UiA Universitetet i Agder

Master

# Ungdomstrinnets reise mot algoritmisk tenkning og programmeringskompetanse

En kvalitativ innholdsanalyse av matematikklæreverk på ungdomstrinnet.

ANNA HOLBEK
MITRA MIRZA-ZADEH

VEILEDERE
Nils Kristian Hansen
Shaista Kanwal

**Universitetet i Agder, 2024**
Fakultet for teknologi og realfag
Institutt for matematiske fag
Emnekode: MA-502

**Forord:**

Det har vært en glede å få mulighet til å skrive masteroppgave sammen, vi vil begge takke hverandre for et godt samarbeid denne våren. Det har vært utfordrende og lærerikt å få mulighet til å fordype seg i et tema over en lengre periode. Vi er også takknemlige for å kunne ha undersøkt et tema som vil være svært relevant for tiden vi går i møte. Vi har lært mye og er mange erfaringer rikere.

Vi vil gjerne takke veilederne våre, Nils Kristian Hansen og Shaista Kanwal for verdifull veiledning og gode diskusjoner. Videre vil vi begge takke familie, venner og kollegaer som har støttet oss i arbeidet med skrivingen.

Innleveringen av denne masteroppgaven markerer slutten på mange fine år på Universitetet i Agder for oss begge, og vi er spente og nysgjerrige på hva som møter oss i arbeidslivet.

Kristiansand, mai 2024

**Sammendrag**
I en stadig mer digitalisert hverdag så vi ved innføringen av Kunnskapsløftet 2020, et tiltak for å øke norske elevers digitale kompetanse. Dette ble gjort gjennom innføringen av algoritmisk tenkning og programmering i læreplanen, i tråd med trender i resten av Europa. Algoritmisk tenkning beskrives i kjerneelementet «Utforskning og problemløsning» som en systematisk problemløsende metode. I sammenhengen med å lære elevene å tenke algoritmisk blir ofte programmering brukt som et pedagogisk verktøy. Formålet med vår studie er å undersøke hvordan elementer av algoritmisk tenkning fremheves i programmeringsinnholdet på ungdomstrinnet, og kommentere på hvorvidt dette er dekkende for bestillingen i kjerneelementet.

Studiens empiri er basert på en kvalitativ analyse, med kvantitative innslag. Vi har gjennomført en innholdsanalyse på læreverk fra ungdomstrinnet, ved bruk av et rammeverk utviklet av forskerne Bråting og Kilhamn (2022). De har funnet frem til seks handlinger knyttet til hvordan utvikle elevenes evne til algoritmisk tenkning, hvor vi har knyttet disse handlingene opp mot elementer funnet i beskrivelsen av algoritmisk tenkning hos Utdanningsdirektoratet. Vi undersøkte til sammen 178 oppgaver, og fant 550 handlinger.

Funnene fra analysen viser at læreverkene på ungdomstrinnet trolig gir elevene på norsk skole mulighet til å utvikle evne til algoritmisk tenkning, men at flere av verkene utelater viktig elementer i betydelig omfang. Omfattende bruk av elementer som utforskning og feilsøking, og å finne mønster og sammenhenger mangler i flere verk. Spesielt delen om utforskning nevnes både i kjerneelement og i kompetansemål for ungdomstrinnet. Større vektlegging på dette kan være fordelaktig for elevenes evne til algoritmisk tenkning. Det vi ser er at det er en overvekt av oppgaver som ber elevene forme og skape i arbeidet med programmeringsoppgaver. Dette er en handling som kan kreve mye av både elever og lærere, og ettersom det er mange lærer som har gitt ytring for at de mangler tilstrekkelig kompetanse innen programmering kan oppgaver som dette vise seg å være utfordrende i klasserommet. Vi konkluderer med at programmeringsinnholdet i læreverk på ungdomstrinnet delvis dekker bestillingen fra kjerneelementet, men at flere av dem utelater tilstrekkelig bruk av viktige elementer.

**Summary**

In an increasingly digitalized everyday life, implementation of the new curricula "Kunnskapsløftet 2020" initiative aimed to enhance Norwegian students' digital competence. This was achieved through the introduction of algorithmic thinking and programming in the curriculum, aligning with trends across Europe. Algorithmic thinking is described in the core element "Exploration and Problem Solving" as a systematic problem-solving method. Often, programming serves as a pedagogical tool to teach students algorithmic thinking. The purpose of our study is to analyze how elements of algorithmic thinking are emphasized in the programming content at secondary level and to comment on whether this aligns with the requirements in the core element.

The empiricism of our study is based on a qualitative analysis with some quantitative components. We conducted a content analysis of textbooks used in secondary education, using a framework developed by researchers Bråting and Kilhamn (2022). They identified six actions related to developing students' ability in algorithmic thinking, which we correlated with elements found in the description of algorithmic thinking provided by the Norwegian Directorate for Education. In total, we examined 178 tasks, and found 550 actions.

The findings from our analysis indicate that the textbooks at the secondary level likely provide Norwegian students with opportunities to develop algorithmic thinking skills. However, several of these textbooks omit crucial elements to a significant extent. Notably absent are use of explore and debugging, as well as the ability to identify patterns and connections. Greater emphasis on explore could be advantageous for students' algorithmic thinking abilities. We observe an overrepresentation of tasks that require students to shape & create solutions in programming assignments. Such tasks can be demanding for both students and teachers, especially considering that many teachers have expressed a lack of sufficient programming competence. In conclusion, the programming content in secondary textbooks partially aligns with the core element's objectives, but several of them fall short in adequately incorporating important elements.

# Innholdsfortegnelse

# 1 Innledning

Med en skolehverdag preget av iPad, Chromebook og kunstig intelligens i klasserommet blir livet til dagens barn og ungdom stadig mer digitalisert. Dette er en trend i utvikling, og de som vokser opp i dag kan trolig se frem mot en stadig mer digitalisert fremtid (Kaufmann & Stenseth, 2021, s. 1029; Sanne et al., 2016, s. 8). I dagens teknologidrevne samfunn kan forståelse av algoritmisk tenkning og programmering være essensielt for å kunne forstå og bidra til denne pågående teknologiske utviklingen (Sanne et al., 2016, s. 7). Algoritmisk tenkning har blitt fornyet som forskningsfelt, i stor grad av Jeanette Wing (2006). Wing (2006) mener at algoritmisk tenkning er en evne alle burde ha, og at det burde komme inn i skolen på lik linje som lesing, skriving og regning.

I løpet av det siste tiåret har programmering blitt en del av læreplanen i mange land rundt i Europa (Sevik et al., 2016, s. 22). Innføringen av programmering i norsk skole har ført til en stor debatt, der «Sanne-gruppen» i 2016 leverte en rapport om hvordan programmering burde innføres i skolen, etter bestilling fra Utdanningsdirektoratet. Der mente de, i likhet med flere andre norske forskere at programmering burde bli innført som et eget teknologifag og ikke blitt del av matematikkfaget, men denne anbefalingen ble ikke fulgt (Sanne et al., 2016, s. 8; Sevik et al., 2016). I Kunnskapsløftet 2020 (LK20) ble algoritmisk tenkning og programmering integrert i matematikkfaget i norsk skole (Utdanningsdirektoratet, 2019g).

I LK20 har algoritmisk tenkning fått en sentral plass i et av kjerneelementene i matematikkfaget (Utdanningsdirektoratet, 2019c). I læreplanen har dette ført til en viktig endring i klasserommet. For å utvikle evne til algoritmisk tenkning blir ofte programmering brukt som en pedagogisk tilnærming (Bråting & Kilhamn, 2022, s. 594). Man finner kompetansemål om programmering og begreper knyttet til programmering, helt ned i 5. trinn (Utdanningsdirektoratet, 2019g). Dette betyr at mange lærere som tidligere ikke har trengt å kunne programmere, nå må lære det.

I overordnet del i LK20 fremmes digital kompetanse som en av de fem grunnleggende ferdighetene, med bakgrunn fra verdigrunnlaget i opplæringsloven. Skolen skal legge til rette for at elevene skal kunne utvikle disse ferdighetene gjennom hele skoleløpet sitt (Utdanningsdirektoratet, 2019b). Der er det poengtert at lærere skal støtte elevenes utvikling av digitale ferdigheter, og Stenseth et al. (2019) mener at programmering kan betraktes som en veldig sentral ferdighet for å kunne delta i den digitale verden (Stenseth et al., 2019; Utdanningsdirektoratet, 2019b).

## 1.1 Forskningsspørsmål, motivasjon og forskningens formål

I denne studien ønsker vi å undersøke hvordan læreverk utgitt etter LK20 tilrettelegger for algoritmisk tenkning gjennom programmeringsoppgaver i matematikk på ungdomstrinnet. Formålet med studien er å belyse programmeringsinnholdet i læreverk på ungdomstrinnet, og undersøke hvordan det kobles sammen med ønsket fra LK20 om at elever skal lære seg å tenke algoritmisk. Vi ønsker å se på muligheter og begrensninger i de forskjellige læreverkene, og sammenligne dem for å fremheve likheter og ulikheter. På bakgrunn av dette har vi utformet følgende forskningsspørsmål:

> - *Hvordan fremheves elementer av algoritmisk tenkning i programmeringsoppgavene i matematikklæreverk for ungdomstrinnet utgitt etter Kunnskapsløftet 2020?*

Vi skal ved bruk av rammeverket fra Bråting og Kilhamn (2022) undersøke hvordan elementer av algoritmisk tenkning fremmes i læreverkene på ungdomstrinnet. Vi ser spesielt på handlinger i Bråting og Kilhamn (2022) sitt rammeverk (beskrevet i detalj i kapitlene om teori og metode), og har fordelt forskningsspørsmål i to følgende spørsmål:

1) *Hvordan er fordelingen av handlinger knyttet til elementer av algoritmisk tenkning i læreverkene?*
2) *Hva kan sammensetningen av elementene i programmeringsoppgavene bety for elevenes potensielle mulighet til utvikling av evne til algoritmisk tenkning?*

Lv et al. (2023, s. 8176) påpeker at tidligere forskning innenfor feltet algoritmisk tenkning i skolen er i stor grad rettet mot barnetrinnet, og mener at videre forskning bør sette fokus på høyere trinn. Forskning på ungdomstrinnet representerer da en mangel i forskningsfeltet, og er derfor noe vi tenker er verdt å undersøke grundigere.

En annen grunn til at dette kan være interessant å undersøke er hvor stor plass læreverkene ser ut til å ta i norsk undervisning (Sanne et al., 2016, s. 49). Lærerne støtter seg mye på læreverk, og det kan være viktig å gjøre et dypdykk i hva disse læreverkene krever at elevene arbeider med knyttet til algoritmisk tenkning og programmering (Bråting & Kilhamn, 2022, s. 607). Vi lurer på om oppgavene kun handler om å følge en prosedyre og gjøre små endringer i en kode, eller om elevene får mulighet til å utforske og drive problemløsning gjennom algoritmisk tenkning, slik det står beskrevet i læreplanen.

Lærerens digitale kompetanse har mye å si for hva og hvordan digitale verktøy brukes i klasserommet (Bråting & Kilhamn, 2022; Forsström & Kaufmann, 2018; Sanne et al., 2016). Mange lærere har ikke fått opplæring i programmering gjennom sin utdanning, noe som kan skape usikkerhet når de selv skal undervise dette i klasserommet (Forsström & Kaufmann, 2018; Sanne et al., 2016; Sevik et al, 2016). Det er derfor viktig med gode læreverk, ettersom mange lærere lener seg mye på disse læreverkene når de planlegger og gjennomfører undervisning (Bråting & Kilhamn, 2022, s. 595).

Når det gjelder undersøkelse av fagstoff og artikler knyttet til algoritmisk tenkning og programmering i skolen, står en forskning gjort av Bråting og Kilhamn (2022) frem som sentral, og er noe vi ønsker å bruke som veiledning for vår egen forskning. Bråting og Kilhamn (2022) har undersøkt matematikklæreverk på barnetrinnet i svensk skole, og utviklet et rammeverk knyttet til algoritmisk tenkning og handlinger i programmeringsoppgaver. Dette rammeverket baserer vi vår analyse på.

I neste kapittel vil vi beskrive teoretisk rammeverk, for så å beskrive metode i kapittel 3. I kapittel 4 vil vi presentere resultatene fra analysen vår, for så å drøfte disse resultatene i kapittel 5. Til slutt kommer avslutning og konklusjon i kapittel 6.

# 2 Teori

Vår forskning har til hensikt å undersøke elementer av algoritmisk tenkning i matematikklæreverk på ungdomstrinnet. I denne sammenheng ønsker vi å undersøke to nøkkelbegreper; algoritmisk tenkning og programmering. Dette kapittelet vil handle om hva algoritmisk tenkning og programmering er, og gjøre rede for tre rammeverk; Brennan og Resnick (2012), Benton et al. (2016) og til slutt en kort introduksjon av rammeverket vi skal bruke til vår analyse fra Bråting og Kilhamn (2022). Rammeverket fra Bråting og Kilhamn (2022) vil bli enda grundigere gjennomgått i metodekapittelet.

## 2.1 Algoritmisk tenkning: sentrale elementer

Å forstå begrepet algoritmisk tenkning er essensielt for å kunne bruke det i utdanningssammenheng. I dette delkapittelet utforsker vi sentrale elementer knyttet til begrepet ved å se på ulike definisjoner og perspektiver, og hvordan disse former vår forståelse av begrepet. Det er lite enighet om en formell definisjon av begrepet algoritmisk tenkning, men det er en generell enighet om flere av konseptets sentrale elementer (Román-González et al., 2017, s. 678). Begrepet er den norske oversettelsen av det engelske uttrykket "computational thinking", og er innført gjennom ett av kjerneelementene i matematikkfaget i LK20. Algoritmisk tenkning er en problemløsende metode, og handler om en systematisk måte å tilnærme seg et problem på. Dette er en grunn til at begrepet er inkludert under kjerneelementet «Utforskning og problemløsning» (Utdanningsdirektoratet, 2019c).

### 2.1.1 Forståelse av algoritmisk tenkning: definisjoner og perspektiver

Som nevnt tidligere, er det ikke en generell enighet om definisjonen av algoritmisk tenkning. Román-González et al. (2017, s. 679) har gruppert definisjonene fra faglitteraturen inn i disse tre kategoriene: generiske definisjoner, operasjonelle definisjoner, og definisjoner knyttet til utdanning og læreplaner. En generisk definisjon vil være en definisjon som omfatter generelle konsepter knyttet til algoritmisk tenkning, mens en operasjonell definisjon fokuserer på operasjonene man må gjennomføre for å løse oppgaver, og er nærmere knyttet til bruk av datamaskiner gjennom arbeidsprosessen. Definisjoner knyttet til utdanning og læreplaner

inneholder et større fokus på arbeidsmetoder og begreper for å lære bort algoritmisk tenkning i klasserommet (Román-González et al., 2017, s. 679).

Jeanette Wing er en sentral forsker innenfor feltet algoritmisk tenkning, og blir ofte berømmet som den som plukket opp tråden igjen på forskningen om algoritmisk tenkning etter Papert (1980) på 80-tallet (Bråting & Kilhamn, 2022, s. 594). Ifølge Wing (2006, s. 33) utgjør algoritmisk tenkning en essensiell ferdighet for alle, og ikke bare for fagpersoner innen datavitenskap. Hun mener økt evne til algoritmisk tenkning kan være med på å bedre elevers evne til problemløsning, analysere data, gjenkjenne mønstre, modellere, dekomponere, og designe algoritmer og programvare (Wing, 2006, s. 33-34). Hun definerer algoritmisk tenkning som en tilnærming til problemløsning og systemdesign som bygger på grunnleggende konsepter innen datavitenskap. Wing (2006) er et eksempel på en generisk definisjon.

Wing (2006, s. 33-34) fremhever at algoritmisk tenkning dreier seg om å dekomponere tilsynelatende utfordrende problemer til løsbare delproblemer. Gjennom arbeidsmetoder som reduksjon, integrasjon, transformasjon og simulering er det tenkt at man kan oppnå ulike løsninger. Wing (2006) mener algoritmisk tenkning er en evne alle bruker i hverdagen sin. Hun gir eksempler på dette gjennom at man stadig gjør vurderinger og avveininger mellom tidsbruk og plasseringer. Et eksempel kan være når sønnen din har mistet en vott. Da foreslår du at han følger stegene sine baklengs for å se hvor han har vært, for å finne den igjen. Et annet eksempel kan være hvordan man velger hvilken kø man skal stille seg i når man handler i matbutikken. Her foregår det en tydelig prosess i hjernen vår for å minske tiden man må stå i kø, uten at vi nødvendigvis alltid tenker over dette (Wing, 2006, s. 34).

