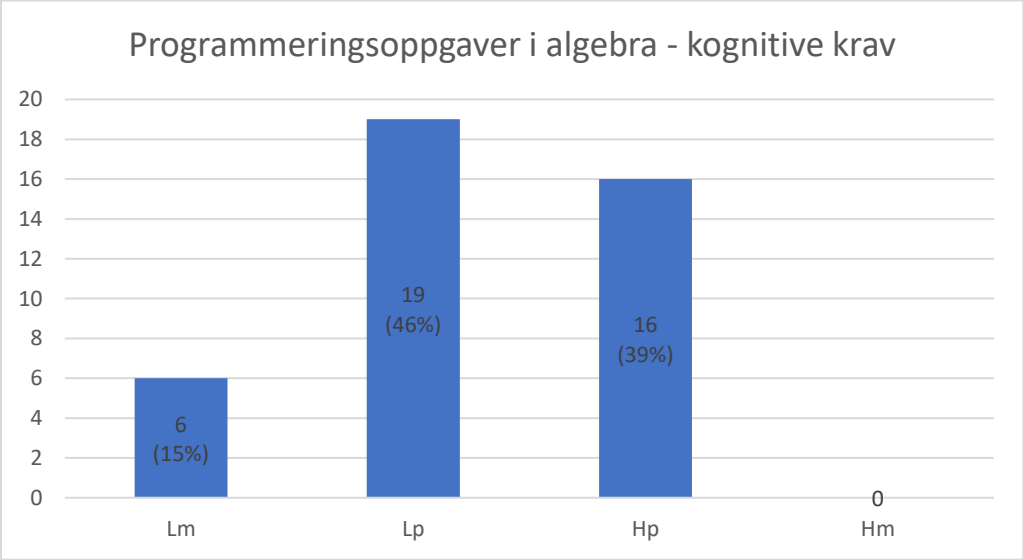
Oppgaver innenfor «Kopier og utvid/endre» er oppgaver der elevene blir gitt en kode, som de deretter skal forbedre ved å endre på programmet, eller utvide ved å tilegne programmet flere koder.

Det ble blant algebraoppgavene ikke identifisert noen oppgaver innenfor programmeringstypen «Feilsøk». Det ble funnet to feilsøkingsoppgaver blant programmeringsoppgavene, men disse var ikke algebraoppgaver. Disse oppgavene er knyttet til kategoriens navn, dette er oppgaver der elevene skal lete etter feil i koder og deretter forklare hva som er feil og/eller påpeke hvordan feilen kan fikses.

Fordelingen av programmeringstype blant algebraoppgavene viser at det er flest oppgaver innenfor programmeringstypen «Lag et program», og færre blant de andre. Dette funnet er ikke unikt for algebraoppgavene, for dette gjelder på tvers av programmeringsoppgavene i læreverkene (Vedlegg 2).

4.4.3 Funn i analysen av kognitive krav

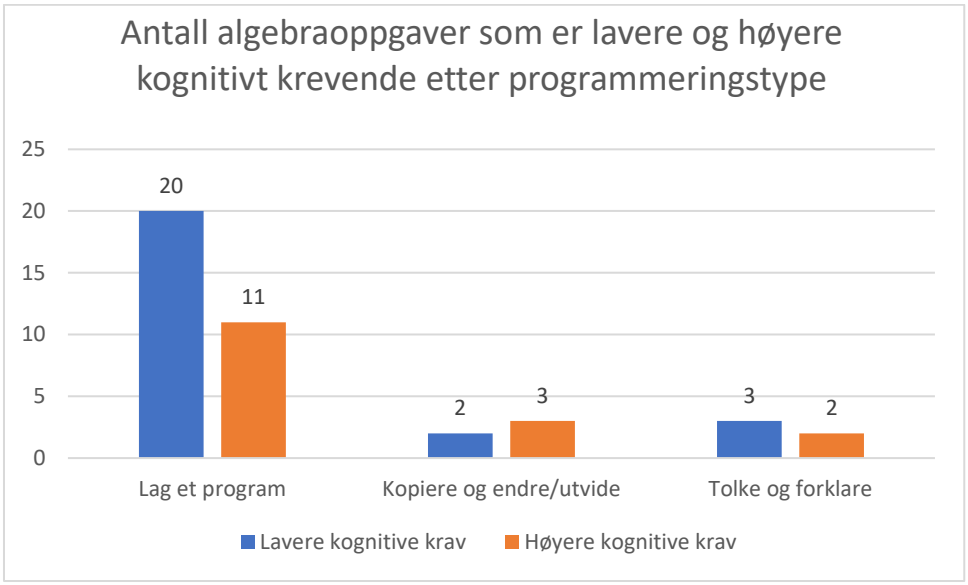
I analysen av kognitive krav i de 41 algebraoppgavene jeg fant innenfor programmering, er det flest (19) oppgaver innenfor «prosedyrer uten forbindelser», altså lavere kognitivt krevende oppgaver. Det er 16 oppgaver innenfor «prosedyrer med forbindelser», de høyere kognitivt krevende oppgavene, og 6 innenfor «memorering», som også er lavt kognitivt krevende oppgaver. Det er ingen oppgaver innenfor «å gjøre matematikk», som er det høyeste nivået av kognitive krav i rammeverket.



Figur 3 - Stolpediagram som viser oversikt over antall oppgaver innenfor de forskjellige kategoriene i rammeverket for kognitive krav til Smith & Stein.

I tillegg til å analysere kognitive krav i alle algebraoppgavene, har jeg deretter sammenlignet typen programmeringsoppgave opp mot hvilken kategori av kognitive krav oppgaver ble kodet til ved bruk av rammeverket til Smith og Stein (1998). Dette er for å se om typen

programmeringsoppgave har sammenheng med hvor kognitivt krevende oppgavene er å løse for elevene. Resultatene er fremstilt i figuren nedenfor.



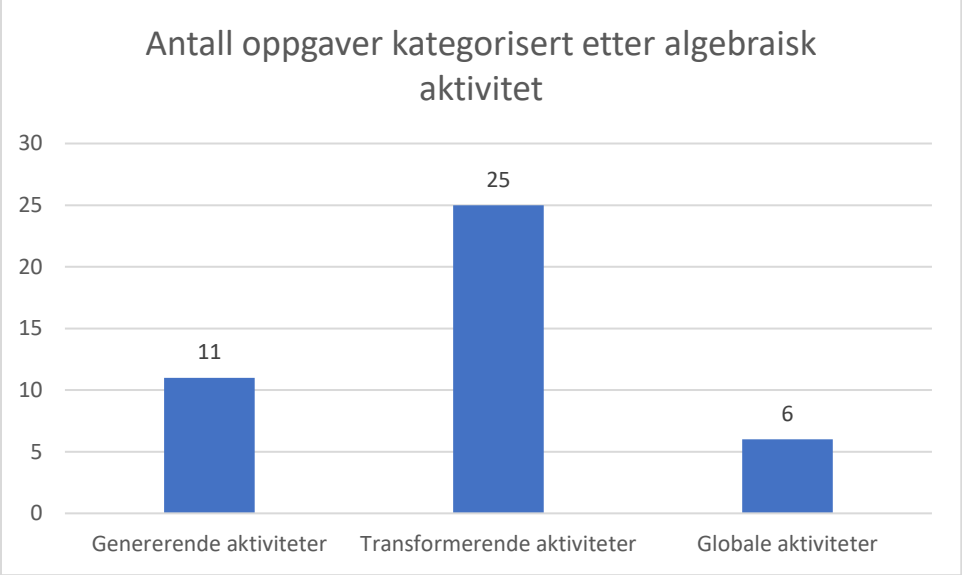
Figur 4 - Stolpediagram som viser sammenheng mellom type programmeringsoppgave og kognitive krav.

Av figuren og resultatene ser vi at oppgaver der elevene først skal kopiere en kode, og deretter utvide eller endre denne koden, potensielt kan stille høyere kognitive krav til elevene enn oppgaver av typen «lag et program». Det samme gjelder for programmeringstypen «tolke og forklare». Det er en større andel lavt kognitivt krevende oppgaver innenfor programmeringstypen «lag et program» hvis en ser på andel kognitivt krevende oppgaver sammenlignet med andelen oppgaver til sammen. Det er også flest oppgaver innenfor «lag et program» blant oppgavene i studien, og få blant de andre typene, som kan ha innvirkning på resultatene. Det var totalt sett 31 oppgaver der elevene skulle lage et program, og 11 av disse stilte høyere kognitive krav. Det var 5 oppgaver der elevene skulle kopiere og endre/utvide, der 3 av disse stilte høyere kognitive krav, og 5 oppgaver der elevene skulle tolke og/eller forklare, der 2 av disse stilte høyere kognitive krav. Ved flere oppgaver innenfor de to sistnevnte programmeringstypene, kunne en fått et bedre bilde av om programmeringstype har innvirkning på kognitive krav.

4.4.4 Funn i analysen av algebraiske aktiviteter

I analysen av ulike typer algebraiske aktiviteter blant oppgavene, identifiserte jeg at det klart er flere oppgaver innenfor den transformerende aktiviteten. Dette er noe Kieran (2004a) også har påpekt i sin forskning, lærebøker og skolealgebraen har ofte lagt mest vekt på de transformerende oppgavene. Det ble i alt identifisert 25 transformerende algebraiske aktiviteter, 11 genererende aktiviteter, og 6 globale aktiviteter. Blant analysen av algebraiske

aktiviteter er det totalt 42 oppgaver, men det er totalt bare analysert 41 algebraoppgaver. Det skyldes at den ene oppgaven har blitt kodet som både en genererende aktivitet og global aktivitet, ettersom at det i deloppgavene ble identifisert to forskjellige algebraiske aktiviteter.



Figur 5 - Stolpediagram som viser antall oppgaver kategorisert etter algebraisk aktivitet ved bruk av GTG-modellen til Kieran.

Blant de transformerende aktivitetene identifiserte jeg oppgaver der elevene ofte skal løse ligninger ved bruk av programmeringsspråk, det var flest slike oppgaver blant algebraoppgavene også.

