Høgskolen i Østfold

MASTEROPPGAVE

Integreringen og relasjonen mellom matematikk og programmering i matematiske lærebøker

Kasper Gustav Skogholt

15.05.2024

Grunnskolelærer 1-7

Fakultetet for lærerutdanninger og språk

Sammendrag

Denne masteravhandlingen ser på programmering i matematikkfaget. Med fagfornyelsen ble det et fokus på å implementere algoritmisk tenkning, noe som gjorde at programmering kom inn i matematikkfaget. Programmering er ment som et verktøy for å utforske matematikk, samt å lære elever sentrale aspekter av algoritmisk tenkning.

Jeg har valgt å fokusere på programmeringsoppgaver i lærebøker og hvorvidt de legger til rette for læring av algoritmisk tenkning. Jeg har valgt meg ut en problemstilling, med to forskningsspørsmål for å svare på problemstillingen:

> På hvilken måte kommer samspillet mellom programmering og matematikk frem i matematiske lærebøker?

Med forskningsspørsmålene:

1) På hvilken måte blir programmering integrert inn i matematiske lærebøker
2) På hvilken måte legger programmeringsoppgavene til rette for å lære matematikk?

For å svare på problemstillingen og forskningsspørsmålene, valgte jeg å se på grunnbøkene til Matemagisk 5-7. Dette ble gjort ved å gjennomføre en horisontal og vertikal analyse, basert på Charalambous et al. (2010) sitt analytiske verktøy. For å støtte opp analysen gjennomførte jeg en relasjonsanalyse basert på Nyman et al. (2024) sitt analytiske rammeverk. Videre gjennomførte jeg en handlingsanalyse og konseptanalyse, basert på Bråting & Kilhamn (2022) sitt analytiske rammeverk.

Gjennom analysen fant jeg at grunnbøkene til Matemagisk 5-7 har 1298 oppgaver, der 1277 inneholder matematiske elementer, og 88 inneholder programmeringselementer. I de delene av bøkene med programmering i seg, ble matematikk oftest brukt i en kontekst for å lære programmering. Videre mangler enkelte aspekter av algoritmisk tenkning i programmeringsoppgavene, som gjør at det potensielle læringsutbytte faller bort.

Abstract

This master's thesis looks at programming in the mathematics subject. With the new national curriculum, there was a focus on implementing computational thinking, leading to the integration of programming in the mathematics subject. Programming is meant as a tool to explore mathematics and teach pupils central concepts of computational thinking.

I chose to focus on programming tasks in textbooks and whether they facilitate computational thinking. I chose a thesis statement, with two research questions to help me answer the thesis statement:

> In what way does the interaction between programming and mathematics appear in mathematical textbooks?

With the following research questions:

1) In what way does programming get integrated into mathematical textbooks
2) In what way does programming tasks facilitate mathematical learning?

To answer the thesis statement and research questions, I chose to look at the textbooks in Matemagisk 5-7. To help with the analysis, I chose to conduct a horizontal and vertical analysis, based on Charalambous et al. (2010) analytical tool. To further support the analysis, I conducted a relation analysis based on Nyman et al. (2024) analytical framework. Furthermore, I conducted an actions analysis and concept analysis, based on Bråting & Kilhamn (2022) analytical framework.

I found a total of 1298 tasks, where 1277 contained mathematical elements, and 88 contained programming elements. In the programming tasks, mathematics was most used as a context to learn programming. There were also some computational elements missing from the tasks, which reduces the potential for both computational learning, and mathematical learning.

Forord

Med denne masteravhandling er 5 år med lærestudie ferdig. Et av temaene som har interessert meg mest gjennom lærestudie har vært programmering, noe jeg har fått utforske mer gjennom denne masteravhandlingen. Programmering har en plass i skolen, og jeg gleder meg til å se hvordan programmering videre blir integrert inn i skolehverdagen.

Jeg er takknemlig for alle venner jeg har fått gjennom studiet, og alle jeg har bodd med, som har gjort min studietid uforglemmelig.

Videre ønsker jeg å takke min veileder Shipra Sachdeva for all veiledningen gjennom masteroppgaven. Takk for å la meg prøve og feile gjennom skriveprosessen, gi grundige og konstruktive tilbakemeldinger, og for å ha vært en god sparringspartner.

Videre ønsker jeg å takke alle i familien min for støtte gjennom skriveprosessen. Takk for alle middagsinvitasjoner, og statusoppdateringer gjennom skriveprosessen. Spesielt ønsker jeg å takke min bror Joakim for å ha vært en inspirasjon når jeg har forsket innenfor matematikkdidaktikk.

Til slutt ønsker jeg å takke min samboer Malene som også har skrevet master i denne perioden. Takk for å ha tolerert litt ekstra rot i dette halvåret, og for all støtte gjennom skriveprosessen.

Innholdsfortegnelse

Tabbeliste

Figurliste

1. Innledning

Allerede på ungdomsskolen var jeg heldig nok til å få prøve ut programmering i skolen, noe som var med på å bidra til økt interesse for algoritmisk tenkning. Videre på lærerstudiet skrev jeg også en FOU om læreres erfaringer innenfor programmering i matematikken. Der var et av hovedfunnene at lærerne hentet mye av informasjonen sin gjennom læreverk og digitale ressurser, noe som også har blitt funnet i annen forskning (Stigberg & Stigberg, 2020). Med dette ble jeg nysgjerrig på hvordan matematikklæreverk presenterer programmering, og fremmer videre læring innenfor programmering.

I arbeid med studiet leste jeg en setning av Kaufmann & Stenseth (2021), som fikk meg til å tenke over dagens situasjon med programmering og matematikk: «Have we designed a problem that utilizes programming to give a better understanding of mathematics, or are we using mathematics to improve the problem-solving skills in programming?» (s. 1045). De konkluderer selv med at oppgaven de lagde var en variasjon av begge, men dette fikk med til å undre om hvordan oppgaver med programmering blir presentert i matematiske lærebøker. Er oppgavene lagd for å gi en bedre forståelse av matematikk, eller brukes de for å lære programmering? Eller klarer oppgavene kanskje å gi en bedre forståelse innenfor både matematikk og programmering?

Gjennom denne masteroppgaven vil jeg se på hvordan programmering brukes i matematikkfaget. Dette vil bli gjort gjennom å se på lærebøker i matematikkfaget, og se hvordan programmering brukes både for å lære matematikk, men også algoritmisk tenkning. Dette vil bli gjort ved å se på hvordan relevante utrykk og konsepter blir introdusert til elevene, og innholdet i oppgavene.

1.1.Bakgrunn for programmering i skolen

I 2015 presenterte Ludvigsenutvalget sitt forslag om hva fagfornyelsen burde basere seg på. De har tatt utgangspunkt i hva de mener vil være relevant for arbeidslivet og elevene selv i fremtiden, og fremmer følgende kompetanseområder:

- Fagspesifikk kompetanse
- Kompetanse i å lære
- Kompetanse i å kommunisere, samhandle og delta
- Kompetanse i å utforske og skape

(NOU 2015:8, s. 22)

Ser vi videre på Ludvingsenutvalget sin rapport fremmer de også viktigheten av problemløsning, der de beskriver denne prosessen som å prøve forskjellige løsninger, justere underveis, og akseptere at man ikke finner riktig løsning med en gang (s. 34). Denne tankegangen om å løse problemer ble videreutviklet til det som kalles «Den algoritmiske tenkeren», der disse elementene fremmes, og tydeliggjøres. Utdanningsforbundet (2019) fordeler algoritmisk tenkning i *nøkkelbegrep* og *arbeidsmåter*:

Nøkkelbegrep:

1) Logikk – Analysere og forutse
2) Algoritmer – Regler og steg-for-steg
3) Dekomposisjon – Bryte ned i mindre deler
4) Mønstre – Finne og bruke likheter
5) Abstraksjon – Fjerne unødvendige detaljer
6) Evaluering – Gjøre vurderinger

Arbeidsmåter:

- Fikle – utforske og eksperimentere
- Skape – Designe og lage
- Feilsøke – Oppdage og rette feil
- Holde ut – Fortsette og prøve igjen
- Samarbeide – Dele og jobbe sammen

(Utdanningsdirektoratet, 2019)

Videre blir også viktigheten av digitalisering i skolen, som en sentral ferdighet både i arbeidslivet, samfunnslivet, privat og i skolen (NOU 2015:8, s. 26). Med dette fokuset av digitaliseringen i skolen, samt fokuset på problemløsning, ble det naturlig å vurdere å implementere programmering inn i den norske skole, om det enten var et eget fag, eller implementert inn i eksisterende fag (Sevik, 2016, s. 25).

I LK20 har programmering blitt implementert som en del av flere fag, inkludert matematikken. Dette innebærer både kompetansemål som har aspekter av algoritmisk tenkning, samt kompetansemål som krever konkret programmering, derav disse som er hentet fra kompetansemålene til 5-7 trinn:

- Lage og programmere algoritmer med bruk av variabler, vilkår og løkker
- Bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre
- Bruke programmering til å utforske data i tabeller og datasett

(Kunnskapsdepartementet, 2019)

1.2.Problemstilling

Ettersom min interesse for temaet ble vekket av refleksjonen til Kaufmann & Stenseth (2021), vil det være sentralt å orientere om hva de viser til i refleksjonen deres. Når de så på oppgaven de hadde konstruert for studien, bemerket de seg at formuleringen av oppgaven var rettet mot å forbedre ferdighetene til elevene innenfor programmet, samtidig som utbytte ville være å få en forsterket evne til å implementere matematikk inn i problemløsning. (Kaufmann & Stenseth, 2021, s. 1045). Jeg tolker dette som at programmering kan tenkes på som et verktøy, der ved riktig bruk kan eleven bruke algoritmisk tenkning for å løse et problem, og få et nytt perspektiv på matematikken i oppgaven.

Målet for denne masteroppgaven er å se på hvordan programmering blir brukt i matematiske lærebøker, og om de legger opp til potensielt læringsutbytte innenfor programmering, eller matematikk. For å gjøre dette, har jeg valgt ut følgende problemstilling:

> På hvilken måte kommer samspillet mellom programmering og matematikk frem i matematiske lærebøker?

For å svare på problemstillingen, har jeg valgt ut to forskningsspørsmål:

3) På hvilken måte blir programmering integrert inn i matematiske lærebøker
4) På hvilken måte legger programmeringsoppgavene til rette for å lære matematikk?

1.3. Disposisjon i masteravhandlingen

For å tydeliggjøre strukturen av denne masteravhandlingen, har jeg valgt å presentere disposisjonen. Dette er for å tydeliggjøre fordelingen av kapitler, innholdene i kapitlene, og begrunne strukturen.

I kapittel 1 ble bakgrunnen for valget av tema tydeliggjort. Videre blir bakgrunn for programmering i norsk skole presentert, med problemstilling og forskningsspørsmål.

I kapittel 2 vil det bli utført en gjennomgang av teorien og den tidligere forskningen som har blitt valgt ut for denne masteroppgaven. Først vil utrykket programmering bli drøftet, og presisert. Videre vil ulike former av programmering bli sett på, og hva som er likt og ulikt mellom disse formene. Etter programmeringen er gjort rede for, blir algoritmisk tenkning, og sammenhengen mellom algoritmisk tenkning og programmering drøftet. Til slutt vil lærebøker som en ressurs i matematikken bli sett på.

Videre i kapittel 3 blir metoden for masteravhandlingen presentert. Først vil valget av metode bli drøftet, og begrunnet. Videre vil valget av lærebokserie bli presentert, og relevant informasjon om læreverket. Når dette er gjort vil rammeverkene for masteravhandlingen bli presentert, og begrunnet. Bruken av rammeverkene, og relevante avgrensninger vil bli presentert og drøftet. Til slutt vil validiteten og reliabiliteten til oppgavene bli presentert, og drøftet.

For kapittel 4 har jeg valgt å kombinere både resultater av analysen, samt drøftingen. Dette ble gjort for å kunne presentere funnene fra analysen, og oppfølge funnene med relevant drøfting. Først blir resultatene fra den horisontale analysen presentert, og drøftet. Videre blir funnene fra den vertikale analysen presentert og drøftet. Videre vil relevante funn fra både horisontal- og vertikal analyse bli drøftet med hverandre, i relevante underkapitler.

Masteravhandlingens siste kapittel, altså kapittel 5, vil være konklusjonen. Her vil forskningsspørsmålenes svar konkluderes, og problemstillingen bli svart på. Videre vil forslag til videre forskning bli presentert.

2. Teori

2.1.Programmering

Som nevnt i innledningen, har programmering blitt en del av både skolehverdagen, og matematikk. For å kunne se på samspillet mellom programmeringsoppgaver og matematiske oppgaver, vil programmering bli drøftet og definert, ved hjelp av tidligere forskning.

Papert (1980) var en av de første til å se potensialet til programmering i matematikk, gjennom programmeringsspråket LOGO (Benton et al., 2018; Bråting & Kilhamn, 2021). Papert (1980, s. viii) reflekterer over at måten han tilegner seg matematisk kunnskap som ung, var gjennom å tenke på matematikken som ulike tannhjul, for å se sammenhengen i matematikken. Ønsket hans er at programmet LOGO, skulle hjelpe andre barn med å forstå matematikk, på samme måte som han gjorde med tannhjulene. Programmering ble etter hvert borte fra skolene igjen, der syntaksen med programmering hindret for læringen (Bråting & Kilhamn, 2022, s. 596).

Når man først tenker på ordet programmering er det lett å tenke at dette er kun å skrive kode inn på en datamaskin. Men det er flere ulike definisjoner av hva programmering er, både basert på et norsk perspektiv, men også mellom forskere. Stenseth et al. (2019) definerer programmering som «[…] prosessen knyttet til utvikling og implementering av instruksjoner for dataprogrammer slik at datamaskinen kan utføre spesifikke oppgaver, løse problemer og støtte menneskelige interaksjoner.» (Stenseth et al., 2019, s. 7). Ut fra denne definisjonen kan man tolke at programmering gjennomføres primært gjennom datamaskiner, for at datamaskinen skal utføre en spesifikk oppgave. Sevik (2016) har en relativt lik definisjon, men inkluderer også tankegangen til å skrive koden inn i en datamaskin:

> «Programmering […] omfatter mer enn å bare skrive programkode som kan kjøres på en datamaskin, det inkluderer også prosessen med å komme fram til denne koden. Det vil si prosessen fra 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) sin definisjon inkluderer også kode som ikke nødvendigvis er skrevet på datamaskin, noe Stenseth et al. (2019) sin definisjon ikke tar høyde for. Programmering uten datamaskin, eller analog programmering, baserer seg på arbeid med algoritmisk tankegang og IKT ved bruk av kun analoge hjelpemidler (Bell et al., 2009). Analog programmering er ment for å unngå de potensielle problemene som kan oppstå enten med datamaskinen eller å lære seg et språk for å programmere, og heller fokusere på hvordan man kan utvikle sin algoritmiske tenkning (Bell et al., 2009, s. 2). I denne avhandlingen vil Sevik (2016) sin definisjon bli brukt som utgangspunkt for hva programmering er.

2.1.1. Former av programmering

I kapittel 2.2 ble analog programmering nevnt som en måte å jobbe med programmering, som ikke inneholdt å skrive kode inn i en datamaskin (Bell et al., 2009). Analog programmering er et eksempel på en variant av programmering uten bruk av digitale verktøy, men det er også flere varianter av programmering ved bruk av digitale verktøy. To av disse variantene blir ofte omtalt som tekstbasert programmering (TBP), og blokkbasert programmering (BBP). I Norge blir BBP ofte brukt fra 1. trinn til 7. trinn, der det blir en overgang til TBP i 8. trinn (Munthe, 2022, s. 35). Ettersom denne mastergraden blir skrevet i grunnskolelærer 1-7, vil det derfor være mest relevant å fokusere på BBP.

BBP, som navnet tilsier, omhandler å bruke blokker for å programmere. I motsetning til tradisjonell TBP, der man manuelt skriver inn kode for å lage programmene, er koden representert som blokker. Disse blokkene har samme betydning som deres ekvivalent i TBP, men gjør syntaksen i programmeringen lettere (Benton et al., 2016, s. 7). Syntaksen i programmering omhandler forståelsen av å bygge kode, ved bruk av f.eks. parenteser, likhetstegn og semikolon (Munthe, 2022, s. 25). Ettersom BBP ikke ber brukeren om å skrive disse tegnene, blir flere av utfordringene rundt syntaksen borte. BBP har vist seg å være effektivt for å lære matematikk, der eleven (Lv et al., 2023, s. 8184).

Av BBP språk i skolen, er Scratch et av de mest populære språkene i Skandinavia, og i Norge (Bocconi et al., 2018, s. 22). Mye forskning rundt BBP i matematikk er også basert på Scratch (Lv et al., 2023, s.8182; e.g. Benton et al., 2017; Brennan & Resnick, 2012; Bråting & Kilhamn, 2021). I denne oppgaven skal lærebokserien Matemagisk 5-7 bli sett på, der BBP språket Trinket blir brukt. Trinket og Scratch er relativt like med måten de arbeides på, men skilles av at Scratch baserer seg på Javascript, og Trinket baserer seg på Python (Trinket, u.å.). Bruken av Trinket passer Matemagisk serien, der de bytter over til Python ved 8. trinn (Kongsnes & Wallace, 2020).

Arbeid med regneark, eller Excel, kan også regnes som programmering. Blant annet nevner Nagy et al. (2021, s. 25) at kan brukes for å arbeide med programmering, gjennom programmeringsrettete oppgaver. Ser man tilbake på definisjonen av programmering i 2.2, omhandler programmering å blant annet løse et problem ved å skrive kode som en datamaskin kan forstå. Videre kan regneark ses på som et programmeringsspråk, der man må bruke korrekt syntaks for å løse problemer, men har mindre potensialet en tradisjonelle programmeringsspråk (Cunha et al., 2014, s. 255). Ettersom Excel krever feilsøking for å løse syntaks problemer, og feilsøking er en sentral del av programmering (Bråting & Kilhamn, 2022; Nyman et al., 2024; Sevik, 2016), kan man anerkjenne Excel som arbeid med programmering.

Samtidig er det også viktig å nevne at Excel har et *potensiale* til å bli brukt som programmering. Det er nettopp Excel *kan* brukes som programmering, men oppgavene må da være lagd med utgangspunkt for å arbeide med programmering (Nagy et al., 2021, s. 25). På lik måte som at algoritmisk tenkning kan være programmering, så kan Excel være programmering. Men dette krever at oppgavene i Excel blir lagd på en måte som bruker verktøyene i Excel som programmering. Ettersom nivået på både matematikken, programmeringen og Excel er relativt lavt, i forhold til hva Nagy et al. (2021) skriver om, vil bruken av Excel ikke bli regnet som programmering i denne masteravhandlingen.