### 2.1.2 Algoritmisk tenkning i skolen: nøkkelbegreper og arbeidsmåter

Utdanningsdirektoratet (2019a) har belyst "Den algoritmiske tenkeren" i sin beskrivelse om hva det betyr å arbeide med algoritmisk tenkning. Under kjerneelementet «Utforskning og problemløsning» blir det beskrevet at «*algoritmisk tenkning er viktig i prosessen med å utvikle strategier og framgangsmåter for å løse problemer og innebærer å bryte ned et problem i delproblemer som kan løses systematisk. Videre innebærer det å vurdere om delproblemene*

*best kan løses med eller uten digitale verktøy*» (Utdanningsdirektoratet, 2019c). «Den algoritmiske tenkeren» fremhever noen nøkkelbegreper som logikk, algoritmer, dekomposisjon, mønstre, abstraksjon og evaluering, og arbeidsmåter som fikle, skape, feilsøke, holde ut, og samarbeide (Utdanningsdirektoratet, 2019a). Dette er et eksempel på en definisjon knyttet til utdanning og læreplan.


*Figur 1: Grafikk fra Utdanningsdirektoratets nettside om algoritmisk tenkning (2019). Figuren er tilpasset fra Barefoot Computing (UK) som er publisert med åpen lisens (OGL) (Utdanningsdirektoratet, 2019a).*

Når elevene undervises i å tenke algoritmisk, er det en tett pedagogisk knytning til programmering (Bråting & Kilhamn, 2022, s. 597). Derfor er det naturlig at når elevene tenker algoritmisk ved bruk av de nevnte arbeidsmetodene, også skal vurdere om en datamaskin kan løse hele eller deler av problemet (Utdanningsdirektoratet, 2019a). Sammenhengen mellom algoritmisk tenkning og programmering utdypes senere i dette kapittelet.

### 2.1.3 Algoritmiske begreper

Etter beskrivelsen av algoritmisk tenkning over, kan det være hensiktsmessig å gå nærmere inn på noen av begrepene vi finner i «den algoritmiske tenkeren». Gjøvik og Torkildsen (2019, s. 33) har kommet med noen nyttige definisjoner. De skriver at algoritmer og algoritmebehandling

handler om å følge og forklare hvert av stegene i en instruksjon. Her handler det da om å kunne lese en oppskrift, gjenkjenne stegene, for så å utføre stegene. Videre nevnes generalisering, som vil si å gjenkjenne mønstre og sammenhenger, for så å lage generelle regler og metoder som vil være gyldig for en hel gruppe av eksempler. Begrepet abstraksjon handler om å trekke ut essensen av et eller flere tilfeller, og å se bort ifra irrelevant informasjon. Eksempler på dette kan være at elevene må trekke ut viktig informasjon fra tekstoppgaver for så å bruke denne informasjonen i oppgaveløsningen sin. Dekomponering vil si å kunne bryte opp et problem i mindre deler, for så å kunne løse disse delproblemene, som til slutt vil løse hovedproblemet (Gjøvik & Torkildsen, 2019, s. 33-34).

## 2.2 Programmering: sentrale elementer

Programmering er en viktig komponent i utviklingen av algoritmisk tenkning. I dette delkapittelet ser vi nærmere på sentrale elementer knyttet til programmering, ulike tilnærminger til programmeringsundervisning og programmering sin rolle i læreplanen i norsk skole. Stenseth et al. (2019, s. 7) mener programmering er en grunnleggende ferdighet for å delta effektivt i dagens digitale verden. Som nevnt i innledningen, møter barn i dag på en stadig mer digitalisert hverdag. Román-González et al. (2017) skriver om «programmér eller bli programmert», og påpeker viktigheten av barns kodekyndighet i arbeidet med å lære, lese, og skrive kode i programmeringsspråk og tenke algoritmisk.

Papert (1996) skriver om hvor viktig det er å bruke programmering og algoritmisk tenkning når man underviser i matematikk. Han sier at programmering kan være et kraftfullt verktøy for å løse problemer og være kreativ innenfor matematikkfaget. Papert (1996, s. 101, 120) mener også at det er bra å lære barn og programmere når de er unge, fordi det kan hjelpe dem å forstå matematikk på en enklere måte. Bråting og Kilhamn (2022, s. 597) knytter sammen begrepene algoritmisk tenkning og programmering ved å beskrive programmering som et ofte brukt pedagogisk verktøy når elever skal lære seg å tenke algoritmisk.

En generell definisjon av programmering er å utforme en rekke instruksjoner for å oppnå et ønsket resultat. Disse instruksjonene skal være på et språk som både datamaskiner og mennesker kan forstå og utføre (Sevik et al., 2016, s. 9; Stenseth et al., 2019, s. 7; Vihovde, 2024). Vi finner også en utvidet bruk av programmeringsbegrepet som omfatter mer enn bare det å skrive programkode som kan kjøres på en datamaskin. Programmering er ikke bare å skrive en kode, men inkluderer også å analysere, forstå, og løse problemer. Programmeringsprosessen inkluderer utformingen av detaljerte instruksjoner som datamaskinen må følge for å løse spesifikke oppgaver og støtte menneskelig interaksjon. Det å utvikle algoritmer og bruke programmeringsspråk for å anvende disse algoritmene er en del av denne programmeringsprosessen. Ferdigheten med å utvikle algoritmer er ofte forbundet med logikk og matematisk tenkning (Sevik et al., 2016, s. 9; Stenseth et al., 2019, s. 7).

Denne beskrivelsen av programmering er begrenset til arbeid med datamaskiner. I datamaterialet vårt, finner vi en type programmering som ikke innebærer bruk av datamaskin; analog programmering. Denne type programmering beskriver vi detalj senere i dette kapittelet. Vi mener derfor at å begrense programmering til å kun omfatte aktiviteter utført på datamaskin, er for snevert for programmeringsinnholdet på ungdomstrinnet. Vi ønsker derfor å utvide bruken av definisjonen over til å inkludere arbeid med analog programmering.

### 2.2.1 Programmeringskunnskap og programmeringsprosessen

For å kunne programmere kreves det spesifikke kunnskaper. Eksempler på denne typen kunnskap kan være; kjennskap til programmeringsspråk, evne til å analysere, forstå og løse problemer ved å verifisere algoritmiske krav, og kunne vurdere korrekthet i koding av algoritmen i et bestemt programmeringsspråk. Programmering innebærer også evnen til å dekonstruere komplekse problemer til mindre deler og deretter utvikle systematiske metoder for å løse dem (Sevik et al., 2016, s. 22; Stenseth et al., 2019, s. 7). Dette samsvarer med Utdanningsdirektoratet (2019a) beskrivelse av algoritmisk tenkning, og vi ser en tydelig kobling mellom algoritmisk tenkning og programmering. Programmering handler om å skape noe nytt eller forbedre eksisterende produkter, og har derfor en kreativ komponent. Elevene skal kunne

bruke tilgjengelige verktøy utforskende på forskjellige måter for å løse oppgaver og problemer. Det krever en balanse mellom improvisasjon og systematikk, samt evnen til å anvende algoritmisk tenkning (Sevik et al., 2016, s. 16).

Programmering kan mulig være en motiverende faktor når elevene arbeider med matematikk. Forsström og Kaufmann (2018, s. 25) har konkludert med at programmering i noen tilfeller kan være med på å øke elevenes motivasjon når de kunne knytte den aktuelle matematikken til virkeligheten. Flere av studiene de har undersøkt viste også at enkelte grupper med elever ble flinkere i matematikk gjennom programmering (Forsström & Kaufmann, 2018, s. 21-26).

### 2.2.2 Feilretting og iterativ arbeidsmetode

I tillegg til kreativitet og utforskning, arbeider man mye med feilretting, korreksjon eller debugging når man programmerer. Feilretting, korreksjon og debugging er også sentrale arbeidsmetoder når man tenker algoritmisk (Utdanningsdirektoratet, 2019a). Det er veldig sjeldent et program kjører feilfritt første gangen, og programmet kan ha ulike feil som syntaksfeil, kjøretidsfeil, eller at programmet ikke gir ønsket resultat. Mengden av mulige problemer som kan oppstå kan være svært krevende for elevene, spesielt i oppstarten av å lære seg programmering (Stenseth et al., 2019, s. 8). Å arbeide med feilretting eller debugging krever ofte en iterativ innfallsvinkel til problemløsning. En iterativ innfallsvinkel vil si at eleven arbeider med problemet på en gjentakende måte, hvor arbeidet går i en syklus som repeteres (Bråting & Kilhamn, 2022, s. 597).

Stenseth et al. (2019) beskriver tre ulike typer iterasjoner eller feilrettingsmetoder:

- Den intuitive – her modifiseres programmet litt tilfeldig, og eleven prøver og feiler.
- Den kodefokuserte – her tolker eleven utfallet av variabler og kode-sekvenser nøyere, og eleven gjør mer gjennomtenkte endringer.
- Den teoretisk funderte – her brukere eleven i større grad matematikken til å analysere og forstå hva som skjer i programmet og formulerer en hypotese før den gjør endringer.

Stenseth et al. (2019) mener at «den teoretisk funderte» er et ideal man ønsker at eleven skal strekke seg mot. Når elevene har lært seg tilstrekkelig med programmering er det ønskelig at de mestrer og overfører deres matematiske kunnskaper inn i programmeringen (Stenseth et al., 2019, s. 8).

### 2.2.3 Tre typer av programmering

Tradisjonelt er det mange som forbinder programmering med tekstbasert programmering og koder på en dataskjerm, men det trenger ikke nødvendigvis være slik. Gjennom vårt datamateriale finner vi hovedsakelig tre typer programmering; analog-, blokk-, og tekstbasert programmering.

Analog programmering er en tilnærming til programmeringsoppgaver som utføres uten bruk av datamaskin. Denne metoden kan være særlig verdifull for å introdusere elever til grunnleggende konsepter knyttet til programmering, og gi dem mulighet til å visualisere fagstoffet ved bruk av konkrete objekter og bilder (Statped.no, 2021a). Selv om det kan virke litt underlig å undervise i programmering uten en datamaskin, så har det gjennom forskning vist seg å være en effektiv måte å introdusere og forklare programmering og algoritmisk tenkning på. Ved å tilnærme læringen på denne måten får elevene et innblikk i programmeringslogikken og strukturen før de møter på eventuelle utfordrende koder på en datamaskin (Munasinghe et al., 2023, s. 56-57).

Munasinghe et al. (2023) mener dette kan legge grunnlaget for en dypere forståelse og kan hjelpe elevene med å tydeligere se sammenhenger, da tekstbasert programmering kan virke overveldende ved første møte. Det nevnes også at dette kan bidra med å sikre elevene langsiktig engasjement og motivasjon, da det kan være med på å få elevene til å føle seg mer komfortable med de grunnleggende konseptene innenfor programmering og øke deres selvtillit i møte med nye utfordringer (Munasinghe et al., 2023, s. 58).

Blokkprogrammering er en type programmering hvor man har forenklet programmeringsfunksjoner fra tekst til blokker, derav navnet. Denne typen programmering

innebærer å programmere ved å sette sammen visuelle blokker som representerer ulike handlinger og konsepter. Dette er en tilnærming som gir visuell støtte og er spesielt egnet for nybegynnere og yngre elever som skal lære grunnleggende programmeringskonsepter (Statped.no, 2021b). Scratch er et eksempel på et blokkbasert programmeringsverktøy. Scratch er spesielt utviklet for å gjøre programmering tilgjengelig for yngre elevgrupper, og kan bli brukt til å løse programmeringsoppgaver uten behov for å skrive avansert tekstbasert kode (Statped.no, 2021b).

Som tredje og kanskje den vanligste programmeringstypen man finner i datamaterialet, kommer tekstbasert programmering. Dette er det mange tenker på som tradisjonell programmering, og som involverer manuell skriving av kode ved hjelp av tekstbaserte instruksjoner (Statped.no, 2021b). Dette er en vanlig tilnærming blant mer erfarne programmerer, og eldre elevgrupper som ønsker å arbeide med mer avanserte programmeringsoppgaver. Eksempler på tekstprogrammeringsspråk inkluderer Java, C, og Python. Denne typen programmering gir større grad av kontroll og fleksibilitet i programmeringen (Statped.no, 2021b).

Ved å tilby ulike oppgavetyper som analog-, blokk-, og tekstbasert programmering, kan lærere tilpasse undervisningen etter elevenes ferdighetsnivå og preferanser (Statped.no, 2021c). Dette mangfoldet i tilnærminger til programmeringsundervisningen legger til rette for en variert læringsopplevelse. Lærere kan bruke de forskjellige programmeringsmetodene til å engasjere og motivere elever med varierende evne til programmering (Statped.no, 2021c).

### 2.2.4 Programmering i læreplanen

I læreplanen finnes det tre kompetansemål knyttet til programmering på ungdomstrinnet. På 8. trinn skal elevene «*utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering*» (Utdanningsdirektoratet, 2019d). På 9. trinn skal elevene kunne «*simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering*» (Utdanningsdirektoratet, 2019e). Og på 10. trinn skal elevene kunne «*utforske*

*matematiske egenskaper og sammenhenger ved å bruke programmering*»
(Utdanningsdirektoratet, 2019f).

## 2.3 Algoritmisk tenkning og oppgavedesign – et grunnlag for analytisk rammeverk

I vår analyse skal vi bruke rammeverket utviklet av Bråting og Kilhamn (2022). De presenterer et analytisk rammeverk bestående av 6 handlinger som beskriver ulike måter elever kan utvikle sin algoritmiske tenkning på, og ulike handlinger de kan bli bedt om å utføre når de jobber med oppgaver i programmering (Bråting & Kilhamn, 2022, s. 598). Under kommer vi med en kort introduksjon til grunnlaget for rammeverket og rammeverket selv. Dette gjør vi i dette kapittelet for å kunne kommentere på hvordan handlingene er knyttet til elementene av algoritmisk tenkning vi finner hos Utdanningsdirektoratet (2019a).

### 2.3.1 Bentons 5E-er

Bråting og Kilhamns (2022) analytiske rammeverk er basert på to andre rammeverk; (1) Benton et al. (2016) 5E-er, og (2) Brennan og Resnicks (2012) tredelte forståelse av algoritmisk tenkning (Bråting & Kilhamn, 2022). Under vil vi undersøke disse to rammeverkene.

Benton et al. (2016) har undersøkt spenningen mellom verktøy og matematikkoppgaver, gjennom det blokkbaserte programmeringsverktøyet Scratch. Deres analytiske rammeverk for matematikkoppgaver var ute etter å skape et rammeverk for «actions», handlinger i oppgavene. Fra dette skapte de et rammeverk spesielt utviklet for oppgavedesign, som de kalte de 5E-ene (Benton et al., 2016, s.2-4). En av årsakene til at Bråting og Kilhamn (2022) har tatt utgangspunkt i dette rammeverket er fordi de mener at «actions» bidrar til å forstå læringsprosessen i sammenheng med utviklingen av algoritmisk tenkning (Bråting & Kilhamn, 2022, s.599).

Dette er Bentons 5E-er: (Benton et al., 2016, s. 5-7).

- **Explore** (utforske): Oppgaver som er formulert smed «explore» eller «utforske», kan legge til rette for å utvikle elevenes evne til å eksperimentere, feilsøke og variere måter elevene «angriper» oppgaver på ved bruk av forskjellige idéer og verktøy.

- **Explain** (forklare): Oppgaver formulert med «forklare» legger til rette for at elevene skal kunne vise kunnskap på forskjellige måter, og forklare sin egen fremgangsmåte og tenkning. Det kan oppfordre til reflekterende spørsmål og diskusjon med medelever både under og etter løst oppgave.
- **Envisage** (forutse)**:** Oppgaver formulert med «forutse» legger til rette for å utvikle elevenes evne til å forutse utfall, og utvikle hypoteser før de kjører et program.
- **Exchange** (utveksle): Utveksle idéer og arbeide sammen om løsninger kan fremme læring og være med på å utfolde elevens måte å tenke på knyttet til programmering.
- Bridg**E** (bro)**:** Den siste E-en handler om å finne sammenhenger mellom programmering og matematiske ideer, og understreke viktigheten med å knytte matematikken inn i programmeringsundervisningen.