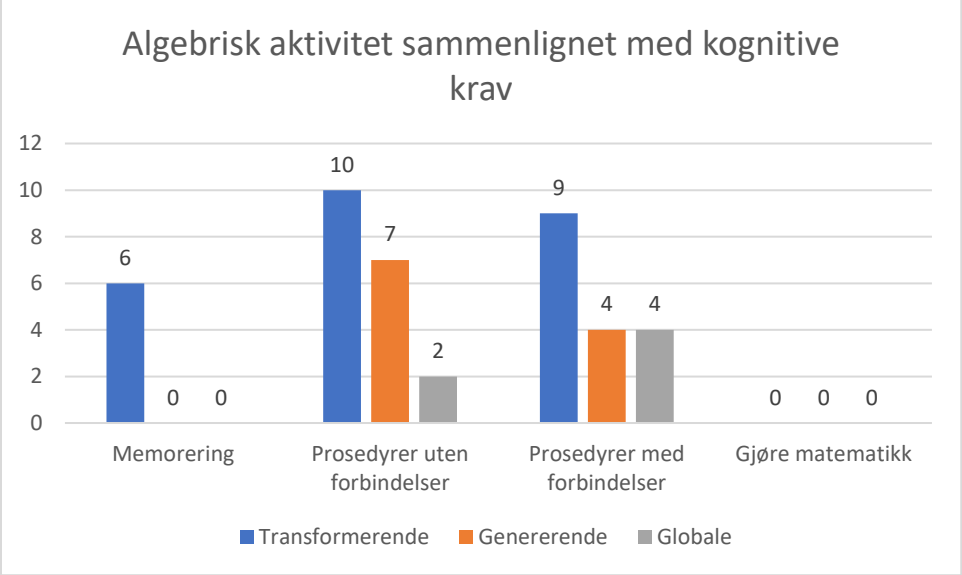
Blant de genererende aktivitetene identifiserte jeg blant annet oppgaver der elevene skal tolke tekstoppgaver, og representere det de tolker algebraisk ved hjelp av eksempelvis ligninger. Det var også mange oppgaver som omhandlet figurtall og geometriske mønstre, der elever blant annet skal finne antall kuler i et gitt figurnummer når de blir presentert et geometrisk mønster. Dette er typiske kjennetegn på genererende aktiviteter i henhold til Kierans (2004a) beskrivelser.

De globale aktivitetene kjennetegner oppgaver der elever bruker algebra som et verktøy, men algebra ikke er nødt til å brukes. Blant disse aktivitetene identifiserte jeg oppgaver der elever skal bevise matematiske sammenhenger ved bruk av algebra og oppgaver som kan løses ved bruk av resonnementer og ikke-algebraiske løsninger, men der algebra fungerer som støtte og er hensiktsmessig å bruke som et verktøy.

4.4.5 Algebraisk aktivitet og kognitive krav

For å kunne diskutere og si noe om sammenhengen mellom kognitive krav oppgavene stiller, og hva slags algebraiske aktiviteter oppgavene inneholder, har jeg identifisert hva slags

algebraiske aktiviteter som finnes innenfor de forskjellige nivåene av kognitive krav i oppgavene. Også her er oppgaveantallet 42, ettersom den ene algebraoppgaven som nevnt har blitt analysert som to algebraiske aktiviteter.



Figur 6 - Stolpediagram som viser sammenligning av algebraisk aktivitet og kognitive krav.

Her fremstår det av funnene som at det blant de høyt kognitivt krevende oppgavene er flest globale algebraiske aktiviteter. Vi ser også at det kun finnes transformerende algebraiske aktiviteter blant memoreringsoppgavene, som er den lavest kognitivt krevende kategorien. Blant prosedyrer uten forbindelser, finner vi flest genererende aktiviteter, det er også den kognitive kategorien med flest oppgaver. Derimot, ser vi at det blant alle algebraiske aktiviteter, finnes forskjellige nivåer av kognitive krav i oppgavene. Blant prosedyrer med og uten forbindelser, er også alle de algebraiske aktivitetene identifisert.

5. Diskusjon

I dette kapittelet vil jeg først trekke frem hovedfunnene fra studiens analyse av algebraiske aktiviteter, kognitive krav og programmeringstyper. Deretter vil jeg drøfte funnene knyttet til algebraisk aktivitet, etterfulgt av kognitive krav. Funnene vil også knyttes opp mot og drøftes med den tidligere forskningen som er presentert i oppgaven. Sammenheng mellom funn vil også diskuteres. Det diskuteres også rundt programmeringstypene som er funnet, og det diskuteres rundt sammenhenger mellom kognitive krav, algebraiske aktiviteter og programmeringstype.

5.1 Diskusjon av resultater

I denne studien har programmeringsoppgaver i to læreverk innenfor temaet algebra blitt analysert på tre forskjellige måter. Først har algebraoppgavene blitt kodet og identifisert etter matematikken i oppgavene, og deretter analysert igjen og kategorisert etter programmeringstype oppgaven setter elevene til å bruke. Deretter har de kognitive kravene i oppgavene blitt analysert og kategorisert, og til sist har de algebraiske aktivitetene som finnes i oppgavene blitt analysert og kategorisert. Det har i oppgaven blitt brukt tre analytiske rammeverk, ett for programmeringstype, ett for kognitive krav og ett for algebraisk aktivitet. Dette ble gjort for å få en helhetlig analyse av både det matematiske innholdet og programmeringskomponenten i oppgavene. Under analysen ble tematisk analyse brukt sammen med rammeverkene for programmeringstype, kognitive krav og algebraisk aktivitet.

Det analytiske rammeverket som baserer seg på programmeringstype, gjør det mulig å sammenligne resultatene med hva tidligere forskning har funnet om sammenheng mellom programmeringstype og matematikk. Rammeverket som ser på kognitive krav, bidrar til å si noe om det matematiske innholdet i oppgavene er høyere kognitivt krevende, eller om det er lavere kognitivt krevende eller prosedyrepreget. Det analytiske rammeverket som tar for seg algebraiske aktiviteter bidrar til å si noe om det matematiske algebrainnholdet i oppgavene, av hvilken type dette er, og om alle aktivitetene finnes blant oppgavene. Ved å bruke alle rammeverkene sammen, sier analysen noe om både programmeringskomponenten, det matematiske innholdets kvaliteter i henhold til kognitive krav og om det matematisk algebraiske innholdet i form av om alle algebraiske aktiviteter finnes i oppgavene.

For analysen av de kognitive kravene kom det frem at det var flest oppgaver som stiller lavere kognitive krav til elever. Selv om disse oppgavene også er viktige, er det i henhold til forskning ønskelig at det finnes flere oppgaver som stiller høyere kognitive krav enn lavere kognitive krav (Stein et al., 1996). Vi ser også blant funnene for algebraiske aktiviteter at det er en overvekt av transformerende aktiviteter, noe Kieran (2004a) også argumenterer for i sin forskning at det ofte finnes mest av i skolealgebraen. Dette kan tyde på at oppgavene er prosedyrepreget, som i mindre grad utvikler begrepsmessig forståelse og i større grad utvikler prosedyrekunnskap (Stein et al., 1996; Smith & Stein, 1998; Kieran 2004a; Kieran 2019). Det ble i analysen identifisert algebraoppgaver innenfor alle de tre algebraiske aktivitetene.

5.2 Algebraiske aktiviteter

Forskning på algebraiske aktiviteter fremhever viktigheten av å også inkludere generende og globale aktiviteter i skolealgebraen (Kieran 2004b; Wilson, Ainley & Bills, 2003), fordi

disse oppgavene er viktige for å utvikle algebraisk forståelse og matematisk tenkning, samtidig som at de bidrar til økt forståelse for de transformerende aktivitetene. Det påpekes derimot også at genererende og globale aktiviteter er vanskeligere for elever (Kieran 2004b; Wilson, Ainley & Bills, 2003). Dette kan være en grunn til at vi ser flere lavt kognitivt krevende oppgaver og transformerende aktiviteter i læreverk. Det er dermed ikke et overraskende funn, da det er vanlig at oppgaver i læreverk legges på et nivå som flere elever klarer å mestre.

Funnene fra analysen av algebraiske aktiviteter blant programmeringsoppgavene i algebra, viser at det er flest transformerende algebraiske aktiviteter blant oppgavene (25). Det var mindre genererende aktiviteter (11), og færrest globale aktiviteter (6). Dette ser vi også av Kierans (2004a) og Wilson, Ainley & Bills (2003) forskning, der de også kom frem til at det er de transformerende aktivitetene det ofte finnes flest av i lærebøker. Kieran (2004a, s.24) mener at når oppmerksomheten rettes mot transformerende aktiviteter og det å manipulere symbolske uttrykk og ligninger, rettes oppmerksomheten mot prosedyrekunnskap fremfor begrepsmessig forståelse. Kieran (2013) har samtidig kritisert skillet mellom begrepsmessig forståelse og prosedyrekunnskap, og ment at det har vært skadelig i algebra, fordi dette har ført til at algebra blir sett på som et prosedyredominert domene. Kieran forklarer kritikken sin med at det å lære algebra rent prosedyremessig, er negativt, og at det å utarbeide og endre uttrykk er konseptuelt av natur. Kieran (2004a) skriver også at de globale aktivitetene har blitt forsømt i skolealgebraen, selv om de er kritiske for å lykkes med algebra, da de bidrar til algebraisk forståelse og er viktige for å lykkes med transformerende og genererende aktiviteter. Dette er noe som også fremkommer av denne studiens analyse og resultater, det er nemlig færrest globale aktiviteter, og flest transformerende aktiviteter. De fleste oppgavene er også prosedyrepreget, noe som fremkommer av analysen av de kognitive kravene.