2.1.2. Programmering i matematikk

Å bruke programmering i matematikk er ikke en ny idé, der Papert (1980) presenterte tanken i sin bok Mindstorms. Programmering i seg selv hjelper ikke med å lære matematikk, men må brukes i en matematisk kontekst for å få et matematisk læringsutbytte (Benton et al., 2018, s. 599). Morten Munthe (2022, s. 13) skriver at i motsetning til CAS verktøy, som ikke viser hvordan utregninger blir utført, kan programmering gjøre utregningene tydeligere. Han utdyper at basert på designet av den matematiske oppgaven, kan programmering legge til rette for øvrig forståelse av den matematiske løsningen. Dette vil altså si at med arbeid med programmering i en oppgave, må eleven forstå den matematiske oppgaven for å kunne lage programmet.

Kaufmann & Stenseth (2021) så på hvordan elever i grupper arbeider med programmering i matematikk. De tok utgangspunkt i en geometribasert oppgave, der elevene måtte både feilsøke og finne den matematiske koblingen mellom programmet og temaet de arbeidet med. De fant at elever i grupper ofte bruker en prøve-og-feile teknikk, der de skriver inn ulike verdier for å se om det løser problemet. Dette gjør at den matematiske innvirkningen får negative konsekvenser, der det er mer fokus på om se om programmet kjører riktig med nye verdier, istedenfor å tenke nøyere gjennom den matematiske sammenhengen (Kaufmann & Stenseth, 2021).

Ettersom programmering kan ansees som en arbeidsmetode, kan man stille seg spørsmålet om hvilke matematiske temaer programmering oppstår i. Lv et al. (2023, s. 8178) skriver i sin litteraturanalyse at geometri og aritmetikk er de mest vanlige matematiske temaene å forske på. Dette kan vi også se i lærebøker, der Bråting & Kilhamn (2022, s. 603) fant at majoriteten av oppgaver omhandlet det som Lv et al. (2023, s. 8178) definerer som geometriske konsepter og aritmetikk. At geometri er et av de matematiske temaene som er mest populære er kanskje ikke overraskende, der Papert (1980, s. 51) sitt program Logo omhandlet å utforske geometriske former.

Å designe oppgaver innenfor matematikk som omhandler programmering er vanskelig, fordi det er flere andre erstattere som gjør jobben lettere (Munthe, 2022, s. 13). Det er også flere ulike symboler og begrep som er visuelt like mellom matematikk og programmering, men har ulik betydning. Bråting & Kilhamn (2021) ser på noen av disse begrepene, og hvordan de er ulike i algebra og programmering. Det var flere ulikheter, men spesielt med begrepet *variabel*. Variabel i en matematisk kontekst omhandler en varierende mengde og en plassholder, der variabel i programmering omhandler lagring av verdi som endrer seg underveis i et program (Bråting & Kilhamn, 2021, s. 180–181). Med disse forskjellene, må da eleven forstå hvilken kontekst variabel er ment å brukes i, som kan skape forvirring og unødvendige hindre hvis dette ikke blir tydeliggjort (Bråting & Kilhamn, 2021),

I matematikkfaget er lærerens rolle sentral for å formidle innholdet, og oppklare materialet som blir lært (Fan, 2013; Fan et al., 2013). Derfor er det også relevant å se på lærernes perspektiv av programmering, og hvordan de ser på forholdet mellom matematikk og programmering. Stigberg & Stigberg (2020) fant i sin undersøkelse at lærere sliter med å se sammenheng mellom matematikk og programmering, og elever på 1., og 6. trinn sliter med å se hvorfor man jobber med programmering i matematikk. Stigberg & Stigberg (2020, s. 494) nevner to forslag for å introdusere programmering inn i matematikk læreplanen:

1. Læreplanen må støttes opp med en solid infrastruktur, der lærere har ressursene til å undervise programmering i matematikk på en god måte
2. Forholdet mellom programmering og matematikk må være tydelig, der lærere og elever forstår hvorfor programmering er i matematikk, og hvordan det kan brukes i matematikk.

Som en del av et større forskningsprosjekt, så Bråting & Kilhamn (2022) på hvordan programmeringsinnhold blir karakterisert i svenske matematikkbøker på 1. trinn til 6. trinn, og hvordan oppgavene knytter sammen matematikk og programmering. Der fant de at et fåtall av oppgaver ba elevene om å feilsøke, forklare og forestille seg, noe som er sentralt innenfor læring av AT (Ezeamuzie & Leung, 2022; Lodi & Martini, 2021). Dette gjør at de stiller spørsmål til læringsutbyttet innenfor algoritmisk tenkning (Bråting & Kilhamn, 2022, s. 607). Videre ba flertallet av oppgavene elevene om å følge instrukser, som gjorde at det potensielle

læringsutbytte i oppgavene falt bort (Bråting & Kilhamn, 2022, s. 604). Forfatterne fant at begrep som *algoritme* mistet sitt matematiske bruksområde, og ble heller gjort om til et begrep innenfor algoritmisk tenkning (Bråting & Kilhamn, 2022, s. 606). De reflekterer videre at oppgavene krever at elevene følger instruksene som forfatterne setter, og lar ikke elevene utforske matematikken fritt nok (Bråting & Kilhamn, 2022).

Nyman et al. (2024) gjennomførte en oppfølgingsstudie til Bråting & Kilhamn (2022). I tillegg til å se på handlingene og konseptene i studien, så Nyman et al. (2024, s.7) også på hvordan programmering ble brukt sammen med matematikk i ungdomsskolen. Av 9 lærebøker og 3 spesialbøker om programmering i matematikk, fant forskerne at rundt 74% av enhetene (altså flere oppgaver under samme tema) brukte matematikk som en kontekst for å lære programmering. Dette vil altså si at matematikken ble brukt i en kontekst for å lære programmering, der matematikken ikke var relevant for aldersgruppen, og fokuset var primært på å lære programmering. Forfatterne nevner at dette kan føre til at programmeringen skaper flere hindre enn muligheter for å lære matematikk, og at matematikken blir nedprioritert for å lære programmering (Nyman et al., 2024).

2.2. Algoritmisk tenking

Algoritmisk tenkning er nevnt i kapittel 1.2, der formålet er å bryte ned større problemer, til mindre problemer. Algoritmisk tenkning er det norske ordet med mest likheter til det engelske begrepet Computational Thinking (Bocconi et al., 2018, s. 8). Det er viktig å bemerke at det norske utrykket algoritmisk tenkning også innebærer å være en digital medborger (Bocconi et al., 2018, s. 9), men vil i denne oppgaven primært omhandle tankeprosessen med å dele opp større problemer til mindre problemer. Derfor vil computational thinking bli skrevet som algoritmisk tenkning (AT) videre i denne masteravhandlingen.

AT ble først nevnt av Seymour Papert (1980, s. 182), som en måte å tenke som en datamaskin. Det skal sies at Papert (1980) sitt bruk av ordet ikke var ment som en definisjon, men ble heller brukt tilfeldig (Lodi & Martini, 2021, s. 890). Den moderne definisjonen av algoritmisk tenkning stammer fra Jeannette M. Wing (2006). Hun nevner at algoritmisk tenkning innebærer flere ting, blant annet å angripe et større problem ved å dele opp problemet inn i løsbare biter (Wing, 2006, s. 33). Dette er noe som også er med i den norske definisjonen av algoritmisk tenkning.

Wing (2006) presiserer at algoritmisk tenkning ikke er programmering med datamaskin, men heller å tenkte på flere abstraksjonsnivåer (Wing, 2006, s. 35). Selv om programmering kan ansees som arbeid med algoritmisk tenkning, betyr ikke det at arbeid med algoritmisk tenkning er programmering. Wing (2006, s. 36) nevner at «Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction». Wing (2006) skiller da AT fra programmering, der hun definerer AT som en tankeprosess, og programmering som en handling man gjør.

Wing (2006) gjorde AT relevant igjen, og argumenterte for at tankeprosessen skulle likestilles med lesing, skriving og aritmetikk (Bråting & Kilhamn, 2021, s. 170). Gjennom årene har flere bygget videre på Wing (2006) sine ideer om hva AT omhandler. Lodi (2020) analyserte definisjoner fra flere som bygger videre på Wing (2006) sine originale tanker om AT, der Lodi & Martini (2021, s. 896) fremmer fire nøkkelpunkter:

- Mental prosess: mentale strategier som er nyttige til å løse problemer.
- Metoder: Operasjonelle tilnærminger ofte brukt av informatikere.
- Praksiser: Typiske praksiser brukt i implementering av løsninger basert på datamaskiner.
- Tverrfaglige ferdigheter: Generelle måter å se og fungere i verdenen på, fremmet av tenking som informatikere; nyttige livsferdigheter som kan forsterke tenkingen som en informatiker.

Ser man tilbake på «Den algoritmiske tenkeren» (Utdanningsdirektoratet, 2019), er det flere likheter mellom modellen, og punktene som Lodi & Martini (2021) fremmer. «Mental prosess» ligger i navnet «Den algoritmiske *tenkeren*». AT er nettopp en tankemåte å bryte ned problemer, gjennom dekomposisjon og abstraksjon (Ezeamuzie & Leung, 2022; Utdanningsdirektoratet, 2019). Videre blir også flere arbeidsmetoder fremmet, blant annet *dekomposisjon* og *abstraksjon* (Utdanningsdirektoratet, 2019). Dette er flere av de sentrale metodene som informatikere arbeider med, som viser utbytte innenfor AT (Ezeamuzie & Leung, 2022; Lodi & Martini, 2021). Videre blir også praksiser fremmet som et sentralt punk, der AT består blant annet av å bruke algoritmer, og logikk (Lodi & Martini, 2021; Utdanningsdirektoratet, 2019). Løsningene er altså inspirert av datamaskiner, noe som er en av grunntankene Wing (2006) fremmet.

Det siste punktet Lodi & Martini (2021, s. 896) fremmer er de tverrfaglige ferdighetene som stammer fra AT. Som nevnt i introduksjonen, ble AT implementert i den norske skolen, for å dyrke den fremtidige medborgeren som eleven skal bli (NOU 2015:8). Tverrfagligheten handler ikke bare om å bruke AT i ulike skolefag, men også i forskjellige kontekster, der det er relevant å tenke som en informatiker (Ezeamuzie & Leung, 2022; Lodi & Martini, 2021). Ettersom AT skal være en av ferdighetene elevene skal kunne for å være en fremtidig medborger, vil det da være relevant å bruke programmering for å lære dette. Videre er AT tverrfaglig, som vil si at AT skal også kunne brukes innenfor matematikk.

2.2.1. Sammenhengen mellom AT og programmering

Formålet med dette delkapittelet er ikke å prøve å skille AT og programmering fra hverandre, men heller vise hva som gjør de ulike fra hverandre. I Ezeamuzie & Leung (2022) sin litteraturanalyse analyserte de 81 artikler om definisjonen av AT (s. 488). Der fant forskerne at flere av forfatterne i de analyserte artiklene skriver at AT er en tankemåte for å løse problemer med eller uten programmering. Men forfatterne i artiklene analysert definerte AT på en annen måte enn hvordan utrykket faktisk ble brukt i forskningen deres, og vurderte tilegningen av AT kunnskap gjennom programmering (Ezeamuzie & Leung, 2022, s. 501). Dette kan tolkes som at det ikke er lett å fastsette den tilegnede kunnskapen gjennom AT ferdigheter, men å heller se på programmeringsferdighetene som har kommet av arbeidet gjennom AT. Ezeamuzie & Leung (2022) fremmer to prinsipper av AT: at AT er underbygget informatikk domene, og at AT ferdigheter er overførbare i flere domener. Det faktumet at AT er overførbare i flere domener er det som skiller AT fra programmering (Ezeamuzie & Leung, 2022, s. 502). Dette vil altså si at programmering er et av domenene som AT kan overføres til.

Men da komme spørsmålet om hvilke domener som skiller programmering og AT. Ezeamuzie & Leung (2022) skriver at domener gjelder forskjellige faglige domener f.eks. teknologi, ingeniør, matematikk osv. (s. 493). Dette støtter opp formålet bak den algoritmiske tenkeren (Utdanningsforbundet, 2019), med at AT er en problemløsningsstrategi. Videre definerer Ezeamuzie & Leung (2022) programmering som ikke bare koding, men aktivitetene som omhandler informatikk rundt idemyldring, produksjon og problemløsning (s.501).

Ser man tilbake på Wing (2006) sitt synspunkt om at AT ikke nødvendigvis omhandler arbeid gjennom programmering (s. 35), så er ikke AT nødvendigvis kun programmering. Programmering er da heller en måte å arbeide med AT, gjennom å løse problemer gjennom kode. All AT er nødvendigvis ikke programmering, men all programmering er AT.

2.3. Bruken av lærebøker i matematikk

Til slutt i teorikapittelet, vil lærebokens rolle i matematikkfaget bli sett på. Ettersom problemstillingen skal vares på med hjelp av lærebøker, vil lærebokens rolle i matematikk være relevant å se på. Lærebøker er en av hovedkildene lærere bruker for å strukturere undervisningsøktene i klasserommet (Lepik et al., 2015, s. 132), og blir ofte prioritert høyere enn kompetansemålene i læreplanen (s.136). Lærebøker blir også brukt for å lage lekser, og påvirke læringsstilen til lærere (Fan et al., 2013). Derfor er det også viktig å se på innholdet til lærebøkene, for å se på kvaliteten av innholdet, og se hvordan innholdet i læreboken blir kommunisert til eleven (Norberg, 2019, s. 54). Viktigheten av lærebøker i matematikken er også flere forskere enige om (Fan et al., 2013), selv om det ikke har vært mye forskning før de siste par tiårene (Fan, 2013). Mye av forskningen innenfor programmering omhandler geometri og aritmetikk (Lv et al., 2023, s. 8182), som betyr at det kan være enkelte temaer det ikke har blitt forsket like mye på.

3. Metode

3.1.Valg av metode

Formålet med denne masteravhandlingen er å se på samspillet mellom programmering og matematikk, i matematikkfaget. Dette kunne ha blitt gjort på flere måter, der jeg f.eks. kunne ha gjennomført en observasjon. Da kunne jeg ha sett konkret arbeid med programmeringsoppgaver, og sett om elevene klarte å forstå koblingen mellom programmering og matematikk, som Kaufmann & Stenseth (2021) gjorde i sin undersøkelse. Da ville jeg ha kunne ha observert elevenes erfaringer med arbeidet, og kunne ha fokusert på elevenes grad av mestring under oppgaveløsning (Postholm & Jacobsen, 2018, s. 54–55). Ettersom denne masteroppgaven fokuserer på flere matematiske emner ville en nyttig bruk av observasjon behøve mer ressurser i form av tid og informanter enn det denne oppgaven kan romme.

Et annet alternativ hadde vært å gjennomføre semistrukturerte intervjuer, altså intervjuer med rom for oppfølgingsspørsmål (Kvale & Brinkmann, 2015, s. 45) med lærere i matematikk. Dette ville ha gitt meg lærernes perspektiver på hvordan de tolker elevenes læringsutbytte mellom programmering og matematikk, og deres tanker rundt temaet. Her kunne jeg også ha hørt om de har arbeidet med programmering i ulike matematiske temaer. I dette tilfellet hadde jeg hatt mulighet til å undersøke refleksjoner rundt programmeringsundervisning, men jeg ville ikke hatt mulighet til å vurdere innholdet i undervisningsmaterialet i den grad jeg ønsker. Derav har jeg valgt å bruke metoden tekstanalyse for å svare på problemstillingen.

3.2. Tekstanalyse

Ved å velge å gjennomføre en tekstanalyse, måtte jeg velge mellom å gjennomføre en kvalitativ eller kvantitativ tekstanalyse. En kvantitativ tekstanalyse ville ha gitt meg muligheten til å presentere funnene ved hjelp av nummer og statistikk samtidig som en kvalitativ tekstanalyse ville latt meg fokusere mer på innholdet (Morgan, 2022, s. 65). Samtidig er det nødvendigvis ikke bare fremleggingen som har en betydning, men også forskningsspørsmålet som skal besvares. Ofte ved kvantitative innholdsanalyser omhandler problemstillingene større spørsmål, som krever flere ulike datasett for å svare på problemstillingen (Clark et al., 2021, s. 271).

En tekstanalyse kan også gjennomføres kvalitativt, der man kan være mer objektiv, og systematisk (Clark et al., 2021, s. 271). Hvis disse to kriteriene blir møtt, vil hvem som helst kunne gjennomføre samme analyse, noe som støtter opp reliabiliteten til oppgaven (Clark et al., 2021, s. 271; Postholm & Jacobsen, 2018, s. 223). Primært vil jeg ta utgangspunkt i å gjennomføre en kvalitativ analyse, men ha med enkelte elementer av kvantitativ metode. Dette vil si at jeg primært vil se dypere på innholdet i lærebøkene, men vil også presentere data i form av tall og tabeller, som er vanlig i kvantitativ metode (Morgan, 2022, s. 65).

Å gjennomføre en kvalitativ tekstanalyse vil si å ta kvalitativ data, og forsøke å finne konsistens og mening i teksten (Patton, 2015, s. 541). Kvalitativ tekstanalyse kan gjennomføres på to måter;

- Deduktivt, der man vurderer om dataen passer en generell oppfatning, forklaringer, resultater og/eller teorier.
- Induktivt, der man generere nye konsepter, forklaringer, resultater og/eller teorier fra innhentet data fra den kvalitative studien

(Patton, 2015, s. 541)

Dette vil altså si at dersom man bruker en induktiv tilnærming vil man se etter nye teorier og mønstre i datamaterialet man analyserer, mens dersom man bruker en deduktiv tilnærming vil man benytte seg av etablerte analytiske verktøy (Patton, 2015, s. 542). I tillegg til disse tilnærmingene er det også mulig å bruke begge metodene om hverandre, dette blir kalt for en abduktiv tilnærming (Postholm & Jacobsen, 2018, s. 102). Det vil altså si at forskeren går frem og tilbake mellom teori og empiri, gjennom forskningsprosessen (Postholm & Jacobsen, 2018, s. 103). Ettersom dette forskningsstudiet ble inspirert av et spørsmål stilt av Kaufmann & Stenseth (2021) vil metoden som er benyttet i denne masteroppgaven være hovedsakelig deduktiv. Dette er også grunnet at datamaterialet blir analysert gjennom analytiske verktøy, noe som også viser til en deduktiv tilnærming. Allikevel skriver Postholm & Jacobsen (2018) at en metode sjeldent er helt induktiv eller deduktiv, noe som også vil gjelde denne studien.