### 2.3.2 Brennan og Resnicks tredeling

Det andre rammeverket Bråting og Kilhamn (2022) har brukt som inspirasjon for utformingen av deres eget rammeverk, er Brennan og Resnicks (2012) tredeling av definisjonen om algoritmisk tenking.

Brennan og Resnick (2012) har delt definisjonen av algoritmisk tenkning inn i tre deler; (Brennan & Resnick, s.1, 3, 2012).

1. **Computational concepts**: konseptene brukt når man programmerer som sekvenser, løkker, betingelser, og operatører.
2. **Computational practices**: praksisene man utvikler når man bruker arbeidsmetoder som inkrementell og gjentagende handling, feilsøking, se på andres arbeid, gjenbruke og endre tidligere arbeid, abstrahere og modellere.
3. **Computational perspectives**: elevenes syn på verden og hvordan dette påvirker deres holdning til programmering.

De har i sin artikkel beskrevet algoritmisk tenkning som et verktøy når de har undersøkt hvordan barn utvikler programmeringsegenskaper i blokkbaserte programmeringsverktøy som

Scratch (Brennan & Resnick, s.2, 2012). De har understreket elevenes behov for å kunne formidle og implementere idéene sine når de arbeider med algoritmisk tenkning, og for å samarbeide med andre gjennom sosiale læringsmiljøer (Brennan & Resnick, s.2, 2012). Vi ser at det Bråting og Kilhamn (2022) har tatt med seg videre fra dette rammeverket er de to første delene av definisjonen, men utelater den tredje. Dette er trolig fordi det er vanskelig å si noe om elevers holdning og hva som påvirker den, uten å observere eller intervjue elevene.

Som Bråting og Kilhamn (2022) selv understreker, er ingen av disse rammeverkene alene passende for en innholdsanalyse av oppgaver i læreverk. Hver for seg er Benton et al. (2016) utviklet for oppgavedesign, og Brennan og Resnick (2012) for undersøkelse av algoritmisk tenking (Bråting & Kilhamn, 2022, s.598). Vi har i metodekapittelet gått nærmere inn på hvordan disse to rammeverket direkte har påvirket utformingen av Bråting og Kilhamns (2022) rammeverk.

### 2.3.3 Bråting og Kilhamns analytiske rammeverk

Ut fra de to rammeverkene beskrevet over, har Bråting og Kilhamn (2022) utviklet et eget analytisk rammeverk. Deres forskningsspørsmål omhandler hva som kategoriserer programmeringsinnhold i svenske matematikklæreverk, og hvordan programmering og matematikk knyttes sammen gjennom programmering (Bråting & Kilhamn, 2022, s.595).

Bråting og Kilhamn (2022) har utformet seks mulige handlingskategorier basert på handlingene læreverkene ber elevene gjennomføre i oppgavene de har undersøkt: (Bråting & Kilhamn, 2022, s.599).

a) **Følge prosedyre:** følg stegvise instruksjoner, repetere eller fortsette et mønster.
b) **Finne regel:** finn ut av prosedyren, regelen eller mønsteret som skaper et utfall, som for eksempel en tallsekvens.
c) **Feilsøke:** feilsøke i en kode. Undersøk hva som gjør at programmet ikke fungerer slik det skal eller er tenkt.

d) **Forme og skape:** gi instruksjoner, lag et mønster, skriv kode, representer konsepter med symboler.
e) **Forklare:** forklar ved bruk av naturlig språk, ved bruk av ord for å beskrive en prosedyre, regel, mønster eller konsept.
f) **Forestille seg:** forutse hva som vil skje, reflekter rundt mulige utfall når betingelser eller verdier blir forandret.

Handlingskategoriene i dette rammeverket samsvarer på flere punkter med «Den algoritmiske tenkeren» vi finner hos Utdanningsdirektoratet. «Den algoritmiske tenkeren» beskriver ulike nøkkelbegreper og arbeidsmetoder knyttet til algoritmisk tenkning (Utdanningsdirektoratet, 2019a). Vi ønsker å undersøke hvordan elementer av algoritmisk tenkning fremmes i matematikklæreverk på ungdomstrinnet, og nøkkelbegrepene og arbeidsmetodene fra «den algoritmiske tenkeren» er slike elementer.

Etter undersøkelse av formuleringer i flere rammeverk (Benton et al., 2016; Brennan & Resnick, 2012), begrepsbeskrivelser, og Utdanningsdirektoratets beskrivelse av algoritmisk tenkning, mener vi at flere av handlingene fra Bråting og Kilhamns (2022) rammeverk samsvarer med flere av elementene fra «den algoritmiske tenkeren». Handling a) *følge prosedyre* kan knyttes til nøkkelbegrepet «algoritmer», som består av å følge regler og arbeide stegvis. Handling b) *finne regel* kan knyttes til nøkkelbegrepene «logikk» og «mønstre», hvor man arbeider med å analysere, og finne og bruke likheter. Handling c) *feilsøke* samsvarer med arbeidsmetoden «feilsøke». Handling d) *forme og skape* kan vi knytte til arbeidsmetodene «fikle» og «skape», hvor elevene skal utforske, eksperimentere, designe, og lage. Nøkkelbegrepet «dekomposisjon» er også relevant her, ettersom når elevene løser oppgaver ved bruk av d) *forme og skape*, er de ofte nødt til å dele opp oppgaven og løse den i flere forskjellige deloppgaver. Elevene må også gjøre kontinuerlige «evalueringer» når de former og skaper, så dette nøkkelbegrepet er også relevant (Sevik et al., 2016, s.15-16). Arbeidsmetoden «holde ut» inngår mulig under både c) *feilsøke* og d) *forme og skape*. Denne sammenhengen trekker vi ut fra Stenseth et al. s (2019) beskrivelse av iterasjon og programmering. Det er sjeldent at et program kjører feilfritt første

gang, og for å lykkes med utviklingen av programmer kreves det utholdenhet og tålmodighet (Stenseth et al., 2019, s. 8). Handling e) *forklare* kan inngå i å gjøre «vurderinger», følge «logikk», og finne «mønstre». Handling f) *forestille seg* går inn under nøkkelbegreper som «logikk», og evnen til å forutse (Utdanningsdirektoratet, 2019a).

# 3 Metode

Etter arbeidet med utviklingen av forskningsspørsmålet vårt og undersøkelse av lignende forskning, ønsket vi å gjennomføre en innholdsanalyse av læreverk ved bruk av det analytiske rammeverket fra Bråting og Kilhamn (2022). I dette kapittelet skal vi beskrive valg av forskningsmetode, begrunne metodiske valg, beskrivelse og bruk av rammeverk, og beskrive forskningsprosessen vår. Vi skal også begrunne datautvalg, og undersøke spørsmålet om validitet og relabilitet i vår forskning.

## 3.1 Innholdsanalyse

Det er flere grunner til at vi har valgt å foreta en innholdsanalyse av læreverk. Som nevnt i kapittelet om teori, fremheves det i faglitteraturen at mange lærere støtter seg mye til læreverk når det kommer til matematikkundervisning. Derfor mener vi det er viktig å undersøke hvordan læreverkene som blir brukt i norsk ungdomsskole er utformet. Mange lærere gir også utrykk for at de mangler tilstrekkelig kompetanse for å undervise programmering i skolen (Bråting & Kilhamn, 2022, s. 596). På bakgrunn av dette ønsker vi å få et innblikk i hvordan programmering og algoritmisk tenkning fremmes i læreverkene. Dette mener vi kan være nyttig for lærere som underviser på ungdomstrinnet. Integreringen av programmering i matematikkfaget er såpass nytt at det kan være interessant å se på hvordan læreverk og læreplan samspiller.

Vi ønsker å gjøre en innholdsanalyse, blant annet på bakgrunn av at vi har sett at lignende forskning har brukt denne metoden (Bråting & Kilhamn, 2022). Vi har derfor i denne forskningen også valgt å gjøre det. Innholdsanalyse er en metode for å analysere dokumenter og tekster som søker å kvantifisere innholdet i henhold til forhåndsbestemte kategorier, på en systematisk og gjentagende måte. Dette inkluderer alt fra lærebøker og digitale ressurser, til intervjuer og annen relevant tekstbasert informasjon (Bryman, 2012, s. 285; Lune & Berg, 2017, s. 182).

Hensikten med innholdsanalyse er å identifisere og kvantifisere spesifikke egenskaper ved innholdet, for eksempel temaer, mønstre, holdninger, eller verdier. Dette kan gi forskere innsikt i hvordan bestemte temaer eller holdninger blir presentert i læringsmateriale, og hvordan

spesifikke grupper uttrykker seg om visse emner (Lune & Berg, 2017, s.181-217; Bryman, 2012, s. 287-288).

### 3.1.1 Kvalitativ innholdsanalyse

Innholdsanalyse blir i følge Bryman (2012, s.285) oftest brukt i kvantitativ forskning, men kan også brukes i kvalitativ forskning. I vår forskning har vi valgt å anvende kvalitativ innholdsanalyse. Dette er for å dyptgående undersøke og forstå programmeringsoppgavene i matematikklæreverkene. Vi mener at kvalitativ innholdsanalyse som en tilnærming kan styrke vår forståelse av hvordan temaer og mønstre kommer frem i læreverkene etter innføringen av LK20. Noe som har påvirket vårt valg av metodisk tilnærming er hvordan forskningsspørsmålet vårt har formet seg under utforskningsprosessen og hvilken analyse vi mener er nødvendig for å svare på dette. Kvalitativ innholdsanalyse kan fremheve vår undersøkelse av elementer knyttet til algoritmisk tenkning i programmeringsinnholdet på ungdomstrinnet.

## 3.2 Analytisk rammeverk

Uavhengig av metodevalg er det hensiktsmessig å følge en systematisk tilnærming til analysen, og tydelig definere kriteriene for hva som skal analyseres og på hvilken måte. Dette kan sikre en grundig og pålitelig analyse av innholdet i læreverk for å belyse hvordan algoritmisk tenkning fremstår gjennom programmeringsoppgaver i datamaterialet (Lune & Berg, 2017, s.182). Bråting og Kilhamn (2022, S.598) presenterer et analytisk rammeverk som beskriver seks ulike handlingskategorier. Disse er, som nevnt tidligere: a) *følge prosedyre*, b) *finne regel*, c) *feilsøke*, d) *forme og skape*, e) *forklare*, og f) *forestille seg*. Rammeverket kan deles inn i flere tydelig definerte deler, og hver av disse spiller en viktig rolle for å besvare forskningsspørsmålet.

### 3.2.1 Bakgrunn for rammeverk

Bråting og Kilhamn (2022, s. 598) presenterer et analytisk rammeverk som undersøker forskjellige læreverks tilnærminger for utvikling av elevers evne til algoritmisk tenkning. I sin forskning har de brukt dette rammeverket til å undersøke programmeringsoppgaver på svensk

barnetrinn. Disse programmeringsoppgavene var tenkt utført gjennom de tre typer programmeringsoppgaver som nevnt i teorikapittelet.

Rammeverket er basert på både (1) Benton et al. s (2016) 5E-er knyttet til «actions», og (2) Brennan og Resnicks (2012) ideer om algoritmisk tenkning. Ettersom disse to rammeverkene ble utviklet for (1) oppgavedesign og (2) algoritmisk tenkning, gjorde Bråting og Kilhamn (2022) noen tilpasninger da de utformet deres rammeverk. De endret og tilpasset rammeverket sitt ved å legge til en måte å snakke om matematiske ideer på, knyttet til oppgaver om programmering (Bråting & Kilhamn, 2022, s.599). Spor av Benton et al. s (2016) rammeverk kommer frem gjennom handlingene b) *finne regel*, c) *feilsøke*, e) *forklare,* og f) *forestille seg.* Brennan og Resnicks (2012) rammeverk finner man i handlingene b) *finne regel*, c) *feilsøke*, og d) *forme og skape*. Handling a) *følge prosedyre,* var noe Bråting og Kilhamn (2022) så seg nødt til å tilføye rammeverket sitt, ettersom den forekom vesentlig mye i deres datamateriale.

Bråting og Kilhamns (2022, s.595) forskning handler om hva som karakteriserer programmeringsinnhold, og hvordan programmering knyttes sammen med matematikk i svenske læreverk. Med vår forskning ønsker vi å undersøke hvordan elementer av algoritmisk tenkning fremheves i matematikklæreverk, utgitt etter LK20. Det er betydelig overlapping i vår og Bråting og Kilhamns (2022) forskningsområde og fokus, at det passer seg å anvende deres rammeverk.

Forskningen vår skiller seg fra deres på noen punkter. Vi unnlater å undersøke det matematiske innholdet i oppgavene, og å se sammenhengen mellom dette innholdet og programmering. Bråting og Kilhamn (2022, s. 599) har i sin forskning analysert «actions» og «concepts» separat, for så å se nærmere på den matematiske sammenhengen mellom disse to. «Concepts» refererer til programmeringsbegreper som løkker, variabler, og betingelser. Vi har valgt å kun se på «actions», handlinger, og utelatt «concepts». Dette valget har vi tatt for å beholdet et større fokus på algoritmisk tenkning, og ikke gå i dybden på programmeringskonsepter. Vi tar et dypdykk på forekomst av handlinger, og hvordan fordeling av handlingene er på tvers av de

forskjellige forlagene og trinnene. Noe annet som skiller vår forskning fra Bråting og Kilhamns (2022), er aldersgruppen vi har valgt å undersøke; ungdomstrinnet i norsk skole som alternativ til barnetrinnet i svensk skole.

### 3.2.2 Tilpasning av rammeverk

Analysen vår vil undersøke forekomsten av Bråting og Kilhamns (2022) seks handlingene i de forskjellige læreverkene. Dette har vi gjort gjennom bruk av tabell 1, som illustrerer hvordan de forskjellige oppgavene blir formulert i læreverkene. Tabell 1 presenteres senere i dette kapittelet. Det er flere grunner til at vi har laget en slik tabell. Vi har sett at det varierer mellom verkene hvordan de formulerer handlingene i oppgavene sine, og ved å lage en slik tabell ønsker vi å unngå og måtte vurdere hver handling enkeltvis. En kategorisering av handlingene kan være med på å fremme en mer konsekvent analyse, og kan også bidra til at vi unngår å gjøre store forskjeller fra verk til verk.

Under analysen undersøkte vi oppgavene ved å se på hvilke handlinger som krevdes av eleven. Dersom noen av handlingene var tvetydig og vanskelig å plassere, undersøkte vi de to analytiske rammeverkene Bråting og Kilhamn (2022) hadde tatt utgangspunkt i, nærmere. Her så vi om vi kunne finne formuleringer som tydet på at det var en handlingskategori handlingen kunne passe bedre inn under. Et eksempel på dette i vår analyse er at vi var usikre på hvor vi skulle plassere handlingen «utforsk». Da undersøkte vi rammeverket fra Benton et al. (2016), de 5 E-ene, for å se om det var noe der som pekte på hvilken handlingskategori den kunne passe inn under. De 5 E-ene finner man utdypning på i teorikapittelet. Der beskriver de handlingen «explore» som noe som innebærer å feilsøke, og det var da naturlig å kategorisere de oppgavene som inneholdte handlingen «utforsk» under c) *feilsøke*.

Et annet eksempel på en handling vi syntes var krevende å plassere, var «endre». Først tenkte vi dette kunne passe under c) *feilsøke*, men etter videre analyse oppdaget vi at den kunne passe bedre under d) *forme og skape*. Dette er fordi når man endrer noe, så krever det at man skaper noe nytt. Når man feilsøker, handler oppgavene mer om å finne en eventuell feil, og påpeke denne. Senere kan oppgaven be eleven om å endre feilen.

Dersom vi ikke fant svar i hverken Benton et al. (2016) eller Brennan og Resnicks (2012) rammeverk, undersøkte vi handlingenes generelle definisjon for å se hvilken handlingskategori vi skulle tildele dem. Et eksempel på dette er «undersøke», som vi første plasserte under c) *feilsøke,* ettersom det kan innebære utforskning. Etter å ha undersøkt verbet og oppgavene som inneholdt «å undersøke» nøyere, mener vi det handler om å se nøye på noe for å finne et mønster eller en sammenheng i en oppgave, ikke nødvendigvis bare en feil. Derfor endret vi dets kategori til handling b) *finne regel*.