Kieran & Yreushalmy (2004) argumenterer for at det å jobbe med ligninger ved bruk av teknologiske verktøy kan bidra til begrepsmessig forståelse av algebraiske uttrykk, ligninger og ekvivalens. Det har i min studie blitt identifisert at de transformerende aktivitetene ofte dreier seg om nettopp dette, elevene skal lage programmer som løser ligninger. For at begrepsmessig forståelse skal oppstå sammen med teknikk når elever bruker digitale verktøy, skriver Kieran (2019, s.275) at det er avgjørende at de transformerende aktivitetene i algebra knyttes til de globale aktivitetene. Denne tilknytningen ble ikke identifisert i denne studien.

Det er i skolealgebraen nødvendig med både transformerende, genererende og globale algebraiske aktiviteter for at elever skal lære seg algebra. Dette argumenterer Kieran (2004a)

for i sin forskning. Selv om de globale aktivitetene regnes som mer givende for matematisk forståelse, så er det nødvendig med transformerende og genererende aktiviteter slik at elevene utvikler både prosedyrekunnskap og begrepsmessig forståelse, og møter på alle aspektene ved algebra (Kieran, 2004a).

5.2.1 Genererende aktiviteter

De genererende aktivitetene kjennetegnes ofte ved at elever skal formulere uttrykk og ligninger ut ifra noe de tolker, for eksempel tekstoppgaver eller figurtallmønster og geometriske mønstre (Kieran, 2004b). Elevene skal tolke noe, og deretter representere det algebraisk. Blant de genererende aktivitetene som ble identifisert i analysen, er det matematiske innholdet blant flertallet av oppgavene figurtallmønstre og problemsituasjoner i form av tekstoppgaver. Elevene skal lage programmer som kan finne gitte figurtall og figurnummer når et geometrisk figurtallmønster er gitt. Elevene skal også formulere algebraiske uttrykk ut fra problemsituasjoner i form av blant annet tekstoppgaver.

Kieran (2017) argumenterer for at struktur og strukturelle aktiviteter i algebra er viktig for den algebraiske tenkningen, og Radford (2015) argumenterer videre for at oppgaver med figurtallmønstre bidrar til dette, samtidig som han skriver at det finnes lite forskning på hvordan dataverktøy utnyttes i utviklingen av strukturell tenkning. De genererende aktivitetene i disse læreverkene baserer seg på figurtallmønstre, og kan være gode eksempler på et utgangspunkt for slike aktiviteter, nemlig fordi de i forskning karakteriseres som gode for elevers matematiske og algebraiske forståelse i form av strukturell tenkning (Radford, 2015; Kieran, 2017). De genererende aktivitetene som er analysert i denne studien består av både lavt og høyt kognitivt krevende oppgaver.

5.2.2 Transformerende aktiviteter

Det matematiske innholdet i flertallet av de transformerende aktivitetene i datamaterialet går ut på at elevene skal løse ligninger ved hjelp av programmering, og programmeringstypen er for det meste «lag et program». Det er også noen få oppgaver der elever skal tolke og forklare programkoder og kopiere og utvide/endre programmer. Dette samsvarer med det Kieran (2004b) skriver om transformerende aktiviteter, nemlig at de ofte baserer seg på det å endre uttrykk og ligninger for å bevare ekvivalens, så dette er i henhold til tidligere forskning ikke særegent for verken programmeringsoppgaver eller oppgavene i disse læreverkene.

Forskjellen mellom matematisk notasjon og programmering ved løsning av ligninger er at elevene ved tradisjonelle matematikkoppgaver bruker algebraisk notasjon, men ved programmeringsoppgaver brukes programmering. Dermed kan potensielt noe av det

matematiske innholdet forsvinne på veien, for eksempel det å sette opp ligninger og bruke «gjett og sjekk»-metoden eller «legge til og trekke fra» på begge sider. Dette gjør programmene for deg. Bråting og Kilhamn (2021a) påpeker også at det ved algebra i programmering finnes forskjeller fra matematisk notasjon som en må være obs på i en læringssammenheng. Kieran og Yerushalmy (2004) argumenterer derimot for at det å løse ligninger ved bruk av dataprogrammer, kan føre til at elever får begrepsmessig forståelse for ligninger og ekvivalens. Dermed kan det være hensiktsmessig at programmeringsoppgaver som er transformerende av aktivitet, baserer seg på det å løse ligninger for å bevare ekvivalens. Eksempler på slike oppgaver er når elevene skal lage programmer eller tolke programmer der en ligning blir løst eller settes opp slik at vi har like verdier på begge sider av likhetstegnet.

5.2.3 Globale aktiviteter

De globale aktivitetene er oppgaver som ofte består av problemløsning, modellering, generalisering og bevisføring (Kieran, 2004a). Globale aktiviteter kjennetegnes også ved at algebra brukes som et verktøy i oppgavene. Det har i analysen blitt identifisert 6 globale aktiviteter. Disse oppgavene går ut på at elevene skal lage programmer som løser matematiske problemer som også kan løses uten bruk av algebra. Det gis eksempler på disse oppgavene i analysen under kapittel 4.3.3. Det som karakteriserer de globale aktivitetene jeg har identifisert, er eksempelvis problemløsningsoppgaver i form av tekst der elever skal finne et ukjent tall eller uttrykke matematiske sammenhenger algebraisk. Algebra er ikke en nødvendighet for å løse disse oppgavene, men det er hensiktsmessig for å komme frem til en løsning eller bevise noe matematisk. Andre eksempler på globale aktiviteter er når elever skal bruke algebra for å bevise at det et program utfører matematisk, er korrekt. Programmeringstypene som er identifisert for disse oppgavene er «lag et program» og «tolke og forklare». De globale aktivitetene finnes både blant lavt og høyt kognitivt krevende oppgaver.

5.2.4 Algebraisk aktivitet og programmeringstype

Funnene rundt programmeringstyper i min studie er i tråd med funnene som Bråting og Kilhamn (2021b) har gjort på oppgaver i svenske lærebøker på barneskolenivå. De kom frem til at det var flest oppgaver der elevene skal følge instruksjoner eller prosedyrer, som i min studie har fått navnet «lag et program». Det er også i min studie identifisert flest oppgaver der elever skal lage programmer ved å følge prosedyrer (Bråting & Kilhamn, 2021b; Wu & Yang, 2022). Videre pekes feilsøkingsoppgaver i programmering av Bråting og Kilhamn (2021b) på

som noe av det fundamentale for å lære programmering og matematikk av programmering. Feilsøking ble også dratt frem som noe som bidrar til å lære matematikk i tidlig forskning på programmering (Papert, 1993). Det ble blant algebraoppgavene ikke identifisert noen feilsøkingsoppgaver. Videre peker Ye et al. (2023) på at det å reflektere over, undersøke og generalisere matematisk under programmering bidrar til matematisk forståelse. Det vil derfor i henhold til tidligere forskning være positivt for elevers læringsutbytte å designe flere slike oppgaver blant programmeringsoppgaver i algebra.

Programmeringstypene som ble identifisert blant de forskjellige aktivitetene, varierer, og det fremkommer ikke av analysen som at programmeringstypen er med på å påvirke hva slags algebraisk aktivitet oppgaver ligger innenfor. Dette er i veldig stor grad påvirket av det matematiske og algebraiske innholdet i oppgaven, og hvordan oppgavene er designet og formulert. Det ble blant de globale aktivitetene også identifisert forskjellige typer programmering, og dermed er det ikke en konkret programmeringstype som gjør oppgavene globale, men det matematiske innholdet i oppgavene. Det dukket i analysen også opp oppgaver som konkret viser at programmeringstype ikke har sammenheng med algebraisk aktivitet.

Et eksempel på en slik oppgave var en oppgave som ble kategorisert som programmeringstypen «tolke og forklare», men derimot også kategorisert som en transformerende aktivitet og en memoreringsoppgave i nivå av kognitive krav. Selv om generende algebraiske aktiviteter er karakterisert av at elever først skal tolke noe, var ikke denne oppgaven genererende selv om programmeringstypen var «tolke og forklare» (Bilde 21). Blant de algebraiske aktivitetene, har ikke programmeringstype hatt noe innvirkning på typen algebraisk aktivitet. Det er som sagt identifisert forskjellige programmeringstyper i alle de forskjellige algebraiske aktivitetene.

Variasjon i typer oppgaver er generelt sett på som viktig i matematikken, og Bråting og Kilhamn (2021b) peker på at en større variasjon i hva slags programmeringsoppgaver som finnes i lærebøkene vil bidra til å gjøre programmering rikere og danne en klarere sammenheng mellom programmering og matematikk. I min studie fremkom det at ingen av algebraoppgavene bestod av programmeringstypen feilsøking, og ettersom forskning på programmering i matematikk har argumentert for at feilsøking er fundamentalt for å lære matematikk ved programmering, er dette et uforventet funn. Det er også i tidligere forskning blitt identifisert få oppgaver der elever skal feilsøke (Bråting & Kilhamn, 2021b; Wu & Yang, 2022).

Programmeringsoppgavene kan ut ifra algebraiske aktiviteter som er identifisert, se ut til å virke som et supplement til andre typer algebraoppgaver i læreverkene. Samtidig ser det ut til at flertallet av oppgavene er prosedyrepreget i lys av de algebraiske aktivitetene og kognitive kravene som er identifisert. De fleste oppgavene er på tvers av alle algebraiske aktiviteter, av programmeringstypen «lag et program», og dermed har ikke typen programmering hatt en like stor påvirkning på kategoriseringen som det matematiske og algebraiske innholdet har hatt.

5.3 Kognitive krav

Vi ser av funnene i analysen der algebraisk aktivitet ses i sammenheng med nivå av kognitive krav blant oppgavene (Figur 6), at det under memoreringsoppgaver kun er funnet transformerende aktiviteter. Det kan tyde på at det er vanskelig å integrere genererende og globale aktiviteter blant de lavest kognitivt krevende oppgavene. Blant oppgavene under prosedyrer med og uten forbindelser, er alle algebraiske aktiviteter identifisert. Det ble funnet fire og flest globale aktiviteter blant prosedyrer med forbindelser, det kan tyde på at når den algebraiske aktiviteten er global, stiller oppgaven høyere kognitive krav. Det er derimot små tall, noe som gjør at det også kan være tilfeldigheter. Det ble ikke funnet noen oppgaver av typen kognitive krav «gjøre matematikk», som er det høyeste nivået av kognitive krav, det kan dermed være interessant å undersøke om den typen kognitive krav kunne blitt koblet til en bestemt algebraisk aktivitet.