3.3. Valg av lærebok

For å velge et læreverk å analysere var det flere kriterier som måtte oppfylles. Det første kriteriet var at læreboken var skrevet i henhold til LK20 (Kunnskapsdepartementet, 2019). Dette var for å sikre at bokserien hadde blitt skrevet til de oppdaterte kompetansemålene, spesielt kompetansemålene basert på programmering. Videre trengte bokserien å inneholde programmeringsoppgaver som omhandlet BBP eller TBP. Dette er for å forsikre at oppgavene i lærebøkene inneholdt programmering, og ikke bare AT-oppgaver. Til slutt trengte læreverket å ha programmeringsoppgaver gjennom boka, og ikke bare isolert i egne dedikerte kapitler. Dette var for å forsikre at programmeringen ble brukt som et verktøy slik læreplanen etterlyser, og ble brukt i ulike matematiske temaer. Matemagisk serien oppfylte alle disse kravene, og jeg har personlig opplevd at lærere i skolen er positive til bokserien, noe som gjorde at jeg valgte å analysere akkurat dette læreverket. Videre har jeg noe tidligere erfaring med Matemagisk bokserien, men det gjelder ikke bøkene gått igjennom i denne masteravhandlingen.

3.3.1. Matemagisk

Matemagisk er en lærebokserie publisert av Aschehoug undervisning. Bokserien har to grunnbøker med en parallellbok per grunnbok, oppgavebok, lærerveiledning og nettressurser. I denne masteravhandlingen ble det valgt å kun se på grunnbøkene til trinnene. Parallellbøkene ble valgt bort, ettersom oppgavene var enten like som de i grunnboken, eller hadde mindre omformuleringer. Videre ble oppgaveboken valgt bort, der temaene er like som de i grunnbøkene. Originalt ble de digitale ressursene vurdert å se på, men dette ble valgt bort basert på mengden med data. Til slutt ble lærerveiledningen valgt bort, der denne masteravhandlingens mål er å se på elevenes potensielle læringsutbytte, vil det være mest relevant å se på oppgavene elevene møter på.

Grunnbøkene i Matemagisk 5-7 deler kapitlene opp inn i 5 ulike deler. Et kapittel starter med en «Fellesløype», som er ment som en introduksjon til temaet. I hvert kapittel er det også én «Kontekstoppgave» som baserer oppgavene på et scenario, som elevene skal hjelpe med å løse. Videre er det tre fordelinger til, som omhandler vanskelighetsgrad. Disse fordelingene blir kalt for «spor», og blir beskrevet slik av forfatterne:

1. Følg stien – Oppgaver der du får trent mer på det klassen har arbeidet med i fellesskap. Her trener du på én ting om gangen.
2. Terrengløypa – Oppgaver som bygger videre på det klassen har arbeidet med i fellesskap. Her kan du få sammensatte utfordringer, også fra flere temaer på en gang.
3. Topptur – Oppgaver som er svært utfordrende. Jobb med toppturen hvis du mestrer oppgavene i terrengløypa godt.

(Raen et al., 2020, s. 2)

De ulike «sporene» kan ansees som et hierarki av vanskelighetsgrader, der «følg stien» tilbyr flere oppgaver ment til fellesskapet, «Terrengløypa» bygger videre på temaene i oppgavene, og «Topptur» som det vanskeligste i kapittelet. Ofte kommer det flere seksjoner med «fellesløype» og «følg stien» gjennom kapittelet, der «Terrengløypa» og «Topptur» kommer bare på slutten av kapittelet.

3.4.Bruk av rammeverk

For å kunne se på hvordan samspillet mellom programmering og matematikk kommer frem i programmeringsoppgaver, har jeg valgt å gjennomføre en horisontal og vertikale tekstbokanalyse (Charalambous et al., 2010, s. 123). Å gjennomføre en horisontal analyse vil gi et overblikk over lærebøkene, for å få et overblikk over innholdet i boken, og kunne finne de relevante oppgavene å gjennomføre en vertikal analyse (Charalambous et al., 2010, s. 122). En vertikal analyse innebærer å ta et dypere innblikk i oppgaven, der man blant annet ser hva som blir formidlet til eleven, hva som kreves av eleven, og sammenhengen til eleven (Charalambous et al., 2010, s. 123). De analytiske rammeverkene som vil bli brukt i horisontal og vertikal analyse vil videre bli presentert.

3.4.1. Horisontal analyse

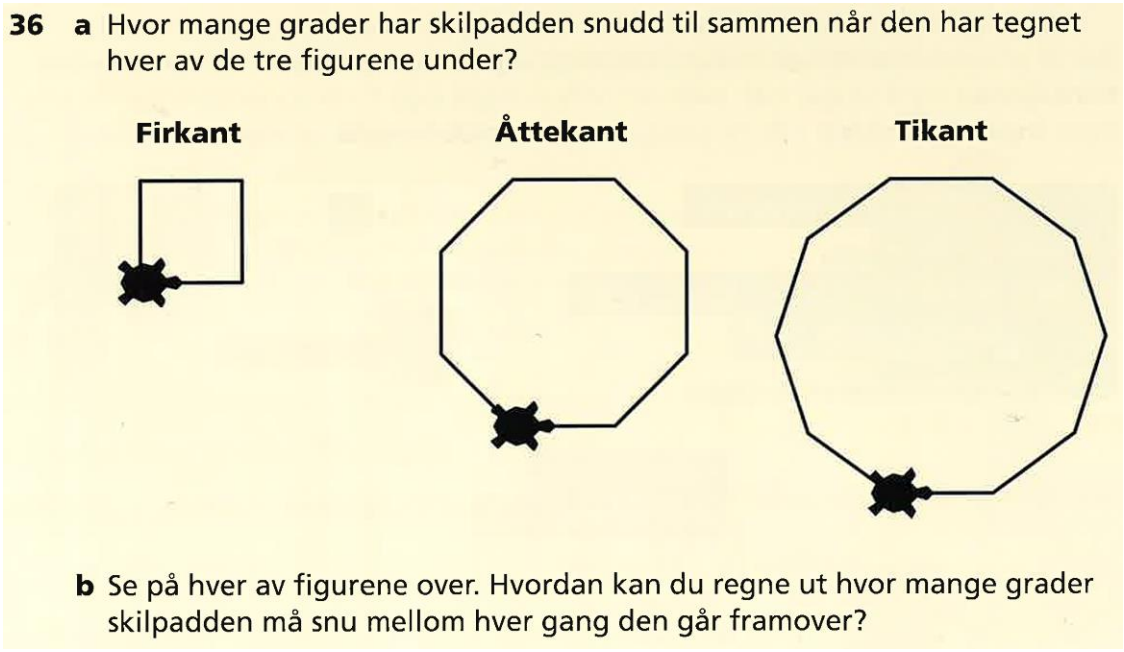
Å gjennomføre en horisontal analyse handler om å få et overblikk over strukturen til læreboken (Charalambous et al., 2010, s. 122). Dette kan være sideantall, struktur av boken, rekkefølge, og emneoversikt (Charalambous et al., 2010, s. 122). Ved å få en oversikt over hvor mange oppgaver som både har elementer av matematikk og programmering, vil utgangspunktet til den vertikale analysen bli fremmet. For å gjennomføre den horisontale analysen har jeg valgt å se på hver oppgave, og se om oppgaven inneholder matematiske og/eller programmerings konsepter. Dette er for å både se strukturen av boken, og få et totalt overblikk over hvor mange oppgaver som det vil bli gjennomført en vertikal analyse på. Videre blir også alt rundt oppgavene sett på, for å få et helhetlig perspektiv over innholdet i lærebøkene.

For å få en oversikt over programmeringsoppgavene i lærebøkene, har jeg laget en tabell som kategoriserer oppgavene. Ved å ha en tabell for å kategorisere vil dette både vise hvilke oppgaver som er relevante for en vertikal analyse, og for å unngå å bli overveldet av mengden med data (Schreier, 2012, s. 58). Ved å også sortere basert på flere kriterier, vil det være enklere å gjennomføre en dypere analyse ved at datainnsamling fra den horisontale analysen fungerer som en byggestein for den vertikale analysen (Schreier, 2012, s. 65).

Inneholder matematiske elementer		
I stor grad	I mindre grad	I ingen grad
Inneholder programmering elementer		
I stor grad	I mindre grad	I ingen grad

Tabell 3.1 – Matematiske elementer og programmeringselementer

En oppgave ble definert som *i stor grad* innenfor matematikk, hvis hovedtemaet i oppgaven var matematikk. Den ble kategorisert som *i mindre grad* hvis den inneholdt matematikk, men som ikke var relevant for temaet i kapittelet. Til slutt ble den kategorisert som *i ingen grad* hvis det ikke var matematikk, eller nivået på matematikken var betydelig lavere enn hva som er forventet på trinnet. En oppgave ble ansett som *i stor grad* innenfor programmering, hvis den omhandler BBP. Videre ble den kategorisert som *i mindre grad* hvis den hadde et fokus på AT, eller beskrev programmeringskonsepter. Til slutt ble den kategorisert som *i ingen grad* hvis det var verken programmeringsinnhold eller AT i oppgaven. I eksempelet under blir markeringen av oppgaven vist, og tydeliggjort:



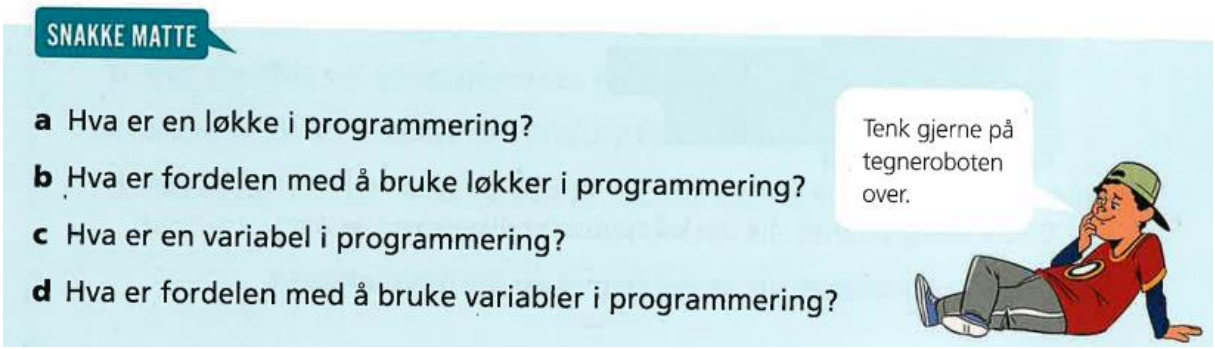
Figur 3.1 – Oppgave 36 Kapittel 7. Fra *Matemagisk 6B: Grunnbok* (s. 118), av A.L. Kongsnes et al., 2021b Aschehoug Undervisning

Oppgave: 36 a + b		
Inneholder matematiske konsepter		
I stor grad	I noe grad	I ingen grad
x		
Inneholder programmering konsepter		
I stor grad	I noe grad	I ingen grad
	x	

Tabell 3.2 – Eksempel av oppgave med matematiske- og programmeringselementer

I oppgave 36 har det blitt notert ned «i stor grad» for matematikk, og «i noe grad» for programmering. Her ble det notert *i stor grad* i matematikk fordi det handler spesifikt om grader, og ber eleven om å regne ut antall grader skilpadden snur seg. Videre ble det notert ned «i noe grad» på programmering konsepter. Selv om det ikke er noe i teksten som tyder til at oppgaven omhandler programmering, så er det flere konsepter i spill. Oppgaven er tydelig inspirert av Papert (1980) sitt program LOGO, og omhandler å forutse et utfall, noe som er en sentral del av programmering (Bråting & Kilhamn, 2022, s. 604). Videre er det tydelige AT elementer, blant annet dekomposisjon samt logikk (Ezeamuzie & Leung, 2022).

Det er enkelte krav til hva jeg definerer som en oppgave i Matemagisk. Gjennom grunnbøkene i Matemagisk er det flere markeringer på innholdet i bøkene. Dette inkluderer blant annet *Snakke matte* bokser, som inneholder spørsmål om det relevante temaet.



Figur 3.2 – Snakke matte. *Fra Matemagisk 6B: Grunnbok* (s. 105), av A. L. Kongsnes et al., 2021b, Aschehoug undervisning.

I figur 3.2 er en av *Snakk matte* boksene, på starten av delkapittelet *Å utforske mønstre med programmering*. Innholdet i boksen er relatert til programmering, og er skrevet opp som oppgaver fra a-d. Men tekstens innhold viser at spørsmålene er mer ment til å reflektere om hva de ulike begrepene betyr. Dette støttes opp i forfatternes forklaring av *snakke matte*, der de skriver at «… der dere skal snakke med hverandre. Her trener dere på å forklare hvordan dere tenker» (Kongsnes et al., 2020, s. 2). *Snakke matte* kan tenkes på mer som en tankeøvelse, og for at elevene skal tenke over begrepene i de ulike temaene. Videre er oppgavene ment å gjøres i plenum, og kan ikke gjøres individuelt. Derfor vil ikke *snakke matte* innslagene kategoriseres som oppgaver i den horisontale analysen. Det er også lignende innhold i Matemagisk serien, f.eks. *brettspill*, som heller ikke vil bli ansett som en oppgave, for samme grunn.

3.4.2. Relasjonsanalyse

For å supplementere den horisontale analysen, ble det valgt å gjennomføre en relasjonsanalyse av kapitlene med programmeringsoppgaver i seg. En relasjonsanalyse ser på sammenhengen mellom matematikk og programmering, med nivået av de matematiske oppgavene (Nyman et al., 2024, s. 7). Nyman, Bråting & Kilhamn (2024, s. 2) gjennomførte en innholdsanalyse om hvordan programmering blir implementert inn i matematikk, og hvordan programmeringsinnholdet påvirker muligheten til å lære matematikk. Dette ble gjort ved å se på læreboksenheter, altså alle oppgavene som blir skilt med en introduksjon, tittel og eksempler (Nyman et al., 2024, s. 6). For å gjøre dette brukte de handlings- og konseptanalyse (Bråting & Kilhamn, 2022), men også relasjonsrammeverket (Nyman et al., 2024, s. 7). Rammeverket baserer seg på forholdet mellom matematikk (M) og programmering (P), på ulike nivåer:

MP1) *Programmering uten tilknytning til matematikk* - Tekstbok enhet som bare omhandler programmering

MP2) *Matematikk som en kontekst for programmering* - Tekstbok enhet der matematikk er til stede, men ikke med relevant faginnhold for trinnet, og hvor målet med læringen er i forhold til programmering

MP3) *Programmering som et verktøy for effektiv kalkulering* – Tekstbok enhet der koder og algoritmer blir brukt til å gjøre kalkulasjoner raskere og mer effektivt, der eleven får muligheten til å inkludere en bredere nummerområde, flere operasjoner eller større data sett ved arbeid med matematikk

MP4) *Programmering som et verktøy for å utforske matematikk* – Tekstbok enhet der programmering blir brukt for å utforske matematikk relevant for elevgruppen, og hvor målet er tydelig presisert i forhold til læring av matematikk

(Nyman et al., 2024, s. 7)

De forskjellige gradene av relasjoner viser til matematikkens relevans i oppgavene, fra 1 (liten relevans, til 4 (høy relevans). Det var kun kapitler med programmeringsenheter i seg, som ble sett på i relasjonsanalysen. Dette kunne jeg gjøre ettersom jeg allerede hadde funnet programmeringsoppgavene gjennom en tidligere analyse, der artikkelen til Nyman et al. (2024, s. i) ble publisert etter den originale analysen ble gjennomført.

Enhetene i Matemagisk 5-7 ble skilt på basert på flere førdefinerte kriterier. Ved nye underoverskrifter i bøkene, starter en ny enhet. Denne enheten varer enten til det kommer et nytt underkapittel, i et nytt «spor» (oppgaver for å trene videre på temaet i kapittelet), eller en kontekstoppgave oppstår.

3.4.3. Vertikal analyse

En vertikal analyse baserer seg på å få et mer fokusert og utdypet perspektiv på det matematiske innholdet (Charalambous et al., 2010, s. 122). Dette vil primært innebære hva som blir formidlet til eleven, hva som kreves av eleven, og sammenhengen til tidligere tilegnet kunnskap (Charalambous et al., 2010, s. 123). I den vertikale analysen vil oppgavene funnet i den horisontale analysen, bli analysert. Den vertikale analysen vil bestå av tre deler, basert på Bråting & Kilhamn (2022) sitt analytiske rammeverk.

3.4.4. Bakgrunn for vertikalt analyserammeverk

Benton et al. (2016, 2017) arbeidet med et forskningsprosjekt kalt ScratchMaths i over to år, i et forsøk på å bygge matematisk kunnskap gjennom programmering i Scratch (s. 26). Formålet med prosjektet var å introdusere en ny måte å arbeide med matematikk på, og gi lærere et verktøy for å se de ulike pedagogiske strategiene de kan bruke for å lære matematikk ved hjelp av programmering (Benton et al., 2016, s. 29). Disse pedagogiske strategiene gjorde de om til et rammeverk, som kalles de fem E-ene (5Es):

- Explore (Utforske) – Utforske ideer, prøve ut ting på egenhånd og feilsøke strukturelle og tekniske problemer når nødvendig. Oppfordre elevene til å prøve nye metoder for å løse problemer.
- Explain (Forklare) – Forklare hva elevene har lært, og begrunne valgene de har gjort underveis. Forklare hva deler av et program gjør, og hvorfor programmet er lagd som det er.
- Envisage (Forestille seg) – Forutse utfallet til et program før man kjører det. Istedenfor å kjøre et program for å se hva som skjer, bruker elevene tid på å forstå alt i programmet, for å kunne forestille seg hva som skjer.
- Exchange (Dele erfaringer) – Dele erfaringer med hverandre, for å se på et problem fra forskjellige perspektiver. Elever med høye ferdighetsnivåer kan lære bort til andre medelever, for at elevene kan få det forklart på en ny måte, samtidig som de erfarne elevene må artikulere tankegangen deres.
- bridgE (Knytte kunnskap) – Aktivitetene er designet på en måte at elevene får læringsutbytte i flere temaer, f.eks. i matematikk og programmering.

(Benton et al., 2016, 2017).

Brennan & Resnick (2012) er to av skaperne av BBP språket Scratch, og presenterer i sin studie et rammeverk om læringen av AT gjennom Scratch. Det er 3 dimensjoner som omhandler de ulike typene læringsutbytte som kan oppstå ved arbeid gjennom BBP:

1) AT konsepter (Konsepter som skaperne bruker når de programmerer)
2) AT øvinger (Øvingen designeren får når de programmerer)
3) AT perspektiver (Perspektivet designeren får av verden rundt dem, og dem selv)

(Brennan & Resnick, 2012, s. 3)

AT konsepter vil si konsepter som skaperne møter på i Scratch, som er like for andre programmeringsspråk. Konseptene de fremhever er sekvenser, løkker, parallellisering, hendelser, betingelser, operatør, og data (Brennan & Resnick, 2012, s. 3). Videre fordeler Brennan & Resnick (2012, s. 7) AT øvinger inn i fire deler; være inkrementell og iterativ, testing og feilsøking, gjenbruke og tilpasse, og abstrahering og modulisering. Den tredje dimensjonen omhandler perspektivene til designerne de har tilegnet seg gjennom intervju, og er derfor ikke like relevant for videre presentering.

