Masteroppgåve

NTNU
Noregs teknisk-naturvitskaplege universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

Marit Hystad

Elevar sin bruk av algoritmisk tenkning ved løysing av oppgåver basert på vilkår

Ein kvalitativ casestudie om algoritmisk tenkning og vilkår i ein programmeringskontekst

Masteroppgåve i matematikkdidaktikk
Rettleiar: Hermund André Torkildsen
Juni 2022

NTNU
Kunnskap for en bedre verden

Marit Hystad

Elevar sin bruk av algoritmisk tenkning ved løysing av oppgåver basert på vilkår

Ein kvalitativ casestudie om algoritmisk tenkning og vilkår i ein programmeringskontekst

Masteroppgåve i matematikkdidaktikk
Rettleiar: Hermund André Torkildsen
Juni 2022

Noregs teknisk-naturvitskaplege universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

NTNU
Kunnskap for ei betre verd

Samandrag

I lys av dei nye kompetansemåla om programmering i skulen, har kunnskap om algoritmisk tenking aldri vore meir dagsaktuell. Føremålet med denne studien er å undersøke kva som kjenneteiknar elevane på 9. trinn sin bruk av algoritmisk tenking når dei arbeider med oppgåver basert på vilkår. Masteroppgåva grunner i et konstruktivistisk læringssyn og bygger på Brennan & Resnick (2012) sitt rammeverk om algoritmisk tenking.

Dette er ein kvalitativ studie der observasjon og elevsvar på oppgåva på papir og ved hjelp av programmering er grunnlaget for datamaterialet, og der oppgåvene var basert på konseptet algoritmisk vilkår. Ein casestudie vart valt som metode slik at ein kunne undersøke kvar gruppe av elever individuelt, og deretter samanlikne dei. Ved å buke ein case metode, var det mogleg å undersøke dei ulike hendingane under observasjonen nøye.

Resultata viste eit skilje mellom korleis elevane brukte algoritmisk tenking når dei arbeidde med oppgåver på papir, samanlikna med når dei arbeidde med oppgåver i ein programmeringskontekst. Når elevane arbeide med oppgåver i ein programmeringskontekst så var deira algoritmiske tilnærmingar mykje meir tydeleg enn når dei arbeidde med oppgåver på papir.

I drøftinga diskuterte eg desse funna opp mot problemstillinga i oppgåva og studerte deretter korleis algoritmisk tenking kan vera med å styrke dei grunnleggjande ferdighetane til elevane og kjerneelement i matematikk faga. Til slutt vert det reflektert rundt forskingspørsmåla, og belyser nokre interessante tema og spørsmål som kan vera interessant for vidare forsking.

Abstract

Considering the new competence goals for programming the knowledge about computational thinking never been more important. The goal of this study is to find which characteristics 9th grade students' display using computational thinking when they are working with conditional problems. The thesis is based on a constructivist view of learning and is built on Brennan & Resnick's framework for computational thinking.

This is a qualitative study where observation and student answers to the problems by hand and with the help of programming tools is the basis of our data. The problems were based on the computational concept of conditionals. To evaluate the students, we chose to do a case study so we could evaluate both groups individually and compare them. By using a case study, it is possible to examine the separate events during observation carefully.

The results showed a difference between how the students used computational thinking when they worked with problems by hand and when they used programming. When the students worked on problems with programming their computational thinking was a lot more evident than when they worked with problems by hand.

I discussed these findings against my research question and then if computational thinking can strengthen the fundamental skills of the students and contribute to the education of the core principles of the mathematic subject. In the end I reflected on the research questions and shed some light on interesting subjects and questions that could be interesting for further research.

Forord

I løpet av denne masterutdanninga har eg fått moglegheitene til å utvikle meg som lærer og student. Dette har vært eit svært lærerikt løp med mange oppturar og nedturar. Eg vil med dette takke rettleiar for verdifull hjelp, konstruktive samtaler og eit hyggeleg samarbeid. Eg ønsker og å' takke medelevane på lesesalen på Kalvskinnet for eit trygt og godt arbeidsmiljø i ei utfordrande Pandemi tid. Til slutt vil eg takke for hjelp frå familie og venner som stilte opp som korrekturlesarar.

Trondheim, juni 2022
Marit Hystad

Innholdsfortegnelse

Figurliste

1 Innleiing

1.1 Læreplanen og programmering

I lys av dei nye kompetansemåla om programmering har kunnskap om algoritmisk tenking aldri vore meir dagsaktuelt. I den nye læreplanen som kom hausten 2020 blei programmering introdusert i fleire fag. Eit av dei var matematikk. Kunnskapsdepartementet skildrar faget og dei sentrale verdiane slik:

> *Matematikk er eit sentralt fag for å kunne forstå mønster og samanhengar i samfunnet og naturen gjennom modellering og anvendingar. Matematikk skal bidra til at elevane utviklar eit presist språk for resonnering, kritisk tenking og kommunikasjon gjennom abstraksjon og generalisering. Matematikk skal førebu elevane på eit samfunn og arbeidsliv i utvikling ved å gi dei kompetanse i utforsking og problemløysing(Kunnskapsdepartementet, 2019, s. 2).*

Dei trekk fram at målet for faget er å forberede eleven til «*et samfunn og arbeidsliv i utvikling ved å gi dem kompetanse i utforsking og problemløsing».* Teknologiane er i dag i stor endring. Dette fører til auka viktigheit for digitale ferdigheiter og kompetanse. Denne utviklinga har gjort at i land som England, Frankrike, Irland og Malta, er programmering og algoritmisk tenking (Computional thinking) ei prioritering i den obligatoriske undervisninga og skulegangen (Bocconi et al., 2018). Programmering er eit verktøy som hjelp elevane i deira utvikling av eit presist språk for resonnering, kritisk tenking og kommunikasjon gjennom abstraksjon og generalisering (Hemnes, 2018).

Kjerneelementa i kompetansemåla for matematikkfaget skal representere dei viktigaste elementa i faget og dei skal skildre det elevane skal meistre i løpet av skulegangen. Desse kjerneelementa står i seg sjølv uavhengig av klassetrinn og følgjer elevane frå første til tiande trinn. Forsking frå andre land der dei har meir erfaring med programmering i skulen enn det me har i den norske skulen, er ikkje nødvendigvis overførbar til det norske klasserommet. Ulike rammefaktorar skil klasseromma i ulike land ifrå kvarandre (Bocconi et al., 2018). Difor er det viktig å utforske programmeringsfaget og dei rammene og moglegheitene som det introduserer i norske klasserom.

Dei grunnlegane ferdigheitane er fem ferdigheiter som Kunnskapsdepartementet (2019) har avgjort skal vera gjennomgåande i alle fag. Utvikingen av desse ferdigheitane har betying for heile opplæringsløpet og er derfor viktige å utvkile i samband med eleven sin utvikling. Kunnskapsdepartementet (2019) fortel at desse ferdigheitane er del av den fagelege kompetansen og nødvendige redskaper for læring og forståelse. Dei er og med på å utvkile eleven sin identitet og sosiale relasjonar, og for å kunne delta i utdanning, arbeid og samfunnsliv.

Programmering omhandlar blant anna å lage ein programkode som kan løyse eit problem. Å utføre programmering er ikkje avgrensa til berre skrivinga av programkoden, men inneheld også prosessen før- og etter skrivinga av koden. Dette inneber å identifisere eit problem, finne moglege løysingar og feilsøking i koden, men også å kontinuerleg forbetre denne koden (Sevik, 2016, s. 9).

Programmering inngår og i Kunnskapsdepartementet si liste over digitale ferdigheiter som ein skal arbeide med i skulen. Kunnskapsdepartementet slår fast at kunnskap om verktøyet og når det bør nyttast, er vel så viktig som korleis ein skal bruke det(Kunnskapsdepartementet, 2019).

1.2 Skulefornying av fag og kompetanse i framtida

NOU-meldinga *Fremtidens skole, Fornyelse av fag og kompetanser* (NOU 2015:8, 2015) var eit svar på kva kompetansar ein har behov for i framtida og eit grunnlag for den nye læreplanen. Her kartla utvalet ulike kompetansar og kom fram til fire kompetanseområde som dei meinte var av stor tyding for elevane si utvikling i framtida. Media har ofte referert til denne meldinga som «Ludvigsen-utvalget».

Den første kompetansen dei presenterer er fagspesifikk kompetanse der det vert fokusert på fagområda matematikk, naturfag og teknologi, språk, praktiske og estetiske fag, og samfunns- og etikkfag. Den fagspesifikke kompetansen er spesielt viktig i samband med yrkes- og utdanningsval, men den er også sentral i allmennutdanninga. Under denne kompetansen vert det spesielt vist til problemløysing og kritisk tenking:

> *«Vitenskapelige metoder og tenkemåter er relevante for fremtiden, og dette ses i sammenheng med behovet for å kunne tenke kritisk og løse problemer» (NOU & 2015:8, 2015, s. 25)*

Med utgangspunkt i dette utdraget der problemløysing og kritisk tenking vert understreka, er det naturleg å knytte programmering og algoritmisk tenking opp mot denne kompetansen.

«Å kunne lære» er det neste kompetanseområdet som blir behandla i Ludvigsen-utvalet. Dette omfattar metakognisjon og sjølvregulering, som er nødvendig for kontinuerleg læring. Dette kompetanseområdet vert utvikla i samband med dei andre kompetanseområda (NOU 2015:8).

> *Metakognisjon handler om å kunne reflektere over egen tenkning og læring. I læringssammenheng handler det om at elevene reflekterer over hvorfor de lærer, hva de har lært, og hvordan de lærer. Metakognisjon betyr også å kunne bruke tenkemåter og læringsstrategier aktivt og målrettet for å fremme egen læring. ("NOU 2015: 8 - regjeringen.no") (NOU 2015:8, s.25)*

Ofte vert metakognisjon og sjølvregulering simulert i programmeringsaktivitetar i skulen, dersom rammeverket og miljøet rundt eleven er lagt til rette for dette (Hemnes, 2018). Programmering kan også knytast til dette kompetanseområdet.

Det tredje kompetanseområdet som utvalet skildrar, er å kommunisere, samhandle og å delta. For det meste omfamnar dette dei munnlege ferdigheitene, i tillegg til ferdigheiter innan lesing og skriving. Utvalet trekk fram kompetanse innanfor å argumentere, diskutere og samarbeide som nødvendige ferdigheiter for framtida. Dersom ein arbeider med programmering i eit klassefellesskap, kan ein trene opp desse ferdigheitene. Dette

kan til dømes gjerast gjennom at elevane skal argumentere for val av løysingar, i tillegg til å hjelpe kvarandre, og ikkje minst spørje og formulere seg korrekt.

Det siste kompetanseområdet som utvalet skildrar, er å utforske og skape. Dei anbefaler at: «kreativitet, innovasjon, kritisk tenkning og problemløsning er kompetanser skolen bør bidra til at elevene utvikler» (NOU 2015:8, s.24). Utvalet argumenterer for at eit skapande menneske er nødvendig for arbeids- og samfunnslivet, både nasjonalt og internasjonalt. Dersom ein trenar på å finne løysingar på ulike problem i skulen og i arbeidssamanheng, kan ein bruke dei same strategiane i andre deler av livet. Her kan ein naturleg kople til programmering og evna til å skape situasjonar der ein må vera innovativ for å skape funksjonar som skal å løyse problem.

> *«Nysgjerrighet, utholdenhet, åpenhet for å se ting på nye måter og evne til å ta initiativ er viktige sider ved kompetansene. Unge mennesker er av natur undrende og utforskende, men nysgjerrighet må stimuleres for å utvikles» (NOU 2015:8, s.26).*

Kunnskapsdepartementet (KD) avgjorde i samarbeid med lærarar, pedagogar og andre fagfolk i 2019 at kompetanse innanfor programmering skulle bli innført i skulen frå 2020.

1.3 Algoritmisk tenking

Kjerneelementa i læreplanen blir presentert med algoritmisk tenking og programmering som undertema (Kunnskapsdepartementet, 2019). I kjerneelementa «utforsking og problemløysing» vert det framheva eit fokus på strategiar og framgangsmåtar, framføre berre det å finne løysinga på problemet eller oppgåva:

> *«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(Kunnskapsdepartementet, 2019, s. 2).»*

I dette utsegna frå utvalet vert og algoritmisk tenking vist til som ein viktig prosess for å utvikle desse strategiane og framgangsmåtane for problemløysing (Kunnskapsdepartementet, 2019). Denne prosessen inneber- og er avhengig av om desse problema kan løysast med eller utan digitale verktøy.