Av resultatene ser vi at cirka 15 % av oppgavene er under kategorien memorering, cirka 46 % under prosedyrer uten forbindelser, og 39 % under prosedyrer med forbindelser. Dette betyr at 61% av oppgavene som er analysert i studien stiller lave kognitive krav til elever, og 39 % stiller høye kognitive krav til elever. Det er i henhold til Stein et al. (1996) og Smith og Stein (1998) mer hensiktsmessig med en høyere andel høyt kognitivt krevende oppgaver, da en overvekt av lavere kognitivt krevende oppgaver kan ha negativ innvirkning på den matematiske forståelsen og læringsutbyttet elever potensielt får av oppgaver. Læringsutbyttet retter seg mer mot prosedyrekunnskap enn begrepsmessig forståelse i matematikk. Det skal poengteres at dette gjelder for programmeringsoppgavene, elevene møter også på andre typer oppgaver innenfor algebra i læreverkene som ikke har blitt analysert.

5.3.1 Kognitive krav og algebraoppgaver

Det er ikke gjort noen forskning på kognitive krav i programmeringsoppgaver tidligere, men det er blitt gjort på algebraoppgaver generelt, for eksempel av Ubuz et al. (2010) og Yang og Sianturi (2022). Det ble identifisert flere høyt kognitivt krevende oppgaver i studien til Ubuz et al. (2010), der ble det også identifisert flere oppgaver innenfor «gjøre matematikk».

Funnene i min studie er mer i tråd med funnene fra algebraoppgavene i studien til Yang og Sianturi.

For detaljer knyttet til Ubuz et al. (2010) og Yang og Sianturi (2022) sine funn og tall fra deres studier, se delkapittel 2.1.3. Resultatene i studien til Ubuz et al. (2010) viser sammenlignet med mine resultater, forskjeller. Sammenligner jeg derimot resultatene mine med Yang og Siantura (2022) sine funn, er det litt flere likheter basert på prosentvis antall oppgaver innenfor hver kognitive kategori. Denne studien er derimot gjort på barneskolen, der elevene er mindre kjent med algebra, noe som kan føre til at oppgavene stiller lavere kognitive krav enn på ungdomstrinnene. Disse resultatene kan tyde på at det kan være vanskelig å integrere et matematisk innhold som er høyere kognitivt krevende i programmeringsoppgaver, spesielt av typen gjøre matematikk. Dette ser vi av resultatene i studiene av algebraoppgaver som skal løses ved bruk av matematisk notasjon (Ubuz et al., 2010), der er det nemlig en større andel høyere kognitivt krevende oppgaver, og at mine resultater for programmeringsoppgaver på ungdomstrinnet samsvarer med resultatene og funnene til Yang og Siantura (2022) for matematikkoppgaver i algebra fra barneskolen.

5.3.2 Hva gjør oppgavene kognitivt krevende?

Eksempler på høyt kognitivt krevende oppgaver er at det ikke gis noen konkret fremgangsmåte for hvordan en skal komme frem til en løsning. Dermed rettes elevers oppmerksomhet mot å tenke på bruken av prosedyrer og fremgangsmåter, der hensikten er å utvikle forståelse for disse ved å klare å benytte disse i nye problemsituasjoner og komme frem til et svar ved hjelp av dem. Prosedyrer elever har lært tidligere kan brukes, men elevene må tenke over hva slags prosedyrer de skal bruke og hvordan de skal brukes. Når matematikken i oppgaven endres, og det kognitive nivået som kreves blir høyere, fører dette også til at måten elevene må programmere på, endres. Det er altså i de analyserte oppgavene i denne studien det matematiske innholdet som gjør oppgavene kognitivt krevende, og ikke programmeringstypen. Elevene kan i oppgavene som er kategorisert som høyt kognitivt krevende, der fremgangsmåter ikke eksplisitt gis, anvende tidligere lært matematikk og prosedyrer, hvis elevene har forstått matematikken eller prosedyrene.

Det ble i Ubuz et al. (2010) sin studie identifisert oppgaver av typen «gjøre matematikk», og jeg vil nå diskutere fraværet av «gjøre matematikk»-oppgaver i min studie nærmere.

5.3.3 Programmering er algoritmisk

Det er ingen overraskelse at det ikke er noen programmeringsoppgaver i min studie som klassifiseres som «gjøre matematikk». Det er heller ingen overraskelse at det er funnet flest

programmeringsoppgaver blant de prosedyrerettede kognitive kategoriene og algebraiske aktivitetene. Programmering er per definisjon algoritmisk, og handler om å gjennomføre prosedyrer. Smith og Steins (1998) karakteristikker for «gjøre matematikk»-oppgaver er blant annet at oppgavene krever ikke-algoritmisk tenkning og at elevene skal utforske. Det kan være vanskelig å inkorporere eksempelvis ikke-algoritmiske oppgaver der elevene skal bevise og utforske matematikk i programmeringsoppgaver, spesielt når programmering er algoritmisk. Et av formålene med programmering i matematikk er nemlig at elevene skal utvikle algoritmisk tenkning (Bråting & Kilhamn, 2021a), det fremkommer både av forskning og læreplanen (Kunnskapsdepartementet, 2019). Slike funn kom også Bråting og Kilhamn (2021b) frem til, flertallet av oppgavene de analyserte innenfor programmering i studien sin, baserte seg på at elevene skulle følge prosedyrer eller fremgangsmåter. Dette kan være en indikator på at visse typer oppgaver er vanskeligere å integrere når matematikk blandes med programmering. Spesielt ser det ut som at disse utforskende og ikke prosedyrerettede eller ikke-algoritmiske oppgavene som kategoriseres som «gjøre matematikk», er vanskelige å integrere.

Selv om det ikke ble identifisert noen oppgaver innenfor kategorien «gjøre matematikk», ble det faktisk blant algebraoppgavene i studien identifisert problemløsningsoppgaver der elevene skal sette sammen prosedyrer. Kieran (2013) påpekte at det å endre prosedyrer eller sette sammen prosedyrer, er konseptuelt av natur, og at dette bidrar til begrepsmessig matematisk forståelse (Kieran, 2013). Hun argumenterer også for at det å lære algebra rent prosedyremessig er negativt, og viser til at det finnes både begrepsmessige og prosedyremessige komponenter innenfor algebra, der blant annet problemløsning er en viktig komponent (Kieran, 2013). Dette resultatet gjør det også spennende å undersøke om det er mulig å lage programmeringsoppgaver i algebra som kan komme inn under kategorien «gjøre matematikk» i Smith og Steins (1998) rammeverk for kognitive krav. Det er mulig at det kanskje er nødvendig med et nytt eller et oppdatert rammeverk for kognitive krav i programmeringsoppgaver i matematikkfaget. Resultatene for algebraoppgavene i programmering i denne studien, og at programmering er algoritmisk, antyder at det potensielt kan være nødvendig.

5.3.4 Programmeringstype og kognitive krav

For å si noe om hvordan programmering og programmeringstypen kan påvirke oppgavenes kognitive krav, uten å ha observert og analysert elevers løsning av oppgaver ved programmering, undersøkte jeg hvor mange lavt og høyt kognitivt krevende oppgaver det var

knyttet til type programmering (Figur 4**)**. Vi ser av programmeringstypen «lag et program» at det er en større andel lavere kognitivt krevende enn høyt kognitivt krevende oppgaver. Dette kan blant annet komme av at fremgangsmåten «lag et program» i matematikk kan anses som regnetekniske og prosedyrerettede oppgaver der elevene skal følge fremgangsmåter gitt av oppgaver. Ved en overvekt av disse, får elevene mengdetrening og prosedyrekunnskap som utbytte. Elevene skal nemlig følge en fremgangsmåte for å komme frem til et svar, uten å vise til en forklaring eller beskrive hvordan en har kommet frem til en løsning eller hvordan prosedyrene er brukt. Dette er noe Smith og Stein (1998) også mener kjennetegner lavere kognitivt krevende oppgaver, og kan være en grunn til at oppgavene innenfor denne programmeringstypen stiller lavere kognitive krav.

Der vi finner flest oppgaver som stiller høyere kognitive krav, knyttet til programmeringstype, er ved programmeringstypen «tolke og forklare». Ved disse oppgavene skal elevene enten tolke programkoder, forklare programkoder eller forklare hvordan de har løst oppgaver og vise at de dermed forstår matematikken og programmering. I og med at Smith og Stein (1998) mener at lavere kognitivt krevende oppgaver ikke krever forklaringer, kan det tyde på at oppgaver der elever faktisk må forklare hva som er blitt gjort, og fokuset i oppgaven ligger på prosedyren eller fremgangsmåten, og ikke løsningen, stiller høyere kognitive krav. Det er få oppgaver blant noen av programmeringstypene, som gjør at dette er en begrensning som kan gi tilfeldigheter i resultatene der kognitive krav sammenlignes med programmeringstype.

Det som gjør oppgavene høyt kognitivt krevende, er det matematiske innholdet. Dette kan være i form av blant annet oppgaveformuleringen, hva slags fremgangsmåter som skal brukes og hvordan fremgangsmåtene gis av oppgaven, samt hvorvidt oppgaver er like eller ulike tidligere gitte oppgaver eller eksempler elevene presenteres for i læreverket. Andre ting som gjør oppgavene kognitivt krevende er om elevene bare skal komme frem til en løsning, eller om de skal forklare hvordan de kom frem til denne løsningen, om de skal generalisere og om de skal bevise matematisk. Det er i analysen gitt to eksempler på høyt kognitivt krevende oppgaver, og det som er felles for de høyt kognitivt krevende oppgavene i analysen og i datamaterialet ellers, er at det er matematikken som er på et høyere kognitivt krevende nivå. Det er altså ikke programmeringen eller programmeringstypen som avgjør om de kognitive kravene er høye, men matematikkinnholdet i oppgavene, oppgavenes plassering og hvor like oppgaver elever skal løse er tidligere presenterte oppgaver.