3.4.5 Handlingsanalyse og konseptanalyse

Bråting & Kilhamn (2022) undersøkte i sin studie sammenhengen mellom matematikk og programmering. Der brukte de en kombinasjon av Brennan & Resnick (2012) sitt teoretiske rammeverk om algoritmisk tenkning i oppgaver, og Benton et al (2016, 2017) sitt teoretiske rammeverk med de fem E-ene i deres artikkel. Ettersom Brennan & Resnick (2012) sitt rammeverk er ment for tenkning, og Benton et al (2016, 2017) sine fem E-ene er designprinsipper, ønsket de å kombinere rammeverkene, for å lage et som passer til å kunne analysere programmeringsoppgaver i lærebøker (Bråting & Kilhamn, 2022, s. 598). For å kombinere disse fordelte de rammeverkene inn i to forskjellige deler, et for å se på *handlingene* oppgavene krever, og et for å se på hvilke matematiske og programmerings *konsepter* som er i oppgavene. Rammeverket for handlinger er som følger:

- **Følge** – Følge stegvise instrukser, repetere eller fortsette et mønster
- **Finne regel** – utarbeide prosedyren, regelen eller mønsteret som genererer et utfall, f.eks. en tallrekke.
- **Feilsøke** – Fikse en kode
- **Forme og skape** – Gi instrukser, skape et mønster, skrive kode, representere med symboler.
- **Forklare** – Forklare ved hjelp av naturlig språk, bruke ord som beskriver en prosedyre, en regel, et mønster eller et konsept.
- **Forestille seg** – Forutse hva som vil skje, reflektere over mulige utfall når vilkår og variabler blir endret

(Bråting & Kilhamn, 2022, s. 599)

Videre fordeler de konsepter inn i matematiske konsepter og konsepter innenfor algoritmisk tenkning. Bråting & Kilhamn (2022) definerer et konsept som matematisk hvis det er funnet i tradisjonell skolematematikk og formidler en viktig matematisk idé (s. 599). Videre definerer de et konsept som algoritmisk tenkning hvis konseptet bærer en spesifikk mening innenfor programmering (s. 600). Dette vil da tilsi konsepter som kode, løkke, vilkår og feilsøking. Enkelte ord har forskjellig mening i programmering og matematikk, f.eks. vilkår (Bråting & Kilhamn, 2021).

For valget av denne masteroppgavens algoritmiske konsepter ble det vurdert å bruke Bocconi et al. (2018, s. 22) sin liste over nøkkeltemaer brukt i lærerutdanningen innenfor programmering. Dette ble valgt bort primært for to grunner, der den ene er at artikkelen ble skrevet før LK20 ble implementert, og programmering kom inn i skolen. Videre omhandler Bråting & Kilhamns (2022) konseptliste flere begrep, som åpner opp til en dypere forståelse av hvilke konsepter oppgavene inneholder. Konseptene fremmet av Bråting & Kilhamns (2022) sin analyse er som følger:

Matematiske konsepter	Algoritmiske konsepter
Mønstre (inkludert nummer sekvenser)	Stegvise instrukser
Geometriske konsepter (Triangel, sirkel etc.)	Algoritme
Aritmetiske konsepter	Løkke, iterasjon, repetisjon
Rotasjon	Regel
Koordinater	Kode
	Vilkår
	Feilkode, feilsøking

Tabell 3.3 – Liste over matematiske- og programmeringskonsepter

Det siste steget Bråting & Kilhamn (2022) har i sin analyse er å se på sammenhengen mellom matematikk og programmering. Formålet med dette er å se om elevene blir spurt om å reformulere programmeringsideer ved hjelp av matematisk terminologi eller matematisk notasjon. Dette gjelder også for å utforske matematiske ideer gjennom programmering (s. 600). Det er ikke et spesifikt rammeverk for å se på sammenhengen mellom programmering og matematikk, men heller å se på sammenhengen mellom resultatene fra handlingsanalysen og konseptanalysen. Her vil annen relevant forskning nevnt i teoridelen av masteroppgaven være relevant, for å drøfte de ulike funnene og sammenhengen mellom matematikk og programmering.

Ved bruk av Bråting & Kilhamns (2022) rammeverk, vil jeg både få en mulighet til å se de ulike type programmeringsoppgaver som er i lærebøkene, samtidig som jeg får muligheten til å konkret se på hvilke matematiske konsepter som blir brukt i programmeringsoppgavene. Videre vil resultatene i Bråting & Kilhamns (2022) analyse være en referanse for drøftingen av resultatene, ved å ha et utgangspunkt i hvilke typer oppgaver andre læreverk har.

Videre har jeg også inkludert en *andre kommentarer* tabell, for å kunne gi kommentarer om f.eks. oppklarninger på siden, forklart i en annen oppgave osv. Ved å kunne komme med egne tanker om oppgaven er det mulig å oppdage mønstre som ikke blir fremhevet i det analytiske rammeverket, og kunne bruke dette som videre data(Cohen et al., 2018, s. 469).

3.5. Validitet og reliabilitet

Validitet deles ofte inn i to typer, indre validitet og ytre validitet (Postholm & Jacobsen, 2018, s. 223). Indre validitet baserer seg oftest på om det er sammenhengen mellom virkeligheten vi analyserer & teori, og om man kan kommentere årsak og virkning basert på studien man har gjennomført (Postholm & Jacobsen, 2018, s. 229). Ytre validitet baserer seg på om resultatet i forskningen kan passer for andre kontekster enn hva som ble gjort i selve forskningsstudiet, altså om de er overførbare (Postholm & Jacobsen, 2018, s. 238).

Denne avhandlingens indre validitet vil primært basere seg på hvordan oppgavene i Matemagisk serien blir presentert, og kategorisert. Ved å gi en tydelig beskrivelse av hva jeg definerer som programmeringsoppgaver, og se på oppgavene ved bruk av relevant teori, er formålet å gjøre funnene valide. Videre har oppgavene blitt gjennomgått flere ganger, for å passe på at jeg kan stå for markeringene av oppgavene, og for å få så nøyaktig data som mulig. Nyman et al. (2024) sin artikkel ble publisert i april, som var etter datainnsamlingen til analysen ble ferdig. Ettersom artikkelen presenterte et rammeverk som passet utmerket til denne masteravhandlingen, valgte jeg å gå igjennom dataen en gang til, for å implementere dette rammeverket inn i analysen. Ved å gjøre dette fikk jeg gått igjennom dataen med et nytt perspektiv, som støtter opp validiteten til oppgaven (Clark et al., 2021; Patton, 2015).

Overførbarheten, eller den ytre validiteten, til denne avhandlingen er også relevant å reflektere over. Ettersom jeg ser kun på en enkel bokserie, vil ikke mine funn nødvendigvis gjelde for andre lærebokserier. Samtidig er ikke formålet med denne masteroppgaven å se på hvilke type programmeringsoppgaver som er i Matemagisk 5-7, men heller å se på eksempler av den eventuelle brobyggingen mellom programmering og matematikk. Jeg kunne ha brukt flere ulike lærebokserier, og sammenlignet oppgavetypene mellom dem, men da ville jeg ikke ha fått en oversikt over de ulike temaene programmering kan eventuelt brukes i. Ved å heller

fokusere på enkelte oppgaver og innholdet til Matemagisk 5-7, vil det være mulig å få en dypere innsikt i hvilke matematiske temaer som inneholder programmering.

Reliabiliteten til et forskningsprosjekt er ment for å se om funnene i forskningsprosjektet kan gjenskapes av andre forskere, for å se om resultatet vil være likt i en senere tid (Kvale & Brinkmann, 2015, s. 276; Postholm & Jacobsen, 2018, s. 223). En av fordelene med å gjennomføre en tekstanalyse er at datamaterialet som forskningen er stabil, og vil ikke endres (Morgan, 2022, s. 66). Dette gjør at dataen samlet inn, altså oppgavene fra Matemagisk 5-7, vil være de samme, og vil ikke endres på. Videre har jeg forsøkt å beskrive analyseprosessen så nøyaktig som mulig, for at forskningen skal kunne være pålitelig.

4. Resultat og drøfting

I dette kapittelet vil resultatet fra analysen bli presentert og drøftet sammen med relevant teori. Dette blir gjort for å kunne presentere funnene, og kommentere på funnene underveis. Først vil resultatene fra den horisontale analysen bli presentert og drøftet. Det vil si at det totale antallet programmeringsoppgaver i lærebøkene vil bli presentert, og fordelingen av programmeringsinnholdet i lærebøkene. Samtidig vil jeg analysere enhetene, altså grupperinger av oppgaver, av bøkene bli analysert og drøftet, for å gjennomføre en relasjonsanalyse, basert på Nyman et al. (2024) sitt analytiske rammeverk.

Videre i den vertikale analysen vil resultatene bli analysert ved hjelp av Bråting & Kilhamn (2022) sitt analytiske rammeverk bli presentert, og drøftet. Dette vil bli gjort ved å først se en total oversikt over de ulike handlingene og konseptene i oppgavene, for å se på programmeringsinnholdet i lærebøkene. Videre vil disse funnene bli sett på sammen, for å se på brobyggingen (bridging) mellom programmeringen og matematikken. Til slutt vil eksempeloppgaver fra lærebøkene bli presentert, og drøftet.

4.1.Horisontal analyse

Gjennom den horisontale analysen, ble totalt 1298 oppgaver gått igjennom i grunnbøkene til Matemagisk 5-7. Av disse oppgavene var det 1277 oppgaver med matematiske elementer, og 88 oppgaver med programmeringselementer. Dersom oppgavene omhandlet AT eller BBP ble disse definert som å inneholde elementer av programmering. Dette ble gjort for å gjennomføre den vertikale analysen, på lik måte som Bråting & Kilhamn (2022) gjorde i sin undersøkelse. Videre vil mer detaljer om denne statistikken bli presentert.

4.1.1. Overordnet oversikt over oppgavene

I den overordnede oversikten vil først funnene fra den horisontale analysen bli presentert. Dette vil altså si hvor mange programmeringsoppgaver som var i lærebøkene, og hvor disse oppgavene er fordelt. Videre vil funnene fra relasjonsanalysen jeg har utført bli presentert, og drøftet.

Matemagisk 5 består av to ulike grunnbøker, A og B. Grunnbok A består av kapittel 0-4, og grunnbok B består av kapittel 5-10. I grunnbok A er det ingen oppgaver med programmeringselementer, noe som betyr at grunnbok A ikke vil være med videre i analysen. I grunnbok 5B er det flere enkeltoppgaver med programmeringselementer, og et kapittel er dedikert til introduksjon av programmering. Av de 260 oppgavene, er det 47 programmeringsoppgaver. Det vil si at i Matemagisk 5B er 18% som omhandler programmering. Total oversikt kan sees i tabell 4.1. Tabellen er fordelt inn i kapitler og temaer. I kolonnen «tema» er hovedkapittelet nevnt først, etterfulgt av alle underkapitler som har programmeringsoppgaver i seg. Underkapitlene er listet opp under hovedkapitlet. Videre er det totale antallet oppgaver, antall oppgaver som omhandler matematikk, og til slutt antall oppgaver som omhandler programmering. En oppgave kan både ha matematiske elementer, og programmering i seg. Ser man på kapittel 8, har ikke alle oppgaver matematiske elementer i seg, noe som betyr at antallet oppgaver med matematiske elementer, er lavere enn det totale antallet oppgaver. Dette gjelder for alle tabellene.

Kapittel	Underkapitler	Oppgaver	Matematikk	Programmering
Kapittel 5 Multiplikasjon, brøk og prosent	- Heltall ganget med brøk - Brøkdelen av et tall	51	51	4
Kapittel 6 Sannsynlighet	- Sannsynlighet	33	33	1
Kapittel 8 Programmering	- Tenke som en robot - Blokkprogrammering - Bruke løkker for å gjenta - Bruke variabler	42	21	42

Tabell 4.1 – Oversikt over programmeringsoppgaver i Matemagisk 5B

Matemagisk 6 har to grunnbøker, fordelt på A og B. Grunnbok A inkluderer kapitlene 0-4, og grunnbok B inkluderer kapitlene 5-9. I grunnbøkene er det totalt fem kapitler som har oppgaver med programmeringselementer, og et delkapittel dedikert til programmering. Totalt er det 409 oppgaver i Matemagisk 6, der 32 har programmering i seg, som vil si at 8% av oppgavene har programmering i seg. En oversikt over hvor mange oppgaver de enkelte kapitlene har kan sees i tabell 4.2:

Kapittel	Tema	Oppgaver	Matematikk	Programmering
Kapittel 0	Vårt Matematiske klasserom	1	1	1
Kapittel 1	Variabler og formler: Figurtall – Sammenhenger	28	28	8
Kapittel 3	Måling: Måleenheter	44	44	1
Kapittel 5	Vinkler og parallelle linjer: Vinkler som en del av en sirkel	39	39	1
Kapittel 7	Geometriske mønstre: Å utforske mønstre med programmering	48	48	21

Tabell 4.2 – Oversikt over programmeringsoppgaver i Matemagisk 6A & 6B

Matemagisk 7 har også to grunnbøker fordelt på A & B. Bøkene til trinnet har ingen enkelt- eller delkapitler dedikert til programmering, der det er kun 9 oppgaver som omhandler programmering. Dette vil altså si at av 393 totale oppgaver, er det kun 2% som omhandler programmering. Programmeringsoppgavene i bøkene er heller enkeltoppgaver, fordelt over tre ulike kapitler. Den totale oversikten kan sees i tabellen under:

Kapittel	Tema	Oppgaver	Matematikk	Programmering
Kapittel 1	Statistikk: Sentralmål	46	46	2
Kapittel 3	Brøk, desimaltall og prosent: Tall med mange desimaler	63	63	6
Kapittel 5	Negative tall: Å legge til og trekke fra negative tall	57	57	1

Tabell 4.3 – Oversikt over programmeringsoppgaver i Matemagisk 7A & 7B

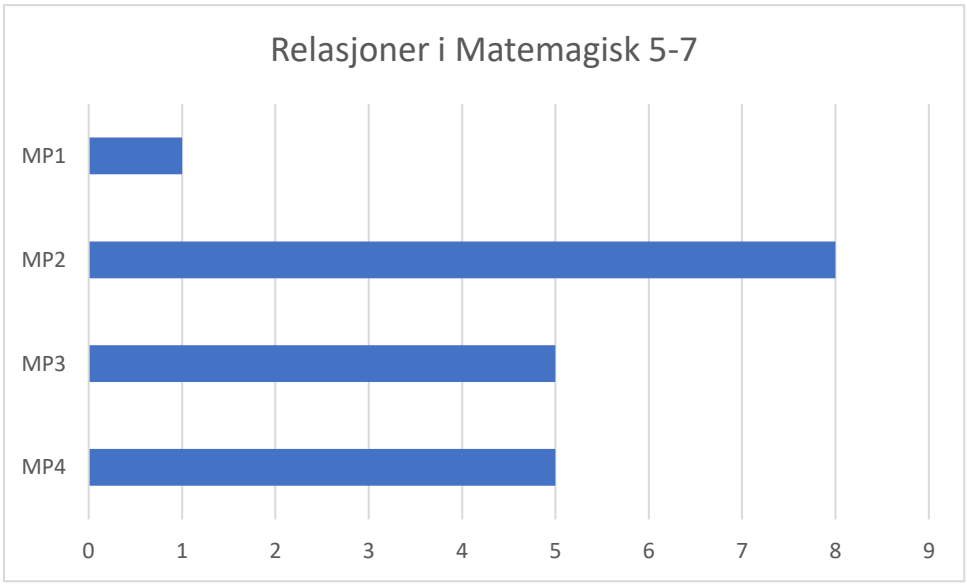
I innledningen til denne masteravhandlingen ble enkelte kompetansemål nevnt, blant annet «Bruke programmering til å utforske data i tabeller og datasett» (Kunnskapsdepartementet, 2019). I oppgaveoversikten i tabell 4.3 er det ingen dedikerte del- eller helkapitler dedikert til programmering, som det har vært på de tidligere trinnene. Dette vil originalt tyde til at Matemagisk 7 ikke oppfyller kompetansemålet om programmering. Men Matemagisk 7 har et kapittel som omhandler bruken av regneark, og utforske data i tabeller og datasett ved hjelp av regneark. Det kan argumenteres for at kompetansemålet blir oppnådd gjennom bruken av Excel, men det er et annet kompetansemål som omhandler bruken av regneark (Kunnskapsdepartementet, 2019). I delkapittel 2.2.1 ble det konkludert at Excel ikke ble regnet som programmering, ettersom det krever oppgaver som er spesifikt lagd som å løses gjennom programmering (Nagy et al., 2021), noe oppgavene i Matemagisk 7 ikke er.

Mange av Excel oppgavene omhandler enten å skrive ulike formler inn i Excel, og lese av verdier fra regneark. Oppgavene fremmer ikke bruk av *dekomponering*, *algoritmer*, eller andre sentrale aspekter av programmering. I seg selv vil

Samtidig må en også lure på hvordan Matemagisk 7 skal lage programmeringsoppgaver som utforsker data i tabeller og datasett. Å lage denne type oppgave for aldersgruppen er vanskelig (Munthe, 2022, s. 13), der matematikken fort blir nedprioritert (Nyman et al., 2024). Det skal sies at en lærer kan velge å finne oppgaver som hjelper med å oppfylle kompetansemålet. Men jeg ble fortsatt overrasket, ettersom Matemagisk 5-7 inneholder flere programmeringsoppgaver til ulike temaer.