Wing (2008) definerer algoritmisk tenking som: «design systems, solve problems, and understand human behavior, by drawing on the concepts fundamental to computer science» (Wing, 2008, s.371). Denne definisjonen vert brukt av fleire forskarar i deira artiklar (Brennan & Resnick, 2012; Cui & Ng, 2021; Fessakis et al., 2013; Israel & Lash, 2020). Wing (2008) definere algoritmar som ein steg-for-steg prosedyre som tek informasjon og produserer noko som «output» eller eit resultat. Når ein set dette saman med øving blir det ein steg-for-seg prosedyre som kan brukast til å løyse problem, samstundes som ein kan bruke digitale verktøy for å løyse dei.

I denne masteroppgåva vel eg å bruke Brennan og Resnick(2012) sitt rammeverk for algoritmisk tenking som grunnlag for det teoretiske rammeverket. Dette rammeverket gir nokre tydelege rammer for å utforske temaet, og det gir moglegheiter for å samanlikne med andre tilsvarande studiar (Brennan & Resnick, 2012). Dette rammeverket byggjer

på ein tredeling av algoritmisk tenking, bestående av algoritmiske konsept, algoritmiske tilnærmingar og algoritmiske perspektiv.

1.4 Problemstilling

Føremålet med denne oppgåva er å bidra med kunnskap knytt til algoritmisk tenking og elevane sine arbeidsmetodar for løysing av oppgåver basert på det algoritmisk konseptet vilkår. Studien søkjer å gi innsikt i kva som kjenneteiknar elevane sitt arbeid når dei nyttar algoritmisk tenking. Dette blir gjort for å få innsikt i tankegangen til elevane ved å kategorisere dei ulike kjenneteikna som ein kan observere i elevane sitt arbeid, med det føremålet å knytte dette til den etterspurde kunnskapen om programmering som kom då den vart ein del av læreplanen. Meir kunnskap om kva som kjenneteiknar elevar sitt arbeid med programmering vil bidra til å lette lærarane si tilrettelegging for elevane i faget. Dette føremålet skal belysast gjennom to spørsmål:

1. **Kva kjenneteiknar 5 elevar sitt arbeid med oppgåver basert på vilkår?**

Dette er mi hovudproblemstilling, som både metodekapittelet og teorien vert basert på. Eg ynskjer å undersøke kva som kjenneteiknar elevane sitt arbeid med oppgåver basert på vilkår. Å kunne kjenne igjen og kategorisere desse kjenneteikna kan gi innsikt i elevane si tenking og den framgangsmåten dei bruker. Eg valte å observere fem elever då dei gjorde oppgåvene både på papir og i ein programmeringskontekst, slik at eg fekk eit breiare bilete av elevane sine kjenneteikn på algoritmisk tenking. Desse kjenneteikna vert kategorisert basert på Brennan og Resnick (2012) sine algoritmiske tilnærmingar. Eg vel i denne studien å bruke algoritmiske tilnærmingar som kjenneteikn, fordi Kunnskapsdepartementet viser til algoritmisk tenking i forklaringa til den nye læreplanen.

2. **Kva spor ser vi av algoritmisk tenking i eleven sitt arbeid både i ein programmeringskontekst og i samband med løysing av oppgåver på papir?**

Valet om å undersøke elevane sine løysingsmetoder både på papir og i ein programmeringskontekst, gav moglegheita til å undersøkje kva spor av algoritmisk tenking som kunne identifiserast og kategoriserast. Korleis dette viste seg i oppgåveløysinga på papir og ved arbeid på ein pc, var spørsmål som dukka opp undervegs i arbeidet og gav ein vidare mogelegheit til undersøking.

1.5 Metodisk val

Ved å bruke ein casestudie for sju elever på 9. trinn etablerte eg eit grunnlag for å kunne svare på problemstillinga. Elevane gjennomførte oppgåveløysinga saman i grupper på 3 elever, først arbeidde dei med eit oppgåvesett på papir med vilkår baserte oppgåver. Skuletimen vart observert av meg og elevane sine diskusjonar vart tatt opp og seinare transkribert. Deretter observerte eg to grupper på to og tre elever som hadde løyst oppgåver på dataprogrammet Scratch med fokus på vilkår. Elevane brukte i gjennomsnitt 25 min på papiroppgåvene og 35 min på oppgåvene på pc.
Ved å bruke det blokkbaserte programmeringsspråket Scratch som kontekst for oppgåvene elevane skulle løyse, får eleven noko visuelt å knytte til oppgåvene. Det blokkbaserte programmet fungerte som representasjonar for dei matematiske elementa i

oppgåvene. Elevane hadde også erfaringar med dette programmet, men nivået på dette grunnlaget varierte mellom gruppene. Utdanningsdirektoratet(2016) var tidleg ute med å anbefale dette programmet for bruk i programmering i skulen. Brennan & Resnick (2012)sitt rammeverk bruker også Scratch i sin studie om algoritmisk tenking.
Scratch er eit blokkbasert programmeringsspråk som vert laga av "the Lifelong Kindergarten group" ved «the MIT Lab». I dette programmet er dei ulike kommandoane visualisert ved ei blokk som ein kan legge saman med andre blokker. Til dømes i « viss X så» blokka kan ein bytte ut X med eit vilkår, og moglegheitene til å legge til hendingar etterpå. I oppgåvesettet som elevane arbeide med skulle dei lage eit program som skulle spørje spørsmål og deretter utførte ulike kalkulasjonar ut i frå svaret.
For å undersøke kjenneteikna til elevane sin løysingsmetode, bruke eg ei rekke ulike oppgåver basert på det algoritmiske konseptet vilkår i tillegg til programmeringsoppgåver som hadde liknande oppsett der eg såg etter utsegn og handlingar som passer inn i dei ulike kategoriane. Eg baserte kjenneteikna på Brennan og Resnick(2012) sitt rammeverk (Brennan & Resnick, 2012).

1.6 Oppgåveoppbygging

Teorigrunnlaget i denne masteroppgåva er basert på Brennan og Resnick(2012) sitt rammeverk. Dette gav grunnlaget for val av analysemetode og seinare analysen i oppgåva (Brennan & Resnick, 2012). For å bygge opp eit grundig teorigrunnlag tok eg først utganspunkt i Kong(2019) sin litteraturstudie over fagfeltet med algoritmisk tenking. Her skal eg presentere relevante omgrepet og kva som vert lagt i desse omgrepa. Vidare i dette teorikapittelet tek eg grundigare føre meg dei ulike algoritmiske tankemønstera (Computational thinking practices) basert på Brennan og Resnick(2012)sitt rammeverk.

I metodekapittelet grunngjev eg forskingsdesignet mitt for oppgåva. Her argumenterer eg for val av datamaterialet, som observasjon og innsamling av elevarbeid. Etter å ha presentert data som eg har samla inn, skal eg analysere dei. Til slutt presenterer eg spørsmålet om pålitelegheit og gyldigheit i oppgåva.

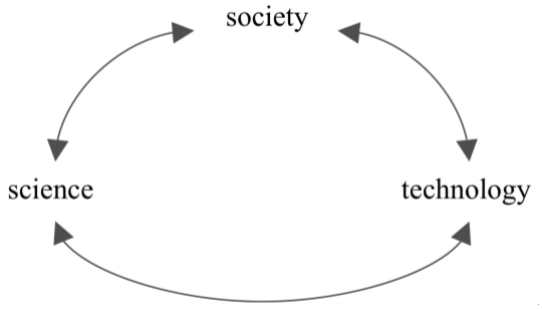
Etter metodekapittelet følgjer eit kapittel som består av resultata frå analysen. Her legg eg fram ulike hendingar der elevane sine utsegn fell innanfor dei ulike kjenneteikna som tidlegare vart definert. Her tek eg føre meg dei ulike hendingane som vert kategorisert og skilder dei meir inngåande. Til slutt samanliknar eg elevarbeidet som vart gjort på papir og elevarbeidet som dei gjorde ved bruk av Scratch, og ser etter likskapar og ulikskapar. I dei siste kapitla i denne oppgåva diskuterer eg funna opp mot teorien til Brennan og Resnick(2012)for deretter å drøfte dette opp mot forskingsspørsmåla mine.

2 Teori

2.1 Algoritmisk tenking

«Algoritmisk tenking bygger på krafta og begrensingane til dataprosessar, enten dei blir utført av eit menneske eller av ein maskin» (Wing, 2006 s 33). Algoritmisk tenking er altså knytt til dataprosess, både når dei vert utført på papir og ved hjelp av datamaskiner. I sin essens er algoritmisk tenking ein abstraksjon (Wing, 2008). Wing viser til at denne abstraheringsprosessen er lagvis og at ein arbeider ofte med minst to ulike lag. Desse to laga kan og skildrast som parallell prosessering, der ein må tenke på kode som data og data som kode.

Innanfor «computing» er det tre drivarar som har ein tovegs innverknad på kvarandre, nemleg samfunnet, teknologien og vitskapen (Wing, 2008). Modellen til Wing kan sjåast på som både ein sirkulær modell og ein sekvensiell modell. Dei ulike drivkreftene verkar på kvarandre og dannar ulike løkker. Modellen er sirkulær fordi ein kan for flytte seg frå samfunn til teknologi deretter til forsking og til slutt tilbake til samfunnet. Dette gjer at ein følger eit løp som går i ring. Den er og sekvensiell fordi det er relasjonar mellom dei ulike elementa kvar for seg.



Figur 1 Tre drivarar for databehandling: samfunnet, forsking og teknologi (Wing, 2008, s.3722)

2.1.1 Definisjon av algoritmisk tenking

Det er mange ulike definisjonar på algoritmisk tenking, men Israel og Lash(2020) trekk fram kjernen til algoritmisk tenking som «*ein måte å tenke på som involvere formuleringa av problemet, bryte dei ned, og strukturere og kommunisere ei løysing som menneske kan forstå og maskiner kan prosessere*» frå Waterman et al.(2018 s. 283). Vidare har Wing (2006) presentert sin definisjon av algoritmisk tenking som fokus på informatikken sitt bidrag til å forbetre og utvide det menneskelege arbeidet: *«algoritmisk tenking involverer løysing av problem, lage system og forstå den annleis åtferda, ved å bruke fundamentale konsept frå informatikk*(Wing, 2006, s.3722).». Algoritmisk tenking er altså ei tilnærming til korleis ein kan løyse problem, lage system og forstå menneskeleg åtferd ved å bruke konsepta frå algoritmar (computing) (Wing, 2006). Informatikk viser til vitskapen om strukturen, drifta og bruken av datamaskiner og datamaskinsystema(Rossen, 2022).

Weintrop et al. 2016 speglar Wing sine definisjonar ved sin eigen definisjon som også legg vekt på informatikk: *«Prosessen ved å kjenne att aspekt ved algoritmar i verden rundt oss, og bruke verktøy og teknikkar frå både naturlege og kunstige system og prosesser»* (Wing, 2008, s. 3717).Weintrop et al. 2016 legg fram desse definisjonane i sin artikkel, der dei understrekar at sjølv om det er lagt vekt på å forstå algoritmisk tenking, presenterer definisjonen framleis ein del utfordringar. Ein av desse utfordringane er korleis ein skal definere algoritmisk tenking i ein læringsprosess. Dette inkluderer til dømes kva som skal inn i pensum, evaluering av elevarbeid, forberede lærarar og sikra tilpassa opplæring til dei elevane som treng det.

Ved opplæring i algoritmisk tenking, er det fleire ulike utfordringar som Wing(2008) trekk fram. Ein av dei er kor nært og avhengig algoritmisk tenking ligg verktøya sine, slik som ulike programmeringsverktøy, og at det ved utføring av «computing» er nødvendig å lære både konsept og korleis ein skal bruke dei ulike verktøya. Slike verktøy kan vere ei utfordring, i følgje Wing (2008). Den første utfordringa som vert trekt fram er at ein ynskjer at eleven skal lære konsept, ikkje berre korleis ein bruker verktøyet. Til dømes det å bruke ein kalkulator utan å forstå aritmetikk. Enno verre er det dersom verktøya kjem i vegen for å lære dei ulike matematiske eller algoritmiske konsepta. Den siste utfordringa som Wing trekk fram er korleis ein skal spore korleis læring føregår når elevane bruker desse verktøya, når lærer elevane å bruke verktøyet og når lærer dei konsepta? Sjølv om verktøy representerer ein del utfordringar, er det viktig å hugse at dei er eit verktøy som kan brukast til å støtte elevane i læringa av algoritmar (Wing, 2008).