## 3.3 Sammenligning: en todelt tilnærming

I tillegg til innholdsanalyse, skal vi gjøre to sammenligninger av verkene. Disse sammenligningene vil både undersøke verkene overfladisk, og gå i dybden og undersøke deres innhold knyttet til algoritmisk tenkning og programmering. Dette er fordi vi mener det er interessant å se på både likheter og ulikheter på tvers av forlagene, og samtidig gå i dybden på verkene og se hva som skiller dem fra hverandre. Charalambous et al. (2010) beskriver hvordan vi kan gjøre en slik sammenligning, med begrepene horisontal og vertikal sammenligning. Videre kommer en forklaring på hvordan slike sammenligninger kan gjøres.
Ved kombinasjonen av disse to sammenligningene kan vi presentere en grundig analyse av hvordan fagstoffet er presentert i læreverkene, forskjeller og likheter på tvers av forlagene, og hvordan handlingene presenteres for elevene. Denne metoden er spesielt tilpasset undersøkelse av læreverk (Charalambous et al., 2010, s. 145).

### 3.3.1 Horisontal sammenligning

En horisontal sammenligning involverer å se på hvordan temaer er bygd opp, strukturen til læreverkene, og pedagogisk tilnærming i verkene. I mer detalj kan dette komme frem i elementer som tittel på verk, antall bøker (grunnbok, oppgavebok, lærerveiledning), antall sider i bøkene, utgivelsesår, forfattere, og ekstramateriale (Charalambous et al., 2010, s. 123). For oss vil ikke den horisontale sammenligningen dekke hele læreverket, men kun de delene som handler om programmering og algoritmisk tenkning. I vår horisontale sammenligning vil vi se på

antall verk, antall oppgaver knyttet til programmering og algoritmisk tenkning, utgivelsesår, og forfattere.

### 3.3.2 Vertikal sammenligning

For å søke dypere i innholdet av læreverkene vil vi også gjennomføre en vertikal sammenligning. En vertikal sammenligning kan involvere å se på spesifikt matematisk innhold, definisjoner og regler, eksempler på oppgaver, modellert tenkning, potensielle kognitive krav, og mulige læringsstrategier gitt gjennom forklaringer og eksempler på oppgaver (Charalambous et al., 2010, s. 122). I vår vertikale sammenligning vil eksempler på oppgaver, og fordeling av handlinger være sentralt. Her kommer funn fra analysen gjort ved bruk av rammeverket fra Bråting og Kilhamn (2022) nyttig til.

## 3.4 Analyseprosessen

I dette delkapittelet skal vi beskrive analyseprosessen. Vi gjennomførte først en undersøkelse av nøkkelbegrepene «algoritmisk tenkning» og «programmering» for å få en bedre forståelse og oversikt av forskningsfeltet. Videre som en innledende utforskning i bruken av det analytiske rammeverket, gjennomførte vi en pilotanalyse med et utdrag fra Cappelen Damms hefte om programmering. Dette var for å evaluere om valg av analytisk rammeverk var egnet for analyse av oppgavene. Ut fra denne pilotanalysen fikk vi også kategorisert ulike handlingers synonymer under passende kategorier i rammeverket. Deretter samlet vi inn programmeringsoppgaver fra læreverk for ungdomstrinnet fra de tre største forlagene i Norge; Aschehoug, Cappelen Damm, og Gyldendal (Opsahl et al., 2024).

Oppgavene i *Utdrag fra Programmering 8-10* er ikke direkte del av vårt datamateriale, men brukes som eksempler på oppgaver senere i kapittelet. Grunnen til at vi bruker *Utdrag fra Programmering 8-10* som eksempler og vedlegg er fordi vi har undersøkt muligheter for å bruke læreverkene fra datamaterialet vårt, men fått avslag fra forlagene om å bruke dem i oppgaven. *Utdrag fra Programmering 8-10* fra Cappelen Damm er tilgjengelig for alle på internett.

Gjennom denne pilotanalysen kom det frem flere handlinger vi måtte ta stilling til, og plassere inn under de satte handlingskategoriene. Dette kunne til tider være krevende. Det var viktig at vi fikk en tilstrekkelig kontroll på dette før vi startet en mer omfattende analyse. Dette har vært med på å forebygge irregulariteter i den senere analysen. For å ytterligere sikre en jevn og presis analyse, gjorde vi store deler av analysen over en kort tidsperiode. Vi gjorde det slik for å holde strukturen vi har utformet, og tolkningen av rammeverket så lik som mulig over alle læreverkene.

Vi tok for oss en vekslende rekkefølge av bøkene fra de forskjellige forlagene. Grunnen til dette var for å unngå å gå inn i «ett spor» for ett forlag, og senere oppdage at mye må endres, på grunn av formuleringsforskjeller mellom verkene. Vi følte også et behov for å gå gjennom læreverkene flere ganger, for å unngå feil og irregulariteter i analysen. Analysen ble gjennomført i par, noe som gjorde det mulig for oss å diskutere tolkninger vi var usikre på. Vi mener dette mulig var med på å hindre en ensidig tolkning av datamaterialet. For å besvare forskningsspørsmålet gikk vi deretter i dybden ved å anvende det analytiske rammeverket på de utvalgte læreverkene. Analysen av datamaterialet ble gjort sammen i par, og gjennomført flere ganger, for å sikre at vi har kategorisert handlingene likt over alle læreverkene.

## 3.5 Analyse av programmeringsoppgaver med elementer av algoritmisk tenkning

I tabell 1 under, finner man en mer beskrivende oversikt over hvordan vi har brukt det analytiske rammeverket fra Bråting og Kilhamn (2022), med eksempler på formuleringer. I arbeidet med å undersøke hvordan elementer av algoritmisk tenkning kommer frem i læreverkene, har dette skjemaet vært med på å legge tydelige rammer på hvordan vi har kategorisert de forskjellige handlingene. Under analysen har vi brukt fargekoder for å kategorisere handlingene. I teoridelen har vi begrunnet vårt syn på hvordan disse handlingene er knyttet til elementer av algoritmisk tenkning, ved bruk av «den algoritmiske tenkeren».

| Handling | Beskrivelse | Eksempel på formulering | Eksempel fra datamaterialet |
|---|---|---|---|
| a) Følge prosedyre (rosa) | Å følge prosedyrer innebærer å følge stegvise instruksjoner, gjenta eller fortsette et mønster. | Kan inneholde formuleringer som «gjør x, så y, så z. …», «kjør koden/programmet» eller «bruk». | Vi fant ikke eksempel på denne i *Utdrag fra Programmering 8-10.* |
| b) Finne regel (gul) | Finne frem til framgangsmåten, regelen eller mønsteret som produserer det resultatet man er ute etter eller passer inn i situasjonen. | Kan inneholde formuleringer som «finn mønsteret», «finn regelen», «undersøk», «passer til», og «finn ut ...». | Se vedlegg 1. |
| c) Feilsøke (oransje) | Feilsøke i en kode, enten gitt eller noe de har lagd selv. | Kan inneholde formuleringer som «let etter», «juster», «test», «utforsk», og «prøv». | Se vedlegg 2. |
| d) Forme og skape (grønn) | Gi steg for steg instruksjoner, lag et mønster, skriv kode, og representere konsepter med symboler. | Kan inneholde formuleringer som «lag», «tilpass», «skriv», «endre», og «utvid». | Se vedlegg 3. |
| e) Forklare (blå) | Forklare med et naturlig språk, hvor man må beskrive en prosedyre, en regel, mønster eller et konsept. | Kan inneholde formuleringer som «forklar», «gi eksempel på», «sammenlign», «vurder», «diskuter», «beskriv» «betyr», og «begrunn». | Se vedlegg 4. |
| f) Forestille seg (lilla) | Forutse hva som vil skje, reflekter over mulige løsninger når betingelser eller verdier endres. | Kan inneholde formuleringer som «forestill deg», «i hvilke situasjoner», «finnes det», «hva skjer» og «tenke». | Se vedlegg 5. |

Tabell 1: oversikt over handlingenes formuleringer i læreverkene.

## 3.6 Utvalg av datamateriale

I vår analyse har vi valgt å undersøke syv læreverk for ungdomstrinnet, utgitt etter innførelsen av LK20. Dette er begrenset til ungdomstrinnet ettersom vårt forskningsspørsmål kun omhandler ungdomstrinnet. Denne begrensningen har vi gjort fordi vi mener det er en passende mengde å undersøke med tanke på omfanget av vår masteroppgave. Vi har valgt å se på både fysiske og utvalgte digitale læreverk, ettersom et av forlagene har utelatt programmering fra grunnbøkene. Vi har analysert oppgaver fra Aschehoug, Cappelen Damm, og Gyldendals læreverk.

### 3.6.1 Avgrensning og analyseenheter

Datamaterialet blir delvis avgrenset av forskningsspørsmålet. Videre avgrenser vi det med å begrense det til tre forlag; Aschehoug, Cappelen Damm, og Gyldendal. I følge Store Norske Leksikon (snl.no) er disse tre de største forlagene i Norge (Opsahl et al., 2020).

Vi har utelatt læringsressurser som lærerveiledning, digitale ressurser, oppgavebøker fra de nevnte forlagene, og læringsplattformer som Campus Inkrement. Grunnen til dette er at datautvalget hadde mulig blitt for uoversiktlig og omfattende. Cappelen Damm har valgt å utgi et tilleggsmateriale i form av en PDF-fil, som kun dekker programmeringsoppgaver. Dette materialet har de valgt å utelate fra de fysiske og digitale versjonene av grunnbøkene sine.
En grunn til at vi har utelatt forlagenes digitale plattformer er fordi de varierer mye i hvordan de er bygget opp. Aschehoug har A-univers, Cappelen Damm har Skolenmin, og Gyldendal har Skolestudio, som alle er lagt opp på forskjellige måter.

Læreverkene vi har analysert har en lignende oppbygning, som passer seg for en analyse. Alle verkene er grunnbøker i matematikk for ungdomstrinnet. Som nevnt over, er dette med unntak av datamateriale fra Cappelen Damm, hvor programmeringsmaterialet kommer i tillegg til grunnboken. Oppgavene om programmering er utformet i lik stil som oppgavene i Cappelen Damms grunnbøker, og vi har derfor valgt å behandle dem på lik måte som oppgavene i grunnbøkene fra Aschehoug og Gyldendal.

Vi har kun behandlet nummererte oppgaver i de utvalgte læreverkene, og ikke eksempler og forklaringer. Dette er fordi eksemplene ikke er noe elevene skal arbeide med, men heller kan bruke veiledende. Vi har definert en nummerert oppgave som en analyseenhet. En annen grunn til at det er passende å se på grunnbøker er, som nevnt tidligere, at mange lærere støtter seg mye til disse læreverkene, og de vil da ha stor innvirkning for planlegging og gjennomføring av undervisning i klasserommet (Bråting & Kilhamn, 2022, s. 596). Mange av eksemplene gjenspeiler også de oppgavene elevene senere vil arbeide med, og vil derfor ikke tilføye noe nytt til analysen. Vi har heller ikke sett på foreslåtte diskusjonsoppgaver som «snakk matte», og «utforsk», som man kan finne i flere av verkene. Disse finnes ikke i alle verkene, og vil være vanskelig inkludere i analysen ettersom vi tolker de som muntlige oppgaver og de skiller seg mye fra de nummererte oppgavene.

Det vi heller har valgt å inkludere, er oppgaver som er ment for elever som yter over evne. Aschehoug beskriver oppgavene sine «Topptur» og «Ekspedisjon» som utfordrende oppgaver som skal treffe de elevene som yter over evne (Kongsnes et al., 2020). Dette er fordi disse ikke skiller seg like mye fra de andre oppgavene i grunnbøkene, men har lignende oppbygning som de nummererte oppgavene.

Læreverkene fra Gyldendal og Aschehoug har i stor grad implementert programmering gradvis gjennom kapitlene, uten et eget kapittel for programmering. Cappelen Damm har et eget tilleggsdokument for programmering som skal dekke alle tre trinnene på ungdomsskolen. Dermed er datagrunnlaget for vår analyse:

- Matemagisk 8 (Kongsnes et al., 2020),
- Matemagisk 9 (Kongsnes et al., 2020),
- Matemagisk 10 (Kongsnes et al., 2021),
- Maximum 8 (Tofteberg et al., 2020),
- Maximum 9 (Tofteberg et al., 2021),
- Maximum 10 (Tofteberg et al., 2021).
- Programmering 8-10 (Hjardar, 2020)

## 3.7 Validitet og reliabilitet

I dette delkapittelet vil vi beskrive validiteten og reliabiliteten i denne forskningen. I følge Bryman (2012, s.41) så handler validitet og reliabilitet i forskningssammenheng om hvor gyldig og pålitelig en forskning og dens resultater er. Vår forskning undersøker matematikkoppgaver i læreverk, og i gjennomføringen av denne forskningen er det viktig at vi er bevisste på våre egne forkunnskaper, fordommer og vår forståelse av matematikk og programmering. Dette vil, uten at vi kan gjøre noe med det, påvirke vår analyse. Det vi kan gjøre, er å prøve og minimere i hvor stor grad det påvirker den. Vi har satt oss inn i relevant teori og forskning for å få en oversikt over fagfeltet, og for kunne ta bedre beslutninger med hensyn til utfordringer som kan oppstå før, under, og etter analysen. Vi har også tatt i bruk et veldefinert og konkret analytisk rammeverk for å minimere risikoen for at våre subjektive oppfatninger og individuelle tolkninger skal ha stor innvirkning på resultatene.

### 3.7.1.Validitet

I likhet med Bryman (2012), mener også Cohen et al. (2011, s.133), at det er fare for at forskernes forkunnskaper, fordommer og forståelse kan spille inn ved bruk av kvalitativ forskningsmetode. Dette kan påvirke forskningens validitet. Validitet i kvalitativ forskning har tidligere vært knyttet til forskerens ærlighet, dybde i datamaterialet, og hvordan eventuelle deltakere i forskningen er blitt behandlet. For å øke validiteten kan man i kvalitativ forskning være nøye ved utvalg av data. Det er umulig å få hundre prosent validitet i en slik forskning, ettersom subjektiviteten av mennesker involvert utvilsomt påvirker forskningen gjennom flere aspekter. Validitet burde derfor sees på som en skala, og ikke noe man kan oppnå (Cohen et al., 2011, s. 133-134).

I korte trekk, vil validitet i en kvalitativ forskning handle om 1) overførbarhet, 2) troverdighet, 3) pålitelighet, og 4) bekreftbarhet (Cohen et al., 2011, s. 134). Disse fire punktene er på utkikk etter om forskningen kan:

1. overføre resultater til lignende forskning,
2. gjenspeiler virkeligheten på en god måte,
3. er gjort på en pålitelig måte, uten å skjule deler av forskningsprosessen,

4. bekreftes gjennom en lignende undersøkelse.

For å minimere hvordan våre egne evner kan påvirke vår analyse og resultat, har vi beskrevet flere tiltak tidligere i kapittelet. Eksempler på dette er at vi har valgt å bruke et allerede utviklet og utprøvd rammeverk, at vi har analysert i par, og at vi har satt oss inn i relevant fagstoff forut vår undersøkelse. Dette er med på å øke validiteten til forskningen.

Vi har videre gjort datautvalget på grunnlag av læreverk fra de største forlagene i Norge, for å dekke mye av det materialet som blir brukt i norske klasserom. En av årsakene til at vi undersøker flere læreverk fra forskjellige forlag er for å sikre at vi har tilstrekkelig med data, for å kunne svare mer dekkende på forskningsspørsmålet vårt. Vi kan ikke hevde å dekke alt, ettersom vi kun har inkludert grunnbøkene fra disse forlagene. Vi vet av erfaring at plattformer som Campus Inkrement også blir brukt på mange skoler.

### 3.7.2 Relabilitet

Relabilitet i kvalitativ forskning kan brukes som et paraplybegrep for pålitelighet, kontinuitet, og replikerbarhet. Dette er spesielt knyttet til resultatene i forskningen, og spørsmålet: *«kan andre replisere din forskning å få lignende resultater ved å gjøre de samme undersøkelsene?».* Dette kan være krevende når man gjør kvalitativ forskning, grunnet forskernes egen involvering i forskningen. Relabilitet handler også om datamaterialet som brukes, hvordan dette datamaterialet samles inn, og hvordan det tolkes (Cohen et al., 2011, s. 146).

På tross av hvordan relabilitet beskrives over, presiserer Cohen et al. (2011, s.148) at to lignende forskninger, gjort av forskjellige parter, kan være reliabel selv om de får forskjellige resultater. Målet er ikke uniformitet, men at begge forskningene skal presentere sitt arbeid med transparens og gjøre rede for sine valg (Cohen et al., 2011, s.148). Dette betyr at dersom noen andre velger å gjøre lignende forskning, så vil nødvendigvis ikke dette påvirke relabiliteten

i vår forskning, skulle de få andre resultater. Dette gjelder så lenge vi er nøye med å vise transparens med hvordan vi har foretatt vår forskning.