4.1.2. Relasjoner mellom matematikk og programmering

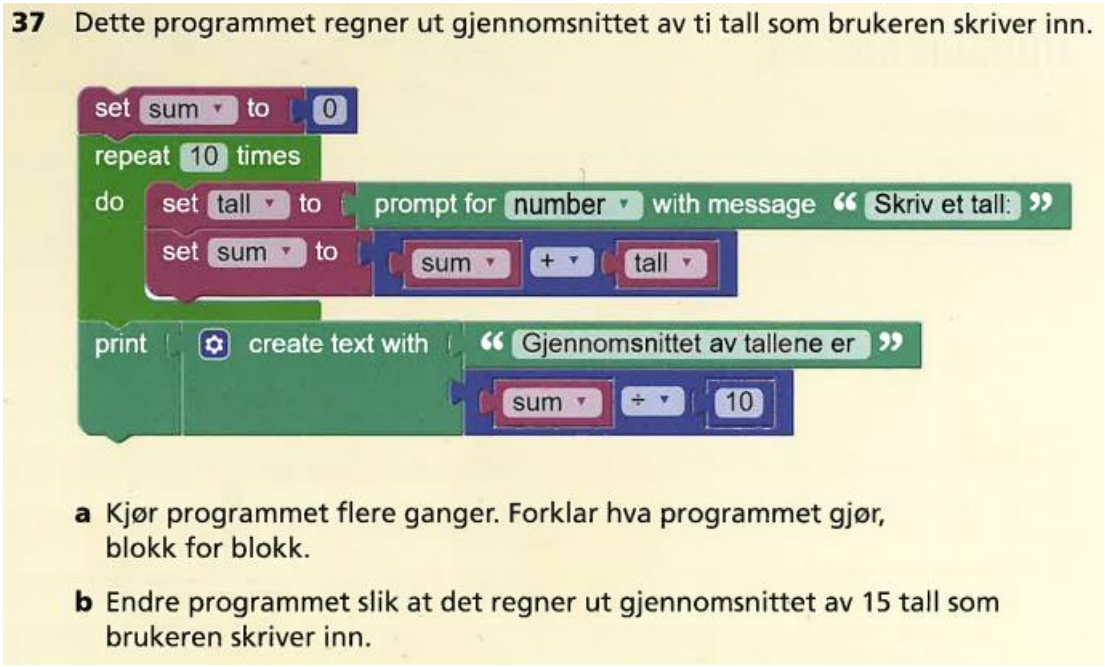
I relasjonsanalysen blir graden av matematisk (M) i programmeringsoppgaver (P) sett på, og rangert, ut ifra Nyman et al. (2024, s.7) relasjonsrammeverk. Rammeverket inkluderer fire ulike relasjoner, som blir kalt for MP1 til MP4. MP1 vil si at det ikke er matematikk i programmeringsoppgavene, MP2 er det ikke relevant matematikk for elevgruppen, MP3 der programmeringen blir brukt for å regne matematikk mer effektivt, og til slutt MP4, der programmering blir brukt for å utforske matematikk. Totalt var det 19 enheter som hadde programmering i seg. Resultatet fra relasjonsanalysen kan sees i figur 4.1:



Figur 4.1 – Relasjoner i Matemagisk 5-7

Relasjonene funnet i min analyse avviker noe fra Nyman et al. (2024) sine funn. I deres funn var det desidert flest enheter som bruker matematikk i en kontekst for å lære programmering, med enkelte MP1 og MP4 enheter, og få MP3 enheter. En forklaring på dette kan være måten enheter blir fremstilt på, nemlig som enheter (oppgaver som henger sammen). Enhetene Nyman et al (2024, s. 6) var om lag 1-3 sider, mens enhetene jeg identifiserte var på 2-6 sider, som gjør enhetene mine lengre. I de fleste enheter var det enkeltoppgaver som kun fokuserte på programmering, noe som ville blitt kategorisert som MP1. Men fordi enhetene ofte hadde oppgaver med noe matematikk, ble enheten kategorisert som MP2 eller høyere.

Sett bort fra avvikene, er jeg overrasket over fåtallet av enheter innenfor kategorien MP1. Sett bort ifra refleksjonen rundt avvikene med Nyman et al. (2024) sine funn, var det bare en enhet som var MP1. Denne enheten var i det første programmeringskapittelet i Matemagisk 5B, som vil altså si første gangen elevene blir møtt med programmering. Det skal sies at alle MP1 enheter som Nyman et al. (2024, s.9) fant i sin undersøkelse, stammet fra bøker laget spesifikt for programmering. Videre var det også vanskelig å analysere enkelte oppgaver, som f.eks. oppgaven i figur 4.2:



Figur 4.2 – Gjennomsnitt i programmering. Fra *Matemagisk 7A: Grunnbok* (s. 51), av A. L. Kongsnes et al., 2021c, Aschehoug Undervisning.

I figur 4.2 oppsto et problem med kategoriseringen av MP. Programmet ber brukeren om å skrive in et tall, 10 ganger. Når dette er gjort 10 ganger, vil programmet vise gjennomsnittet av tallene. Isolerer man oppgaven kan man tenke at dette er en fin måte for eleven å lære gjennomsnitt, samtidig som dette øker forståelsen av programmering for eleven. Men oppgaven er i «terrengløypa», og oppgave 37 i temaet statistikk. En kan anta at eleven som møter på denne oppgaven har en grunnleggende forståelse av hva programmering er, etter å ha gjort 36 tidligere oppgaver om dette temaet.

Sett bort fra plasseringen av oppgaven, er det også annen problematikk som dukker opp. I øverste del av teksten får eleven beskrevet hva programmet gjør, og får i deloppgave a) beskjed om å forklare hva programmet gjør, «blokk for blokk». En observant elev kan se på beskrivelsen av programmet og gjette seg frem til hva hver blokk i programmet gjør. Videre i deloppgave b) skal eleven endre programmet, slik at den regner ut gjennomsnittet av 15 tall. Denne endringen kan gjøres ved å finne alle tallene der det står 10, og skrive det om til å bli 15.

Med beskrivelsene over kan det tenkes at oppgaven har mest fokus på programmering. Men ser man tilbake på rammeverket til Nyman et al. (2024, s. 7), blir det vanskelig å kategorisere den. Selv om nivået på matematikken er lavt, basert på plasseringen av oppgaven, så er matematikken i oppgaven relevant for aldersgruppen. Samtidig er det tydelig at i oppgaveteksten er fokuset på programmeringen, og å forstå programmet. Dette vil si at oppgaven passer i MP4 fordi matematikken er relevant for aldersgruppen, men det matematiske målet er ikke presisert. Samtidig kan den passe i MP2, der oppgaveteksten er rettet mot å lære programmering, men det matematiske innholdet er relevant til alderstrinnet. Jeg har valgt å kategorisere denne oppgaven som MP2, ettersom oppgaven har matematikk som er *relevant* for trinnet, betyr ikke nødvendigvis dette at det matematiske nivået er bra nok.

4.1.3. Oppsummering horisontal analyse

Med utgangspunkt i dataen presentert i oppsummeringen av den horisontale analysen, inneholder rundt 7% av alle oppgavene i Matemagisk 5-7 elementer av programmering. Dette inkluderer også oppgaver som omhandler AT. Når jeg først så denne prosentdelen syntes jeg at det var litt lite, ettersom hvor stor del AT har fått i LK20 (Utdanningsdirektoratet, 2019). Videre er prosentandelen av kompetansemål i LK20 (Utdanningsdirektoratet, 2019) som omhandler programmering eller regneark rundt 17%, noe som også får det er få programmeringsoppgaver. Men tenker man på arbeidsmengden som går inn i en programmeringsoppgave i motsetning til f.eks. en aritmetisk oppgave, er det mer grunnlag for at fordelingen er jevn. Videre har Matemagisk også oppgavebøker samt nettressurser, som kan inneholde flere programmeringsoppgaver enn i grunnboken.

Gjennom Matemagisk 5-7 er flertallet av programmeringsoppgavene kondensert i enkelte kapitler gjennom bøkene. Man kan anse dette som et problem med tanke på at programmering læres best gjennom jevnt arbeid (Stigberg & Stigberg, 2020). Samtidig er det naturlig at enkelte matematiske emner er fordelt inn i enkelte kapitler, der programmering blir ansett som et matematisk emne. Videre er det enkelte programmeringsoppgaver i de fleste kapitlene, noe som også fremmer arbeidet over tid, og med forskjellige matematiske temaer. Spørsmålet blir da om man ser på programmering som et emne innenfor matematikken som man skal igjennom, eller som en alternativ metode for å løse oppgaver.

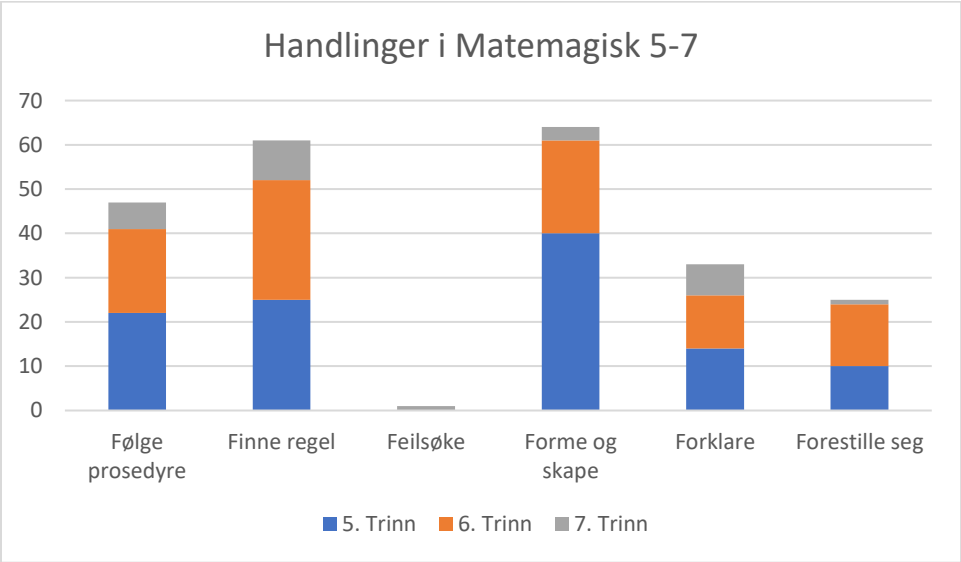
Det var interessant å oppdage hvor mange ulike temaer som inneholdt programmeringsoppgaver. De fleste av oppgavene omhandler geometri og utforskning av figurer, som er ganske vanlig i arbeid med programmering på dette nivået (Bråting & Kilhamn, 2022). Programmering oppstår innenfor blant annet statistikk, brøk negative tall etc. Man kan argumentere for at oppgavene blir kun brukt for å ha programmering med, der det egentlig trengs (Munthe, 2022). Dette kan sees i enkelte av enhetene, der matematikken kun er en kontekst for å arbeide med programmeringen, og bidrar ikke til relevant matematisk læringsutbytte (Nyman et al., 2024). Hvordan det matematiske innholdet blir presentert i programmeringsoppgavene, vil videre bli sett på i den vertikale analysen.

4.2.Vertikal analyse og drøfting

Å gjennomføre en vertikal analyse gir muligheten til å få en mer fokusert og grundig analyse av det matematiske innholdet (Charalambous et al., 2010, s. 122). Den vertikale analysen ble gjennomført vet bruk av Bråting & Kilhamns (2022) handlings- og konseptrammeverk. Handlingene og relasjonene i programmeringsoppgavene i Matemagisk 5-7 vil bli analysert, og drøftet med relevant teori. Videre vil også handlingene og konseptene bli sett på i lys av hverandre, for å se på brobyggingen mellom programmering og matematikk.

4.2.1. Handlingsanalyse

I handlingsanalysen ble Bråting & Kilhamns (2022) sitt rammeverk brukt, for å se hvilke handlinger oppgavene krever at elevene skal gjøre. Handlingsanalysen ble gjennomført med bakgrunn av oppgavene fremhevet gjennom den horisontale analysen. Totalt ble det funnet 47 oppgaver i *følge*, 61 i *finne regel*, 1 *feilsøke*, 64 *forme og skape*, 33 *forklare* og 25 *forestille seg*. Funnene fra handlingsanalysen kan sees i tabell 4.6:



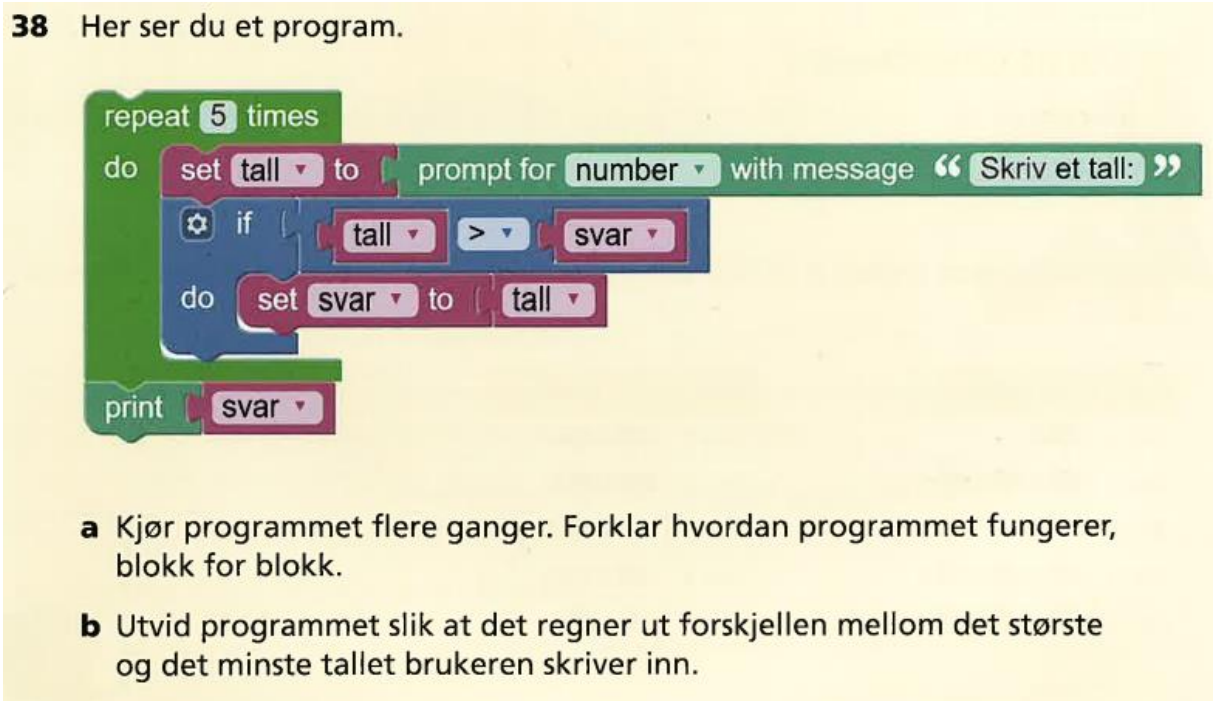
Figur 4.3 – Handlinger i Matemagisk 5-7

Av alle handlingene som elevene ble spurt om å gjennomføre, var «forme og skape», «finne regel» og *følge prosedyre* de vanligste. I 5. trinn ba de fleste oppgavene eleven om å både «forme og skape» og *følge prosedyre.* Dette ble det mindre av i de senere trinnene, der elevene ble bedt om å enten «forme og skape» eller *følge prosedyre*, og ikke begge. Dette kan vise til at når elevene først lærte seg det grunnleggende rundt å lage kode i Trinket, så får

elevene mer frihet til å dele opp problemet inn i mindre biter, og forstå hvordan programmet skal bygges opp (Sevik, 2016; Wing, 2006). Når programmet da blir laget, må eleven forstå matematikken, for å få riktig utfall fra programmet (Benton et al., 2018).

Å «forklare» og «forestille seg» var i rundt en tredjedel av oppgavene. Ofte var disse sammen, der elevene ble spurt om å forestille seg et utfall i programmet, og senere forklare hvordan utfallet oppsto. I de oppgavene der elevene skulle «forklare» men ikke «forestille seg», ble elevene bedt om å endre på verdiene, og forklare hvordan dette endret programmet. Mye av «forklaringen» la til rette for at elevene skulle se sammenhengen mellom matematikken, og verdiene i programmet. Ved å tydeliggjøre det matematiske i programmet, vil eleven potensielt forstå det matematiske læringsutbytte (Nyman et al., 2024; Stigberg & Stigberg, 2020)

Totalt i Matemagisk 5-7 var det én oppgave med feilsøking. Det nærmeste andre oppgaver kom til feilsøking var å forbedre kode, men dette har jeg heller tolket som «forme og skape». Oppgaven i 7. trinn som hadde feilsøking kan sees i figur 4.4:



Figur 4.4 – Oppgave 38 Kapittel 1. Fra *Matemagisk 7A: Grunnbok* (s. 51), av A. L. Kongsnes et al., 2021c, Aschehoug Undervisning

Oppgaven i figur 4.4 er i kapitellet Statistikk, innenfor temaet sentralmål. I programmet blir eleven spurt om å skrive inn et tall, og etter å ha gjort dette fem ganger vil programmet printe det største tallet skrevet. Ved første øyekast virker programmet normalt. Men når elevene lager og kjører programmet vil programmet levere en feilmelding. Denne feilmeldingen skyldes at variabelen «svar» ikke er definert. Ettersom variabelen «svar» ikke har en tall-verdi, vil det være umulig for programmet å sammenligne om elevens input er større enn tallet i variabelen «svar». Hvis eleven definerer variabelen i starten av programmet, noe som blir gjort i en annen oppgave på samme side (se Figur 4.2), vil programmet fungere.

Ser man på oppgaven i figur 4.4, er dette den eneste oppgaven der handlingen feilsøking forekommer. Selv om oppgaven ikke spør eleven direkte om feilsøking, minner dette om måten Kaufmann & Stenseth (2021, s. 1042) brukte feilsøking på i deres studie. Oppgaven i figur 4.4 ber eleven om å forklare hvordan programmet fungerer, istedenfor å spesifikt be eleven om å se etter feil. Altså vil elevene se feilen gjennom å forklare hvordan programmet fungerer. Denne formuleringen gjør at eleven både må forklare hvorfor programmet er feil, i tillegg til å konkretisere hva resultatet av programmet blir. Dette gjør at eleven stimulerer elevens evne til å feilsøke, og testing av hypoteser (Kaufmann & Stenseth, 2021, s. 1042).

Samtidig er det viktig å huske av Kaufmann & Stenseth (2021) sin studie handlet om å se hvordan elever samarbeidet for å løse et programmeringsproblem, der oppgaven var designet for å aktivt feilsøke, og kontinuerlig forbedre programmet. Oppgaven i figur 4.4 mangler kun én definert variabel. Syntaksen i koden er riktig, og rekkefølgen er korrekt. Dermed er det en mulighet at elevene kan tolke den eneste mangelen i koden som en slurvefeil. Dette blir et problem, ettersom målet i oppgaven må være tydelig for å se sammenhengen mellom programmering og matematikk (Stigberg & Stigberg, 2020, s. 493) For å fremheve handlingen «feilsøke» mer, kunne forfatterne av Matemagisk 5-7 ha flere oppgaver som anerkjente at det var en feil i koden, eller at elevene måtte rette opp i matematikken i programmet for å fikse koden.