Kanoknitanunt et al. (2021) legg vekt på at algoritmisk tenking er ei evne til å løyse problem, analysere problema, lage eller utvikle system. Han meiner at evna til å konseptualisere kalkulasjonar og sekvensielle tankeprosessar kan knytast til det daglege livet. Kanoknitanunt et al.(2021) trekk fram det aukande behovet for samarbeid mellom menneske og maskinar, fordi informasjonsmengda ein vert utsett for, blir stadig meir kompleks og større i omfang. Her er algoritmisk tenking viktig for å kunne utvikle system som menneske kan forstå og maskinar kan bruke effektivt.

Brennan og Resnick(2012)legger fram tre dimensjoner for algoritmisk tenking i rammeverket sitt. Det er desse dimensjonane som blir lagt til grunn i denne oppgåva. Dei tre dimensjonane er algoritmisk konsept, algoritmisk tilnærmingar og algoritmisk perspektiv. Algoritmisk konsept er konsept som ein møter når ein programmerar. Algoritmisk tilnærmingar er ulike strategiar eller tilnærmingar ein kan bruke ved problemløysinga. Algoritmisk perspektiv viser til perspektiv som vert påverka av algoritmisk tenking om verden rundt(Brennan & Resnick, 2012).

2.1.2 Algoritmisk tilnærmingar

Det er fire algoritmiske tilnærmingar innafor konseptet algoritmisk tilnærming som Brennan og Resnick(2012) kategoriserer. Dei er «being incremental and iterative», «testing and debugging», «reusing and remixing» og «abstracting and modularizing». Eg har valt å bruke ei norsk oversetting (Sannes, 2020) av desse tilnærmingane (Brennan & Resnick, 2012, s. 7).

- Ei adaptiv og iterativ tilnærming (Being incremental and iterative)
- Ei testande og feilsøkjande tilnærming (Testing and debugging)
- Ei gjenbrukende tilnærming (Reusing and remixing)
- Ei abstraherande og modulariserande tilnærming (Abstracting and modularizing)

2.1.2.1 Ei adaptiv og iterativ tilnærming (Being incremental and iterative)

Ei adaptiv og iterativ tilnærming (Being incremental and iterative) viser til ein sekvensiell prosess. I omsetjinga til Sannes (2020) vel han å bruke ordet adaptiv som omsetjing av «incremental», medan den direkte omsetjinga er «stegvis». Eg vel også å bruke adaptiv fordi det passer betre til Brennan og Resnick(2012) si skildring av denne tilnærminga. Den adaptive prosessen består av å utarbeide planen for programmeringa og designet til prosjektet for deretter å prøve å implementere dette i koden. Denne tilnærminga er stegvis, men fokusere på å endre planen under vegs. Altså er det ei adaptiv, stegvis tilnærming. Difor vel eg å bruke namnet «ei adaptiv og iterativ tilnærming».

Brennan og Resnick (2012) fortel at ein slik plan ikkje alltid er lineær. Ein programmerar/ elev/ scratcher kan først få ein idé og deretter implementere delar av denne ideen i programmet. Ein scratcher er nokon som lager prosjekter innan for programmet Scratch. Dette er ein adaptiv prosess eller ein stegvis prosess som kan endre seg undervegs som ein respons til læringa av ulike problem. Her skildrar dei ein syklus av idear og utvikling av prosjektert. Denne prosessen utviklar seg også i takt med evnene, erfaringane og ideane til utviklaren. Dette dømet kjem frå analyse kapittelet 8. For å illustrere tilnærminga kan ein sjå føre seg ei gruppe elevar som skal løyse eit problem. Dei har fått utdelt ei vekt og tre kuler. Ei blå, ei grøn og ei raud. To av kulene er like tunge, medan ei er lettare enn dei andre. Her avgjorde elevane kva kule som var lettast ved hjelp av ei vekt. Elev 1 føreslår følgjande til gruppa si:

> *Elev 1: [skriv ut første løysingsforslag litt meir forklarande enn det står no].*
>
> *Etter litt diskusjon og visualisering, har dei endra planen. I mellomtida har Elev 1 prøvd å overtyde dei andre i gruppa om at dette er den rette planen. Ikkje det han sa først.*
>
> *Elev 1: Dersom me legg på to kuler og den eine er lettaren enn den andre, så veit me kva som er den lettaste kula. Dersom den andre er tyngre enn den første, så veit du kva som er den lettaste. Dersom du legg to på og dei er like tunge, så veit du at den som ligg igjen er den lettaste. Då finn ein uansett ut kva som er den lettaste på ei vegning».*

Dette dømet visar den adaptive tilnærminga som Brennan og Resnick (2012) diskuterte i artikkelen deira. I dette dømet endrar eleven planen sin etter han fekk ny erfaring og gjorde feilsøking på den gamle planen (Brennan & Resnick, 2012)

2.1.2.2 Ei testande og feilsøkjande tilnærming (Testing and debugging)

Brennan og Resnick (2012) viser til at når ein programmerar vil koden ofte ikkje fungere på første forsøk. Dersom løysinga ikkje tilfredsstilte oppgåva er det viktig å ha strategiar for å kunne handtere slike situasjonar. Det er viktig for elevane å utvikle strategiar for å handtere og føresjå ulike problem som kan komme i vegen(Brennan & Resnick, 2012).

Ulike strategiar kan vera som følger, identifiser problemet, lese igjen koden, eksperimentere med koden, prøv å skrive koden om igjen og finn dømer på koder som fungerer og deretter samanlikne. I studiet til Brennan og Resnick (2012) kom det fram ulike testande og feilsøkjande tilnærmingar, som var utvikla igjennom å prøve seg fram, overføring kunnskap og erfaringar frå ulike aktivitetar, eller av hjelp med kunnskap frå andre(Brennan & Resnick, 2012).

I Israel og Lash (2020) sin studie om integreringa mellom informasjonsteknologifag og matematikkfaget fann dei at feilsøking var ei tilnærming som ikkje var med i progresjonen i studiet deira(Israel & Lash, 2020). Å arbeide med feilsøking var altså ein strategi som ikkje vart undervist i, og gjennom diskusjonar og intervju med lærarane som var med i denne studien kom dei fram til at dette var noko dei burde ha undervist meir. I dømet under ser ein elever som gjennomfører feilsøking på ei oppgåve. Dei har to variablar som dei skal sjekke. Først skal dei sjekke om alderen til den aktuelle personen er over eller under 20 år, og den andre variabelen kor mange timar ein har arbeide. Her går dei nøye igjennom dei fleste moglege kombinasjonane og finn feilen.

> *Gut 2: Skal vi prøve, sjå om den funkar?*
> *(Prøver ut koden med talet 14)*
> *Gut 1: Den går ikkje.*
> *Gut 2: Ikkje så bra.*
> *Elevane bytter ut ei blokk og snur ulikhets teiknet slik at den delen av programmet fungere.*
> *Gut 2: Sånn der funka da. Kor mange timer jobber han i uka? Så seier eg 10 timer i uka kor gammal er han? 14 skriver eg. Då får eg opp, 450.*
> *Gut 1: Koffor sa den ikkje navnet?*
> *Elevane har ikkje laga ein blokk for å gi ut variabelen namnet, berre for å lagre variabelen namnet.*
> *Gut 2: Vent den seier at svaret er 0?*
> *Elevane har ikkje lagra variabelen alder.*
> *Gut 1: Skal den ikkje sei namnet ditt då? Berre ta sei navn og tar bort dei pluss greinene(blokka) og ein sei alder og så tar du sei svar.*
> *Gut 1: Du må ha en sei i blokk.*
> *Gut 2: Skal vi prøve med det same? 14.*

2.1.2.3 Ei gjenbrukende tilnærming (Reusing and remixing)

Ei gjenbrukende tilnærming (Reusing and remixing) er ein praksis som har blitt brukt i lang tid i programmering og har blitt forsterka ved hjelp av nettverk og internettet som gjer at fleire har tilgang til andre personar sitt arbeid og kan dermed bruke det om igjen i følge Brennan og Resnick (2012) Dette kan og bety å bruke opptatt deler av ein kode frå tidlegare oppgåver. Til dømes har elevoppgåve 1 og oppgåve 2 eit liknande oppsett. Her skal ein skrive eit program som kan delast inn i to deler, der elevane i den første delen skal skrive ein kode som spør om alderen til programbrukaren og i den neste delen skal programmet gi eit svar ut i frå dei ulike vilkåra i oppgåva. Her kan elevane bruke oppatt deler frå oppgåve 1 i oppgåve 2 slik at dei ikkje treng å lage heile programmet frå grunnen av. Ved å bruke denne tilnærminga har ein testa og feilsøka den første delen av oppgåva, slik at ein veit at denne fungerer. I dømet under har elevane identifisert kva som likner og kva som dei kan bruke opp att i oppgåva. Dei bruker ulike

funksjonsblokker og har funne ut at dei kan kopiere dei for å gjenbruka desse i ein annan del av oppgåva. Dette gjer dei ved å bruke skjelettet av blokka og bruker denne oppatt.

Gut 1: Litt som i stad?
Gut 2: Litt som i stad ja.
Elevane viser til deler av oppgåva som kan brukast oppatt i den neste oppgåva.
…….
Gut 1: Går det ann å kopiere den eller noko sånt?(Elevane viser til ein del av koden som kan gjenbrukas i ein annan del av programmet)
Gut 1: Ja, både den og den.
Gut 2: Tar den og berre kopiere den då.
Gut 1: Sånn ja, satt du den tilbake Du treng å gjere det ein gang til fordi me skal jo bruke den.
Gut 1: Navn og den der. (Elevane bruker igjen den blokka der Sprite spør om alder og berre endra på spørsmålet)

2.1.2.4 Ei abstraherande og modulariserende tilnærming (Abstracting and modularizing)

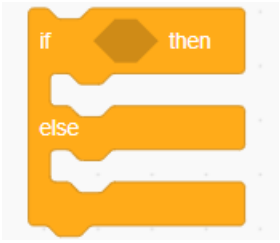
Ei abstraherande og modulariserende tilnærming (Abstracting and modularizing) viser til det å sette saman ein samling av mindre deler til et større heile, og dette er rekna som ei viktig tilnærming for oppgåveløysing og problemløysing(Brennan & Resnick, 2012, s. 9). I forbindelse med programmering med Scratch bruker ein abstrahering og modulisering i fleire nivå frå ide til deretter å konseptualisere denne ideen, til å omsetja konsepta til ulike deler med kode og funksjonsblokker. Dette inkluderer også eleven sin prosess frå å danne seg ein oppfatning av problemet, til planlegginga av løysinga og til slutt gjennomføringa av denne. Dette er ei viktig tilnærming som vert essensiell i andre større prosjekter som problemløysing. I nokre tilfelle bruker elevane den abstraherande og modulariserende tilnærminga for å skape ein prioriteringskø for å avgjera kva som først må fjernast. Deretter gjennomfører dei denne modulen og går så dei vidare. I dømet under ser me at elevane avgjer kva dei treng å gjere i denne delen av oppgåveløysinga og isolere den delen frå resten av oppgåva.

Gut 1: Først må den spør om alderen. Kor gammal er du?
Gut 2: Kvifor har du «if» blokk der då? Du skal jo berre spør om alderen.
Gut 1: Først må me spør kor gammal dei er før me finner ut kor mykje dei tener.
Gut 2: Å ja visst du er over 20 så, okei du kan starte med kor gammal er di og så, nei. Altså visst det var over 20 så gang her du berre med 45 eller 55.
Gut 2: Nei no skjønte eg ikkje...
Gut 1: Først. Spør om du kan han heiter, er. Han skal berre huske på navet. Skal me få da til?
Gut 2: Sikkert den her då?
Gut 1: Så setter me namn til svaret. Så neste tingen blir alderen då?
Gut 2: Spør først om kor gamle dei er, og så må me ha sånn «if».