6. Avslutning og konklusjon

I denne studien har jeg foretatt en tematisk analyse av programmeringsoppgavene i algebra i læreverkene Matemagisk 8-10 og Campus Inkrement. Jeg undersøkte hva som karakteriserer det matematiske innholdet som finnes blant programmeringsoppgaver i algebra i læreverkene i form av algebraiske aktiviteter og kognitive krav. Oppgavene har blitt analysert fire ganger, for å finne ut hva slags kognitive krav oppgavene stiller, hvilke typer programmering og matematisk tema som finnes i oppgavene, og hvilke algebraiske aktiviteter som finnes blant oppgavene. Ettersom det kun var læreverkene og oppgavene som ble analysert, har jeg ikke noe grunnlag for å si noe om hvordan læreverkene eller oppgavenes presenteres for elever i praksis eller brukes i undervisning. Jeg kan heller ikke si noe om hva slags potensiale oppgavene har til å legge opp for matematisk forståelse eller læring hos elever i praksis, da dette i klasserommet også avhenger av mange andre faktorer enn selve oppgavene. Læreverkene har også blitt analysert etter bestemte rammeverk, definisjoner og med bakgrunn i tidligere forskning og teori.

Problemstillingen for studien var:

«Hva karakteriserer det matematiske innholdet i programmeringsoppgaver innenfor algebra i læreverk for ungdomstrinnene»

Denne ble belyst gjennom disse tre følgende forskningsspørsmålene:

- Hvilke kognitive krav stiller algebraiske programmeringsoppgaver i matematikk i to læreverk for ungdomstrinnene til elever?
- Hvilke algebraiske aktiviteter finnes blant programmeringsoppgavene i algebra i disse to læreverkene for ungdomstrinnene?
- Hvilken sammenheng er det mellom det matematiske innholdet og programmeringstypene i programmeringsoppgavene i algebra?

6.1 Konklusjon

I studien har jeg identifisert at det matematiske innholdet blant programmeringsoppgavene bærer preg av lavere kognitive krav, transformerende algebraiske aktiviteter og av en type programmering der elevene skal lage programmer etter gitte instrukser. Blant algebraoppgavene, ble alle algebraiske aktiviteter ble identifisert, samt tre programmeringstyper og tre nivåer av kognitive krav. Av studien fremkommer det at oppgavene i programmering i algebra er prosedyrerettede, noe som sannsynligvis kommer av at programmering og programmeringsspråk er algoritmisk. Dette kan tilsies å være oppgaver

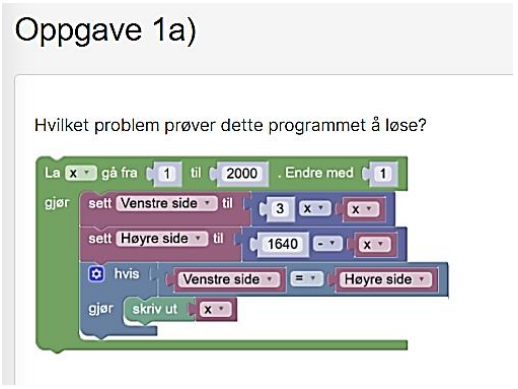
som gir mengdetrening og prosedyrekunnskap. Blant oppgavene, ser det ut til at det er det matematiske innholdet som setter premissene for om en oppgave er kognitivt krevende eller av en type algebraisk aktivitet, og ikke programmeringstypen. Det fremkommer også at det kan være vanskelig å integrere mange komponenter inn i en programmeringsoppgave. Det vil si at hvis en skal ha høyt kognitivt krevende matematikk, globale algebraiske aktiviteter og hensiktsmessige programmeringstyper i en og samme oppgave, kan oppgaven bli for omfattende eller for tidskrevende. Funnene kan illustrere at programmering potensielt kan betraktes som et supplement til andre læringsmetoder og oppgaver i algebra i skolematematikken. Resultatene i studien er ikke generaliserbare til andre læreverk, og for øvrig ikke til andre oppgaver i disse læreverkene. Studien er et bidrag til et forskningsfelt som er relativt nytt, fordi programmering er en nokså ny del av skolematematikken.

6.2 Videre forskning

For å videre undersøke hvordan programmeringsoppgaver kan designes, og hva slags matematisk innhold som finnes blant programmeringsoppgaver og hva som karakteriserer dette innholdet, er det mulig å forske på programmeringsoppgaver i andre læreverk og innenfor andre temaer av matematikken. Videre studier kan også sammenligne tradisjonelle oppgaver i algebra og algebraoppgaver innenfor programmering, for å se hva slags eventuelle likheter og forskjeller som finnes. Det er også mulig å forske på hvordan vi kan få en større bredde i programmeringsoppgaver, både basert på matematisk innhold og type programmering. Det er også mulig å undersøke hva slags fordeler eller ulemper en finner ved bruk av programmering knyttet til spesifikke deler av algebra. En annen mulighet er å undersøke hvordan en kan implementere oppgaver av typen «gjøre matematikk», og flere globale aktiviteter blant programmeringsoppgavene i algebra. Det er også mulig å undersøke om oppgavetypen «gjøre matematikk» kan knyttes til en spesifikk algebraisk aktivitet.

6.2.1 Forslag til feilsøkingsoppgaver

På bakgrunn av at det ikke ble identifisert noen feilsøkingsoppgaver blant algebraoppgavene i læreverkene, ønsker jeg å vise til et forslag for hvordan disse oppgavene kan designes. Noen programmeringsoppgaver er designet slik at programmeringskodene er gitt, og elevene skal deretter tolke denne koden. Et eksempel på en slik oppgave er oppgave 1a. For å implementere feilsøking, slik som teorien argumenter for at er fruktbart for matematisk forståelse ved programmering



Bilde 31 - Oppgave fra Campus Inkrement.

(Bråting & Kilhamn, 2021b; Ye et al., 2023), kunne oppgavene vært designet på samme måte som oppgave 1a. Hvis elevene for eksempel hadde blitt presentert for et program som løser en ligning, men det i koden var innlagt feil, kunne oppgaven vært «finn feilen som gjør at programmet ikke virker eller finner en løsning». Elevene kunne da selv gått inn i koden for å lete etter feil, og prøvd å rette denne feilen slik at programmet virker igjen eller klarer å løse ligningen. Denne oppgaven ville hørt inn under den algebraiske aktiviteten transformerende aktiviteter. Det er også mulighet for å utvide oppgaven slik at elevene deretter kunne forklart hva som var feil og hvordan de rettet på feilen. Elevene vil da i en og samme oppgave tolke en programkode, feilsøke koden, og forklare hva de har gjort. Det ville også vært interessant å forske videre på om det er andre typer feilsøkingsoppgaver som kan knyttes til de andre algebraiske aktivitetene. Slike utvidelser kan være en måte å gjøre det på.

Referanseliste

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

Aunivers. (2021, 1. Juni). *Matematikk.* Aunivers. https://aunivers.no/marked/ungdomsskole/laeremidler-8. -10/matematikk

Balanskat, A., & Engelhardt, K. (2015). Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across Europe. Brussel: European Schoolnet. http://www.eun.org/documents/411753/817341/Computing+our+future_final_2015.pdf/d3780a64-1081-4488-8549-6033200e3c03

Blikstein, P. (2018). Pre-College Computer Science Education: A Survey of the Field. Mountain View, CA: Google LLC. https://goo.gl/gmS1Vm

Bowen, G. A. (2009). Document Analysis as a Qualitative Research Method. *Qualitative research journal*, *9*(2), 27-40. https://doi.org/10.3316/QRJ0902027

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3*(2), 77-101. https://doi.org/10.1191/1478088706qp063oa

Bråting, K. & Kilhamn, C. (2021a). Exploring the intersection of algebraic and computational thinking. *Mathematical thinking and learning*, *23*(2), 170-185. https://doi.org/10.1080/10986065.2020.1779012

Bråting, K. & Kilhamn, C. (2021b). 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

Chan, S.-W., Looi, C.-K., Ho, W. K. & Kim, M. S. (2023). Tools and Approaches for Integrating Computational Thinking and Mathematics: A Scoping Review of Current Empirical Studies. *Journal of educational computing research*, *60*(8), 2036-2080. https://doi.org/10.1177/07356331221098793

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

Cui, Z. & Ng, O.-L. (2021). The Interplay Between Mathematical and Computational Thinking in Primary School Students' Mathematical Problem-Solving Within a Programming Environment. *Journal of educational computing research*, *59*(5), 988-1012. https://doi.org/10.1177/0735633120979930

Denning, P. J. (2017). Viewpoint: Remaining Trouble Spots with Computational Thinking. *Communications of the ACM*, *60*(6), 33.

Fan, L., Zhu, Y. & Miao, Z. (2013). Textbook research in mathematics education: development status and directions. *ZDM*, *45*(5), 633-646. https://doi.org/10.1007/s11858-013-0539-x

Forsström, S. E. & Kaufmann, O. T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18-32. https://hiof.brage.unit.no/hiof-xmlui/handle/11250/2599710

Gilje, Ø., Ingulfsen, L., Dolonen, J. A., Furberg, A., Rasmussen, I., Kluge, A., Knain, E., Mørch, A., Naaslund, M. & Skarpaas, K. G. (2016). *Med ARK&APP: Bruk av læremidler og ressurser for læring på tvers av arbeidsformer.* Universitetet i Oslo. https://www.uv.uio.no/iped/forskning/prosjekter/ark-app/arkapp_syntese_endelig_til_trykk.pdf

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

Hiebert, J., & Lefevre, P. (1986). Conceptual and procedural knowledge in mathematics: An introductory analysis. I J. Hiebert (Ed.), *Conceptual and procedural knowledge: The case of mathematics* (pp. 1–27). Lawrence Erlbaum Associates, Inc.