Ettersom feilsøking er en sentral del av programmering og AT (Brennan & Resnick, 2012, s. 7), er det overraskende at det kun er en oppgave som har feilsøking. Et naturlig argument er at elevene skal lære matematikk, og ikke programmering. Men ser vi tilbake på oppgaven Kaufmann & Stenseth (2021) brukte i sin undersøkelse, var feilsøkingen en del av den matematiske forståelsen. For å kunne lage programmet riktig, måtte elevene forstå matematikken først. Det kunne ha vært oppgaver i Matemagisk som nettopp krevde at elevene forsto matematikken, for at programmet skulle fungere. Feilsøkingen elevene må gjøre i denne oppgaven krever ikke at elevene forstår matematikken bak oppgaven, men heller at variabler må bli definert i et program.

Ved sammenligning av Bråting & Kilhamn (2022) sine funn er det flere forskjeller på antall oppgaver som har enkelte handlinger. Matemagisk 5-7 har flere oppgaver som krever at elevene skal forestille seg et utfall. Videre er det flere oppgaver innenfor å finne regel og forklare. Samtidig er det en relativt lik mengde med feilsøking og følge en prosedyre.

Det er flere ulike grunner til at det er flere avvik mellom mine og Bråting & Kilhamn (2022) sine funn. En av mulighetene til dette avviket kan ha oppstått er at jeg har tolket rammeverket på en annen måte enn hva som var tiltenkt av forfatterne. Dette kan blant annet ha oppstått i handlingene «følge» og «forme og skape», der grensen mellom å få stegvise instrukser i å lage et program, i motsetning til å forme og skape et program, kan ansees som uklar. Samtidig ble oppgavene med handlingen «forestille seg» lettere å kategorisere, fordi oppgaven konkret ba elevene om å forestille seg et utfall eller en situasjon. Bråting & Kilhamn (2022) sin studie tar for seg svenske lærebøker, som har andre kompetansemål enn de norske lærebøkene, som kan ha en innvirkning på de forskjellige resultatene fra min og Bråting & Kilhamn (2022) sine funn.

4.2.2. Konseptsanalyse

I konseptanalysen ble de ulike matematiske og programmeringskonseptene i oppgavene sett på. Et matematisk konsept er et tema som ofte oppstår innenfor matematikk på dette nivået. Et programmeringskonsept er sentrale aspekter av programmering, og AT. Oppgavene sett på er de samme som i handlingsanalysen, og ble gått igjennom flere ganger for å sikre at antallet oppgaver var korrekt. Resultat fra konseptanalysen kan sees i tabell 4.4 og 4.5:

Matematiske konsepter	5. Trinn	6. Trinn	7. Trinn
Mønstre	36 (77%)	31 (97%)	7 (78%)
Geometriske konsepter	31 (66%)	31 (97%)	0 (0%)
Aritmetiske konsepter	9 (19%)	24 (75%)	9 (100%)
Rotering	12 (26%)	20 (63%)	0 (0%)
Koordinater	0 (0%)	1 (3%)	0 (0%)

Tabell 4.4 – Oversikt over matematiske konsepter i Matemagisk 5-7

Av de matematiske konseptene er det *Mønstre* og *Geometriske konsepter* som er de mest vanlige i Matemagisk bøkene. Dette er samsvarer med Bråting & Kilhamns (2022, s. 603) forskning, der mønstre var det mest vanlige konseptet, på de yngre trinnene. Samtidig er det flere konsepter som er i få oppgaver, som *Rotering* og *Koordinater*. Noe som også overrasker, er hvor få matematiske konsepter som er i 7. trinn. De to eneste konseptene i oppgavene, er *Mønstre* og *Aritmetiske konsepter*. En grunn for dette kan være at Matemagisk 5&6 har dedikerte programmeringsoppgaver, som omhandler geometri. Matemagisk 7 har ingen dedikerte programmeringskapitler, der programmeringsoppgavene som oppstår er mer et supplement til temaene.

Videre er det markert kun én oppgave innenfor koordinater. Oppgaven baserer seg på å flytte «skilpadden» basert på x og y koordinater, og ber eleven om å tenke som et rutenett. Utenom i enkelte sideaktiviteter i bøkene var det ingen programmeringsoppgaver som omhandlet tradisjonelle koordinater. Videre er det også enkelte programmeringskonsepter som heller ikke har stor representasjon i oppgavene:

Programmeringskonsepter	5. Trinn	6. Trinn	7. Trinn
Stegvise instrukser	23 (62%)	23 (72%)	8 (89%)
Algoritme	23 (49%)	32 (100%)	9 (100%)
Løkke, iterasjon, repetisjon	36 (77%)	24 (75%)	8 (89%)
Regel	20 (43%)	17 (53%)	7 (78%)
Kode	27 (57%)	20 (63%)	3 (34%)
Betingelse	10 (21%)	8 (25%)	4 (45%)
Feilsøking	0 (0%)	0 (0%)	1 (11%)

Tabell 4.5 – Oversikt over programmeringskonsepter i Matemagisk 5-7

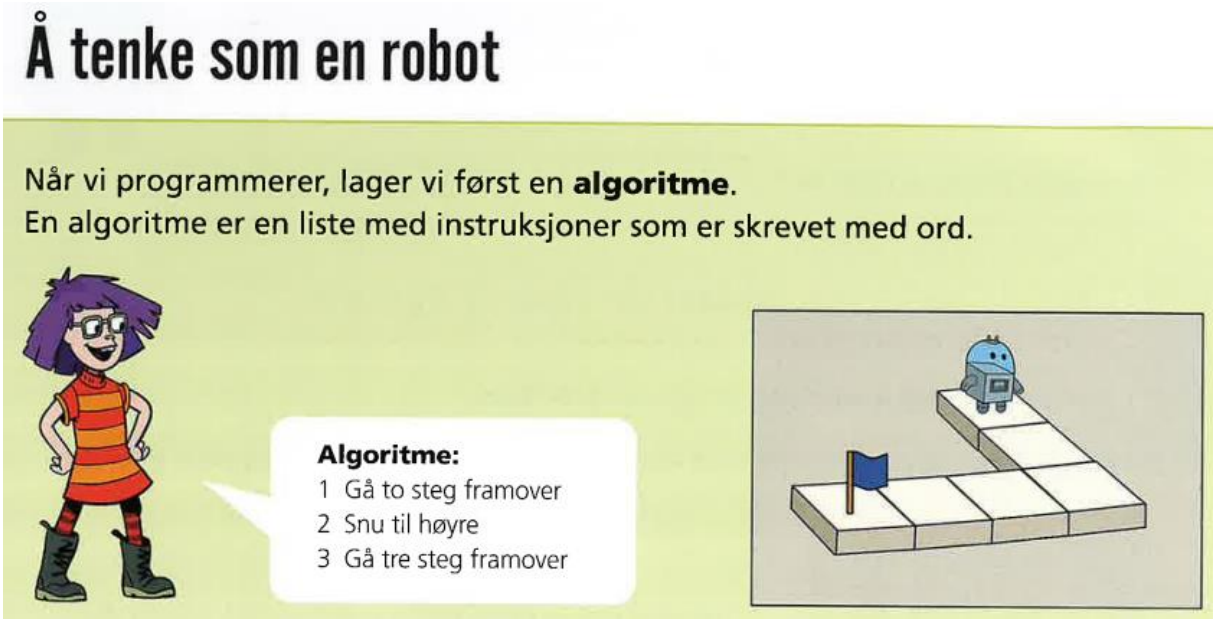
Gjennom programmeringsoppgavene er de fleste begrepene representert. Totalt var det en oppgave som inneholdt feilsøking, og dette er oppgaven fremhevet i figur 4.8. Videre var Algoritme kun i halvparten av oppgavene i 5. trinn, men økte til 100% i 6.- og 7. trinn. Dette kan skyldes at mange av oppgavene i 5. trinn handler om å introdusere programmering som et konsept, og at når elevene forstår dette, vil det være mer naturlig å ha algoritme i hver oppgave.

Basert på prosentdel av konsepter, fant jeg både flere matematiske og programmeringskonsepter i min analyse, enn hva Bråting & Kilhamn (2022) fant i deres, der det var kun 211 av 327 programmeringsoppgaver som inneholdt matematiske konsepter (Bråting & Kilhamn, 2022, s. 603). Dette kan komme av at programmering er tydelig knyttet til arbeid med AT, noe det ikke er i Sverige, som deres studie stammer fra (Bråting & Kilhamn, 2022). Dette kan sees tydelig i programmeringskonseptet algoritme, som er i alle oppgavene etter 5. trinn, og nesten i halvparten av oppgavene i 5. trinn. Samtidig har Matemagisk 5-7 flere programmeringsoppgaver som omhandler mer enn hva som er kompetansemålene i LK20 (Kunnskapsdepartementet, 2019).

4.2.3. Begreper og utrykk

En vertikal analyse omhandler hvordan læreboken fremstiller definisjoner til eleven (Charalambous et al., 2010, s. 123), noe som gjør det relevant å se på hvordan Matemagisk bøkene formidler begrep og utrykk til eleven. Dette gjelder spesielt programmering i matematikk, der det er flere tilfeller begrep og utrykk som kan være like, men ha ulik betydning (Bråting & Kilhamn, 2021). Derfor syntes jeg at dette kan være interessant å se videre på. Dette delkapittelet vil derfor se nøyere på hvordan Matemagisk presenterer ulike begreper og utrykk, til elevene.

I introduksjonen av kapittel 8 blir eleven møtt av konseptet av «å tenke som en robot». Elevene blir møtt av en robot, og skal tenke over hvordan den beveger seg til målet. Videre fremhever de ordet algoritme, og gir en beskrivelse av hva det ordet betyr:



Figur 4.5 – Å tenke som en robot. Fra *Matemagisk 5B: Grunnbok* (s. 97), av K. M. Raen et al., 2020, Aschehoug Undervisning

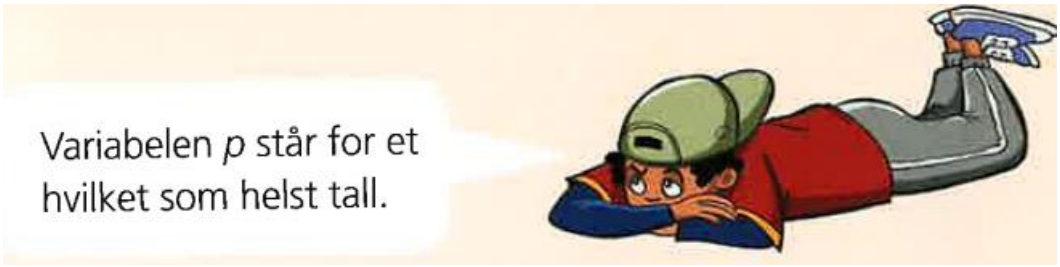
I figur 4.5 blir elevene introdusert til å tenke som en robot, før deres første møte med programmering. Forfatterne forklarer at for å lage et program, lager man først en algoritme, som er et sett med instrukser skrevet med ord. Raen et al. (2020). Ordet algoritme blir presentert innenfor konseptet *stegvise instrukser*. Dette snevrer begrepet algoritme til å kun følge instrukser, istedenfor å fremheve begrep som *dekomposisjon* og *abstraksjon* (Bråting & Kilhamn, 2022; Lodi & Martini, 2021). Dette snevrer inn bruken av AT, der det ikke blir tydeliggjort at problemet kan brytes ned i mindre biter, men heller å gi nøye instrukser.

Videre er begrepet algoritme knyttet til en programmeringskontekst, istedenfor en matematisk kontekst. Der algoritme vanligvis er en matematisk ide for å effektivisere kalkuleringer basert på f.eks. plassverdisystemer, blir en algoritme et programmeringskonsept ved å gi stegvise instrukser som kode (Bråting & Kilhamn, 2022, s. 606). Ordet algoritme blir altså prioritert brukt for å følge *stegvise instrukser*, istedenfor å bygge videre på matematiske ideer. Videre er også AT utbytte lavt, der order ikke inkluderer bruk som *dekomponering* og *abstraksjon* (Bråting & Kilhamn, 2022; Lodi & Martini, 2021).

Et annet problem Bråting & Kilhamn (2021, s. 179) fremmer som et problem med integreringen av programmering inn i matematikk, er ordet *variabel*. I figur 4.6 blir variabel definert i Matemagisk 5B, innenfor programmering. Dette er elevens første møte med utrykket variabel. Variabel blir da definert som noe man lagrer tall eller tekst i, og kan gjøre endringer i flere tall, samtidig.

Figur 4.6 – Variabel i programmering. Fra *Matemagisk 5B: Grunnbok* (s. 105), av K. M. Raen et al., 2020, Aschehoug Undervisning.

Videre, i Matemagisk 6A blir variabel definert på nytt. Denne definisjonen er innenfor temaet variabler og formler, og omhandler å finne et tallmønster. Definisjonen kan sees i figur 4.7, der variabel blir definert som et hvilket som helst tall:

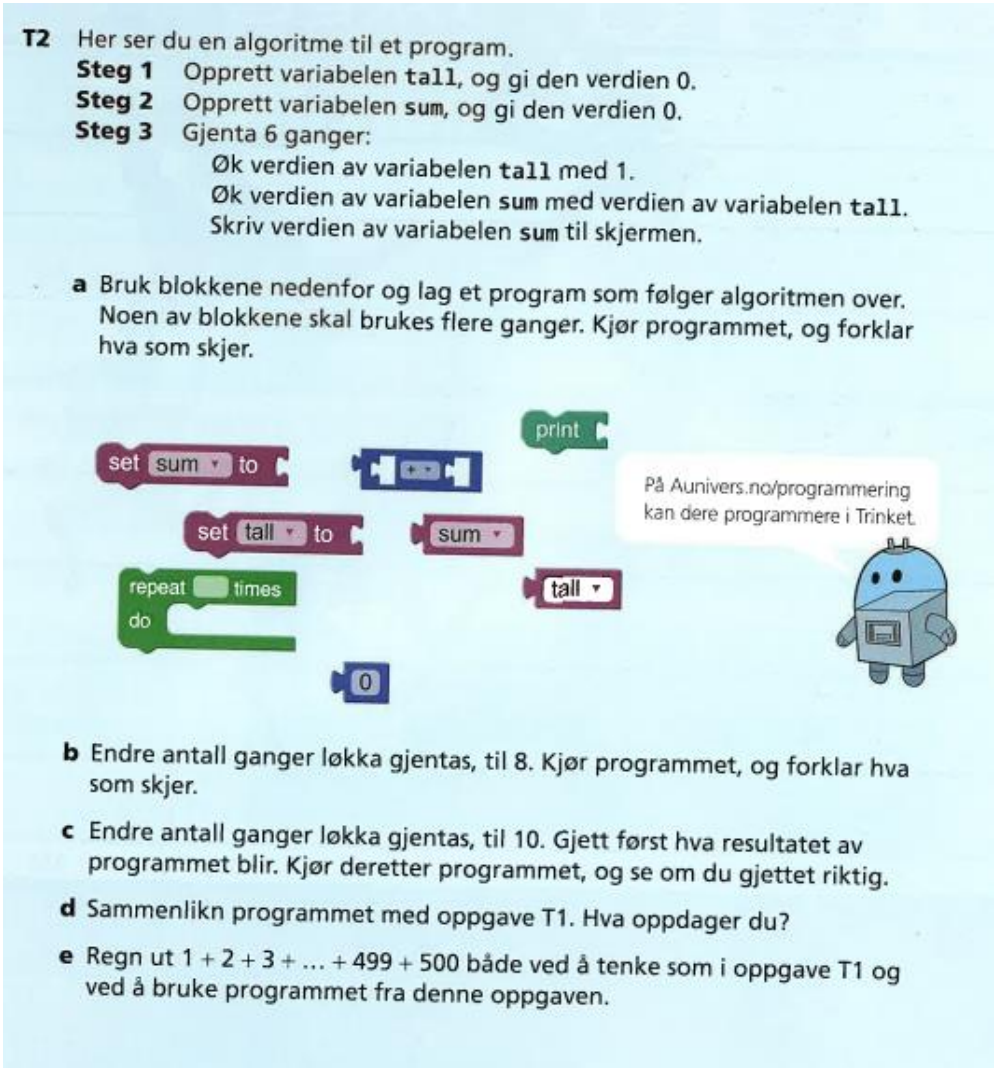


Figur 4.7 – Variabel i matematikk. Fra *Matemagisk 6A: Grunnbok* (s. 20), av K. M. Raen & A. L. Kongsnes, 2021a, Aschehoug Undervisning

Ordet variabel har en ulik betydning mellom matematikk og programmering (Bråting & Kilhamn, 2021), noe som kan også sees i eksemplene over. I figur 4.7 fremhever Raen et al. (2020) at variabel kan gjøre endringer i flere tall samtidig, altså underveis i programmet. Dette skiller seg fra variabel i en matematisk kontekst, der begrepet er ment til å være en ukjent, og en plassholder for et tall (Bråting & Kilhamn, 2021). Selv om eksemplene er i ulike bøker og klassetrinn, kan dette være med på å skape forvirring (Bråting & Kilhamn, 2021). Denne utydeligheten har en mulighet til å gjøre det forvirrende å bytte mellom programmering og matematikk, og gjøre det endelige læringsmålet utydelig (Nyman et al., 2024; Stigberg & Stigberg, 2020).

4.2.4. Dypdykk i enkelte oppgaver

Med overordnede funn presentert, og innhold som ikke er direkte i oppgavene fremhevet, ønsker jeg nå å se nøyere på enkelte oppgaver. Oppgavene plukket ut er på forskjellige trinn, og i forskjellige temaer. Den første oppgaven kan sees i figur 4.8:



T2 Her ser du en algoritme til et program.

Steg 1 Opprett variabelen `tall`, og gi den verdien 0.
Steg 2 Opprett variabelen `sum`, og gi den verdien 0.
Steg 3 Gjenta 6 ganger:
Øk verdien av variabelen `tall` med 1.
Øk verdien av variabelen `sum` med verdien av variabelen `tall`.
Skriv verdien av variabelen `sum` til skjermen.

a Bruk blokkene nedenfor og lag et program som følger algoritmen over. Noen av blokkene skal brukes flere ganger. Kjør programmet, og forklar hva som skjer.

b Endre antall ganger løkka gjentas, til 8. Kjør programmet, og forklar hva som skjer.