Vi gjør leseren tydelig oppmerksom på hvor vi har hentet datamaterialet fra, og lister opp en oversikt over de forskjellige læreverkenes forfattere. Disse læreverkene er bøker som ikke vil endre seg over tid, og andre forskere kan undersøke samme bøker ved et senere tidspunkt. Det eneste som eventuelt kan endre seg og være vanskeligere å finne tilbake til, er Cappelen Damms PDF-fil om programmering. Lærebokanalyse er gjort på et skriftlig datamateriale som ikke endres over tid. Dette kan implisere en høyere grad av relabilitet, ettersom det skriftlig datamateriale kan etterprøves igjen og igjen (Bryman, 2012, s. 156).

Vi har også tydelig lagt frem det analytiske rammeverket vårt, og hvordan vi har brukt dette på datamaterialet. Rammeverket vi har brukt er utviklet av to anerkjente forskere som har forsket mye innenfor programmering og algoritmisktenkning. De har igjen tatt utgangspunkt i to rammeverk fra andre anerkjente forskere i feltet. Vi har lagt til flere eksempler på oppgaver som vedlegg.

Rammeverket fra Bråting og Kilhamn (2022) har klart definerte kategorier av handlinger, og veiledning på hvordan rammeverket skal brukes i sin artikkel. Det som heller mulig kan påvirke forskningen, er hvordan vårt datamateriale har forskjellig formuleringer fra Bråting og Kilhamns (2022) datamateriale. Vi som forskere må tolke disse handlingene og plassere dem i handlingskategorier.

### 3.7.3 Etiske betraktninger

Ved en innholdsanalyse av læreverk er det færre etiske hensyn å ta enn dersom man behandler mennesker, som i for eksempel et intervjuer eller undersøkelser. Vi behandler ingen personopplysninger, og det er ikke krav om søknad til NSD (Norsk Senter for forskningsdata). Det vi må tenke på, er hvordan vi fremstiller læreverkene vi undersøker, med hensyn til

forfatterne. Vår intensjon med forskningen er å undersøke forekomst av handlinger, fremme forskjeller, men ikke å rangere læreverkene.

# 4 Resultater

I dette kapittelet vil vi presentere resultatene fra vår analyse av læreverk i matematikk på ungdomstrinnet. Vi har anvendt det analytiske rammeverket utviklet av Bråting og Kilhamn (2022) som utgangspunkt for vår analyse. Etter å ha undersøkt datamaterialet har vi selv måttet gjøre noen modifikasjoner i rammeverket. Disse tilpasningene har vi redegjort for i metodekapittelet.

Vi har gjennomført en omfattende vertikal-, og horisontal sammenligning av læreverkene, og resultatene er derfor presentert under delkapitlene horisontal og vertikal sammenligning. I den horisontale sammenligningen ser vi på de mer overfladiske variasjonene i de ulike læreverkene. I den vertikale sammenligningen dykker vi dypere ned i hvert enkelt læreverk for å utforske dybden av programmeringsoppgavene og elementer av algoritmisk tenkning som fremheves i de forskjellige verkene. Disse elementene knytter vi til handlinger i vår analyse, som igjen er knyttet til nøkkelbegreper og arbeidsmetoder brukt til å beskrive algoritmisk tenkning av Utdanningsdirektoratet (2019a). Vi undersøker også hvilke typer programmering det forventes at elevene skal anvende i de forskjellige verkene. Dette vil gi oss en innsikt i hvordan læreverkene, gjennom oppgavene, legger til rette for å utvikle elevenes algoritmiske tenkning.

## 4.1 Horisontal sammenligning: i overflaten

Analysen er gjort på syv forskjellige læreverk; tre fra Aschehoug, tre fra Gyldendal, og ett fra Cappelen Damm. Til sammen har vi analysert 178 oppgaver, og funnet 550 handlinger. Tabell 2 under, viser funnene fra den horisontale sammenligning med oversikt over verkene, antall programmeringsoppgaver per verk, antall handlinger per verk, utgivelses år, og de forskjellige verkenes forfattere.

| Horisontal sammenligning | | | | | |
|---|---|---|---|---|---|
| **Forlag** | **Verk** | **Antall programmerings-oppgaver** | **Antall handlinger** | **Utgivelsesår** | **Forfattere** |
| **Aschehoug** | Matemagisk 8 | 65 | 179 | 2020 | A. L. Kongsnes, A. K. Wallace, A. Moholt, E. Ødegaard, M. Hvattum, og K. Sortland |
| | Matemagisk 9 | 26 | 99 | 2020 | A. L. Kongsnes, A. K. Wallace, M. Hvattum, K. Sortland, og E. Ødegaard. |
| | Matemagisk 10 | 27 | 73 | 2021 | A. L. Kongsnes, A. K. Wallace, M. Hvattum, E. Ødegaard, K. Sortland, H. Sjøtrø, og A. Moholt. |
| | Total | 118 | 351 | | |
| **Cappelen Damm** | Programmering 8-10 | 24 | 109 | 2021 | Espen Hjardar |
| **Gyldendal** | Maximum 8 | 16 | 47 | 2021 | L. W. T. Bråthe, J. Tangen, G. N. Tofteberg, I. M. Stedøy, og B. Alseth |
| | Maximum 9 | 11 | 18 | 2021 | L. W. T. Bråthe, J. Tangen, G. N. Tofteberg, I. M. Stedøy og B. Alseth. |
| | Maximum 10 | 9 | 25 | 2020 | J. Tangen, G. N. Tofteberg, I. M. Stedøy og L. W. T. Bråthe. |
| | Total | 36 | 90 | | |

*Tabell 2: oversikt over resultater fra horisontal sammenligning.*

Tabell 2 viser resultatene fra den horisontale sammenligningen. Fra dette kan vi se en tydelig forskjell i mengden programmeringsoppgaver fra de forskjellige forlagene. Verkene fra Aschehoug har flest programmeringsoppgaver, Gyldendal har nest flest, og Cappelen Damm har færrest.

## 4.2 Vertikal sammenligning: i dybden

I den vertikale sammenligningen har vi undersøkt den indre strukturen, og oppgavenes sammensetning i de enkelte læreverkene. Oppgavene vi har undersøkt varierer, der noen av oppgavene består av kun en enkelt oppgave og en enkel handling, mens andre oppgaver er bygget opp av flere deloppgaver, der hver deloppgave også kan inkludere flere forskjellige handlinger. Funnene i den vertikale sammenligningen vil være en oversikt over fordelingen og forekomsten av handlingene i de forskjellige verkene. Tabell 3 under viser oversikt over verkenes antall oppgaver, handlingsfordeling, og gjennomsnittlig handlinger per oppgave.

| **Vertikal sammenligning** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Forlag** | **Bok** | **Antall oppgaver i verk** | **a)** | **b)** | **c)** | **d)** | **e)** | **f)** | **Gj. snitt handlinger per oppg.** |
| **Aschehoug** | Matemagisk 8 | 65 | 27<br>15% | 0<br>0% | 8<br>4.4% | 80<br>44.7 % | 28<br>15.6 % | 36<br>20.1 % | 2.75 |
| | Matemagisk 9 | 26 | 15<br>15% | 3<br>3% | 2<br>2% | 36<br>36.4 % | 24<br>24.1 % | 19<br>19.1 % | 3.80 |
| | Matemagisk 10 | 27 | 8<br>11% | 0<br>0% | 3<br>4.1% | 42<br>57.5 % | 15<br>20.5 % | 5<br>6.8 % | 2.70 |
| | Total | 118 | 50 | 3 | 13 | 158 | 67 | 60 | - |
| | Gjennomsnitt | | 13.7% | 1% | 3.5% | 46.2 % | 20.1 % | 15.3 % | 3.08 |
| **Cappelen Damm** | Programmering 8-10 | 24 | 6<br>5.5% | 12<br>11% | 29<br>26.6 % | 37<br>33.9 % | 10<br>9.2 % | 15<br>13.8 % | 4.55 |
| **Gyldendal** | Maximum 8 | 16 | 1<br>2.1% | 8<br>17% | 6<br>12.8 % | 14<br>30% | 15<br>32% | 3<br>6.4 % | 2.93 |
| | Maximum 9 | 11 | 0<br>0% | 0<br>0% | 0<br>0% | 15<br>83.3 % | 2<br>11.1 % | 1<br>5.6 % | 1.64 |
| | Maximum 10 | 9 | 2<br>8% | 1<br>4% | 3<br>12% | 11<br>44% | 8<br>32% | 0<br>0% | 2.78 |
| | Total | 36 | 9 | 21 | 9 | 40 | 25 | 4 | - |
| | Gjennomsnitt | | 3.4% | 7% | 8.3% | 52.4 % | 25% | 4% | 2.45 |

*Tabell 3: oversikt over resultatene fra den vertikale sammenligningen.*

Av de 550 handlingene som forekommer, er det handling *d) forme og skape* som forekommer flest ganger i læreverkene, etterfulgt av *e) forklare*. Den handlingen som forekommer færrest ganger er *b) finne regel*.

### 4.2.1 Aschehoug

Under kommer funnene fra analysen av verkene til Aschehoug. I disse er det en overvekt av oppgaver som ber om handlingen *d) forme og skape*, etterfulgt av handling *e) forklare*. Gjennomsnittlig er 46,2% av handlingene funnet i verkene fra Aschehoug, handling d).

I *Matemagisk 8* er det 52 oppgaver som tar for seg programmering og algoritmisk tenkning. Handling *d) forme og skape* forekommer flest ganger, etterfulgt av handling *f) forestille seg*. Handling d) er i overvekt i verket, og står for 44,7% av handlingene man finner. Handling *b) finne regel* forekommer ikke i dette verket.

Verket har flere oppgaver enn de andre seks verkene. Vi ser at mange av oppgavene er lignende hverandre, og flere av dem kunne nok heller vært samlet som deloppgaver i en større oppgave. Noen eksempler på dette kommer tydelig frem i *Matemagisk 8*, hvor man finner flere oppgaver med lignende formulering rett etter hverandre:

- «Lag et program som ber brukeren skrive inn alderen sin» (Kongsnes et al., 2020, s. 89).
- «Lag et program som spør brukeren hva favorittfaget hennes er» (Kongsnes et al., 2020, s. 89).
- «Lag et program som spør brukeren hva hun heter, så hva favorittfaget hennes er, og printer svaret på skjermen» (Kongsnes et al., 2020, s. 89).

- «Lag et program som skriver dette: #**#» (Kongsnes et al., 2020, s. 92).
- «Lag et program som skriver dette: ####, #**#, #**#, ####» (Kongsnes et al., 2020, s. 92).

Disse oppgavene blir veldig like hverandre, og handlingen *d) forme og skape* går nesten over til å være *a) følge prosedyre*. Resultatet av mange oppgaver med få handlinger gjenspeiles også i gjennomsnittet av handlinger per oppgave i verket, som er 2,75. Dette gjennomsnittet er lavere enn det totale gjennomsnittet for verkene fra Aschehoug.

Verket inneholder oppgaver som legger til rette for at elevene kan velge mellom blokk- og tekstbasert programmering når de skal løse oppgaver, men tekstbasert programmering blir mest brukt i verkets eksempler på oppgaver. Vi finner ikke oppgaver hvor det forventes at eleven skal programmere analogt (se teorikapittel for en presentasjon av disse formene for programmering).

I *Matemagisk 9* er det 16 oppgaver som tar for seg programmering og algoritmisk tenkning. I dette læreverket er det handling *d) forme og skape* som forekommer flest ganger, etterfulgt av handling *e) forklare*. Handlingen d) står for 36,4% av handlingene som forekommer. Det er to handlinger som forekommer svært lite i dette verket; handling *b) finne regel* finner man i 3% av oppgavene, og handling *c) feilsøke* i kun 2% av oppgavene. Gjennomsnitt av antall handlinger per oppgave i Matemagisk 9 er 3,80 handlinger per oppgave. Dette er det høyeste gjennomsnittet i verkene fra Aschehoug. I *Matemagisk 9* finnes kun oppgaver hvor det forventes at eleven skal bruke tekstbasert programmering når de løser oppgaver.

I *Matemagisk 10* er det 23 oppgaver som tar for seg programmering og algoritmisk tenkning. Igjen forekommer handling *d) forme og skape* flest ganger. Etter handling d), forekommer handling *e) forklare* nest mest. I dette verket er det overvekt av oppgaver som inneholder handlingen *d)*. Hele 57,5% av handlingene man finner i verket ber om dette. Handling *b) finne regel* forekommer ingen ganger i dette verket. Handling *c) feilsøke* forekommer også lite, 4,1%. Gjennomsnitt av handlinger per oppgave i *Matemagisk 10* er lavest for verkene fra Aschehoug. Her er gjennomsnittet av handlinger som forekommer per oppgave 2,70. Samme som i *Matemagisk 9*, finner man kun oppgaver hvor det forventes at eleven skal bruke tekstbasert programmering når de arbeider med oppgavene.

### 4.2.2 Cappelen Damm

Analysen av verket fra Cappelen Damm skiller seg fra de andre læreverkene, ettersom de har samlet alt av programmeringsoppgaver i en PDF-fil kalt *Programmering 8-10*. I denne PDF-filen var det 24 oppgaver som tok for seg programmering og algoritmisk tenkning. Handling *d) forme og skape* forekommer flest ganger med 33,9%, etterfulgt av handling *c) feilsøk*e med 26,6%. Så høy prosentandel av handlingen *c) feilsøke* finner vi ikke i noen andre verk. Handling *a) følge prosedyre* forekommer færrest ganger, med 5,5%.

Dette verket inneholder alle de tre formene for programmering. Verket starter med oppgaver som inkluderer analog programmering, for så å gå over til blokkbasert programmering, og etter blokkbasert programmering introduseres tekstbasert programmering for elevene. Verket har også gjennomsnittlig flere handlinger per oppgave enn de seks andre som er undersøkt. Dette med tilsvarende 1,47 flere handlinger per oppgave enn Aschehoug, og 2,1 flere handlinger per oppgave mer enn i verkene fra Gyldendal.

### 4.2.3 Gyldendal

Totalt i læreverkene fra Gyldendal forkommer handlingen *d) forme og skape* flest ganger, 52,4%, etterfulgt av handling *e) forklare*, 25%. Dette er det forlaget med lavest gjennomsnitt for antall handlinger per oppgave av de tre undersøkte. Her er det gjennomsnittlig 2,45 handlinger per oppgave i verkene.

I *Maximum 8* er det 16 oppgaver som tar for seg programmering og algoritmisk tenkning. I dette læreverket er det handling *e) forklare* som forekommer flest ganger, 32%, etterfulgt av handling *d) forme og skape*, 30%. Dette er det eneste verket som ikke har handling *d)* som den handlingen som forekommer oftest. Oppgavene i verket ber elevene ofte om å forklare det de ser, og beskrive hva som skjer. Alle seks handlingene er til stede i verket, men handling *a) følg prosedyre* forekommer kun en gang. Handling *c) feilsøke* forekommer mer enn i de påfølgende læreverkene fra Gyldendal, 12,8%. Gjennomsnittlig har dette verket 2,93 handlinger per

oppgave. Av Gyldendals verk, er dette det verket med høyest gjennomsnittlig forekomst av handlinger per oppgave. Dette verket inneholder alle de tre formene for programmering.

I *Maximum 9* er det 11 oppgaver som tar for seg programmering og algoritmisk tenkning. I dette læreverket er det handling *d) forme og skape* som forekommer flest ganger, 83,3%, etterfulgt av handling *e) forklare*, 11,1%. Det forekommer ingen tilfeller av de tre første handlingene, *a) følge prosedyre, b) finne regel*, og *c) feilsøke*. Det er en stor overvekt av handling d), hvor 15 av 18 handlinger ber om at elevene skal forme og skape. Dette gjøres ofte med verbet «lag», og gjennom formuleringen «Lag et program som ...». Verket har også det laveste gjennomsnittet av handlinger per oppgave av alle verkene undersøkt. I *Maximum 9* fant vi kun programmeringsoppgaver som ba elevene løse oppgavene gjennom tekstbasert programmering.