Kieran, C. (2004a). The Core of Algebra: Reflections on its Main Activities. I Stacey, K., Chick, H., Kendal, M. (eds) *The Future of the Teaching and Learning of Algebra The 12th ICMI Study.* New ICMI Study Series, vol 8. Springer, Dordrecht. https://doi.org/10.1007/1-4020-8131-6_2

Kieran, C. (2004b). Algebraic thinking in the early grades: what is it? *The mathematics Educator, 8(1), 139-151*. https://www.researchgate.net/publication/228526202_Algebraic_thinking_in_the_early_grades_What_is_it

Kieran, C., Yerushalmy, M. (2004). Research on the Role of Technological Environments in Algebra Learning and Teaching. I Stacey, K., Chick, H., Kendal, M. (eds) *The Future of the Teaching and Learning of Algebra The 12th ICMI Study*. New ICMI Study Series, vol 8. Springer, Dordrecht. https://doi.org/10.1007/1-4020-8131-6_6

Kieran, C. (2013). The False Dichotomy in Mathematics Education Between Conceptual Understanding and Procedural Skills: An Example from Algebra. I (s. 153-171). New York, NY: Springer New York. https://doi.org/10.1007/978-1-4614-6977-3_7

Kieran, C. (2017). *Teaching and Learning Algebraic Thinking with 5- To 12-Year-Olds: The Global Evolution of an Emerging Field of Research and Practice*. Cham: Springer International Publishing AG. https://doi.org/10.1007/978-3-319-68351-5

Kieran, C. (2018). *Teaching and Learning Algebraic Thinking with 5- to 12-Year-Olds : The Global Evolution of an Emerging Field of Research and Practice* (1st 2018. utg.). Springer International Publishing : Imprint: Springer.

Kieran, C. (2019). Task Design Frameworks in Mathematics Education Research: An Example of a Domain-Specific Frame for Algebra Learning with Technological Tools. I Kaiser, G., Presmeg, N. (eds) *Compendium for Early Career Researchers in Mathematics Education* . ICME-13 Monographs. Springer, Cham. https://doi.org/10.1007/978-3-030-15636-7_12

Kongsnes, A. L., Wallace, A. K. (2020). *Matemagisk 8.* Aschehoug undervisning.

Kongsnes, A. L., Wallace, A. K. (2020). *Matemagisk 9.* Aschehoug undervisning.

Kongsnes, A. L., Wallace, A. K. (2021). *Matemagisk 10.* Aschehoug undervisning.

Kunnskapsdepartementet. (2019). *Læreplan i matematikk 1.-10. trinn (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05?lang=nob

Mason, J. (2018). How Early Is Too Early for Thinking Algebraically?. I Kieran, C. (eds) *Teaching and Learning Algebraic Thinking with 5- to 12-Year-Olds.* ICME-13 Monographs. Springer, Cham. https://doi.org/10.1007/978-3-319-68351-5_14

McCormick, R. (1997). Conceptual and Procedural Knowledge. *International journal of technology and design education, 7*(1-2), 141-159. https://doi.org/10.1023/A:1008819912213

NOU 2013: 2. (2013). *Hindre for digital verdiskaping.* Fornyings-, administrasjons- og kirkedepartementet. https://www.regjeringen.no/contentassets/e2f0d5676e144305967f21011b715c16/no/pdfs/nou201320130002000dddpdfs.pdf

NOU 2015: 8. (2015). *Fremtidens skole: Fornyelse av fag og kompetanser*. Kunnskapsdepartementet. https://www.regjeringen.no/contentassets/da148fec8c4a4ab88daa8b677a700292/no/pdfs/nou201520150008000dddpdfs.pdf

Opplæringslova. (1998). *Lov om grunnskolen og den vidaregåande opplæringa (opplæringslova)* (§ 1-1). Lovdata. https://lovdata.no/dokument/NL/lov/1998-07-17-61

Papert, S. (1993). *Mindstorms : children, computers, and powerful ideas* (2. utg.). Basic Books.

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

Radford, L. (2015). Early Algebraic Thinking: Epistemological, Semiotic, and Developmental Issues. I Cho, S. (eds) *The Proceedings of the 12th International Congress on Mathematical Education.* Springer, Cham. https://doi.org/10.1007/978-3-319-12688-3_15

Radford, L. (2017). The Emergence of Symbolic Algebraic Thinking in Primary School. I C. Kieran (Red.), *Teaching and Learning Algebraic Thinking with 5- to 12-Year-Olds* (s.3-26). Springer International Publishing AG.

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., Mørken, K. M., Svorkmo, A., Voll, L. O. (2016). Teknologi og programmering for alle: *En faggjennomgang med forslag til endringer i grunnopplæringen*. Utdanningsdirektoratet.

https://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf

Sevik, K. (2016). Programmering i skolen. https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Smith, M. S. & Stein, M. K. (1998). REFLECTIONS on Practice: Selecting and Creating mathematical Tasks: From Research to Practice. *Mathematics teaching in the middle school, 3*(5), 344-350. https://doi.org/10.5951/MTMS.3.5.0344

Stein, M. K., Grover, B. W. & Henningsen, M. (1996). Building Student Capacity for Mathematical Thinking and Reasoning: An Analysis of Mathematical Tasks Used in Reform Classrooms. *American educational research journal, 33*(2), 455-488. https://doi.org/10.3102/00028312033002455

Stein, M. K., Smith, M. S., Henningsen, M. A. & Silver, E. A. (2000). *Implementing Standards-based Mathematics Instruction: A Casebook for Professional Development Ways of Knowing in Science Series*. Teachers College Press.

Stein, M. K. (2009). Implementing standards-based mathematics instruction: a casebook for professional development. 2. utg. Reston, Va.: New York, National Council of Teachers of Mathematics

Ubuz, B., Erbaş, A. K., Çetinkaya, B. & Özgeldi, M. (2010). Exploring the quality of the mathematical tasks in the new Turkish elementary school mathematics curriculum guidebook: the case of algebra. *ZDM, 42*(5), 483-491. https://doi.org/10.1007/s11858-010-0258-5

Utdanningsdirektoratet (2019, 27. Mars). Algoritmisk tenkning. Udir. https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet (2019, 18. November). Hva er kjerneelementer?. Udir. https://www.udir.no/laring-og-trivsel/lareplanverket/stotte/hva-er-kjerneelementer/

Wilson, K., Ainley, J., & Bills, L. (2003). Comparing Competence in Transformational and Generational Algebraic Activities. *International Group for the Psychology of Mathematics Education, 4, 427-434.* https://files.eric.ed.gov/fulltext/ED501158. pdf

Wing, J. M. (2008). Computational thinking and thinking about computing. *PHIL TRANS R SOC A*, *366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008. 0118

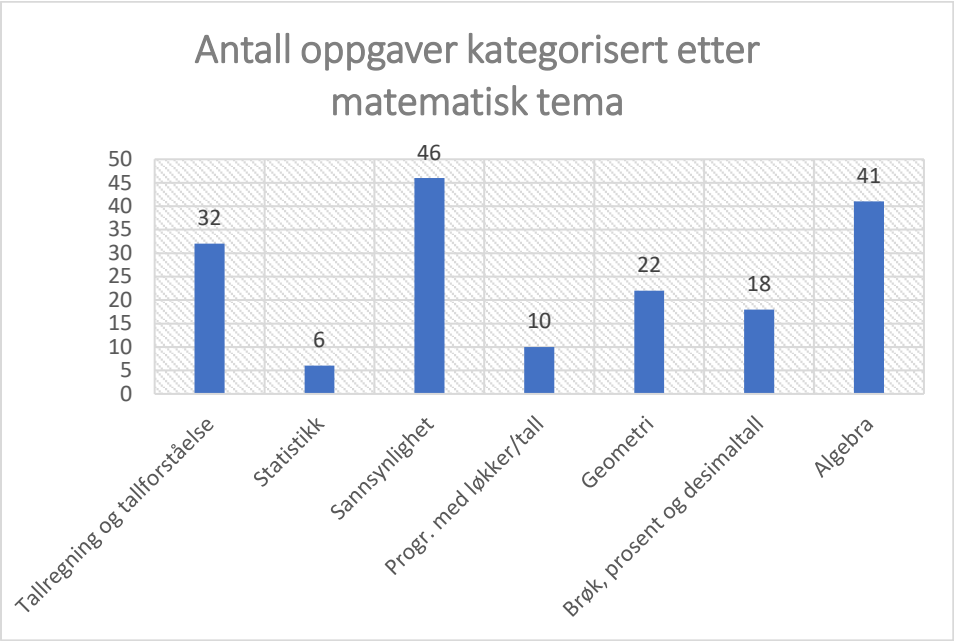
Wu, W.-R. & Yang, K.-L. (2022). The relationships between computational and mathematical thinking: A review study on tasks. *Cogent education*, *9*(1). https://doi.org/10.1080/2331186X.2022.2098929

Yang, D.-C. & Sianturi, I. A. J. (2022). Analysis of algebraic problems intended for elementary graders in Finland, Indonesia, Malaysia, Singapore, and Taiwan. *Educational studies*, *48*(1), 75-97. https://doi.org/10.1080/03055698. 2020.1740977

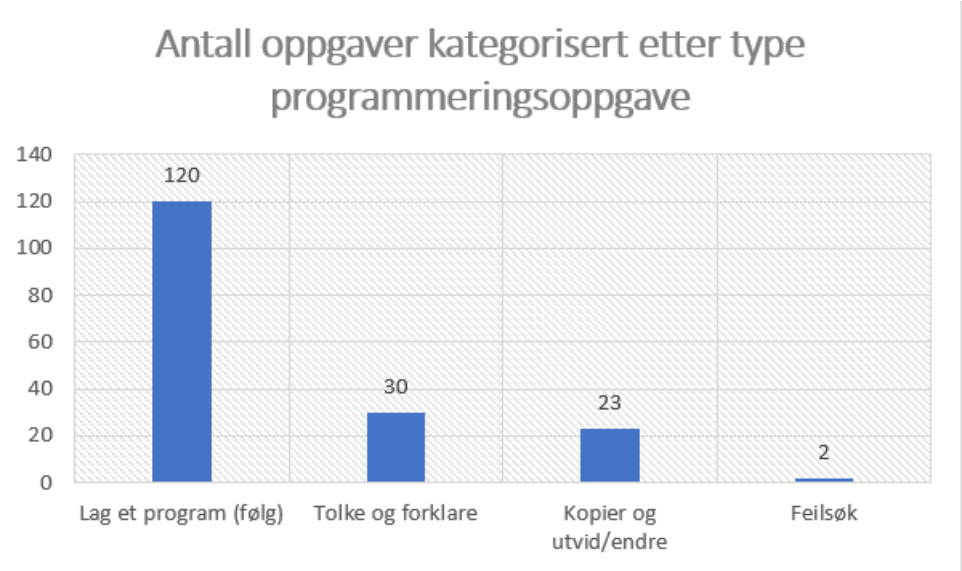
Ye, H., Liang, B., Ng, O.-L. & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: a systematic review on CT-based mathematics instruction and student learning. *International journal of STEM education*, *10*(1), 3-26. https://doi.org/10.1186/s40594-023-00396-w