2.1.3 Algoritmiske konsept

Denne oppgåva fokuserer spesielt på konseptet vilkår. Algoritmiske konsept fokuserer på ulike byggeblokker som ein bruker i programmering, og desse konsepta er basert på

Scratch sine blokker vist i figur 2 og 3. Brennan og Resnick (2012) har satt opp desse byggeblokkene som sekvensar, løkker, parallellisme, hendingar, vilkår, operatørar og data. I Scratch sin programmerings språk ser vilkår ut som figuren under. Den består av ei blokk med moglegheit til å fylle inn ulike vilkår og deretter korresponderande hendingar gitt av vilkåra i blokka. Figur 2 og 3 er illustrasjonar av if og if/else blokkene som Scratch bruker. Desse blokkene kan knytast til det algoritmiske konseptet vilkår frå Brennan og Resnick (2012).



Figur 2 if/else blokk frå Scratch

Figur 3 if blokk frå Scratch

2.1.4 Oppgåver basert på vilkår

Oppgåver som er basert på det algoritmiske konseptet vilkår baserer seg ofte på figur 2 og 3 som er ofte kalla blokker. Oppgåver i programmering omhandlar vilkår eller Conditionals (Horstmann & Necaise, 2019). Slike oppgåver presenterer ein ofte med ein tilstand og på bakgrunn av denne tilstanden skal noko utførast. I døme under er tilstanden alderen og der dei skal lage eit program som gir ut eit svar med omsyn vilkåra til denne tilstanden.

> *«Planlegg og skriv et program der Sprite spør om din alder.*
> *Hvis du er 19 eller yngre, svar at du må være på skolen.*
> *Hvis du er eldre enn 19, svar at du må være på jobb.»*

2.1.5 Algoritmisk perspektiva

Dei algoritmiske perspektiva handlar om eleven sin even til å reflektere over seg sjølv og verden rundt seg. Brennan og Resnick (2012) belyste tre perspektiv som elever vert undervist i Scratch hadde utvikla. Desse tre perspektiva var evna til å uttrykke seg sjølv gjennom nye media, gjennom sosiale koplingar, og å stille spørsmål.

Tabell 1 viser ein oversikt over algoritmisk tenking og inndelinga til Brennan og Resnick (2012). i tre deler, algoritmisk konsept, algoritmiske tilnærmingar og algoritmiske perspektiv.

Tabell 1 Oversikk over Brennan og Resnick(2012)sin inndeling ac algoritmisk tenikng.

Algoritmisk tenking	
Algoritmisk konsept	- Sekvensar (sequences) - Løkker(loops) - Parallellisme(parallelism) - Hendingar(events), - Betingelser eller vilkår (conditionals) - Operatørar (operators) - Data
Algoritmiske perspektiv	- Uttrykke(Expressing) - Kople saman (Connecting) - Stille spørsmål (Questioning)
Algoritmiske tilnærmingar	- Ei adaptiv og iterativ tilnærming (Being incremental and iterative) - Ei testande og feilsøkjande tilnærming (Testing and debugging) - Ei gjenbrukende tilnærming (Reusing and remixing) - Ei abstrakt og modulariserende tilnærming (Abstracting and modularizing)

2.2 Integrasjon mellom algoritmisk tenking, informatikk og matematikkfaget

Ei parallell problemstilling er integreringa av algoritmisk tenking og informatikk i matematikkfaget og andre fag. I denne master oppgåva vil eg fokusere på integrasjonsprosessen mellom informatikk og matematikkfaget, til liks som Isreal og Lash trekk fram i sin artikkel(2020). Dei grunngjev med å argumenter for integreringa mellom faga og spesielt algoritmisk tenking og informatikk inn i andre fag, fordi det er mest relevant for det verkelege liv problem og kan belyse kor samankopla dei ulike faga er. Dette bygger dei på at informatikk er samansett av på mange matematiske konsept og evner, og dette fører til overlapp mellom dei to faga. På den andre sida har det blitt argumentert for at når ein kombinerer ulike fag er det ofte ein utfordring å vedlikehalde faget sin integritet i følge English (2017) i Israel og Lash(2020).

I artikkelen til Israel og Lash presenterer dei tre ulike modeller for integrering i matematikken. Den første dei presenter er Kiray sin modell, der ein kan integrere algoritmisk tenking inn i matematikk faget slik at det ville vera matematisk intensivt med algoritmisk tenking som ein kopling mellom faga(Israel & Lash, 2020). Den andre modellen er Vasquez, Sneider og Comer(2013) sin tilnærming som organiserte integrasjon langs eit kontinuum av aktivitetar. Aktiviteten vart evaluert basert på nivået som kvart av faga vert brukt i undervisninga. Den siste modellen stammar frå Bryan,

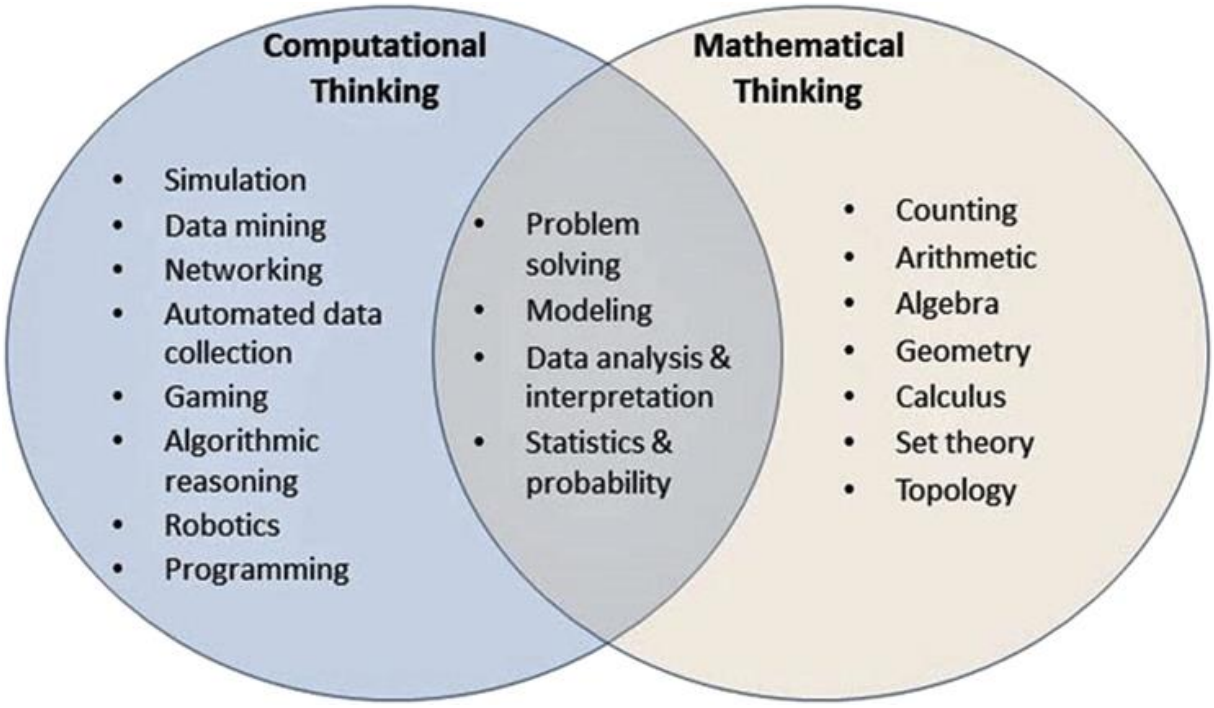
Moore, Johnson og Roehrig (2015) som argumenterte for at integrasjon skal vera bevisst. Dei identifiserte tre ulike typar integrering.

Integrasjon mellom matematikk og informatikk og algoritmisk tenking har fleire forfattarar peikt ut som naturleg. Sneider (2014) har laga eit Venn-diagram som skildrar integrasjonen mellom matematikk og algoritmisk tenking, der problemløysing, modellering, analysering, statistikk og sannsyn er felles for begge to fagområda. Pei et al. (2018) har også skildra overlappinga mellom dei to faga, der dei seier at matematisk tenking og algoritmisk tenking er to ulike områder som gjensidig støtter kvarandre og kan samarbeide saman. Moglegheita for å undervise begge områda samstundes er ein sjeldan hending. Weintrop et al. formulerte ein taksonomi der matematikk og algoritmisk tenking integrerer og inkluderer fire kategoriar: datapraksis, modellering og simuleringspraksis, berekningspraksis for problemløysing og systemtenkingspraksis (Weintrop et al., 2016). Utfordringa ein møter ved integrasjon mellom faga er å finne metodar å undervise på som er både autentisk og har meiningsfullt integrert matematikk og informatikk.

2.3 Tidlegare studiar om matematisk tenking og algoritmisk tenking

Cui og Ng har gjennomført eit studie der dei observerer elever si oppleving under matematisk aktivitet. Dei brukte ein blanding av Weintrop et al (2017) sin taksonomi og Richard et al. (2020) sin klassifisering for å undersøke elevane sin aktivitet under problemløysing og programmering. Weintrop et al (2017) sin taksonomi er delt inn i fire hovudkategoriar:
datapraksis, modellering og simuleringspraksis, berekningspraksis for problemløysing og systemtenking. Dei veljar å bruke ordet praksisar i motsetning til ferdigheiter eller konsept for å understreke at å engasjere seg i vitskaplege undersøkingar ikkje berre krev ferdigheiter men også kunnskap som er meir spesifikk enn for dei enkelte praksisar. Dei presenterer taksonomien sin i eit sett med distinkte kategoriar, men understrekar at desse praksisane heng nært saman og er avhengig av kvarandre. For å analysere deira data som bestod av observasjon så undersøkte dei først kor eleven hadde gjort feil i programmet, og der koden deira ikkje fungerte. I diskusjonane deira etterpå fann dei korleis desse utfordringane kan bli kopla mellom matematisk tenking og algoritmisk tenking. Ved analysering av programmeringsdelen, kom dei fram til at eleven generelt gløymte å presentere svaret som eit nødvendig steg for deira løysing. Ei forklaring Cui og Ng kom fram til, er at i ei matematisk oppgåve presenterer ein svaret på oppgåva, altså er det ikkje eit nødvendig steg ein må ta i tillegg. Eit anna resultat dei kom fram til var elevane sin bruk av vilkår i ei oppgåve med primtal. Eleven fekk problem med å bruke vilkår saman med repetisjon og dette fenomenet kunne stamme frå ulikskapen mellom matematisk tenking og algoritmisk tenking.

Cui og Ng bruker eit Venn-diagram for å skilje mellom matematisk tenking og algoritmisk tenking frå Sneider et al., (2014) sin artikkel. Her skilder dei at ein måte å forstå algoritmisk tenking på er å undersøke deira likskapar og ulikskapar frå matematisk tenking. Figur 4 viser forhaldet mellom matematisk tenking og algoritmisk tenking. Den illustrerer at det er fleire områder som overlappar, i dei er problem løysing, modellering, data analyse og tolking, og statistikk og sannsyn.



Figur 4 Venn-diagram av matematisk og algoritmisk tenking (Sneider et al., 2014)

2.4 Logisk tenking

Logisk tenking er bruken av årsak og verking til resonnering og problemløysing. Denne prosessen ved rasjonalisering kan bli uttrykt ved hjelp av ord. Funksjonen til resonnement finn stad i venstre side av hjernen, som involverer den kontinuerlege prosessen med å overføre strukturerte data til den delen av hjernen som omhandlar den visualiserande talen, og som ikkje er relatert til kvarandre, inntil forståing oppstår og meir komplekse minner blir utvikla (Herrick, 1999). Rasjonell tenking viser til tankeprosessen om sanningar og prinsippa. Logisk tenking er ein eigenskap som krev at evna til å forstå strukturen i eit problem og å komme fram til eit forståeleg svar. Ein kan og argumentere for å lære programmering er ein avhengig av problemløysing, som igjen kan krevje logisk tenking (Kawamoto & Arai,2008). Logisk tenking er altså å tenke systematisk og som ein prosess med start og slutt. Systematisk tenking vil gi eleven rom til å forstå og prioritere idear, oppfordre dei til å lære frå ulike kontekster og løyse problema logisk. I kjernen av dette ligg det at dersom ein kan forstå det grunnleggjande ved problemet og har evner frå tidlegare kunnskap som kan brukast til å løyse dette problemet, kan eleven legge ein strategi for problemløysinga.