I *Maximum 10* er det 9 oppgaver som tar for seg programmering og algoritmisk tenkning. Her er det overvekt av handling *d) forme og skape*, 44%, etterfulgt av *e) forklare*, 32%. Handling *f) forestille seg* fant vi ingen tilfeller av. Handling *b) finne regel* forekommer en enkelt gang. I *Maximum 10* fant vi også kun programmeringsoppgaver som ba elevene løse oppgavene gjennom tekstbasert programmering.

### 4.2.4 Variasjon i sammensetning av handlinger

Gjennom analysen vår har vi sett at det ofte varierer mellom verkene hvor mange handlinger som kreves av eleven i hver oppgave. Dette har betydning for videre drøfting, ettersom det kan ha mye å si for elevens utbytte av oppgavene. I tabell 3 er en oversikt over gjennomsnitt av handlinger per oppgave i de forskjellige verkene. Her ser vi en tydelig forskjell fra Aschehoug og Gyldendals gjennomsnittlige handlinger per oppgave, til Cappelen Damms gjennomsnittlige handlinger per oppgave. Cappelen Damm krever betydelig flere handlinger per oppgave enn de andre to forlagene.

Gjennomsnittlig antall handlinger per oppgave kan mulig knyttes til forskjellig tilnærming til elevveiledning i læreverkene. Denne forskjellen finner vi blant annet i mengden av handlinger

per oppgave, og plassering av handlingen *d) forme og skape* i en oppgave. Vi ser at noen av oppgavene fremstår som mer veilenende for elevene gjennom bruk av forskjellige handlinger fordelt gjennom prosessen med å forme og skape, mens andre oppgaver kan overlate mye av prosessen til elevene selv.

Et eksempel på en oppgave som veileder eleven gjennom prosessen med å forme og skape, finner man i *Programmering 8-10* (Hjardar, 2020, s. 19). Oppgaven skal løses gjennom blokkbasert programmering, men ber først elevene lage ei skisse av en iskrystallarm på papir. Videre blir elevene bedt om å analysere iskrystallarmen, og spør dem om hvordan de ville tegnet den uten å løfte blyanten. Så ber oppgaven elevene om å lage en pseudokode på grunnlag av skissen, og sier at dette skal hjelpe dem med kodingen senere. Mot slutten av oppgaven blir elevene bedt om å lage en algoritme for iskrystallarmen i Scratch og teste den, altså handlingene *d) forme og skape* og *c) feilsøke*. Oppgaven avsluttes med å spørre elevene om de mener algoritmen kan forenkles ved hjelp av en løkke, altså handling *f) forestille seg* (Hjardar, 2020, s. 19).

Et annet eksempel finner man også i *Programmering 8-10*, men denne gangen med tekstbasert programmering (Hjardar, 2020, s. 20). Det er den første oppgaven for tekstbasert programmering i verket, og den viser først kodelinjen: 1 print («Hello world»). Den første handlingen elevene møter på er *f) forestille seg*, og de blir spurt om hva de tror «print» betyr i koden. Etter dette blir de bedt om å skrive inn koden, og teste (feilsøke) den i Python. Videre spør oppgaven om elevene kan få koden til å «printe» ut noe annet (endre→ forme og skape). Til slutt ber den elevene om å lage (forme og skape) en enkel kode som printer ut navnet til eleven (Hjardar, 2020, s. 20).

En lignende oppgave finner man i *Matemagisk 8*. Oppgaven ber eleven om å få Python til å skrive (forme og skape) «Nå skal jeg lære å programmere!». Det er gitt et eksempel over på samme side som oppgaven, som skal veilede dem (Kongsnes et al., 2020, s. 86). Begge

oppgavene har som mål at elevene skal lære å bruke «print» funksjonen i Python, men med forskjellig fremgangsmåte og fokus fra læreverkene.

# 5 Diskusjon

Etter å ha gjennomført en detaljert undersøkelse og sammenligning av læreverkene, har vi lært mye om fordelingen og forventede handlinger av elevene knyttet til elementer av algoritmisk tenkning og programmering på ungdomstrinnet.

I dette kapittelet ønsker vi å drøfte funnene våre med fokus på hvordan de svarer på forskningsspørsmålet vårt: «*hvordan fremheves elementer av algoritmisk tenkning i programmeringsoppgavene i matematikklæreverk for ungdomstrinnet utgitt etter Kunnskapsløftet 2020?».* Dette skal vi svare på gjennom delspørsmålene:

3) *Hvordan er fordelingen av handlinger knyttet til elementer av algoritmisk tenkning i læreverkene?*
4) *Hva kan sammensetningen av elementene i programmeringsoppgavene bety for elevenes potensielle mulighet til utvikling av evne til algoritmisk tenkning?*

Først skal vi drøfte forekomst av handling *d) forme og skape*, så forekomsten av handling *c) feilsøke* og knytte disse to til kjerneelementet "Utforskning og problemløsning". Deretter skal vi drøfte handlinger som forekommer lite i verkene, samt sammensetning av handlinger i programmeringsoppgavene og hvilke typer programmering programmeringsinnholdet i læreverkene ber elevene anvende. I tillegg ønsker vi å undersøke hvordan våre funn samsvarer med annen lignende forskning. Til sist skal vi drøfte funnene våre knyttet til kompetansemålene i programmering på ungdomstrinnet.

## 5.1 Læreverkenes bruk av handlingen d) *forme og skape*

Handling *d) forme og skape* er tydelig den handlingen som forekommer flest ganger i læreverkene. I seks av syv verk finner man denne som den mest forekomne handlingen, ofte med stor overvekt. I et av verkene er det hele 83,3% av handlingene som kategoriseres som *d) forme og skape*.

«Forme og skape» er en omfattende handling. Bråting og Kilhamn (2022) mener denne handlingen innebærer å gi instruksjoner, lage mønster, skrive kode, og representere noe med symboler. Arbeidsmetodene «fikle» og «skape» er elementer av algoritmisk tenkning vi har knyttet til handling d) *forme og skape*, hvor elevene må utforske og eksperimentere, designe, og skape. I beskrivelsen fra Utdanningsdirektoratet av kjerneelementet «Utforskning og problemløsning» skal elevene utforske og problemløse gjennom algoritmisk tenking (Utdanningsdirektoratet, 2019c). Utforskning og feilsøking er en sentral del av det å lære seg å programmere, ettersom å forstå og lære av feil kan føre til en utvidet kunnskap om programmeringsverktøyet elevene bruker. Å systematisk kunne feilsøke er også en stor del av det å tenke algoritmisk gjennom programmering (Bråting & Kilhamn, 2022, s. 607). Selv om vi i vår bruk av Bråting og Kilhamns (2022) rammeverk, har plassert handlingen «utforsk» inn under *c) feilsøke*, inngår det også en del utforskning i handling *d) forme og skape*. Dette er positivt for elevene på flere områder, ettersom vi ser at handling *c) feilsøke* forekommer i adskillig mindre grad enn handling d). Dersom det kun ville vært mulig for elevene å utforske i de tilfellene vi har funnet handling c), ville dette vært for lite med hensyn til hvordan utforskning vektlegges i læreplanen knyttet til algoritmisk tenkning. Dette vil da si at selv om mange av verkene har lavere forekomst av *c) feilsøke*, kan de likevel inneholde en del utforskning.

### 5.1.1 Fra å d) *forme og skape* til å a) *følge prosedyre*

I den vertikale analysen kommer handling d) oftest frem i form av formuleringen «Lag et program som ...». Når mange av slike oppgaver kommer rett etter hverandre, vil det som tidligere var en handling d*) forme og skape* bli om til a) *følge prosedyre* for elevene som jobber med oppgavene.

Bråting og Kilhamn (2022, s. 599) beskriver a) *følge prosedyre* som å følge stegvise instruksjoner, repetere eller fortsette et mønster. Oppgavene beskrevet over i delkapittel 4.2.1 er kategorisert som d) *forme og skape* når man ser på dem som enkeltoppgaver, men dersom elevene gjør flere av dem etter hverandre mener vi at oppgavene får elevene til å repetere eller

fortsette et mønster. Dersom man har løst en av dem, er det ikke utfordrerne for elevene å løse den neste. Lite variasjon i programmeringsoppgavene i et læreverk kan være negativt for elevenes programmeringskompetanse og evne til algoritmisk tenkning, og kan føre til at elevene ikke får tilstrekkelig utfordring. Slike oppgaver kommer tydeligst frem i Matemagisk 8, men vi ser lignende innhold i flere av verkene. Til gjengjeld har Matemagisk 8 flest programmeringsoppgaver av alle verkene vi har undersøkt, og kan mulig tillate seg å ha med slike oppgaver.

Om disse oppgavene går fra å be elevene om å forme og skape til å be dem følge prosedyre avhenger av lærerens utvalg av oppgaver. Vi mener det er mulig at læreren sløyfer flere av disse oppgavene dersom de mener elevene ikke har bruk for så mange av dem. Men igjen er det ikke mange programmeringsoppgaver per verk, så det er mulig lærene trenger alle oppgavene de kan finne. Dette legger ansvar over på læreren i utvalg av oppgaver, og eventuelt om de er nødt til å finne flere oppgaver gjennom andre ressurser.

Slike oppgaver som nevnt over er bare tilfeller i deler av handlingene kategorisert som d) *forme og skape*. Det finnes fremdeles en betydelig mengde med oppgaver som gir elevene utfordring og variasjon, og samsvarer mer med beskrivelsen av handlingen vi finner hos Bråting og Kilhamn (2022).

### 5.1.2 Prosessen av å forme og skape

Selv om et likhetstrekk mellom verkene er at de inneholder mye bruk av handlingen *d) forme og skape*, mener vi det er en forskjell i sammensetningen av handlinger brukt i oppgaver som inkluderer handling d), og hvordan verkene legger opp til at elevene skal forme og skape. Det er et skille mellom oppgaver som kun ber elevene om å forme og skape, og oppgavene som gradvis bygger opp til at eleven mot slutten av oppgaven skal forme og skape. Vi fant også variasjoner av tilfeller mellom disse sammensetningene. Den førstnevnte gir eleven mindre veiledning på hva og hvordan eleven skal gå frem for å løse programmeringsoppgaven. Den andre gir trolig eleven mer veiledning og tilrettelegger mulig for mer læring knyttet til

programmeringsprosessen. Dette mener vi fordi vi tror bruk av flere handlinger mulig gir eleven et bedre bilde på hva som forventes i utformingen av programmet, og en bedre forståelse av funksjonene som brukes under utformingen. Handlingene som observeres før og etter d) blir støttende handlinger i programmeringsprosessen, og kan ofte innebære at eleven skal forklare et begrep og funksjonen av dette begrepet, eller forestille seg noe. Ellers kan det være at oppgavene ber eleven feilsøke i en allerede skrevet kode, for så å lage lignende program eller utvide programmet.

Denne forskjellen i sammensetning av handlinger kommer også til syne gjennom gjennomsnittlig antall handlinger per oppgave i læreverkene. Cappelen Damm krever i gjennomsnitt betydelig flere handlinger av eleven per oppgave enn de andre verkene. Hvorfor dette er tilfellet ser vi tydelig gjennom eksempler på oppgaver som de beskrevet i delkapittelet 4.2.4. Det å gi elevene mer eller mindre veiledning kan være både positivt og negativt. Vi ser at flere handlinger per oppgave ofte gir elevene mulighet til å undersøke og forklare mer, hvor flere av handlingene som observeres i tillegg til *d) forme og skape,* er c) *feilsøke*, e) *forklare*, og f) *forestille seg*. I eksemplene i kapittel 4.2.4 ser vi at det er mulig elevene lærer mer om funksjonen «print», om elevene får arbeidet med flere forskjellige handlinger gjennom oppgaven.

Dersom oppgavene er laget for tekstbasert programmering krever det en større syntakskunnskap for å løse oppgavene, og elevene kan derfor trenge mer veiledning. Vi ser at det er flere læreverk som kun inneholder denne typen programmering. Dersom det blir for vanskelig er det fare for at flere av elevene kan treffe en «vegg» både på kreativitet og kompetanse innenfor programmering. Sevik et al. (2016) fremmer kreativitet som en viktig del av programmeringsprosessen. Et motargument til dette er dog at det med flere handlinger per oppgave muligens kan bli uoversiktlig og for krevende for elevene. I tillegg kan det være mange elever som får utforsket og arbeidet med problemløsning gjennom oppgaven dersom de ikke blir veiledet for mye.

## 5.2 Å kunne c) *feilsøke*

Handling *c) feilsøke* er handlingen som forekommer nest minst i læreverkene. Når elever engasjerer seg i feilsøking, praktiserer de algoritmisk tenkning ved å bryte ned problemer og vurdere løsninger (Wing, 2006). Dette er for å hjelpe elevene med å forstå og lære av feilene de gjør underveis. Algoritmisk tenkning er essensielt i arbeidet med utforskning og problemløsning (Utdanningsdirektoratet, 2019a).

### 5.2.1 Årsaker til lite forekomst av feilsøke

Bråting og Kilhamn (2022) beskriver feilsøking som et sentralt element for å utvikle evnen til å programmere. Med tanke på denne beskrivelsen av feilsøking, er det overraskende at denne handlingen ikke blir vektlagt mer i de svenske læreverkene som ble undersøkt (Bråting & Kilhamn, 2022, s. 607). Dette er også noe vi ser gjenspeiler seg i de norske læreverkene vi har analysert. Bråting og Kilhamn (2022, s.607) argumenterer for at feilsøking bør være en større del av programmeringsoppgaver i matematikkundervisningen for å hjelpe elevene med å forstå og lære av feilene de gjør underveis i programmeringsprosessen.

Bråting og Kilhamn (2022) mener at en mulig årsak til den begrensede bruken av *c) feilsøke* i svenske matematikkbøker kan ligge i hvordan programmering er blitt innført i det svenske skolesystemet. Innføringen av programmering er gjort ved å integrere det i matematikkfaget, og dette er gjort på samsvarende måte i Norge. Feilsøking er tradisjonelt knyttet tett til programmering, men ikke matematikk. Dette er en mulig grunn til at fokuset på feilsøking er mindre i læreverkene, da det er mulig forsøkt å tilpasse programmeringsinnholdet til undervisningen i matematikk, og ikke slik feilsøking tradisjonelt undervises i programmeringsfag (Bråting & Kilhamn, 2022, s. 607). Dette fordi programmering ikke er tenkt som et fag for seg selv, men som verktøy for elevene til å bruke og utforske i matematikkfaget (Stenseth et al., 2019). Vi mener utelatelsen av *c) feilsøke* vil være negativt for elevene ettersom det er sentralt for utviklingen av deres evne til algoritmisk tenkning og programmering, og kan knyttes direkte til kjerneelementet om «Utforskning og problemløsning» i LK20.

I kjerneelementet «Utforskning og problemløsning» knyttes algoritmisk tenkning tett opp til problemløsning. Vi vil påpeke at det er vanskelig for oss i denne forskningen å si om en oppgave er problemløsende, da vi ikke har undersøkt datamaterialet med fokus på problemløsende formuleringer. Det er også vanskelig av en annen grunn ettersom det varierer fra elev til elev hva som er et «problem» (Valbekmo & Stedøy, 2018, s. 4). Likevel er utforskning startfasen til problemløsning, og funn av utforskning i læreverkene kan være til støtte for potensiell problemløsning i klasserommet (Carlson & Bloom, 2005; Liljedahl, 2021). Feilsøking er per begrepets definisjon også problemløsende, ettersom man ser etter feil og problemer i et program og ønsker å løse disse (Stenseth et al., 2019, s.8).

### 5.2.2 Utforskning gjennom både c) *feilsøke* og d) *forme og skape*

Gjennom analysen vår har vi sett forskjeller på formene for utforskning i handling *c) feilsøke* og d) *forme og skape*. I handlingen *c) feilsøke* er utforskningen rettet mot noe spesifikt i programmeringsoppgaven som å teste et program, justere et programs resultat eller utforske en kode, noe som veileder elevenes utforskning. I d) *forme og skape* mener vi utforskningen oppstår mer naturlig og uveiledet gjennom oppgaveløsningen. Ettersom vi har funnet mest av handling d) i læreverkene vi har undersøkt, kommer mesteparten av utforskningen elevene vil foreta seg da være mer uveiledet, enn dersom vi hadde funnet mer av handling c).

Når elevene arbeider med oppgaver som handler om å forme og skape vil det mulig være stor variasjon i fremgangsmåtene de velge å bruke. I en klasse på 20 elever vil det mulig oppstå 20 forskjellige programmer. Når man skriver et eget program er det sjeldent at programmet kjører feilfritt fra første gang (Stenseth et al., 2019, s.8). Det kan være krevende for elever på ungdomstrinnet å programmere «fritt» i tekstbasert programmering med lite veiledning, ettersom det er mye forskjellig som kan være feil med programmet de utvikler. Siden vi har funnet mest av handling d) *forme og skape* vil elevene trolig gjøre mest uveiledet feilsøking og utforskning gjennom at de skaper programmer.