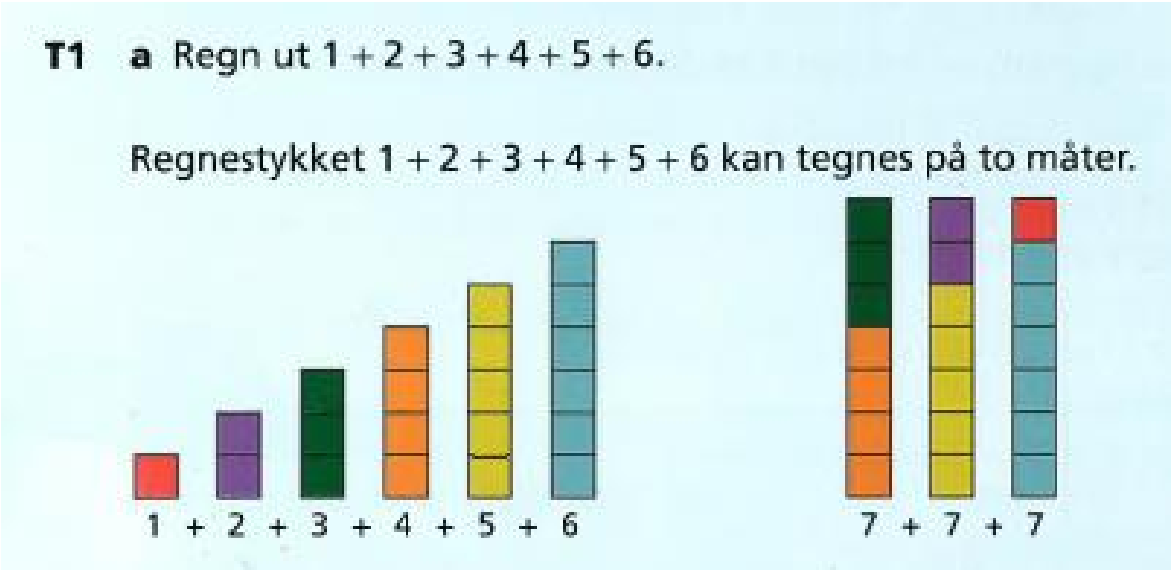
c Endre antall ganger løkka gjentas, til 10. Gjett først hva resultatet av programmet blir. Kjør deretter programmet, og se om du gjettet riktig.

d Sammenlikn programmet med oppgave T1. Hva oppdager du?

e Regn ut $1 + 2 + 3 + \ldots + 499 + 500$ både ved å tenke som i oppgave T1 og ved å bruke programmet fra denne oppgaven.

Figur 4.8 – Topptur 2 – Variabler og formler. Fra *Matemagisk 6A: Grunnbok* (s. 35), av K. M. Raen & A. L. Kongsnes, 2021a, Aschehoug Undervisning

Oppgaven i figur 4.8 er den siste oppgaven i kapittelet «Variabler og formler». Elevene blir presentert med en algoritme, og skal lage et program som utfører algoritmen. Videre skal eleven forklare tallmønstret $\frac{n}{2} * (n+1)$ når programmet kjører 6, og 10 ganger. Eleven blir så spurt om å sammenligne med tankegangen gjort i T1, der eleven tenker på regnestykker som blokker, og blir introdusert til den tidligere nevnte formelen. Til slutt skal eleven regne ut talrekken opp til 500 gjennom programmet og den tidligere tankegangen.



Figur 4.9 – Topptur 1 – Variabler og formler. Fra *Matemagisk 6A: Grunnbok* (s. 34), av K. M. Raen & A. L. Kongsnes, 2021a, Aschehoug Undervisning

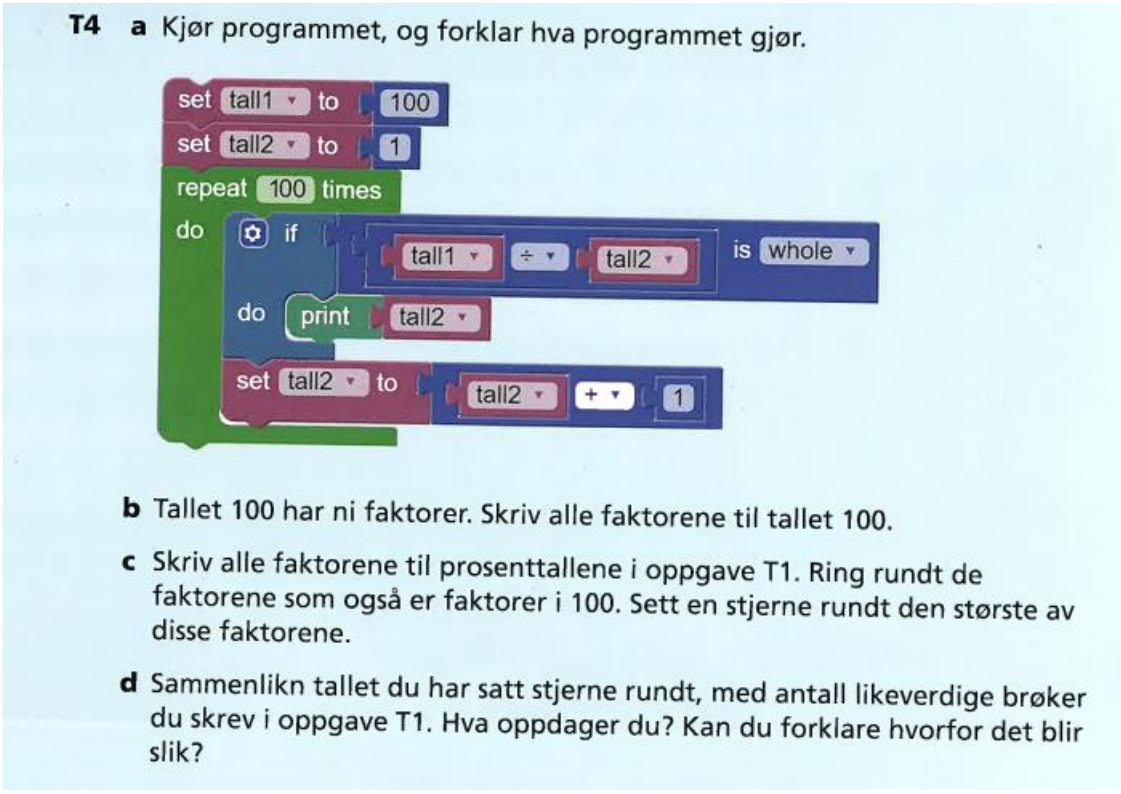
I oppgaven i figur 4.8 skal elevene sammenligne tankegang med oppgave T1, som kan sees i figur 4.9. I oppgave T1 skal elevene regne regnestykker, ved hjelp av klosser. Oppgavene videre omhandler større regnestykker, hvordan utregningene fungerer, og formelen til utregningene.

Tidligere ble oppgaven i Figur 4.2 kritisert for å simplifisere matematikken, og prioritere programmering ovenfor det potensielle matematiske læringsutbytte. Man kan først tenke at oppgaven i Figur 4.7 gjør det samme, men det er noen nøkkelforskjeller. Først og fremst blir ikke programmet presentert i ferdig BBP kode, men heller som en algoritme. Dette gjør eleven må dekomponere *selve matematikken*, istedenfor å dekomponere et ferdig konstruert program. For å lage programmet må altså eleven bryte ned matematikken i mindre biter, og se på den med et nytt lys (Lodi & Martini, 2021; Munthe, 2022). Deloppgave b) og c) ber eleven

om å endre variabler i programmet, og forklare samt forutse utfallet, noe som også er en sentral del av programmering (Bråting & Kilhamn, 2022).

Deloppgave e) ber eleven om å gjøre samme utregning som tidligere, opptil 500. Dette kan tenkes som en MP3 oppgave (Nyman et al., 2024), der oppgaven heller kunne ha blitt regnet med en kalkulator (Munthe, 2022, s. 35). Men da er det også viktig å huske at man nødvendigvis ikke bruker CAS kalkulatorer på 6. trinn, som oppgaven er laget til. Oppgaven fremstiller en algoritme, som eleven må tolke gjennom matematikken de har regnet gjennom den tidligere oppgaven. Videre får eleven beskjed om å sammenligne de ulike tankegangene fremmet i Figur 4.8 og 4.9. Dette gjør at oppgaven kan regnes som en MP4 oppgave, istedenfor kun en MP3 oppgave (Nyman et al., 2024).

Et av grunnlagene for at oppgaven i Figur 4.8 kunne anerkjennes som en MP4 oppgave, var at eleven ikke ble møtt med et ferdigkonstruert program. Videre ønsker jeg å fremheve en annen oppgave som faller innenfor MP4, men som har et ferdig konstruert program. Oppgaven kan sees i figur 4.10:



Figur 4.10 – Oppgave T4 Kapittel 3. Fra *Matemagisk 7A: Grunnbok* (s. 121), av Kongsnes et al, 2021c, Aschehoug Undervisning

Oppgaven i figur 4.10 er den nest siste i kapittelet om Brøk, desimaltall og prosent. Før oppgaven blir elevene presentert med hva en faktor er. Programmet i deloppgave a) viser alle faktorene i tallet som står i variabelen «tall1». Eleven får ikke direkte vite at programmet gjør dette, men blir klar over sammenhengen i deloppgave b). Videre ber oppgaven om at elevene skal finne faktorene i tallene i T1, som kan sees i figur 4.11:

Figur 4.11 – Oppgave T1 Kapittel 3. Fra *Matemagisk 7A: Grunnbok* (s. 120), av Kongsnes et al., 2021c, Aschehoug Undervisning

I oppgave T1 skal elevene finne alle likeverdige brøker av prosentene, der nevneren skal være under 100. Tar vi utgangspunkt i a), altså 35%, vil tallet ha 5 likeverdige brøker;

$$\frac{7}{20} = \frac{14}{40} = \frac{21}{60} = \frac{28}{80} = \frac{35}{100}$$

Totalt er det fem likeverdige brøker av tallet 35, med en maks nevner lik, eller mindre enn 100. Ser vi på deloppgave c) og d) i figur 4.9 skal elevene finne alle faktorene i prosenttallene, og sammenligne det med antallet likeverdige brøker fra T1. Mønsteret som oppstår er at den største faktoren funnet i programmet, er antallet likeverdige brøker fra T1.

Oppgaven i figur 4.10 fremhever et tydelig bruk av programmering for å utforske matematikk (Nyman et al., 2024). I motsetning til andre oppgaver der oppgaven stiller spørsmål om programmet, så omhandler deloppgavene kun det matematiske innholdet. Istedenfor å stille direkte spørsmål om programmets utskrift, spør oppgaven eleven om å forstå *hva* programmet printer ut. Dette gjør at eleven må både forstå hvordan programmet funker, men også hva sammenhengen mellom programmet og matematikken er, som kan være med på å skape en større sammenheng mellom programmering og matematikken (Nyman et al., 2024; Stigberg

& Stigberg, 2020). Videre blir programmet brukt som et verktøy, for å utføre matematikk som hadde vært vanskelig for hånd (Benton et al., 2018; Munthe, 2022).

Men samtidig kan en stille seg spørsmål om dette egentlig er programmering? Eleven blir ikke bedt om å bygge videre på programmet eller endre på variabler, som mange av de andre oppgave i Matemagisk 5-7 ber om. Det oppgaven gjør er å *finne regel*, altså forstå hvordan programmet fungerer. Ved å forstå hva programmet faktisk gjør, vil elevene få svar på spørsmålene i oppgaveteksten. Sammenhengen mellom matematikk og programmering blir tydeliggjort på en naturlig måte, der det ikke pekes direkte på, men heller at eleven selv oppdager den. Dette gjør at jeg anerkjenner oppgaven i figur 4.10 som en oppgave som tydelig viser sammenhengen mellom programmering og matematikk (Bråting & Kilhamn, 2022; Nyman et al., 2024).

4.3. Brobygging mellom programmering og matematikk

Ettersom LK20 oppgir at elevene skal mestre å «bruke programmering til å utforske data i tabeller og datasett» (Kunnskapsdepartementet, 2019) vil elevene trenge en tilstrekkelig mengde relevante oppgaver for å oppnå kompetansemålet. Totalt ble det funnet 9 oppgaver i Matemagisk 7A og 7B som omhandlet programmering, der ingen av programmeringsoppgavene brukte datasett og tabeller. Ettersom læreboken er en av hovedkildene lærerne bruker for å strukturere undervisningen (Fan et al., 2013; Lepik et al., 2015), vil bruken av kun Matemagisk 7 være nok til å oppfylle kompetansemålene i LK20 (Kunnskapsdepartementet, 2019). Oppgavene med programmering ble funnet under ulike temaer i forskjellige kapitler, men det er kun *mønstre* og *aritmetiske konsepter* som oppstår innenfor de matematiske konseptene. Alle programmeringskonseptene som Bråting & Kilhamn (2022) fremhever blir representert i Matemagisk 7. Oppgavene i grunnbøkene til 7. trinn omhandler flere aspekter som er sentrale innenfor AT, men få matematiske konsepter. En kan tenke seg at oppgavene bruker matematikk som en kontekst for å lære programmering (MP2) (Nyman et al., 2024). Men av oppgavene som er i Matemagisk 7, bruker flere av de programmering som et verktøy for å utforske matematikk (MP4) (Nyman et al., 2024). Dette kan tyde til at når elevene har fått en grunnleggende forståelse av programmering, vil forfatterne gi oppgaver som ikke har fokus å lære programmering, men heller å utforske matematikk gjennom programmering (Nyman et al., 2024).

Med ser vi tilbake på funnene til Nyman et al. (2024, s. 15), fant de at oppgavene på 7-9 trinn brukte primært matematikk for å lære programmering. Dette stemmer overens med Bråting & Kilhamn (2022) sine funne, der matematikken ofte blir omgjort eller simplifisert, for å passe programmeringen. Man kan da se for seg når overgangen mellom BBP til TBP som oppstår i 8. trinn (Munthe, 2022) oppstår, kan man tenke at mye av fokuset vil da være å lære TBP språkene, og den tyngre syntaksen rundt dem (Benton et al., 2016; Munthe, 2022). Selv om oppgaver på ungdomsskolen er utenfor denne masteravhandlingens fokus, kan det hende at samme situasjon oppstår igjen. For selv med en økt forståelse av sentrale AT elementer, vil det fortsatt være nye problemer med nye typer språk.

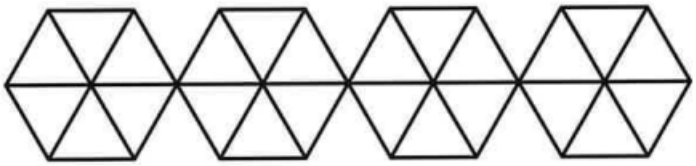
Nesten alle av programmeringsoppgavene på 5.trinn utfører handlingen *forme og skape*, med en kombinasjon av programmeringskonseptet *stegvise instrukser*. Dette vil si at mye av formingen og skapningen som elevene gjør, blir gjort gjennom å enten følge instrukser eller en algoritme som forfatterne presenterer i oppgaven. Det er ikke før i 6.- og 7. trinn at *forme og skape* blir brukt mye uten *stegvise instrukser*. Ofte omhandler disse oppgavene at elevene skal lage spesifikke figurer, uten å gi direkte instrukser om hvordan dette skal gjøres. Eksempel av dette kan sees i figur 4.12:

IV Frida skal bruke programmering til å tegne en del av et spillebrett.

a Lag en funksjon som tegner en likesidet trekant.

b Bruk funksjonen i et program for å tegne figuren under.

c Utvid programmet slik at det tegner mønstret under.



Figur 4.12 – Brettspillfabrikken IV. Fra *Matemagisk 6B: Grunnbok* (s. 114), av Kongsnes et al., 2021b, Aschehoug Undervisning

Oppgaven i Figur 4.12 er en av oppgavene som omhandler *forme og skape*, uten at elevene får *stegvise instrukser*. Elevene får ikke en spesifikk algoritme eller instrukser å følge, men heller former som de skal ta utgangspunkt i. Selv om oppgaven ikke gir *stegvise instrukser* for eleven å følge, så får elevene nøyaktig beskjed om hva de skal lage. Den geometriske figuren som oppgaven vil at eleven skal utforske er en likesidet trekant, noe som elever på 6. trinn er kjent med fra før. Her blir matematikken brukt som en kontekst for å arbeide med programmering, der det potensielle læringsutbyttet baserer seg på om elevene klarer å bruke en funksjon i programmeringsspråket, for å lage en ønsket mønster (Bråting & Kilhamn, 2022, s. 604).

Oppgaven i Figur 4.12 er et eksempel på at det matematiske konseptet *geometri* blir brukt for å lære programmering (Bråting & Kilhamn, 2022; Nyman et al., 2024). Det fleste av oppgavene tillater ikke elevene å utforske fritt med geometriske mønstre, men ber eleven heller om å lage enten en spesifikk figur, eller kjøre et spesifikt program. Det er et fåtall av oppgaver der eleven får muligheten til å utforske geometri fritt, noe som er fremmet som et potensielt læringsutbytte av bruken med programmering (Bråting & Kilhamn, 2022; Kaufmann & Stenseth, 2021; Munthe, 2022). Men i disse oppgavene har eleven ikke et spesifikt mål å arbeide etter. Det er altså ikke noe grunn til å gjøre oppgaven, uten å lage en

figur som er lik de vist i oppgaven. Oppgavene ber ofte elevene om å reflektere hvilket mønster som er deres favoritt (e.g. Raen et al., 2020, s. 107), eller å lage mønstre som eleven syntes er fint (e.g. Kongsnes et al., 2021, s. 114). For at programmering skal være et effektivt verktøy i matematikk, må det matematiske innholdet være tydelig i oppgaveteksten, for eleven (Nyman et al., 2024; Stigberg & Stigberg, 2020). Ved å ikke ha tydelige matematiske læringsmål, vil da det potensielle matematiske læringsutbytte være svakt for eleven. Hadde disse oppgavene bedt eleven om å f.eks. *forklare*, kunne det matematiske læringsutbytte potensielt vært høyere (Bråting & Kilhamn, 2022, s. 604).

Å *forklare* er en handling som oppstår i nesten halvparten av programmeringsoppgavene i Matemagisk 5-7. I Matemagisk 5 blir elevene ofte bedt om å forklare hvordan et program fungerer, eller forklare hva som skjer når verdiene i et program blir endret på. Ofte blir også eleven bedt om å *forestille seg* et utfall med å *forklare*, der eleven skal forestille seg hva et program gjør, og forklare tankeprosessen deres. Videre gjennom trinnene er det et fåtall av oppgaver der elevene skal forklare ulike sammenhenger mellom programmet, og matematikk (e.g. Figur 4.10).

Bruken av å *forklare* i Matemagisk 5-7, er som oftest for å dekomponere, og abstraksjon. Disse tankeprosessene er sentrale innenfor AT (Lodi & Martini, 2021; Utdanningsdirektoratet, 2019). Men måten dette blir gjort er ofte rettet mot programmets funksjonalitet, og få ganger til matematikken som blir gjort. Matematikken blir altså en kontekst for å lære programmering (Nyman et al., 2024), der å dekomponere programmet blir det potensielle læringsutbyttet. Noen enkeltoppgaver ber eleven om å *forklare* sammenhengen mellom programmet de har laget, og hvordan det henger sammen med matematikken relevant for dem. Men disse oppgavene er ofte i «topptur» delen av kapitlene, som er ment for elevene som mestrer resten av kapittelet (Raen et al., 2020, s. 2).