2.5 Problemløysing

Kanoknitanunt et al. (2021) tar utgangspunkt i John Derey sin ide om problemløysingslæring. Han definer problemløysingslæring som elevsentrert læringstilnærming der ein fokuserer på problemløysing og øvinga i å løyse problem, både individuelt og saman med andre. Problemløysingsbasert læring tillet eleven å lære for seg sjølv og i sitt eige tempo, men samtidig ha klasseroms undervisning. Kanoknitanunt viser til Greenwal (2000) som argumenterer for at problemløysing basert læring er

effektivt for elever med ulike nivå og evner. Dette grunngjev han med at eleven kan sjølv velgje problem og løysingsmetode basert på deira utviklingsnivå og interesser.

2.6 Programmering

Programmering er eit omfattande omgrep som inkluderer heile problemløysingsprosessen, i tillegg til å skrive programmeringskode(Sevik,2016). Programmeringsprosessen kan fortsette etter at problemet er løyst, ved at elever ønsker å gjere endringar og forbetringar på sin programkode eller løysingane som denne gir.

«Programmering, slik det er brukt i dette dokumentet, omfattar meir enn berre å skrive ein programkode som kan køyrast på en datamaskin, det inkluderer også prosessen med å komme fram til denne koden. Det vil seia prosessen frå å identifisere [sic] et problem og tenke ut moglege løysningar dette, til å skrive ein kode som kan forståast av en datamaskin, og å feilsøke og kontinuerleg forbetre denne koden.» (Sevik, 2016).

Definisjonen som ligg til grunn for programmering er henta frå Kunnskapsdepartementet sitt skriv om programmering, og som er utvikla av senteret for IKT. Eg vel å bruke denne definisjonen fordi forskinga mi fann plass på ein norsk skule og dermed er det mest naturleg å halde meg innafor rammene som Kunnskapsdepartementet har lagt fram. Algoritmisk tenking er ein måte å tilnærma seg eit problem på og korleis elevane vel å løyse det som er ein viktig del av matematikk faget frå hausten 2020(utdanningsdirektoratet, 2019).

3 Metode

I dette kapittelet skal eg gjere greie for vala mine og korleis dei har påverka oppgåva mi. Vala har handla om å velje mellom kvalitativ forskingsmetode eller kvantitativ forskingsmetode, utvalet, analysemetoden, utforming av oppgåvesett, og korleis eg har forsøkt å sikre kvaliteten på oppgåva. Desse tinga vert belyst i dette kapittelet.
For å kunne gjere greie for forskingsmetoden min bruker eg problemstillinga mi som grunnlag for alle avgjerdene mine. Forskingspørsmålet mitt er del inn i to deler:

1. Kva kjenneteiknar 7 elevar sitt arbeid med oppgåver basert på vilkår?
2. Kva spor av algoritmisk tenking kan ein sjå i elevane sitt arbeid i ein programmeringskontekst og ved løysing av oppgåver på papir?

Begge problemstillingane fokuserer på eleven sitt arbeid, og eg vel å bruke observasjon for å kunne svare på dette. Her ønsker eg å undersøke og sjå korleis arbeidet deira kan kategoriserast i Brenna og Richard sitt rammeverk for algoritmisk tenking.

3.1 Forskingsdesign og metode

For å avgjere kva metode som passar best til problemstillinga mi, må eg fyrst skildre kva omgrepet metode handlar om. Christoffersen og Johannessen seier at metode er å følgje ein fastsett veg mot eit mål (Christoffersen & Johannessen, 2012). I denne oppgåva er målet å svare på problemstillinga mi. Målet går ut på å samle inn informasjon frå elevane og arbeidet deira. Denne informasjonen kjem frå den sosiale røynda til elevane, som vert skapt både av interaksjonen elevane har mellom kvarandre og interaksjonane mellom elevane og oppgåvene dei arbeider med. Christoffersen og Johannesen skildrar samfunnsvitskapleg metode som korleis me hentar inn informasjon om den sosiale røynda, korleis me analyserer han og kva han fortel oss om samfunnsmessige omstende og prosessar.

Ut ifrå denne skildringa er det slike metodar eg vel å bruke for å undersøke problemstillingane mine. Dette er grunnen til at eg ønsker å hente inn informasjon om læringa til elevane og den sosiale røyndommen deira, og analysere denne informasjonen. Forskingsmetodane eg har valt gir meg moglegheit til å gjere det. Viktige eigenskapar ved samfunnsvitskaplege metodar er openheit, gyldigheit og dokumentasjon. Dette vil eg kome nærare inn på seinare i kapitelet. Føremålet til metodekapitelet er å kunne bidra med både dokumentasjon og openheit om kva eg har gjort i denne oppgåva.

3.1.1 Kvalitativ forsking

Samfunnsvitskapleg forsking inneheld både ein kvantitativ og ein kvalitativ del. Kvantitativ forsking er ofte assosiert med tal og mengder. Dette kan gje eit overordna bilete på det generelle og trendar i forskingsmateriala. Kvalitativ forsking er derimot meir fleksibel, og det er denne fleksibiliteten gir rom for meir spontane samtaler som er betre tilpassa situasjonen og problemstillinga. Dette gjer at deltakarane kan utdjupe svara sine betre. Fellestrekk innafor ulike kvalitative orienteringar er ein erkjenning av at menneskelege handlingar ikkje kan vurderast frå utsida med forskaren som ein tilskodar (Kvarv, 2014). Forskaren kan ikkje fri seg frå normene og føresetnadene sine som eit

sosialt menneske. Dette gjer at ein i stor grad legg vekt på det å teoretisere, noko som gjer at teoriar om menneskelege handlingar vert integrert i kvalitative undersøkingar.

3.1.2 Kvalitativ metode

Kvarv fortel at kvalitativ metode handlar om å kategorisere ting (Kvarv, 2014). Ordet kvalitativ viser til eigenskapar eller karakteristiske trekk ved eit fenomen. I oppgåva mi ønsker eg å karakterisere kjenneteikn ved elevane sitt arbeid med oppgåver basert på det algoritmisk konseptet vilkår. Derfor er kvalitativ metode ein passande metode. I kvalitative metodar er det tekst, tale og bilete som er datamaterialet. Metoden gjer at ein undersøker materialet i djupna. Å kunne forstå fenomenet, og ikkje berre omfanget eller utbreiinga av fenomenet ein undersøker, er avgjerande hevdar Kvarv (2014). Ved bruk av kvalitative metodar har ein mogelegheit til å undersøke forskingsobjekta nøye. Kvalitativ metode er basert på situasjonar prega av interaksjon og kommunikasjon. For å svare på problemstillinga mi vil eg undersøke interaksjonen og kommunikasjonen mellom dei ulike elevane i klassen, men også mellom elevane og oppgåvene dei arbeider med.

Kvalitative studiar har ulike stadium i forskingsprosessen sin på lik linje med kvantitative studiar, der ein utarbeider problemstillinga, til tolking og formidlinga av forskingsresultatet. I motsetting til kvantitative forskingsprosessar er dei kvalitative prosessane ikkje like lineære og trinnvise prosessar, men meir ein vekslande sosial prosess der ein justerer opplegget undervegs og nye tolkingar kan kome fram (Kvarv, 2014).

3.1.3 Epistemologi

Postholm og Jacobsen skriv om konstruksjonisme som ei tilnærming som tek utganspunkt i at det ikkje er mogeleg å skilje mellom objektet ein skal studere og personen som studerer dette objektet (Postholm & Jacobsen, 2018). Vidare viser dei til Kant som seier at det einaste me som menneske kan seie noko om, er korleis me oppfattar eit fenomen, ikkje korleis det heilt objektivt sett er. Det er gjennom dette synet konstruksjonisme har vorte danna i den moderne vitskapsteorien. Me ser ikkje nødvendigvis korleis eit objekt faktisk er. Me lagar oss vårt eige bilete av objektet. Dette bilete vil basere seg på vår eigen, subjektive forståing av røyndommen, ikkje røyndommen slik han objektivt sett er. I denne tilnærminga speler kunnskap ein viktig rolle, sidan bileta våre av verda kan endre seg når vi får ny og utvida kunnskap. Omgrepet sosial konstruksjonisme er eit omgrep som er viktig i humaniora og filosofi, men det vert også brukt ein del i grunnlagsteoretiske diskusjonar innafor samfunnsvitskapen. Kvarv deler sosial konstruksjonismen inn i to former: moderat og radikal (2014. s136). Den radikale forma for sosial konstruksjonisme legg vekt på språk. Dei radikale konstruksjonistane meiner at verda og røynda vert skapt gjennom språk og kunnskap (Kvarv, 2014. s.136). Me konstruerer verda vår basert på det folk rundt oss snakkar om, argumenterer for og skriver om han. Den moderate konstruksjonismen legg meir vekt på praktisk fornuft som utgangspunkt for korleis folk forstår verda(Kvarv, 2014).

Det er umogeleg å skilje forskaren frå objektet som studerast når ein studerer sosiale fenomen meiner Postholm og Jacobsen. Det kjem av at forskaren går i dialog med det

han eller ho studerer. Denne interaksjonen vert påverka av omgivnadene og relasjonane rundt interaksjonen (Postholm & Jacobsen, 2018).
Det er fleire ulike variantar av konstruksjonisme, men alle har eit felles utgangspunkt der ein ikkje ser på verda objektivt, men heller som noko vi menneske har konstruert. Det vil seie at det er vanskeleg å seie noko om kva som er sant og kva som er falskt. I denne epistemologien legger forskaren heller vekt på ulike aspektar ved røynda, enn å prøve å finne ei riktig sanning.

Eg valde å bruke ei konstruktivistisk tilnærming i denne studien fordi majoriteten av dei studiane som eg tek utgangspunkt i også bruker denne tilnærminga. Dette gjer det naturleg for meg og bruke same tilnærming, sidan det vert lettare å hente ut informasjon og kunnskap som har relevans for studien min.

3.2 Kvalitativ analyse

Hensikta med kvalitativ dataanalyse er for det fyrste å sortere datamaterialet som er samla inn. Dette hjelpte meg med å skape strukturen og skjelettet for analysen. I mange tillefeller handlar det om å lage eit mønster slik at ein kan kategorisere og tematisere data materialet (Postholm & Jacobsen, 2018). I denne analysen bruker eg Brennan og Resnick(2012)sitt rammeverk for å finne mønster i datamaterialet.
Analysen byrjar allereie under datainnsamlinga, når forskaren prosesserer kva typar informasjon han samlar inn. Allereie her byrjar ein å filtrere ut uviktig informasjon, og ta vare på viktig informasjon.

3.2.1 Tematisk analyse

Tematisk analyse er ein fleksibel tilnærming og kan vere nyttig for kvalitative analyser. Tematisk analyse kan også kombinerast med andre analysemetodar. Denne metoden er ein framgangsmåte som ofte vert oversett fordi han er svært grunnleggande. Johannessen, Rafoss og Rasmussen forklarer metoden nærare i boka si (Johannessen et al., 2018). Som namnet seier er ein tematisk analyse ein analyse som basere seg på temaet. Eg vel å bruke Johannessen, Rafoss, et. al. sin definisjon på tema som ein gruppering av data med viktige fellestrekk. I denne samanhengen kan omgrepa tema og kategori vere brukt om kvarande. Ved til dømes å gruppere datoane i kategoriar kan vi orden i datamaterialet på ein måte som gjer at ein lettare kan identifisere nye samanhengar. Tematisk analyse er teoriavhengig, noko som betyr at ein bør bruke teori til å definere dei ulike kategoriane. Her argumenterer Johannessen, Rafoss, et. al. for at teori gjer deg til ein god oppdagar, sidan du då lettare veit kva du skal sjå etter (Johannessen et al., 2018). I min tematiske analyse bygger eg på Brennan og Resnick(2012)sitt rammeverk for algoritmisk tenking då eg lagde mine kategoriar. Det er derfor ein adaptiv og iterativ tilnærming, ein testande og feilsøkjande tilnærming, ein gjenbrukstilnærming og ein abstrakt og modualiserande tilnærming. Eg byrja med å kode data, for deretter å sortere det inn i kvar sitt tema basert på kategoriane eg satt. Til slutt skal eg samanlikne dei ulike tema og sette dei opp mot problemstillinga mi.

3.3 Rammeverket

Rammeverket eg vel å bruke i denne analysen er Brennan og Resnick(2012)sitt rammeverk for algoritmisk tenking. Rammeverket deira er tredelt, noko som reflekterer definisjonen deira på algoritmisk tenking. Denne definisjonen består av algoritmisk konseptet, algoritmiske tilnærmingar og algoritmisk perspektiv. I denne analysen bruker eg dei algoritmiske konsepta til å lage kategoriane som datamaterialet vert sortert inn i. Oppgåvene eg gav elevane i undersøkinga er også basert på dei vilkåra til det algoritmiske konseptet. I artikkelen til Brennan og Resnick(2012)legg dei vekt på at dei tilnærmingane som fokuserer på prosessane tenking og læring, der ein flytter seg forbi sjølve kunnskapen som er lært og vidare måten du lærte deg kunnskapen på. Då kan ein vidareføre læringa og bruke ho i nye samanhengar (Brennan & Resnick, 2012a).