Denne uveiledede feilsøkingen kan da bli til en stor frustrasjon i klasserommet, og det kan være krevende for læreren å få tid til og hjelpe alle. Forsström og Kaufmann (2018) skriver om viktigheten av lærerens rolle i klasserommet når elevene arbeider med programmering. De merker seg at det er mange lærere som føler de mangler tilstrekkelig kompetanse innen programmering, og det kan da være krevende å hjelpe elever når oppgavene blir for avanserte (Forsström & Kaufmann, 2018, s. 19).

Av de tre formene for feilsøking Stenseth et al. (2019) beskriver, er den mest ønskede formen for feilsøking og feilrettingsforsøk en type hvor elevene er målrettet og bruker deres teoretiske og matematiske kunnskap aktivt. I denne formen for feilsøking skal de blant annet arbeide systematisk og prøve å forutse utfall av endringer de gjør i programmeringsprosessen. Mye uveiledet og tilfeldig feilsøking kan føre til at det blir mer tilfeldige endringer i arbeid med utforming av programmer, noe vi mener kan være mindre lærerikt og effektiv for flere av elevene. Vi mener læreverk med stor overvekt av oppgaver som ber elevene om å forme og skape kan gi lærerne enda større utfordringer i programmeringsundervisning, enn om oppgavene varierte mer mellom forskjellige typer handlinger og elementer.

### 5.3 Lite forekomst av handling a) *følge prosedyre* og b) *finne regel*

Gjennom vår analyse var det lite forekomst av handlingene *a) følge prosedyre* og *b) finne regel*. Bråting og Kilhamn (2022) så seg nødt til å tilføye handling a) *følge prosedyre* til rammeverket etter undersøkelse av datamateriale sitt, ettersom den forekom mye i de svenske læreverkene for barnetrinnet. Det er den eneste handlingen de ikke har tatt fra rammeverkene fra Brennan og Resnick (2012) eller Benton et al. (2016). Vi har ikke funnet denne handlingen i like stor grad som Bråting og Kilhamn (2022) gjorde. Grunnen til dette kan mulig være fordi vi undersøker et høyere alderstrinn, hvor det forventes mer avansert programmeringskunnskap av elevene og det ikke er behov for enkle oppgaver som ber dem følge prosedyre på lik måte som i lavere trinn. Vi ser ikke nødvendigvis negativt på funnet av få handlinger kategorisert som a) *følge prosedyre*, men merker oss forskjellen i læreverkene vi har undersøkt.

Handling *b) finne regel* er heller ikke mye til stede i læreverkene. Læreverkene Matemagisk 8, Matemagisk 10 (Aschehoug), og Maximum 9 (Gyldendal) har av til sammen 263 handlinger kategorisert, ingen som passer under *b) finne regel.* Å lete etter mønster og kunne se sammenhenger er spesifisert i kjerneelementet om utforskning og problemløsning (Utdanningsdirektoratet, 2019c). Her skiller våre funn seg fra funnene i Bråting og Kilhamn (2022), ettersom handling *b) finne regel* er deres tredje mest forekommende handling, mens det er den som forekommer minst i vår analyse. Handlingen b) *finne regel* kan være noe forfatterne ikke har ment er sentralt i oppgaver for å utvikle evne til programmering og algoritmisk tenkning, eller de kan ha ment at elevene får mulighet til dette gjennom andre handlinger. Man kan utforske en sammenheng gjennom c) *feilsøke*, forklare en sammenheng gjennom e) *forklare*, og bruke en sammenheng gjennom d) *forme og skape*. På den andre siden kan mangelen på handling b) *finne regel* skape hull i noen av elevenes evne til algoritmisk tenkning. Det at man kan få muligheten til å se sammenhenger og finne mønster i andre handlinger, betyr ikke at alle elever vil gjøre nettopp dette.

Å få handlingen spesifisert og eksplisitt formulert i oppgaver vil rette fokuset i oppgaven direkte mot elementet og handlingen, og ikke bare være en sekundær handling gjennom *d) forme og skape, e) forklare* eller c) *feilsøke.* Å finne mønster og sammenhenger står sentralt i både Wings (2006) artikkel om algoritmisk tenkning, og som nøkkelbegrep i «den algoritmiske tenkeren» (Utdanningsdirektoratet, 2019a). Utelatelsen av denne handlingen kan frata elevene muligheten for å kunne arbeide med å finne mønstre og bruke disse mønstrene når de skaper og former kode.

## 5.4 Sammensetningen av typer programmering i verkene

I vår analyse har vi funnet ut at *Programmering 8-10* fra Cappelen Damm og *Maximum 8 fra* Gyldendal inneholder alle de tre formene for programmering som vi nevner i teorikapittelet: analog-, blokk-, og tekstbasert programmering. Denne sammensetningen og variasjonen i programmeringsinnholdet i et læreverk kan være fordelaktig da det muligens kan hjelpe elevene å forstå programmering bedre når de ser det fra flere perspektiver (Munasinghe et al.,

2023, s. 59). Derfor kan de verkene som inneholder alle tre formene til programmering være fordelaktige i utvikling av elevens evne til å programmere, og gi en grundigere forståelse av forskjellige programmeringsverktøy. Ved å utelate en eller flere tilnærminger til programmering kan man risikere at det blir mer utfordrende for elevene å lære seg og programmere og tenke algoritmisk (Munasinghe et al., 2023, s. 59-60). Alle forlagene bortsett fra Aschehoug har inkludert alle formene for programmering i et eller flere av læreverkene sine. Valget fra Aschehoug om å utelate noen av programmeringsformene mener vi kan knyttes til at verkene er laget for ungdomstrinnet, og de tenker mulig at analogprogrammering passer bedre for læreverk på lavere trinn.

## 5.5 Kompetansemål sammenlignet med handlinger i læreverk

På hvert trinn i ungdomsskolen finnes det kompetansemål som omhandler programmering. På 8.trinn skal elevene «utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering» (Utdanningsdirektoratet, 2019d). I læreverkene på 8. trinn ser vi at det forekommer mye *d) forme og skape*, men mindre *c) feilsøke*. Vi opplever likevel at programmeringsoppgavene vi har analysert er dekkende for kompetansemålet over.

På 9. trinn skal elevene lære å «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering.» (Utdanningsdirektoratet, 2019e). Vi mener dette ikke er noe vi kan kommentere på etter vår analyse, ettersom vi ikke har undersøkt spesifikt matematisk innhold.

På 10. trinn skal elevene lære å «utforske matematiske egenskaper og sammenhenger ved å bruke programmering.» (Utdanningsdirektoratet, 2019f). Med unntak av Cappelen Damms læreverk, er det lite av handling c) *feilsøke* i verkene på 10. trinn. Imidlertid er det mye av handling *d) forme og skape*, og elevene får da gjennom dette prøvd seg på utforskning. Handling *e) forklare* kan også være støttende for dette kompetansemålet. Denne handlingen finner vi mye av i 10. trinns verkene. Vi kan ikke kommentere på om de får muligheten til å se på matematiske egenskaper gjennom oppgavene. Det vi ser gjennom de handlingene vi har

funnet i 10. trinns verkene, er at de får mulighet til å utforske gjennom handling d) *forme og skape*, og blir mulig bedt om å forklare sammenhenger gjennom handling e) *forklare*.

# 6 Avslutning

Vår forskning har hatt som formål å undersøke læreverk på ungdomstrinnet, innen faget matematikk med spesielt fokus på algoritmisk tenkning og programmeringsinnhold. Vi ønsket å se hvilke elementer av algoritmisk tenkning programmeringsoppgavene vektla, og hvordan de fremmet algoritmisk tenkning. Gjennom vår analyse og bruk av rammeverket fra Bråting og Kilhamn (2022) så vi et klarere bilde på hvordan elementer av algoritmisk tenkning fremmes i læreverkene på ungdomstrinnet. Forskningsspørsmålet vi kom frem til er formulert slik:

- H*vordan fremheves elementer av algoritmisk tenkning i programmeringsoppgavene i matematikklæreverk for ungdomstrinnet utgitt etter Kunnskapsløftet 2020?*

Under skal vi oppsummere forskningen ut ifra to delspørsmål:

5) *Hvordan er fordelingen av handlinger knyttet til elementer av algoritmisk tenkning i læreverkene?*
6) *Hva kan sammensetningen av elementene i programmeringsoppgavene bety for elevenes potensielle mulighet til utvikling av evne til algoritmisk tenkning?*

## 6.1 Konklusjon

Handlingen *forme og skape* kommer klart frem som den mest forekommende handlingen i vår analyse, noe som indikerer at elevene i stor grad oppfordres til å skape og eksperimentere med programmering. Handlingen omfatter en rekke aktiviteter, som å gi instruksjoner, lage mønstre, skrive kode og representere konsepter med symboler. Handlingen er også knyttet til elementer som «fikle» og «skape», hvor eleven skal utforske, eksperimentere, designe og lage. Denne handlingen og elementet kommer oftest frem i form av formuleringen «Lag et program som...», men også gjennom formuleringer som ber elevene «utvide programmet» og «endre programmet». Mens dette er positivt for å stimulere elevenes kreativitet og utforskning, må det bemerkes at vi mener overvekten av denne handlingen kan legge mye ansvar på læreren. Vi mener oppgavene kan være vanskelig å veilede dersom man som lærer føler at man ikke har tilstrekkelig kompetanse.

Videre ser vi at handlingen *forme og skape* fremheves på en annen måte i læreverkene. I diskusjonen har vi poengtert hvordan oppgaver som opprinnelig falt under handlingen *forme og skape*, kan gå over til å være handlingen *følge prosedyre*, når disse oppgavene gjøres rett etter hverandre. Dette kan reduserer graden av utfordring og variasjon for elevene, og det er viktig at læreverkene tilbyr oppgaver som stimulerer elevenes kreative og utforskende ferdigheter.

Selv om handlingen *forme og skape* omhandler mye, og gir elevene rom til å utforske og eksperimentere med programmering, ser man i flere av verkene at det er lite forekomst av handlingen *feilsøke.* Vi mener utforskningen gjennom feilsøking er mer målrettet enn utforskningen elevene gjør gjennom forming og skapning, ettersom den førstnevnte er mer spesifikk. Denne handlingen er sentral for elevenes utforskning og algoritmiske tenkning ettersom det er den handlingen i Bråting og Kilhamns (2022) rammeverk som direkte knyttes til utforskning. Vi ser at tilstrekkelig bruk av formuleringer som feilsøk, test, prøv og juster er manglende i flere av verkene. Vi mener en kombinasjon av utforskning gjennom både handlingene *forme og skape* og *feilsøke* er sentrale for å kunne gi elevene mulighet til å arbeide med algoritmisk tenkning, som kjerneelementet «Utforskning og problemløsning» ønsker.

Oppgavene i de forskjellige læreverkene varierer også i sammensetningen av handlinger per oppgaver. Noen av verkene har lavere gjennomsnitt av handlinger per oppgav, og lar oftere handlinger som ber elevene forme og skape stå alene i en oppgave. Andre læreverk har et høyere gjennomsnitt av handlinger per oppgave, hvor mange av oppgavene som er knyttet til å forme og skape blir støttet opp av andre handlinger som ofte ber elevene undersøke og forklare funksjoner og begreper knyttet til programmeringsprosessen. Her skiller læreverket fra Cappelen Damm seg ut med et høyt gjennomsnitt av handlinger per oppgave.

Det er også variasjon i sammensetningen av hvilke typer programmering læreverkene tar i bruk i programmeringsinnholdet. Alle verkene inneholder oppgaver med tekstbasert programmering, gjennom bruk av programmeringsverktøy som Python. Flere av verkene har også inkludert oppgaver som lar elevene programmere gjennom analog og blokkbasert

programmering, i tillegg til den tekstbasert programmeringen. Vi mener de læreverkene som inneholder alle de tre nevnte formene for programmering mulig kan legge mer til rette for elevenes utvikling av evne til programmering og algoritmisk tenkning.

Samlet sett indikerer våre funn at programmeringsinnholdet i læreverkene på ungdomstrinnet er dekkende for beskrivelsen av algoritmisk tenkning i kjerneelementet «Utforskning og problemløsning», men kan trenge mer variasjon i bruken av handlingen *forme og skape*, og flere tilfeller av handlinger som *feilsøke* og *finne regel* slik at elevene får øvet seg på flere elementer knyttet til algoritmisk tenkning og programmering. Her mener vi at det kunne vært bedre om læreverkene økte bruken av støttende handlinger som ber elevene forklare og forestille seg mer gjennom oppgavene. Utelatelsen av handlingen *finne regel* i flere læreverk gjør at elevene får en begrenset mulighet for å øke evne til å undersøke mønster og se sammenhenger. Den begrensede bruken av handlingen *feilsøke* gir elevene færre muligheter til å utforske. Vi mener dette overlater et større ansvar til læreren for å sørge for at elevene får arbeidet med utforskning, og mønstre og sammenhenger i arbeidet med å lære elevene å tenke algoritmisk.

## 6.2 Studiens begrensninger

I dette delkapittelet skal vi ta for oss begrensninger knyttet til vår forskning. Det er viktig å undersøke beslutninger vi har tatt gjennom forskningsprosessen, med et kritisk blikk. Dette skal vi gjøre gjennom å kommentere på utvalg av datamateriale, valg av metode og rammeverk, og innhold i analysen.

Vi har, som nevnt tidligere, valgt å undersøke de forskjellige forlagenes grunnbøker. Det er derfor en del undervisningsmateriale som er utelatt fra analysen. Utelatelsen av digitale læringsressurser begrenser oppgaven vår til kun grunnbøkene, og representerer derfor ikke alt programmeringsinnholdet som blir brukt på ungdomstrinnet i norsk skole. Dette innebærer da utelatelsen av blant annet de digitale universene fra forlagene. Dersom vi hadde undersøkt dette i tillegg, ville våre resultater mulig sett annerledes ut. Dette gjelder også utelatelsen av

oppgavebøker. Med et digitalt univers følger det med mer fleksibilitet og mulig en større frihet rundt oppgaveutformingen. Det kan også være at noen skoler og lærere kun bruker slike digitale plattformer.

Noe annet som begrenser vår forskning og sammenligning er at det er forskjell i antall bøker fra hvert forlag. Aschehoug og Gyldendal har begge en grunnbok for hvert trinn, mens Cappelen Damm kun har ett læreverk for alle tre trinnene. Dersom Cappelen Damm hadde hatt programmeringsoppgaver fordelt over tre læreverk er det igjen mulig at resultatene våre hadde sett annerledes ut.

Selv om læreverkene som ble analysert ble valgt med omhu og med interesse i å undersøke de mest brukte læreverkene i norsk ungdomsskole, så har vi ikke funnet tall som direkte sier at de vi har valgt er de mest brukte. Vi har heller ikke tall på hvor mange skoler som bruker digitale plattformer til fordel for grunnbøker.

At vi kun har valgt å analysere oppgavene med fokus på handlingene i dem, og ikke det spesifikke matematiske innholdet, gjør at våre resultater mulig er noe overfladiske. Dersom vi hadde undersøkt nærmere hvordan programmering og matematikk knyttes sammen i oppgavene, kunne vi mulig fått et mer helhetlig bilde av handlingenes variasjon og betydning i forskjellige situasjoner.

Selv om vi har anvendt rammeverket fra Bråting og Kilhamn (2022) og deres seks hovedkategorier, så har vi gjort tolkninger av ulike handlinger for å plassere dem under handlingskategorien vi mente passet best. Dette kan ha påvirket resultatet vårt, ettersom dette er knyttet til vår forståelse av handlingene. Andre forskere kan ville plassert dem i andre kategorier enn hva vi har gjort, basert på deres forståelse av handlingene.

## 6.3 Implikasjoner og veien videre

Forskningen vår identifiserer flere områder hvor læreverkene kan forbedres for å bedre støtte elevenes utvikling av algoritmisk tenkning. Videre forskning kan være nødvendig for å undersøke effekten av slike endringer på elevenes læringsutbytte over tid. Det kan også være nødvendig med mer forskning på det matematiske innholdet knyttet til programmeringsinnholdet for å se den fulle effekten av hvordan læreverkene påvirker elevenes mulighet for å utvikle evne til algoritmisk tenkning.