Ser vi tilbake på forordene til Papert (1980, s. viii), så ønsket han å bruke datamaskiner til å erstatte «tannhjulene» han brukte for å lære matematikk. Målet hans var at elevene kunne bruke programmering for å få et nytt perspektiv på å lære matematikken. Oppgavene som gjør nettopp dette, som fremmer matematikken gjennom programmering, og lar elevene utforske matematikk gjennom programmering, blir ofte begrenset til «topptur» delen av kapittelet.

Oppgavene jeg har fremmet, som er på et MP4 nivå (Nyman et al., 2024), er nettopp «topptur» - oppgaver (e.g. Figur 4.8 & Figur 4.10). Det er ikke alle elevene som klarer å komme seg til disse oppgavene, som mister muligheten til å bruke programmering for å lære matematikk på en alternativ måte (Munthe, 2022).

Ser vi på Nyman et al. (2024) sine relasjonsfunn, og Bråting & Kilhamn (2022) sine konsept- og relasjonsfunn, er det flere punkter der jeg finner mer matematikk i programmeringsoppgavene, enn hva forskerne fant i sine undersøkelser. Denne kan være for Norges fokus på å implementere AT gjennom programmering, der det er veldig tydelig hvilke AT elementer elevene skal lære gjennom fagene (Utdanningsdirektoratet, 2019). Forskningen som mine funn har blitt sammenlignet med stammer fra Sverige, der AT ikke er knyttet til den svenske læreplanen (Bråting & Kilhamn, 2021, s. 171, 2022, s. 594; Nyman et al., 2024, s. 15).

Selv om det er tydelig at vi bruker programmering for å lære AT (Utdanningsdirektoratet, 2019), så opplever jeg flere av avvikene som Bråting & Kilhamn (2022) og Nyman et al. (2024) opplever i sine studier. Dette tyder til at selv når AT er definert i de overordnede læringsmålene, betyr nødvendigvis ikke at brobyggingen mellom matematikk og programmering blir tydeligere. Det skal sies at Norge ikke har et eget fag for å lære programmering, som kan være en av grunnene til at det er mye fokus på å lære programmering.

4.4. Begrensninger innenfor drøftingen

En av begrensningene innenfor oppgaven, er at jeg kun ser på en læreboks serie. Vanligvis med kvalitative lærebokanalyser, blir flere ulike lærebokserier sett på sammen (Fan, 2013; Fan et al., 2013). Dette fører til at funnene jeg presenterer stammer fra en kilde, og betyr nødvendigvis ikke at mine funn gjelder alle lærebøker. Samtidig så jeg på det som nødvendig å begrense til en bokserie. Hadde jeg inkludert flere ville datamaterialet vært overveldende, og utenfor omfanget av denne masteravhandlingen. Et argument hadde vært å fokusere på enkelte kapitler, der hovedtemaet er programmering. Hadde jeg gjort dette, ville jeg ikke ha oppdaget flere av MP4 oppgavene i lærebøkene (e.g. Figur 4.8 & Figur 4.10). Videre ser mange lærebokanalyser på enkelte konsepter i enkelte temaer, der det også trengs forskning

som ser mer overordnet (Fan et al., 2013). Dette gjelder også programmering i matematikk, der enkelte studier fokuserer kun på programmering i et tema (e.g. Kaufmann & Stenseth, 2021). For å kunne se på hvordan programmering blir integrert inn i matematikk, anser jeg det som nødvendig å se på implementeringen i hele boka, og ikke bare i dedikerte kapitler.

Videre ble Nyman et al. (2024) sin artikkel publisert nærmere slutten av skriveprosessen. Jeg har brukt relasjonsrammeverket (Nyman et al., 2024, s. 7), men forfatterne presenterte også en oppdatert versjon av handlingsrammeverket (Nyman et al., 2024, s. 8). Den største forskjellen mellom den oppdaterte og Bråting & Kilhamn (2022) sin versjon, er inkluderingen av *tinkering*, altså tilpasse eller utvikle en ferdigprodusert kode, og *no programming* (Nyman et al., 2024, s. 8). I den vertikale analysen noterte jeg ned oppgaver om en oppgave ikke inneholdt programmering, men jeg noterte ikke ned om en oppgave inneholdt *tinkering*. Inkluderingen av *tinkering* kunne ha satt lys på enkelte aspekter av samholdet mellom programmering og matematikk, som ikke kommer frem med Bråting & Kilhamn (2022) sitt verktøy. Hvis jeg skulle analysert dataen på nytt, ville Nyman et al. (2024) sitt oppdaterte handlingsrammeverk blitt brukt.

Ved sammenligning av Nyman et al. (2024) sine funn er det to bemerkelser som må gjøres. Det første er at analysen baserte seg på enheter, altså flere oppgaver under samme tema, og ikke enkelte oppgaver slik som min masteroppgave gjør. Videre er forskjellene mellom de ulike relasjonene litt uklar ved enkelte steder, der f.eks. MP4 kan også inkludere MP3 og MP2 (Nyman et al., 2024, s. 7), noe som kan gjøre det vanskelig å riktig kategorisere enkelte oppgaver. Det skal sies at forfatterne kom til enighet mellom seg selv ved usikkerheter, men ettersom denne masteroppgaven skrives alene, var ikke dette en mulighet.

5. Konklusjon

5.1. Forskningsspørsmål 1 – På hvilken måte blir programmering integrert inn i matematiske lærebøker?

Gjennom grunnbøkene til Matemagisk 5-7 er de fleste programmeringsoppgavene fordelt i enkelte kapitler, som fokuserer på programmering. Det er enkelte programmeringsoppgaver i andre kapitler, blant annet aritmetikk, geometri og statistikk. Matemagisk 5B introduserer flere nøkkelkonsepter innenfor programmering (Raen et al., 2020), blant annet *dekomponering*, *abstraksjon* og *algoritmer* (Lodi & Martini, 2021; Utdanningsdirektoratet, 2019).

Flesteparten av enhetene i grunnbøkene bruker matematikk som en kontekst for å lære programmering. Dette stemmer overens med relasjonene mellom matematikk og programmering, i flere svenske lærebøker (Nyman et al., 2024, s. 9). Dette kan komme av at på lik måte som i Sverige, har ikke Norge et eget fag som lærer elevene programmering, som gjør at fag som matematikk får ansvaret for opplæringen av programmering (Bocconi et al., 2018).

Videre har ikke Matemagisk 7A+7B et dedikert programmeringskapittel som Matemagisk 5 og Matemagisk 6. Dette kan basere seg på at Kongsnes et al. (2021;2022) beregner bruken av Excel som programmering. Excel oppgavene er ikke lagd som programmeringsoppgaver (Nagy et al., 2021), noe som gjør at jeg ikke anerkjenner de som programmeringsoppgaver.

5.2. Forskningsspørsmål 2 – På hvilken måte legger programmeringsoppgavene til rette for å lære matematikk?

Programmeringsoppgavene i Matemagisk 5-7 har flere ulike matematiske konsepter, der *mønstre* og *geometriske konsepter* er de mest vanlige. Samtidig blir disse ofte brukt i en kontekst for å lære programmering, der det matematiske nivået ofte er lavt. Det er enkelte oppgaver som viser potensialet programmering har for å lære matematikk, men disse er ofte på slutten av kapitlene. Disse oppgavene er ment for kun de sterkeste elevene (Raen et al., 2020, s. 2), noe som gjør at flere elever ikke får arbeidet med disse oppgavene.

Oppgavene som de fleste av elevene vil møte på ber oftest eleven om å *følge prosedyre* og *forklare*. Disse handlingene blir oftest gjort i kontekst av selve programmet i oppgaven, og

sjeldent for å forstå matematikk bedre. Enkelte programmeringsoppgaver blir brukt for å introdusere noen temaer (e.g. Kongsnes et al., 2021b, s. 108). Ofte blir ikke det matematiske målet tydeliggjort i disse oppgavene, noe som er sentralt for å få et matematisk læringsutbytte av programmeringsoppgavene (Nyman et al., 2024; Stigberg & Stigberg, 2020). Med mindre endringer har flere av oppgavene potensiale til å fremme et nytt perspektiv på matematisk forståelse (Bråting & Kilhamn, 2022), men oppgavene mangler enkelte oppklarninger. Oppgavene hjelper elevene til å lære flere viktige aspekter av AT, men dette på bekostning av det potensielle matematiske læringsutbytte. Samtidig er det kun en oppgave som ber elevene om å *feilsøke*, som er et av nøkkelkonseptene innenfor programmering (Brennan & Resnick, 2012; Kaufmann & Stenseth, 2021; Sevik, 2016).

5.3. Problemstilling og avsluttende tanker

Etter å ha analysert grunnbøkene i Matemagisk 5-7, har jeg funnet og presentert både muligheter og eventuelle problemer som kan oppstå mellom programmering og matematikk. Det er blitt vist til hvilke ulike matematiske temaer programmering kan brukes i, og hvordan programmeringen brukes. Videre har viktigheten av måten programmeringsoppgavene blir presentert på, blitt vist. Målet har ikke vært å lage en anmeldelse av grunnbøkene i Matemagisk 5-7, men heller bruke bøkene for å vise ulike former av programmeringsoppgaver. Bøkene har samtidig gjort det klarere hva som kan gjøre at enkelte oppgaver fokuserer mer på programmering enn matematikk, men samtidig vist enkelte tilfeller der brobyggingen mellom matematikk og programmering skinner.

Flere av oppgavene oppnår det som blir beskrevet som i *den algoritmiske tenkeren* (Utdanningsdirektoratet, 2019), men ofte går dette på bekostning av det matematiske innholdet. Det er enkelte oppgaver som viser programmering brukt til å utforske matematikk (Nyman et al., 2024), men de fleste av oppgavene bruker kun matematikken for en kontekst å programmere.

Programmering er fortsatt et relativt ferskt tema i den norske skolen, noe som gjør temaet viktig å undersøke. Norge begynner å få mer og mer forskning innenfor programmering (e.g. Kaufmann & Stenseth, 2021; Munthe, 2022), men det er fremdeles behov for mer. I denne masteroppgaven har sammenhengen mellom matematikk og programmering blitt sett på, der lærebøker i matematikk fordelt over tre årsverk har blitt analysert. Læreres erfaringer med

programmering er ikke en del av denne masteroppgaven, men er en relevant del av den realistiske implementeringen av programmering i skolen. Derfor kunne det vært interessant å se på sammenhengen mellom lærebøkenes innhold og lærernes bruk av dette materialet. Videre omhandler mye av forskningen innenfor programmering i matematikk enkeltstående temaer (Lv et al., 2023). Det trengs mer forskning som ser på programmering som en helhet, og ikke bare i enkelte kontekster. AT er implementert som en tankemåte (Utdanningsdirektoratet, 2019; Wing, 2006), og ikke kun som et verktøy for å fokusere på geometri eller andre matematiske elementer. Ettersom programmering kan brukes i flere temaer, trengs det mer forskning på hvordan dette kan gjøres på den beste mulige måten.

Med denne masteravhandlingen har jeg fremmet at programmering kan brukes innenfor mange forskjellige matematiske temaer, for å gi en alternativ innfallsvinkel innenfor temaet. Ved å gå systematisk gjennom alle lærebøkene i en matematikkbokserie, har jeg funnet programmeringsoppgaver i andre kapitler utover de som finnes i programmeringskapitlene. Videre fremmes hvordan programmering kan brukes som et verktøy til å utforske matematikk, og gi et nytt perspektiv på matematikken (Papert, 1980). Det kommer fram i oppgaven at noen aspekter ved programmering og AT ikke kommer tydelig frem i oppgavene (e.g. *feilsøking*) noe forlag kan bli bevisste på, slik at deres utvikling av programmeringsoppgaver kan lede til tydeligere illustreringer av konseptene. Lærere kan også dra nytte av denne tydeligheten slik at de også lærer bort konseptene mer effektivt. Altså belyser denne oppgaven utviklingsområder innenfor programmeringsoppgaver brukt i skolen, som flere instanser kan ha nytte av å utforske.

6. Referanser

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer Science Unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016). *Building mathematical knowledge with programming: Insights from the ScratchMaths project*. Suksapattana Foundation

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging Primary Programming and Mathematics: Some Findings of Design Research in England. *Digital Experiences in Mathematics Education*, *3*(2), 115–138. https://doi.org/10.1007/s40751-017-0028-x

Benton, L., Kalas, I., Saunders, P., Hoyles, C., & Noss, R. (2018). Beyond jam sandwiches and cups of tea: An exploration of primary pupils' algorithm-evaluation strategies. *Journal of Computer Assisted Learning*, *34*(5), 590–601. https://doi.org/10.1111/jcal.12266

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The nordic approach to introducing computational thinking and programming in compulsory education*. https://doi.org/10.17471/54007

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, *1*, 25. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Bråting, K., & Kilhamn, C. (2021). 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. (2022). The Integration of Programming in Swedish School Mathematics: Investigating Elementary Mathematics Textbooks. *Scandinavian Journal of Educational Research*, *66*(4), 594–609. https://doi.org/10.1080/00313831.2021.1897879

Charalambous, C. Y., Delaney, S., Hsu, H.-Y., & Mesa, V. (2010). A Comparative Analysis of the Addition and Subtraction of Fractions in Textbooks from Three Countries. *Mathematical Thinking and Learning*, *12*(2), 117–151. https://doi.org/10.1080/10986060903460070

Clark, T., Foster, L., Sloan, L., & Bryman, A. (2021). *Bryman's social research methods* (Sixth edition.). Oxford University Press.

Cohen, L., Morrison, K., & Manion, L. (2018). *Research methods in education* (8. utg.). Routledge.

Cunha, J., Mendes, J., Saraiva, J., & Visser, J. (2014). Model-based programming environments for spreadsheets. *Science of Computer Programming*, *96*, 254–275. https://doi.org/10.1016/j.scico.2014.02.002

Ezeamuzie, N. O., & Leung, J. S. C. (2022). Computational Thinking Through an Empirical Lens: A Systematic Review of Literature. *Journal of Educational Computing Research*, *60*(2), 481–511. https://doi.org/10.1177/07356331211033158

Fan, L. (2013). Textbook research as scientific research: Towards a common ground on issues and methods of research on mathematics textbooks. *ZDM*, *45*(5), 765–777. https://doi.org/10.1007/s11858-013-0530-6

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

Kaufmann, O. T., & Stenseth, B. (2021). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*, *52*(7), 1029–1048. https://doi.org/10.1080/0020739X.2020.1736349

Kongsnes, A. L., Raen, K. M., & Sørdal, M. (2021a). *Matemagisk 6B: Matematikk for barnetrinnet : Grunnbok* (2.). Aschehoug.

Kongsnes, A. L., Raen, K. M., & Sørdal, M. (2021b). *Matemagisk 7A: Grunnbok* (2). Aschehoug undervisning.

Kongsnes, A. L., Raen, K. M., & Sørdal, M. (2022). *Matemagisk 7B: Matematikk for barnetrinnet : Grunnbok* (2). Aschehoug.

Kongsnes, A. L., & Wallace, A. K. (2020). *Matemagisk 8* (1.). Aschehoug undervisning.

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

Kvale, S., & Brinkmann, S. (2015). *Det kvalitative forskningsintervju* (3). Gyldendal akademisk.

Lepik, M., Grevholm, B., & Viholainen, A. (2015). Using textbooks in the mathematics classroom–the teachers' view. *Nordic Studies in Mathematics Education*, *20*(3–4), 129–156. https://ncm.gu.se/wp-content/uploads/2020/06/20_34_129156_lepik.pdf

Lodi, M. (2020). Informatical Thinking. *OLYMPIADS IN INFORMATICS*, *14*, 113–132. https://doi.org/10.15388/ioi.2020.09

Lodi, M., & Martini, S. (2021). Computational Thinking, Between Papert and Wing. *Science & Education*, *30*(4), 883–908. https://doi.org/10.1007/s11191-021-00202-5

Lv, L., Zhong, B., & Liu, X. (2023). A literature review on the empirical studies of the integration of mathematics and computational thinking. *Education and Information Technologies*, *28*(7), 8171–8193. https://doi.org/10.1007/s10639-022-11518-2

Morgan, H. (2022). Conducting a Qualitative Document Analysis. *Qualitative Report*, *27*(1), 64–77. https://doi.org/10.46743/2160-3715/2022.5044

Munthe, M. (2022). *Press 'run' to improve mathematical expertice (PRIME)* [Norwegian University of Life Sciences, Ås]. http://hdl.handle.net/11250/3045236

Nagy, T., Csernoch, M., & Biró, P. (2021). The Comparison of Students' Self-Assessment, Gender, and Programming-Oriented Spreadsheet Skills. *Education Sciences*, *11*(10), 590-. https://doi.org/10.3390/educsci11100590

Norberg, M. (2019). Potential for Meaning Making in Mathematics Textbooks. *Designs for Learning*, *11*(1), 52–62. https://doi.org/10.16993/dfl.123

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

Nyman, R., Bråting, K., & Kilhamn, C. (2024). Can programming support mathematics learning? An analysis of Swedish lower secondary textbooks. *International Journal of Mathematical Education in Science and Technology*, *0*(0), 1–19. https://doi.org/10.1080/0020739X.2024.2329345

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc. https://dl.acm.org/doi/abs/10.5555/1095592

Patton, M. Q. (2015). *Qualitative research & evaluation methods: Integrating theory and practice* (4). Sage.

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

Raen, K. M., & Kongsnes, A. L. (2021). *Matemagisk 6A: Grunnbok* (2). Aschehoug undervisning.

Raen, K. M., Kongsnes, A. L., Lang-Ree, H. L., & Nyhus, G. (2020). *Matemagisk 5B: Grunnbok* (2). Aschehoug undervisning.

Schreier, M. (2012). *Qualitative content analysis in practice: Margrit Schreier*. SAGE.

Sevik, K. (2016). *Programmering i skolen—Notat fra Senter for IKT i utdanningen*. Senter for IKT i utdaningen.

Stenseth, B., Kaufmann, O. T., & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten (trykt utg.)*, *30*(2).

Stigberg, H., & Stigberg, S. (2020). Teaching programming and mathematics in practice: A case study from a Swedish primary school. *Policy Futures in Education*, *18*(4), 483–496. https://doi.org/10.1177/1478210319894785

Trinket. (2024, april 2). *Frequently Asked Questions*. Trinket.io. https://trinket.io/faq

Utdanningsdirektoratet. (2019). *Algoritmisk tenkning*. Hentet fra https://www.udir.no/kvalitet-og-kompetanse/digitalisering/algoritmisk-tenkning/

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