3.4 Casestudiar

Ordet case kjem frå latin og tydar tilfelle (Christoffersen & Johannessen, 2012). I casestudie er det i forskaren si interesse å forstå kvar enkelt unike case og på den måten skape ein overføringsverdi til andre liknande casar seinare. Ein case kan både vere eit studieobjekt og eit forskingsdesign. I min studie er casen basert på tre elevgrupper sine arbeid med å løyse oppgåver knytt til vilkår på to ulike måtar; både ved hjelp av programmet Scratch og ved å løyse oppgåver på papir. «Fagfornyelsen» har gjort algoritmisk tenking meir aktuelt i skulen. Ein casestudie der ein fokuserer på elevar i norsk skule vil bidra til å belyse dette fenomenet. Målet for ein casestudie er at analysen, tolkinga og rapporten skal gi lesaren ei forståing av tematikken som er undersøkt (Postholm & Jacobsen, 2018).

Casestudiar kan definerast på mange ulike måtar, men det som går igjen er ein tydeleg avgrensing av kva casen vert satt saman av. I mitt tilfelle er casen definert til grupper av elevar som gjennomfører oppgåver basert på vilkår. Kvar case har altso ein del med oppgåveløysing på papir og ein del med oppgåveløysing på programmet Scratch. Dette gjer at eg har tre casar eg skal skildre for å skape eit bilete nært røynda (Postholm & Jacobsen, 2018; Stake, 1995). Casestudiar vert sett saman ved å samle inn mest mogelege informasjon om eit avgrensa fenomen (Postholm & Jacobsen, 2018).

Casestudiar vert mykje brukt innafor utdanning, noko som nok kjem av at eit kjenneteikn ved casestudiar er at dei kan hente inn masse informasjon frå få einingar. Dei bukar ofte kvalitative tilnærmingar for datainnsamling, for å skape ei djupneundersøking. Det er to element som kjenneteiknar casestudiar er avgrensa merksemd og mest mogeleg glidande skildring (Postholm & Jacobsen, 2018).

3.4.1 Analyse av casestudiar

Analysen av casestudien skal bidra til detaljerte skildringar av casen og kontekster til casen. Vidare skal analysen gi meir data og utvikle ei betre forståing av casen forskaren studerer (Postholm & Jacobsen, 2018). Stake hevder at analyseprosessen er prega av kreativitet og er ein intuitiv prosess for å skape meining og forståing rundt casen (1995). Stake legg fram to metodar for analyse i casestudiar. Den fyrste er kategorisk analyse der ein leiter etter repeterande hendingar og plasserer dei i kategoriar. Ein brukar ofte

denne typen analyse for å avkrefte eller bekrefte tidlegare forståingar. Denne typen analysemetode vert ofte brukt der ein studerer fleire casar samstundes. Den andre typen er ein direkte type som handlar om å analysere dei individuelle hendingane. Her er ein ikkje ute etter å bekrefte si eiga forståing, men vil heller byggje ein forståinga av situasjonen som skjer (Stake, 1995).

3.5 Utval

Eit av kjenneteikna til kvalitativ forsking er at tala av informantar er avgrensa (Postholm & Jacobsen, 2018). I teorien skal ein hente ut informantar og informasjon til ein ikkje kan hente ut meir informasjon. Då har ein nådd ein grenseverdi. Her tek ein omsyn til gruppestorleiken også, og om han er den homogen eller heterogen. I praksis vert studien meir avhengig av problemstillinga og kor mykje som er praktisk moglege å gjennomføre, enn at ein skal samle inn mest mogeleg data (Postholm & Jacobsen, 2018).

I mitt prosjekt fekk eg tilgang til to 9. klasser. Alle elevane i klassene fekk tilbod om å vere med. Ni av dei ville vere med på observasjon både under programmering og ved løysing av oppgåver på papir. Eg valte å bruke to av elevane til ein pilot og satt då igjen med 7 elevar som var med begge dagane. Det var ei blanding av gutar og jenter, og av elevar med og utan norsk som morsmål. Sidan dette ikkje har noko innverknad på problemstillinga mi, valde eg å ikkje påpeike kjønn og morsmål. Engelske utsegn har eg omsett til norsk. Ut i frå datamaterialet mitt har eg gjort eit strategisk utval frå målgruppa basert på problemstillinga mi (Postholm & Jacobsen, 2018).

Stake (1995) trekk fram fridommen til å velje kva som vert definert som ein case. Dette kan til dømes vere ei klasse, ein skule, enkeltelevar og studieprogram. For studien mitt har eg valt at kvar gruppe med elever er ein case, og i kvar case er det eitt oppgåvesett på papir og eitt oppgåvesett som løysast ved hjelp av Scratch.

3.6 Observasjon

Rautaskoski legg fram at observasjon er ein passande metode dersom ein ønsker å undersøke kva rolle konkrete materielle omgjevnader har på menneskelege handlingar (Rautaskoski, 2012). Dette vil vere med på å gje innsikt i elevane sitt arbeid. Det passer derfor i studien min fordi eg ønsker å studere elevane sine arbeidsmetodar og strategiar i arbeid med programmering på PC og i arbeid med logiske oppgåver basert på vilkår. Observasjon er ein av dei vanlegaste metodane innafor kvalitativ forsking og er ein av dei mest brukte metodane innan for casestudiar. Det kan brukast både åleine og i kombinasjon med andre metode for datainnsamling (Kvarv, 2014; Mertens, 2010). Dette er ein av dei metodane der forskaren har liten innverknad på åtferda til elevane, noko som gjer det mogeleg å studere direkte kva elevane føretek seg. Ved observasjon er eg ein del av situasjonen og eg kan sjølv sjå elevane sine strategiar og kva metodar dei brukar for å løyse oppgåvene. Ved å observere elevane sine løysingsmetodar i staden for å spørje dei ut, kan eg lettare få eit inntrykk av dei faktiske løysingsmetodane deira og ikkje berre oppfatninga deira av løysinga. Observasjon vil gje informasjonen om kva som føregår i ein situasjon (Mertens, 2010). På den andre sida er observasjon tidkrevjande, og eg som observatør kan påverke elevane og situasjonen under observasjonen.

Observasjon er ein av dei mest grunnleggjande metodane som vert brukt til å hente inn informasjon frå røyndommen, direkte frå situasjonen som vert observert. Resultatet av observasjonen vert ofte ei detaljert skildring av den menneskelege aktiviteten (Adler & Adler, 1987; Christoffersen & Johannessen, 2012). Dette gjer at observasjon ikkje gjennomførast på eit laboratorium, som eit konkret eksperiment, men i naturlege situasjonar der deltakarane får fritt spelerom (Angrosino & Rosenberg, 2011)
Observasjon handlar om å bruke alle sansane til å oppfatte kva som skjer og forstå handlingane, ved å vere open og fange opp det som skjer. Dette vert ofte referert til som fokusert observasjon. Sjølve observasjonen av situasjonen skjer i tre fasar. Fasane er kronologiske og består av å gå inn i ein situasjon, vere til stades i situasjonen og deretter gå ut av situasjonen (Christoffersen & Johannessen, 2012).

I gjennomføringa av observasjonen, ved datainnsamlinga, var desse fasane tydelege. I den fyrste observasjonen var fasane knytt til skuletimen og med ein tydeleg start, arbeidsperiode og slutt. Den andre delen av observasjonsarbeidet hadde ein meir flytande overgang. Her byrja observasjonen då elevane starta på oppgåvene og var over då dei anten var ferdige eller dei hadde brukt meir enn 40 minutt på oppgåvene.

Ved å bruke observasjon som metode får eg direkte tilgang til situasjonen. Dette gjer at eg kan skaffe meg grundig informasjon ved å vere til stade då interessante hendingar skjer. Postholm og Jacobsen (2018) fortel at observasjon er ein tidkrevjande metode, som passer best då mindre grupper skal studerast.

Det er nokre sentrale omgrep knytt til observasjon som metode eg ønsker å avklare her. Det fyrste er observatør. Dette er rolla til den som observerer ein situasjon. Det er viktig å vere obs på observatøren sitt blikk på situasjonen og dette verker inn på det etiske perspektivet ved datainnsamlinga. Ettersom eg er til stades i oppgåvesituasjonen vert situasjonen uvant for elevane, for dei er ikkje er vande med meg eller den type oppgåver eg vil dei skal løyse. Dette gjer at rolla mi som observatør ikkje vert ei nøytral rolle. Eg kan påverke situasjonen. Rolla som observatør kan vere ein utfordrande rolle. Under observasjonen valde eg å vere kjent for elevane og i tillegg ønskte eg at elevane skulle få arbeide med andre elever dei var trygge med. Desse gruppene fekk eg hjelp av kontaktlærar til å lage.

Observasjon viser til det me erfarer gjennom sansane våre (Christoffersen & Johannessen, 2012). Dette kan vera både under sjølve observasjonen og i etterkant då ein går igjennom notata - og i mitt tilfelle lydopptak. Observasjonane ein gjer undervegs vert farga av teorien og erfaringane observatøren har kjennskap til på førehand (Postholm & Jacobsen, 2018). Resultatet av observasjonen er ofte ein detaljert skliding av handlingane som vart observert (Postholm & Jacobsen, 2018).
Det er i felten ein gjer observasjonane. Settinga er der observasjonen skjer. I dette tilfelle er elevane som deltar i undersøkinga felten, og klasserommet deira er settinga. I denne oppgåva kjem eg til å referere til elevane med enten gruppenamnet, elevane eller det anonymiserte namnet eg har gitt dei. Det er viktig at settinga belyser problemstillinga. Postholm og Jacobsen (2018) fortel at setting viser til noko meir enn den fysiske settinga. Her kan til dømes gruppedynamikken vere med på å gje meg informasjon til analysen.

Analysen av observasjonen går ut på å analysere dei elementa som vert observert. Postholm og Jacobsen seier at det er fire ulike element som kan analyserast: aktør, handlingar, meiningar og hendingar. Eg skal analysere elevane sine handlingar får å få innsikt i tankeprosessane og strategiane deira, slik at eg kan identifisere kva algoritmiske tilnærmingar som elevane har brukt.

3.6.1 Deltakande observasjon / aktivt medlem

Observatørrolla kan delast inn i ulike grader av involvering. På den eine sida av denne skalaen har me roller som ikkje er deltakande i aktiviteten som vert observert, og på den andre sidan har med observatørar som er fullstendige medlemmer av gruppa som vert studert (Adler & Adler, 1987; Gold, 1958; Kvarv, 2014). Kvarv viser til den klassiske inndelinga med deltakande og ikkje-deltakande observatørar (2014). Her er ein ikkje-deltakande observatør ein observatør som ikkje deltar på noko under observeringa, han sit berre på sida og ser på, og alt anna vert deltakande observatørar. Gold deler rollene inn i fire ulike roller. Han meiner at dei går frå ein ekstrem til den neste. Graderinga hans går frå fullstendig deltakar, deltakar som observatør, observator som deltakar og til slutt fullstendig deltakar(Gold, 1958).

Aktivt medlemskap er omgrepet Adler og Adler (1987) bruker for å forklare kva som er deltakande i observasjonen deira, og kva som ikkje er fullstendige deltakarar. Adler og Adler har delt inn observatørrollene i tre ulike roller (Adler & Adler, 1987). Dei er perifert medlem, aktivt medlem og komplett medlem. Eit perifert medlem tek ikkje del i observasjonen og observerer hendingane utanfrå, litt som Kvarv sin ikkje-deltakande observatør(1987; 2014). Eit aktivt medlem deltek i aktivitetane, men er ikkje opphavleg ein del av gruppa eller han har ein anna rolle enn han pleier å ha i gruppa. Til slutt har me eit fullstendig medlem. Det er nokon som allereie har ei rolle i gruppa og vedlikehald denne rolla under observasjonen.