Vedlegg

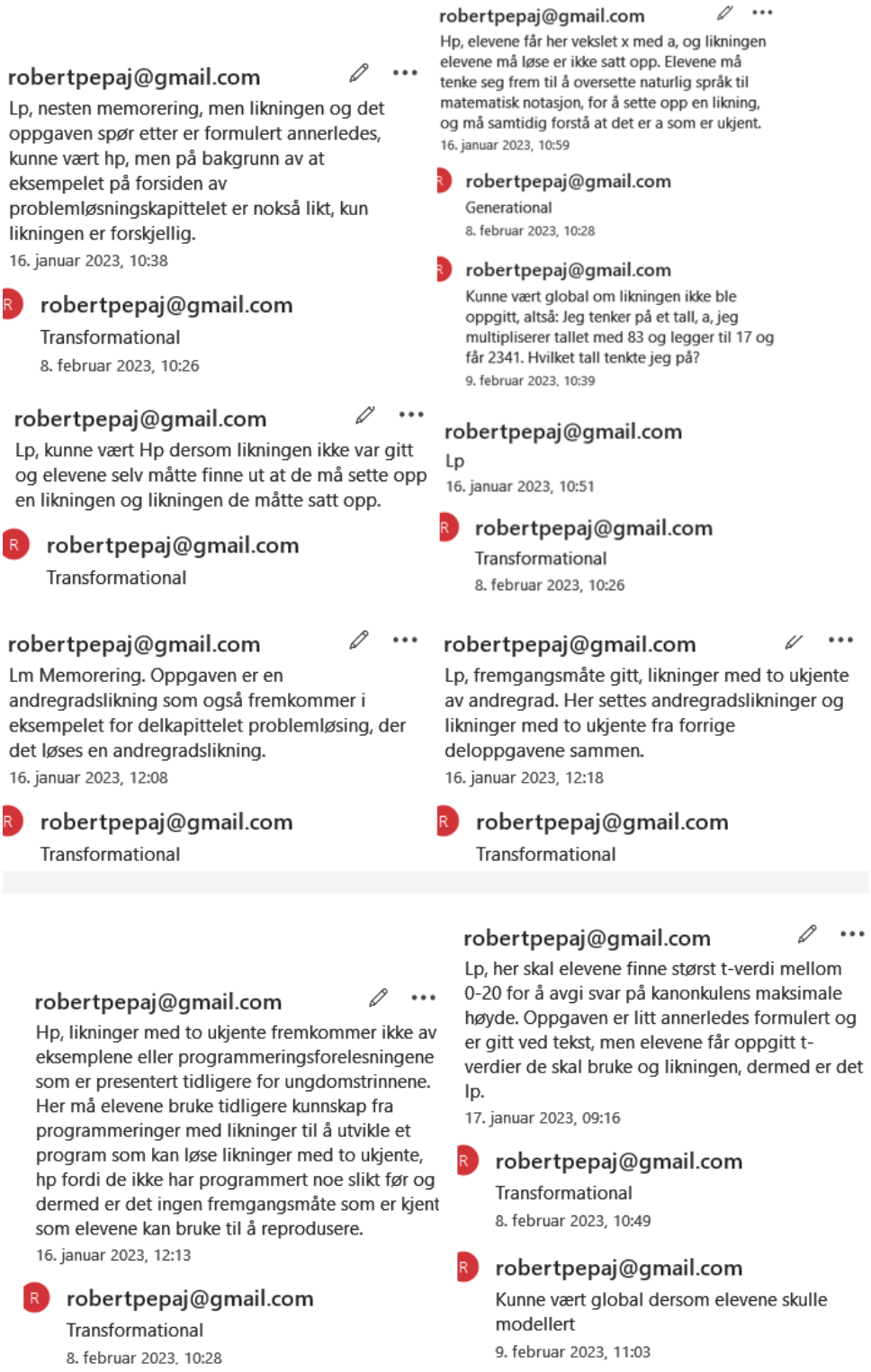
Vedlegg 1 – Statistikk over oppgaver i læreverkene kategorisert etter matematisk tema



Vedlegg 2 – Statistikk over oppgaver i læreverkene kategorisert etter programmeringstype



Vedlegg 3 – Eksempler fra kodingen av kognitive krav og algebraiske aktiviteter i Word



robertpepaj@gmail.com

Lp, nesten memorering, men likningen og det oppgaven spør etter er formulert annerledes, kunne vært hp, men på bakgrunn av at eksempelet på forsiden av problemløsningskapittelet er nokså likt, kun likningen er forskjellig.

16. januar 2023, 10:38

robertpepaj@gmail.com

Transformational

8. februar 2023, 10:26

robertpepaj@gmail.com

Lp, kunne vært Hp dersom likningen ikke var gitt og elevene selv måtte finne ut at de må sette opp en likningen og likningen de måtte satt opp.

robertpepaj@gmail.com

Transformational

robertpepaj@gmail.com

Hp, elevene får her vekslet x med a, og likningen elevene må løse er ikke satt opp. Elevene må tenke seg frem til å oversette naturlig språk til matematisk notasjon, for å sette opp en likning, og må samtidig forstå at det er a som er ukjent.

16. januar 2023, 10:59

robertpepaj@gmail.com

Generational

8. februar 2023, 10:28

robertpepaj@gmail.com

Kunne vært global om likningen ikke ble oppgitt, altså: Jeg tenker på et tall, a, jeg multipliserer tallet med 83 og legger til 17 og får 2341. Hvilket tall tenkte jeg på?

9. februar 2023, 10:39

robertpepaj@gmail.com

Lp

16. januar 2023, 10:51

robertpepaj@gmail.com

Transformational

8. februar 2023, 10:26

robertpepaj@gmail.com

Lm Memorering. Oppgaven er en andregradslikning som også fremkommer i eksempelet for delkapittelet problemløsing, der det løses en andregradslikning.

16. januar 2023, 12:08

robertpepaj@gmail.com

Transformational

robertpepaj@gmail.com

Lp, fremgangsmåte gitt, likninger med to ukjente av andregrad. Her settes andregradslikninger og likninger med to ukjente fra forrige deloppgavene sammen.

16. januar 2023, 12:18

robertpepaj@gmail.com

Transformational

robertpepaj@gmail.com

Hp, likninger med to ukjente fremkommer ikke av eksemplene eller programmeringsforelesningene som er presentert tidligere for ungdomstrinnene. Her må elevene bruke tidligere kunnskap fra programmeringer med likninger til å utvikle et program som kan løse likninger med to ukjente, hp fordi de ikke har programmert noe slikt før og dermed er det ingen fremgangsmåte som er kjent som elevene kan bruke til å reprodusere.

16. januar 2023, 12:13

robertpepaj@gmail.com

Transformational

8. februar 2023, 10:28

robertpepaj@gmail.com

Lp, her skal elevene finne størst t-verdi mellom 0-20 for å avgi svar på kanonkulens maksimale høyde. Oppgaven er litt annerledes formulert og er gitt ved tekst, men elevene får oppgitt t-verdier de skal bruke og likningen, dermed er det lp.

17. januar 2023, 09:16

robertpepaj@gmail.com

Transformational

8. februar 2023, 10:49

robertpepaj@gmail.com

Kunne vært global dersom elevene skulle modellert

9. februar 2023, 11:03

Vedlegg 4 – Alle underkategorier og koder for algebraoppgavene i Nvivo.