Resultatene i forskningen vår kan ha betydelige implikasjoner for lærere som står overfor valg av læreverk for undervisning av programmering og algoritmisk tenkning på ungdomstrinnet. Ved å analysere og sammenligne ulike læreverk har denne forskningen identifisert hvilke elementer som forekommer mye og lite i læreverkene, og sammensetningen av programmeringsinnholdet deres ved bruk av forskjellige typer programmering og oppgaveoppbygning. Lærerne kan ha fordel av å velge læreverk som gir en helhetlig dekning av ulike elementer av algoritmisk tenkning, og som inneholder de typene programmering lærerne vurderer passende for deres elevers behov.

Forfattere av matematikklæreverk på ungdomstrinnet kan også muligens dra nytte av forskningens funn. Ved å analysere hvilke elementer av algoritmisk tenkning som er fremtredende i læreverkene, og knytte dette til relevant teori, kan vi gi forfatterne innsikt i hva som er viktig for lærere og elever. Dette kan mulig veilede forfattere i utviklingen av nye læreverk ved å inkludere en variert og balansert dekning av ulike elementer av algoritmisk tenkning, og tilrettelegge for en progressiv læringsprosess. Videre kan forskningen bidra til å øke bevisstheten blant forfattere om betydningen av å inkludere varierte og utfordrende programmeringsoppgaver som fremmer utviklingen av elevers evne til algoritmiske tenkning.

Forlengelse av denne studien kan innebære å se på de digitale læringsplattformene vi har utelatt i denne studien. Alle tre forlagene har en digital plattform; A-univers, Skolenmin, og Skolestudio, som kan inneholde flere programmeringsoppgaver som potensielt kan inneholde

en annen sammensetning av handlinger. Vi har også unnlatt å undersøke plattformer som Campus Inkrement. Ettersom programmering ofte knyttes til å arbeide på datamaskin kunne det vært interessant å se på variasjonen av oppgaver gjennom bruk av digitale verktøy som disse plattformene.

Det kan også være interessant å nøyere undersøke det matematiske innholdet i programmeringsoppgavene på ungdomstrinnet. Dette mulig med et spesifikt fokus på hvordan oppgavene som inneholder handlingen *forme og skape* behandler spesifikt matematisk innhold, ettersom den er så fremtredende. Undersøkelse av disse punktene over kan gi et enda mer nyansert bilde av programmeringsinnholdet i matematikkfaget på ungdomstrinnet.

Det kan også være interessant å undersøke hvordan elevene løser oppgavene og deres løsningsprosesser. Dersom man skal se den fulle effekten av læreverkene kan det være sentralt å se på elevers bruk av algoritmisk tenkning og vurderinger i prosessen ved å løse programmeringsoppgaver på ulike måter. Dette vil mulig gi oss et bedre bilde på hvordan programmeringsoppgaver knyttet til algoritmisk tenkning kan være problemløsende, og innsikt i elevenes konkrete bruk av elementer knyttet til algoritmisk tenkning.

## Litteraturliste


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

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.

Bryman, A. (2012). *Social Research Methods*. Oxford University Press, USA.

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

Carlson, M. & Bloom, I. (2005). The Cyclic Nature of Problem Solving: An Emergent Multidimensional Problem-Solving Framework. *Educational Studies in Mathematics*, *58*, 45-75. https://doi.org/10.1007/s10649-005-0808-x

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

Cohen, L., Manion, L., Morrison, K. & Bell, R. C. (2011). *Research methods in education* (7th. utg.). Routledge.

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18-32.

Gjøvik, Ø. & Torkildsen, H. A. (2019). Algoritmisk tenkning. *Tangenten–tidsskrift for matematikkundervisning*, *30*(3), 31-37.

Hjardar, E. (2020). *PROGRAMMERING 8–10*
*fra CAPPELEN DAMM*. Cappelen Damm.

Hjardar, E. (2021). *Utdrag fra PROGRAMMERING 8-10* [1-12]. Cappelen Damm. https://issuu.com/cdundervisning/docs/smakebit_p_programmering_i_matematikk_8-10_fra_ca

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., Wallace, A. K., Hvattum, M., Sortland, K. & Ødegaard, E. (2020). *Matemagisk 9*. Aschehoug undervisning.

Kongsnes, A. L., Wallace, A. K., Hvattum, M., Ødegård, E., Sortland, K., Sjøtrø, H. & Moholt, A. (2021). *Matemagisk 10*. Aschehoug undervisning.

Kongsnes, A. L., Wallace, A. K., Sortland, K., Moholt, A., Ødegaard, E. & Hvattum, M. (2020). *Matemagisk 8*. Aschehoug undervisning.

Liljedahl, P. (2021). *Building Thinking Classrooms in Mathematics, Grades K-12 - 14 Teaching Practices for Enhancing Learning*. Corwin Press Inc.

Lune, H. & Berg, B. L. (2017). *Qualitative research methods for the social sciences*. Pearson.

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

Munasinghe, B., Bell, T. & Robins, A. (2023). Unplugged activities as a catalyst when teaching introductory programming. *Journal of Pedagogical Research*, *7*. https://doi.org/10.33902/JPR.202318546

Opsahl, P. C., Johannessen, L. B., Neraal, A. & Røhne, B. (2024). *forlag*. Store norske leksikon på snl.no. Hentet 26.mars fra https://snl.no/forlag

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books, Inc.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123. https://doi.org/10.1007/BF00191473

Román-González, M., Pérez-González, J.-C. & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., Mørken, K. M., Svorkmo, A.-G. & Voll, L. O. (2016). *Teknologi og programmering for alle*

*En faggjennomgang med forslag til endringer i grunnopplæringen- august 2016*. Utdanningsdirektoratet. Utdanningsdirektoratet. https://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf

Sevik, K., et al. (2016). *Programmering i skolen*. Senter for IKT i utdanningen. https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Statped.no. (2021a). *Programmering: Analog programmering*. Hentet Februar fra https://www.statped.no/laringsressurser/teknologitema/programmering-for-barn-med-saerskilte-behov/programmering/analog-programmering/

Statped.no. (2021b). *Programmering: Programmeringsspråk*. Hentet Februar fra https://www.statped.no/laringsressurser/teknologitema/programmering-for-barn-med-saerskilte-behov/programmering/programmeringssprak/

Statped.no. (2021c). *Programmering: Programmeringsverktøy*. Hentet Februar fra https://www.statped.no/laringsressurser/teknologitema/programmering-for-barn-med-saerskilte-behov/programmering/programmeringsverktoy/

Stenseth, B., Kaufmann, O. T. & Forsström, S. (2019). Programmering og matematikk. *Tangenten–tidsskrift for matematikkundervisning*, *30*(2), 7-12.

Tofteberg, G. N., Tangen, J., Bråthe, L. T. & Stedøy, I. (2021). *Maximum 10*. Gyldendal.

Tofteberg, G. N., Tangen, J., Bråthe, L. T., Stedøy, I. & Alseth, B. (2020). *Maximum 8*. Gyldendal.

Tofteberg, G. N., Tangen, J., Bråthe, L. T., Stedøy, I. & Alseth, B. (2021). *Maximum 9*. Gyldendal.

Utdanningsdirektoratet. (2019a). *Algoritmisk tenkning*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/kvalitet-og-kompetanse/digitalisering/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019b). *Grunnleggende ferdigheter (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05/om-faget/grunnleggende-ferdigheter?lang=nob

Utdanningsdirektoratet. (2019c). *Kjerneelementer*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05/om-faget/kjerneelementer?lang=nob

Utdanningsdirektoratet. (2019d). *Kompetansemål og vurdering - Kompetansemål etter 8. trinn (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05/kompetansemaal-og-vurdering/kv16?lang=nob

Utdanningsdirektoratet. (2019e). *Kompetansemål og vurdering - Kompetansemål etter 9. trinn (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05/kompetansemaal-og-vurdering/kv15?lang=nob

Utdanningsdirektoratet. (2019f). *Kompetansemål og vurdering - Kompetansemål etter 10. trinn (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05/kompetansemaal-og-vurdering/kv14?lang=nob

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

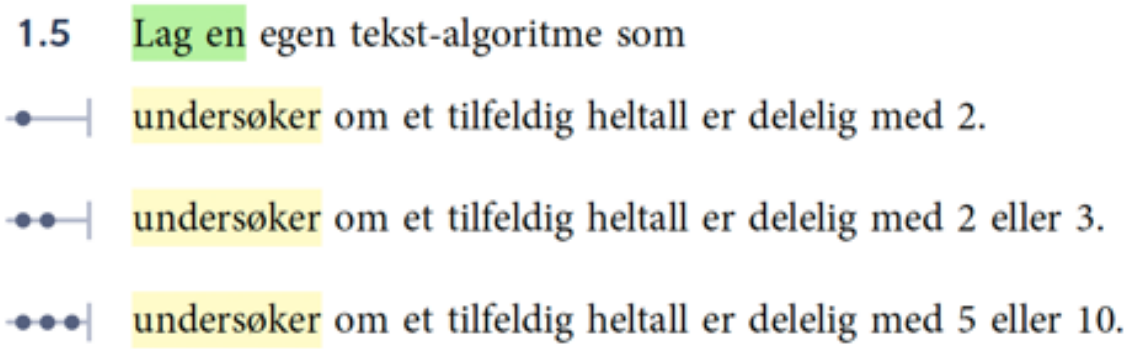
Valbekmo, I. & Stedøy, I. M. (2018). Problemløsing Hva er problemløsing, og hvordan skiller det seg fra arbeid med vanlige matematikkoppgaver? Hva kjennetegner en god problemløser? *Realfagsloyper.no*. https://www.matematikksenteret.no/sites/default/files/2022-04/Probleml%C3%B8sing.pdf

Vihovde, E. H. (2024). *programmering (IT)*. Hentet 25.mars fra https://snl.no/programmering_-_IT

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

# Vedlegg

## Vedlegg 1

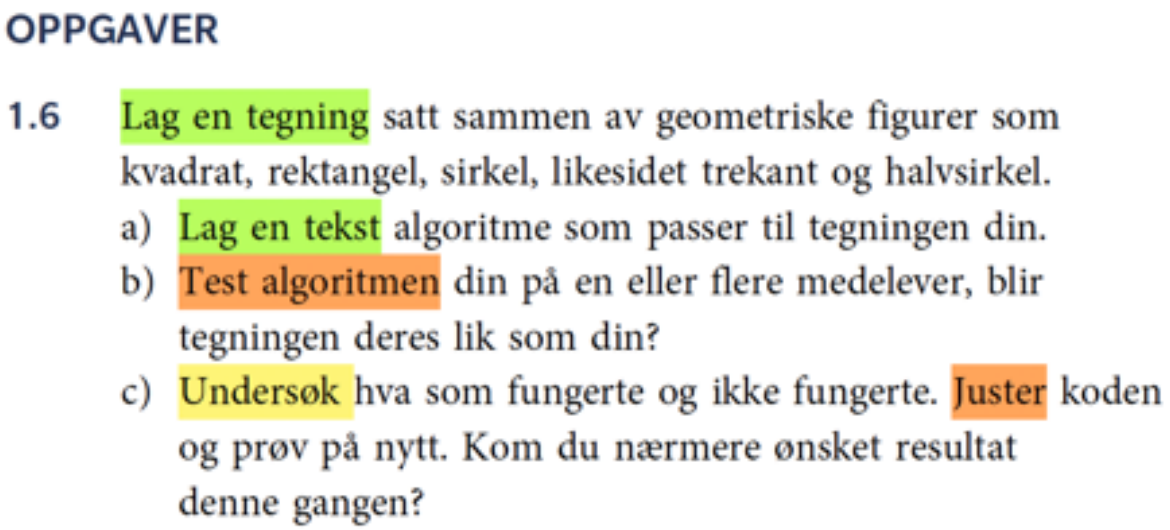


**1.5** Lag en egen tekst-algoritme som

- undersøker om et tilfeldig heltall er delelig med 2.
- undersøker om et tilfeldig heltall er delelig med 2 eller 3.
- undersøker om et tilfeldig heltall er delelig med 5 eller 10.

Vedlegg 1: Bilde av oppgave 1.5 hentet fra *Utdrag fra Programmering 8-10 (Hjardar, 2021, s. 7)*

## Vedlegg 2



**OPPGAVER**

**1.6** Lag en tegning satt sammen av geometriske figurer som kvadrat, rektangel, sirkel, likesidet trekant og halvsirkel.

a) Lag en tekst algoritme som passer til tegningen din.

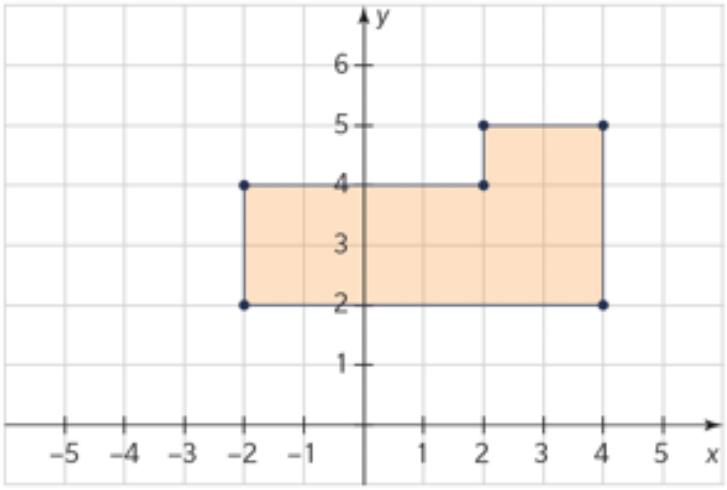
b) Test algoritmen din på en eller flere medelever, blir tegningen deres lik som din?

c) Undersøk hva som fungerte og ikke fungerte. Juster koden og prøv på nytt. Kom du nærmere ønsket resultat denne gangen?

Vedlegg 2: Bilde av oppgave 1.6 hentet fra *Utdrag fra Programmering 8-10* (Hjardar, 2021, s. 10)

## Vedlegg 3

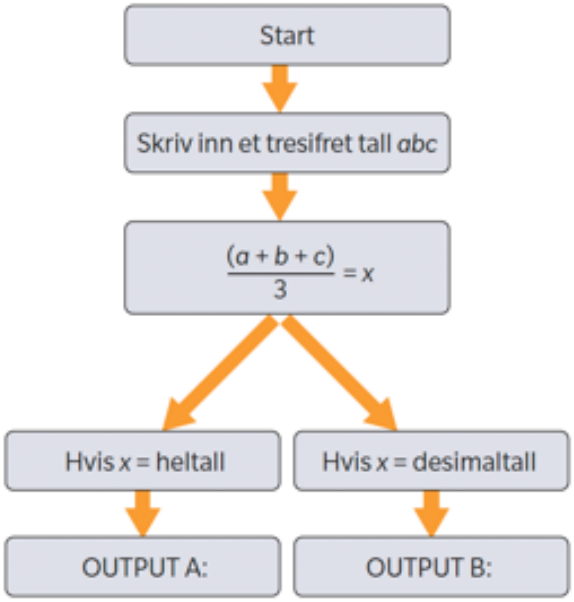
**1.7** Lag en algoritme som styrer en penn som tegner figuren under.
Du angir selv startpunkt, startretning og stoppunkt.




Vedlegg 3: Bilde av oppgave 1.7 hentet fra *Utdrag fra Programmering 8-10* (Hjardar, 2021, s. 10)

## Vedlegg 4

**1.4** Se på algoritmen når du svarer på oppgaven.




a) Undersøk og forklar hva algoritmen gjør eller undersøker.
b) Gi et eksempel på hva output A og hva output B kan være.

Vedlegg 4: Bilde av oppgave 1.4 hentet fra *Utdrag fra Programmering 8-10* (Hjardar, 2021, s. 6)

## Vedlegg 5

**1.1** Du skal nå lage en algoritme/kode gjør at du kan gå fra et startpunkt til et stoppunkt. Du velger selv hvor de to punktene skal være.

a) Lag en algoritme og test den på deg selv. La så en medelev teste algoritmen og se om eleven stopper på samme sted som deg.

b) Hvis medeleven ikke stoppet på samme sted som deg juster du algoritmen og prøver den på en annen medelev.

c) I hvilke situasjoner trenger du å legge inn en hvis-setning i algoritmen din?

d) Legg inn en hvis-setning i algoritmen din og test den på en medelev.

Vedlegg 5: Bilde av oppgave 1.1 hentet fra *Utdrag fra Programmering 8-10* (Hjardar, 2021, s. 5).