Eg var til stades under begge situasjonane der elevane løyste oppgåver, og hadde ein aktiv rolle i observasjonen. Sidan eg ikkje hadde kjennskap til elevane frå før, fann eg at Adler og Adler sin skliding av aktivt medlemskap passa best for min observatørrolle. Denne rolla kom med vitende at eg skulle delta i oppgåvene, men halde i tankane at eg ikkje var ein del av fellesskapet (Adler & Adler, 1987). Dette påverka kor mykje eg involverte meg i elevane sin prosess og eg svarte hovudsakleg på spørsmål.

3.7 Datainnsamling

Empirisk forsking er samansett av å samle inn data om røynda ein forskar på. Data kan vere fysiske ting som spørjeskjema, videoopptak eller transkripsjonar av intervju. Data er noko me skaper og er bindeleddet mellom analysen og røynda og tolkinga av røynda. I samfunnsvitskapleg forsking er data ofte ei handling som vert utøvd, medan innan andre forskingsområde kan data vise til noko som vert gitt. Når røynda vert observert og på ein eller anen måte vert registrert, vert dette data, til dømes ein transkripsjon av ein samtale mellom elever når dei arbeider med oppgåver basert på vilkår. Dette datamaterialet er ikkje røyndommen, men ein representasjon av røyndommen.

3.7.1 Planlegging av oppgåver

Under planlegginga av datainnsamlinga var utforminga av oppgåvene i fokus hjå meg. Ved å fokusere på ein av Brennan og Resnick (2012) sine konsept kunne eg avgrense omfanget av oppgåvene og spesialisere dei meir. Vidare i planleggingsfasen måtte eg avgjere korleis eg ønskte å samle inn data og kva som ville vere passande metode for å få svart på problemstillinga mi. Til slutt måtte eg avgjere kva type data eg ønskte å samle inn. Desse vala vart grunnlaget for vidare gjennomføring og oppsett av datainnsamlinga.

Under planlegginga av oppgåvene tok eg kontakt med elevane sin matematikklærar for å undersøke kva forkunnskapar dei hadde. Mattematikklæraren deira informerte meg om at dei hadde ikkje undersøkt eller arbeide med oppgåver som brukte vilkår og bygde på logikk med vilkår. Dette gjorde at eg planla ein introduksjon på vilkår for alle elevane, slik dei hadde den nødvendige kunnskapen til å løyse papiroppgåvene. Introduksjonen gjekk ut på gje dei eit par dømer og forklare korleis desse oppgåvene var bygd opp.

Vidare hadde eg ein samtale med både elevane og læraren deira. Læraren informerte meg om at dei hadde erfaring med ulike blokkbaserte program. Dei hadde også brukt Scratch tidlegare, men brukte det til å lage animasjonar til eit kunst-og-handverksprosjekt. Læraren informert meg og om at elevane som deltok hadde ulike erfaringar med programmering. Nokre hadde ikkje hatt programmering som valfag, medan andre hadde ei ekstra interessa for programmering og dreiv på med det på fritida.

3.7.2 Oppgåvesett

For å finne programmeringsoppgåvene søkte eg både på internett og leitte i bøker. Kriteria mine var at oppgåvene skulle vere opne for fleire løysingsmetodar, passande for elevane sin alder og nivå og det måtte gå an å bruke Scratch til å løyse oppgåvene. Eit viktig kriteria for oppgåvene var det ei skulle handle om vilkår. Eg valte å ha opne oppgåver slik at elevane kunne diskutere ulike løysingar utan å føle seg bundne til å bruke omgrep som er nært knytt til programmeringsspråket. Eg enda med oppgåver frå boka til Leving om diskret matematikk (Levin, 2019). Etter litt tilpassing i lag med rettleiar og etter piloten kom eg fram til eit oppgåvesett på tre oppgåver. Oppgåvene, sjå vedlegg 1, hadde liknande struktur slik at elevane kunne bygge løysingsmetodane på kvarandre. Oppgåve 1 og 2 hadde nokså like tal steg dei måtte ta for å nå løysingen, medan oppgåve 3 var litt meir kompleks og var bygd opp av fleire deler.

Den fyrste oppgåva elevane skulle løyse var nok den lettaste, fordi den hadde berre ein tilstand dei skulle ta omsyn til - det var berre to utfall som kunne skje. Denne oppgåva gav mogelegheit for nøye testing sidan det var berre to mogelege utfall.

Oppgave 1 for Programming

Planlegg og skriv et program der Sprite spør om din alder.
Hvis du er 19 eller yngre, svar at du må være på skolen.
Hvis du er eldre enn 19, svar at du må være på jobb.

Den andre programmerings oppgåva er laga på ein slik måte der elevane kan bruke opp att deler av den fyrste oppgåva. Det vart gjort slik for at dei skulle få mogelegheit til å bruke den gjenbrukende tilnærminga og for å undersøke om dei kjende det igjen.

Oppgave 2 for Programming

> *Skriv et program der Sprite spør noen om alderen deres.*
> *Hvis svaret mindre enn 18 år, få Sprite til å si "Du er for ung til å kjøre", og hvis de er over 18, få Sprite til å spørre om de har tatt lappen.*

I den siste programmeringsoppgåva skal elevane gjennomføre fleire operasjonar. Her må dei ta vare på tre variablar og deretter utfører ulike utrekningar basert på «personens» alder og timer. Denne oppgåva er laga slik at elevane må dele opp oppgåva og arbeide med fleire variablar samstundes.

Oppgave 3 for Programming

> *Du jobber i en butikk, og der får man betalt 45 kr per time frem til fylte 20 år. Når du er 20 år eller eldre får du betalt 55 kr per time. Planlegg og skriv et program der Sprite spør om en persons navn, alder og antall timer jobbet i løpet av en uke. Sprite skal da regne ut lønnen og skrive den ut samen med personens navn og alder.*

Å finne oppgåver som kan løysast på papir i grupper syntest eg var meir utfordrande. Den fyrste utfordringa eg møtte var at logiske oppgåver ofte er retta mot studentar på vidaregåande eller høgare utdanning. Desse oppgåvene vart luka ut av fleire grunner, der dei viktigaste var at det ofte var komplekse oppgåver som ville krevje meir forkunnskapar enn elevane hadde, og at oppgåvene ofte var ein samansetting av fleire ulike konsept, medan eg ønskte å fokusere på konseptet vilkår. Oppgåver som var i matematikkbøker retta mot ungdomsskulen inkluderte oppgåver om vilkår men dei var meir retta mot sannsyn, medan oppgåver innan programmering var retta mot å lære seg algoritmar og programmeringsspråket. Til slutt enda eg med oppgåver om logikk frå læreboka *Tall og Tanke* og oppgåvehefte deira (Smestad, 2021; Solem, 2017). Oppgåvene fokuserte på likningar og var hovudsakleg utsegn som basert på sant eller usant.

Dei tre fyrste oppgåvene elevane løyste på papir er den som er nærast oppgåvene dei løyste ved hjelp av programmering. Dei bruker tilstand og gir vilkår. Det spørjast etter kva utfall som stemmer. Under er eit døme på oppgåve 1.

Oppgave 1 oppgåve på papir

> *Hvis Helle har tre eller flere epler, så kan hun lage en eplekake.*
> *Helle har fem epler og et gresskar. Kan hun lage en eplekake?*
> *Ja eller Nei*

Oppgåve 4 og 5 består av tabellar der elevane skulle diskutere samanhengen mellom dei ulike utsegna. Her skulle elevane avgjere om utsegna var avhengige av kvarande og, dersom ein snudde på rekkefølgene på utsegna, om dei framleis var logisk avhengige av kvarandre.

a)	Tromsø er en katt.	Tromsø er en gråstripete katt.
b)	Figuren har fire hjørner.	Figuren er et kvadrat.
c)	Sol har fire ben.	Sol er en katt.
d)	Sekken veier mindre enn 20 kg.	Sekken veier mer enn 10 kg.
e)	$x^2 = 25$.	$x = 5$.
f)	Sarakka bor i Karasjok.	Sarakka bor i Norge.

Figur 5 Oppvåve 4 frå vedlegg 1(Smestad, 2021)

Oppgåve 6 er den oppgåva elevane opplevde som mest utfordane. Dei skulle finne dei kontrapositive utsegna til utsegna som var vist. Elevane fekk utdelt eit døme men ikkje noko meir informasjon enn det. Eit utsegn P → Q I er logisk likt til sitt kontrapositive utsegn ¬Q → ¬P (Levin, 2019). Oppgåva bruker uttrykket motsette utsegn fordi utrykket kontrapositive utsegn er ukjend for elevane. Målet med denne oppgåva var at elevane skulle utforske noko dei ikkje hadde kjennskap til og dermed bruke ulike strategiar for å finne fram til svaret.

Oppgave 6 oppgåve på papir

Diskuter samanhengen mellom desse to utsagnene.
Hvis figuren er et kvadrat, er alle vinkelen rette. Hvis ikke alle vinkelen er rette, er figuren ikke et kvadrat.
Hva ville vert det motsatte for dette utsagnet?
Hvis x er mindre enn 100, så er x mindre enn 1 000.

3.7.3 Pilot

Under gjennomføringa av piloten erfarte eg eit par utfordringar som gjorde at eg endra gjennomføringa av observasjonen med elevane då dei arbeide med oppgåvene på PC. Piloten vart gjennomført på ein gruppe med to elevar. Den fyrste utfordringa var at elevane hadde kvar sin PC. Dette gjorde at dei retta merksemda si mot meg og ikkje hadde ein samtale seg imellom. Elevane retta spørsmåla og utfordringane sine mot meg i staden for til kvarandre, dette gjorde at diskusjonen mellom elevane vart minimal. Dei arbeidde også med oppgåvene i ulikt tempo, noko som gjorde at lydopptaka og diskusjonen var ikkje strukturerte. Desse utfordringane løyste eg ved at elevane brukte same PC og løyste oppgåvene saman. Eg oppfordra og elevane til å diskutere og forklare for kvarandre kva dei gjorde og tenkte.

Den neste utfordringa var at pilotgruppa viste stor usikkerheit ved bruk av programmet Scratch. Dette førte igjen til ein del usikkerheit og samtale om programmet, og ikkje om oppgåvene og løysingsmetoden. Dette valte eg å løyse ved å lage ei liste over ulike blokker dei kunne bruke og eg hadde fyrst ein samtale med elevane om desse oppgåve. Denne lista fungerte som eit oppslagsverk for elevane.

Den siste utfordringa eg møtte i pilotundersøkinga var at elevane vart usikre på formuleringa av oppgåvene. Oppgåvene var formulert meir som ei generell programmeringsoppgåve, ikkje spesifikt retta mot Scratch. Dette gjorde at elevane ikkje heilt viste kva som var forventa av dei. Dette løyse eg ved å spesifisere at ikonet «sprite» skulle spørje og gi svar på spørsmål.

3.7.4 Gjennomføring

Gjennomføringa av observasjonane vart gjort over to dagar. I forkant hadde eg samen med læraren delt ut og samla inn samtykkeskjema. Med fråfall på dagen på grunn av sjukdom var det ni elevar som ville vere med. To av elevane deltok på pilotundersøkinga som vart haldt den fyrste dagen. Resten gjennomføre dataoppgåver dagen etter. Den fyrste dagen fekk alle elevane ei innføring i oppgåvene dei skulle gjennomføre. Her fokuserte eg på å vilkår og korleis oppgåver basert på vilkår kan løysast. Her fekk elevane prøvd seg på eit par oppgåver og me diskuterte korleis det fungerte. Deretter delte me dei i grupper der dei elevane som ønskte å vere med på observasjonen vart verande på eitt klasserom, medan resten var på eit anna klasserom. Elevane vart delt inn i tre grupper med tre personar på kvar. Elevane brukte eitt sett med oppgåver der dei samarbeidde med å løyse dei. Dei fekk også utdelt eit opptaksband som tok opp samtala deira. Elevane brukte mellom 20 og 30 minutt på å løyse oppgåvene.
Den neste dagen deltok tre grupper med dei resterande sju elevane på observasjon då dei gjennomførte programmerings oppgåvene. Denne observasjonen vart gjennomført ved at elevane fyrst fekk ein introduksjon i dei mest brukte blokkene på Scratch. Deretter brukte dei programmet til å løyse oppgåvene. Elevane sin samtale vart tatt opp og transkribert. Me gjennomførte også skjermopptak slik at eg kunne sjå kva slags løysingar dei brukte og korleis dei kom fram til desse. Elevane brukte mellom 30 og 40 minutt på oppgåvene. Ei av gruppene gjennomførte berre to av tre oppgåver på grunn av tidspress.