Algebra	2	41	RP	15.12.2	RP	20.12
Figurtall og tallmønster	2	9	RP	04.01.2	RP	04.01
Lag et program som finner omkretsen til en gitt figur i et figurtallmønster	1	1	RP	08.12.2	RP	20.12
Lag et program som finner summen av omkretsen til figur 1 til 53 i et gitt figurtallmø	1	1	RP	08.12.2	RP	20.12
Lag et program som finner ut antall fyrstikker i et figurnummer brukeren skriver inn i	1	1	RP	08.12.2	RP	20.12
Lag et program som finner ut antall kuler fra og med figur 1 til og med figur 78 i et g	1	1	RP	08.12.2	RP	20.12
Lag et program som finner ut antall kuler i en gitt figur i en figurtallmønster	1	1	RP	08.12.2	RP	20.12
Lag et program som finner ut summen av fyrstikker vi trenger for å lage de 15 første	1	1	RP	08.12.2	RP	20.12
Lag et program som fortsetter et gitt figurtallmønster bestående av fyrstikker, progra	1	1	RP	08.12.2	RP	20.12
Lag et program som regner ut hvor mange kuler vi har igjen dersom vi lager de først	1	1	RP	08.12.2	RP	20.12
Tolke en programkode som lar brukeren skrive inn et tall og antall påfølgende tall, pr	1	1	RP	14.12.2	RP	20.12
Lag eller tolk et program som løser likninger (algebra)	2	17	RP	14.12.2	RP	20.12
Forklar en programkode som lar brukeren skrive inn en x verdi for en likning, progra	1	1	RP	14.12.2	RP	20.12
Følg instruksjonene og kopier en programkode som lar brukeren skrive inn et interva	1	1	RP	14.12.2	RP	20.12
Følg instruksjonene og kopier en programkode som løser en likning, brukeren skrive	1	1	RP	14.12.2	RP	20.12
Følg instruksjonene og kopier en programkode som løser en likning, forklar hvordan	1	1	RP	14.12.2	RP	20.12
Følg instruksjonene og kopier en programkode som løser en likning, og forklar hva s	1	1	RP	14.12.2	RP	20.12
Lag et program som finner det ukjente tallet a i et gitt regnestykke, med et gitt svar	1	1	RP	07.12.2	RP	20.12
Lag et program som finner ut hva en pakke makaroni og en pakke spaghetti koster,	1	1	RP	07.12.2	RP	20.12
Lage et program som finner et ukjent positivt heltall x i en gitt likning som gjør at lik	1	2	RP	07.12.2	RP	20.12
Lage et program som finner to ukjente positive heltall, a, og b, som gir rett løsning i	1	1	RP	07.12.2	RP	20.12
Lage et program som finner to ukjente positive heltall, x og y, som gir rett løsning i e	1	2	RP	07.12.2	RP	20.12
Lage et program som løser en gitt likning	1	1	RP	07.12.2	RP	20.12
Lage et program som løser en tekstoppgave der likningen er gitt	1	1	RP	07.12.2	RP	20.12
Tolke en programkode som finner den største y-verdien i en likning	1	1	RP	08.12.2	RP	20.12
Tolke en programkode som løser en likning, gjenkjenne likningen programmet løser.	1	1	RP	07.12.2	RP	20.12
Tolke en programkode som regner ut en gitt likning for ulike x-verdier og skriver ut	1	1	RP	08.12.2	RP	20.12
Programmer som finner en ukjent og problemsituasjoner	1	5	RP	04.01.2	RP	25.04
Lage et program som finner et ukjent tall, det ukjente tallet skal adderes med 12 og	1	1	RP	07.12.2	RP	20.12
Lage et program som finner et ukjent tall, det ukjente tallet skal divideres med 18 og	1	1	RP	07.12.2	RP	20.12
Lage et program som finner ut et ukjent antall elever i en klasse når det er kjent at el	1	1	RP	07.12.2	RP	20.12
Lage et program som finner ut hvor mange ganger en person spiste middag på cafe,	1	1	RP	07.12.2	RP	20.12
Lage et program som finner ut hvor mange katter det er i en by, når det er kjent at al	1	1	RP	07.12.2	RP	20.12
Funksjoner	2	10	RP	15.12.2	RP	20.12
Lag eller tolk et program som regner ut verdier i funksjomer	2	10	RP	14.12.2	RP	20.12
Lag eller tolk et program som finner ekstremalverdier og nullpunkt	1	3	RP	15.12.2	RP	20.12
Lag et program som definerer en gitt funksjon og bruk funksjonen til å finne	1	1	RP	08.12.2	RP	20.12
Lag et program som finner falkens minimale høyde over bakken for t sekund	1	1	RP	08.12.2	RP	20.12
Lag et program som simulerer en kanon som skyter en kule, der funksjonen f	1	1	RP	08.12.2	RP	20.12
Lag en programkode som bruker tallet brukeren skriver inn og multipliserer det	1	1	RP	14.12.2	RP	20.12
Lag en programkode som kalles en funksjonsmaskin, programmet skal bruke et	1	1	RP	14.12.2	RP	20.12
Lag en programkode som regner ut funksjonsverdier for en gitt funksjon, progra	1	1	RP	14.12.2	RP	20.12
Lag et program som lager en funksjon som beregner stigningstallet til en linje so	1	1	RP	08.12.2	RP	20.12
Lag et program som regner ut funksjonsverdier fra 1 til 10 i en gitt funksjon	1	1	RP	08.12.2	RP	20.12
Lag et program som regner ut funksjonsverdier fra 1 til 100 i en gitt funksjon, pr	1	1	RP	08.12.2	RP	20.12
Tolke en programkode som skriver ut funksjonsverdier for en gitt funksjon, utvid	1	1	RP	14.12.2	RP	20.12

Vedlegg 5 – Epost fra Aschehoug angående bruk av læreverket Matemagisk i masteroppgaven

Rebekka Næss <rebekka.nass@aschehoug.no> man. 3. okt. 2022, 12:53
til meg

Hei! Så gøy at du vil se nærmere på Matemagisk. Du kan selvfølgelig bruke skjermdumper og oppgaver fra bøkene i masteroppgaven din. Vi vil veldig gjerne lese den når du er ferdig, slik at vi kan lære av den og forbedre oss. Har du tilgang til Aunivers.no slik at du kan få tilgang til de digitale programmeringsressursene våre på 4.-10.trinn? Hvis ikke kan du be om en prøvelisens og henvise til meg når du snakker med kundeservice. Lykke til med masteren.

Med vennlig hilsen

Rebekka Næss
forlagsredaktør realfag, Aschehoug skole

H. Aschehoug & Co (W. Nygaard) AS

+47 93035459
aschehoug.no

ASCHEHOUG

Vedlegg 6 – Epost fra Campus Inkrement angående bruk av læreverket Campus Inkrement i masteroppgaven

Lars Unneberg <campus@inkrement.no> ons. 7. des. 2022, 10:57
til meg

Hei Robert,
det er selvsagt helt i orden. Forutsetter at det er enkeltbilder og ikke en full dokumentasjon av alt i Campus i dette emnet.

Vennlig hilsen
Lars Unneberg
Inkrement

Vedlegg 7 – Epost fra Aunivers angående bruken av Matemagisk i skolen

Kontakt aunivers <kontakt@aunivers.no> fre. 11. nov. 2022 på 12:53
Til: Robert Pepaj

Hei,
vi oppgir ikke markedsandeler, men kan bekrefte at Matemagisk 8-10 er mye brukt og har en betydelig markedsandel.
--
Prefer fewer emails from me? Click here

Vis opprinnelig melding

Vedlegg 8 – Samtykkeskjema med medstudent

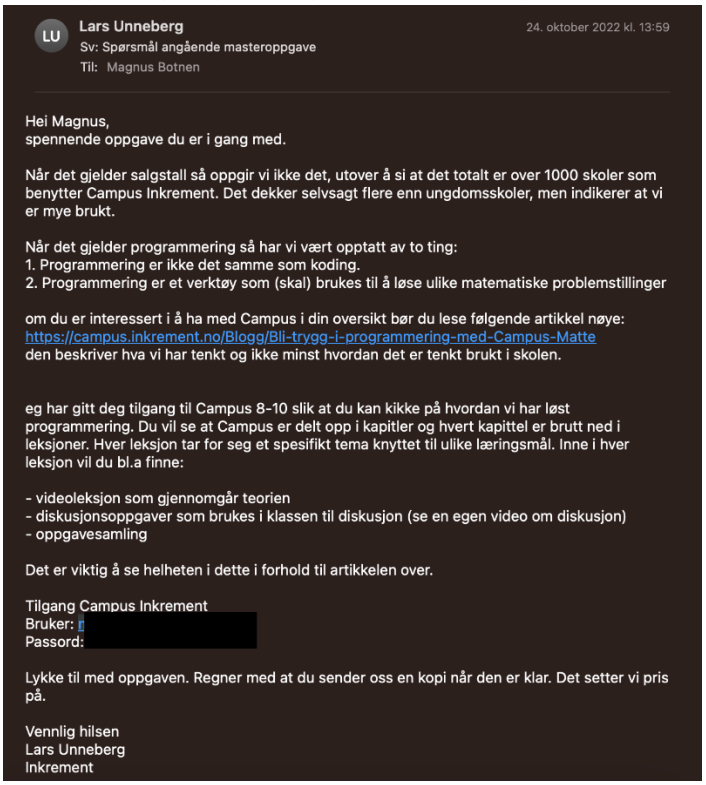
Vi, Robert Pepaj og Magnus Botnen, samtykker om å dele og bruke hverandres informasjon og data angående spørreundersøkelse for læreverk, som lærere og skoler bruker til å hente programmeringsoppgaver i matematikk i masteroppgaven. Vi samtykker også til å dele eventuelle eposter fra forlag/læreverk angående brukstall.

Bergen, 08.05.2023

Underskrift: Robert Pepaj
Robert Pepaj

Underskrift: Magnus Botnen
Magnus Botnen

Vedlegg 9 – Epost fra Campus Inkrement angående bruken av Campus Inkrement i skolen



Lars Unneberg 24. oktober 2022 kl. 13:59
Sv: Spørsmål angående masteroppgave
Til: Magnus Botnen

Hei Magnus,
spennende oppgave du er i gang med.

Når det gjelder salgstall så oppgir vi ikke det, utover å si at det totalt er over 1000 skoler som benytter Campus Inkrement. Det dekker selvsagt flere enn ungdomsskoler, men indikerer at vi er mye brukt.

Når det gjelder programmering så har vi vært opptatt av to ting:
1. Programmering er ikke det samme som koding.
2. Programmering er et verktøy som (skal) brukes til å løse ulike matematiske problemstillinger

om du er interessert i å ha med Campus i din oversikt bør du lese følgende artikkel nøye:
https://campus.inkrement.no/Blogg/Bli-trygg-i-programmering-med-Campus-Matte
den beskriver hva vi har tenkt og ikke minst hvordan det er tenkt brukt i skolen.

eg har gitt deg tilgang til Campus 8-10 slik at du kan kikke på hvordan vi har løst programmering. Du vil se at Campus er delt opp i kapitler og hvert kapittel er brutt ned i leksjoner. Hver leksjon tar for seg et spesifikt tema knyttet til ulike læringsmål. Inne i hver leksjon vil du bl.a finne:

- videoleksjon som gjennomgår teorien
- diskusjonsoppgaver som brukes i klassen til diskusjon (se en egen video om diskusjon)
- oppgavesamling

Det er viktig å se helheten i dette i forhold til artikkelen over.

Tilgang Campus Inkrement
Bruker: [redacted]
Passord:

Lykke til med oppgaven. Regner med at du sender oss en kopi når den er klar. Det setter vi pris på.

Vennlig hilsen
Lars Unneberg
Inkrement