3.8 Kvalitet i forsking

Postholm og Jacobsen viser til Fox (1958) når dei forklarer behovet for forskingskvalitet (Postholm & Jacobsen, 2018). Jo nærare me kjem oppdaginga av deler av aninga, desto større er omfanget av det vi ikkje veit (Postholm & Jacobsen, 2018, s.219)
Dette sitatet understrekar viktigheita av forskingskvalitet. Forskingskvalitet er altså ikkje berre knytt til resultata forskaren kjem fram til, men avgjerast heller av korleis kunnskapen vert produsert. Funna som ein utarbeider i ein forskingskontekst er forskaren si forståing utvikla i settingane og situasjonane som vert studert. Desse funna vert ramma inn av problemstillinga. Kunnskap som vert utarbeidd av desse funna vil vere kontekstuell fordi han er konstruert i møtet mellom forskaren og settinga til forskinga. Funna mine vert dermed avgrensa til kjenneteikna til algoritmisk tenking fordi problemstillinga mi fokuserer på det. Både settinga og kunnskapen min vil vera med på å påverke funna. Dette gjer at kunnskapen ikkje kan vere objektiv.

3.8.1 Pålitelegheit og gyldigheit

I kvalitative studer vert reliabilitet erstatta med pålitelegheit, dette er fordi det er vanskeleg å reflektere over møtet mellom forskaren, forskingsfeltet og mennesket som deltek i studien. Menneske kan fortone seg ulike, og ulike forskarar vil ta med seg ulike subjektive og individuelle teoriar inn i forskinga si. Reliabilitet handlar meir om studien kan etterprøvast, altså at dersom den same studien vert gjennomført av nokon andre, vil dei få same resultat. Pålitelegheit er meir knytt til ein refleksjon over undersøkinga og korleis forskaren verker inn på resultata (Postholm & Jacobsen, 2018). I ein kvar diskusjon om pålitelegheit må forskaren kunne skildre for seg sjølv og andre relasjonen mellom forskaren og deltakarane, forholdet mellom problemstillinga og deltakarane, forskingskonteksten, kva grupper som ikkje vert dekka av utvalet og argumenter for at ein har fått med alt som er viktig (Postholm & Jacobsen, 2018). Forholdet mellom meg og deltakarane, forskingskonteksten og utvalet vart diskutert tidlegare i dette kapittelet. Problemstillinga vart justert etter kor mange elevar som kunne delta i denne studien. Dermed var problemstillinga avhengig av elevane. Avgrensinga av datamaterialet vart diskutert i 3.10 Avgrensing.

Gyldigheit kan delast inn i indre og ytre gyldigheit. Indre gyldigheit handlar om to forhold. Det fyrste er samsvar mellom det ein påstår ein studerer og analyserer, og teorien som ein nyttar for skildre dette fenomenet. Det andre forholdet er om ein har grunnlag for å uttale seg om kausalitet ut i frå studien ein har gjort (Postholm & Jacobsen, 2018). Samsvaret mellom det ein studere og omgrepa som skilder dei. Postholm og Jacobsen viser til Kerlinger (1979) sitt kjente sitat «måler du det du trur du måler?» (s.138), for å understreke nødvendigheita av at omgrepa representer empirien. Ved å bruke eit rammeverk og følgje dette tett opp har eg eit bilete eg kan samanlikne datamaterialet mitt med. Dette biletet kjem frå det originale rammeverket, men også andre artiklar som har brukt det (Brennan & Resnick, 2012; Cui & Ng, 2021; Israel & Lash, 2020;).

Kausalitet er det andre forholdet som Postholm og Jacobsen(2018) brukte for å skildre indre gyldigheit. Innafor samfunnsvitskap vert ikkje kausalitet sett på same måte som innafor naturvitskapen. Her er det ikkje nokre tydeleg rette svar eller kausale lover som ein kan diskutere. I dette prosjektet er kausalitet ikkje noko særleg relevant på grunn av omfanget og metodane som vert brukt. Det er mogeleg å diskutere kva som skjedde i dei tilfella eg undersøkte, og trekke lærdom ut frå dei i samband med teori, men å påpeike noko kausalitet er ikkje nødvendig.

Den ytre gyldigheit handlar om i kor stor grad ein kan overføre funna til andre kontekstar, eller med andre ord om det er mogeleg å generalisere funna (Postholm & Jacobsen, 2018). Sidan eg arbeide med sopass få elever og fokuserte på casen deira, kan generalisering verte ei utfording. Eg kan fortelje om trekk eg såg ved fleire av casane mine, men sidan alle casane er satt saman av elever frå same klasse, er det vanskeleg å seie noko meir ut i frå det.

3.8.2 Triangulering

Triangulering er ein måte å stryke både pålitelegheita og gyldigheita på. Triangulering handlar om å skaffe seg eit mangfald av datakjelder. Ein bør altso hente informasjon frå ulike forskarar, som brukar ulike forskingsdesign, og datainnsamlingsmetodar. I oppgåva mi tek eg i bruk ulike former for både datakjelder og metodar for datainnsamling (Postholm & Jacobsen, 2018). Eg har både lydopptak av elevane, elevsvar på ark og skjermopptak frå det dei gjorde på PC som mine datakjelder. I tillegg gjennomførte eg observasjonen min to ulike gongar. Triangulering er altså ein prosedyre som har intensjon å skildre røynda frå leire ulike vinklar og slik skape eit meir heilskapleg bilete av eit samansett fenomen (Postholm & Jacobsen, 2018). Innafor konstruksjonisme paradigmet er triangulering ei prosedyre som kan fange opp røyndommen der og då, medan forskinga føregår, sjølv om settinga, menneska og forskaren hele tida endrast. Postholm og Jacobsen legg vekt på at triangulering er ein tidkrevjande praksis og at det kan vere utfordrande for masterstudentar å gjennomføre sidan ein kan gå seg vil innafor fleire datakjelder og metodar. Dette kan gjere at ein mistar fokus på sjølve prosjektet ein skal gjennomføre.

3.9 Etikk

I studiet mitt har eg arbeide etter NESH (2021) sine retningslinjer for etiske overveiingar. NESH anerkjenner at observasjon av elevar kan gjere dei sårbare, spesielt om det vert tatt opp lyd av dei. For å ivareta elevane sine interesser, integritet og personvern informerte eg dei tydeleg om kva som var hensikta med forskinga og korleis eg kom til å arbeide med prosjektet. Eg hadde også tett kontakt med både kontaktlærarane deira og faglæraren deira, slik at elevane ikkje følte på noko press ifrå meg. Sidan elevane ikkje kunne gje eit aktivt samtykke, fekk dei med seg informasjonsskriv som informerte om forskingsprosjektet og at det vert godkjent av NSD, samt eit samtykkeskjema som føresette kunne skrive under på. I etterkant kunne elevane og lærarane trekke seg ut av studien når som helst. Dette skjedde under datainnsamlinga på grunn av ein del sjukdom som gjorde at nokre elever ikkje kunne delta som planlagt. Berre elevar som hadde gitt både aktivt samtykke sjølv og hadde løyve frå føresette er ein del av dette prosjektet. Elevane har fått tildelt tilfeldige alias spesielt for kvar elev. Skulen vert heller ikkje omtalt i studien slik at det ikkje er nokre som kan knytte det tilbake til elevane. All data som vert samla vert lagra på ei spesiell sky som krypterte alt innhaldet.

3.10 Avgrensing

Analysen vert avgrensa til kategoriane basert på Brennan og Resnick (2012) sine tilnærmingar for algoritmisk tenking. I analysen ser eg berre etter desse kategoriane og ikkje noko meir. Ved å avgrense analyseringa kan eg fokusere på å undersøke og tolke diskusjonane til elevane for å finne svar på forskingsspørsmåla mine. For å kunne gjennomføre ein kvalitativ analyse må ein bryte ned datamaterialet i mindre delar og undersøke dei nærare. For å kunne vite kva ein so skal halde fram med, må ein avgrense datamaterialet og luke ut det som vert unødvendig (Postholm & Jacobsen, 2011). Dette blei gjort ved å berre bruke det algoritmiske konseptet vilkår, og bruke dei eksisterande kategoriane til Brennan og Resnick(2012).

4 Analyse

I dette kapittelet skal eg presentere analysen av det innsamla datamaterialet, som er bygd på kvalitative prinsipp for analyse. Her har eg valt å bruke ein kategorisk analyse basert på Stake(1995) si skildring av analysar innan casestudiar. Eg nyttar Brennan og Resnick(2012) sitt rammeverk for å analysere datamaterialet med grunnlag i problemstillinga min.

4.1 Kva kjenneteikn kan me sjå i elevane sitt arbeid?

I den første delen av analysen undersøker eg kva som kjenneteiknar elevane sitt arbeid då dei arbeider med oppgåver basert på vilkår. Korleis kan me kjenna att algoritmisk tenking då dei arbeida med oppgåver basert på det algoritmisk konseptet vilkår? I avsnitta under undersøker eg alle dei ulike algoritmiske tilnærmingane for å sjå korleis dei kom fram i elevane sitt arbeid, for deretter å sjå etter fellesnemnarar for tilnærmingane.

4.1.1 Ei adaptiv og iterativ tilnærming

Den adaptive og iterative tilnærminga består av at elevane tilpassar arbeidsmetodane sine ut i frå oppgåva. Dette kan skje idet dei byrjar å arbeide med oppgåva, men også undervegs i oppgåveløysinga. Denne tilnærminga vart observert i fleire av casane, men for det meste i dei casane der oppgåvene vart gjennomført i ein programmeringskontekst. Det kom tydleg fram i desse casane at elevane prøvde å lage ein plan som etter kvart vart vidareutvikla undervegs. Mange av tilfella der den adaptive og iterative tilnærminga var til stades, hende i byringa av oppgåveløysinga, og oftast under eller rett etter opplesinga av oppgåveteksten. Her observerte eg at elevane prøvde å isolere kva element det var behov for i oppgåveløysinga, og dermed starta først med dei. Dei arbeidde vidare med løysinga ved å adaptere planen undervegs, dersom første utkastet av løysinga ikkje fungerte.

Den adaptive og iterative tilnærminga vart også synleggjort i oppgåveløysinga til Gruppe 1 då dei arbeida med programmeringsoppgåvene. Elevane diskuterte kort ulike løysingsstrategiar og kva blokkar dei ville bruke, ofte etter dei leste opp oppgåva. I oppgåve nummer 2 som liknar oppgåve nummer 1, uttalte elevane at dei kunne bruke det dei hadde gjort i den førre oppgåva. Dette har eg valt å kategorisere som ei gjenbrukende tilnærming, sjølv om denne utsegna viser til elevane si planlegging av programmet dei skal lage, og moglegheitene til å tenke framover i programmeringa. I den første oppgåva når dei jobba med programmering, las gut 1 opp oppgåva og deretter svarer gut 2 med nokre utsegn at oppgåva likner på det dei gjorde tidlegare. Herifrå byrjar dei planlegginga av oppgåva. Dei oversett oppgåveteksten til kode som kan bli forstått av ein pc. I dette tilfellet diskuterte dei oppgåve 1, der dei skal skrive eit program som undersøker alderen til eleven og deretter fortel om ein skal vera på skulen eller ikkje. Elevane har på dette tidspunktet ikkje lagt inn noko på Scratch programmet sitt, men vurderer ulike blokker som dei kan bruke til å løyse oppgåva.

Gut 1: Er det ikkje «if» og «else» du må buke.
Gut 1: Er så hvis du er det eine du være på skolen eller visst du er det andre så

må du vera på jobb?
Gut 2: Og så må du bruke ein sånn her.

Gruppe 1 brukte den adaptive og iterative tilnærminga mykje meir enn den andre gruppa og dette kunne ein sjå ved at dei nemnte planen for oppgåveløysinga og endra den undervegs. Den andre gruppa brukte også denne tilnærminga i sin løysingsmetodikk. Her var det igjen mest tydeleg då dei skulle gjennomføre oppgåvene i ein programmeringskontekst. Her hadde dei ein liknande tilnærming som gruppe 1 der dei planla og oversett oppgåva ved hjelp av å bruke Scratch sitt språk som bru mellom oppgåva og maskinen.

Gruppe 2 brukte denne tilnærminga under oppgåveløysing på papir ved å endre på planen når dei løyste oppgåve 6. Denne oppgåva bestod av eit utsegn om tre kuler og ei vekt, her var to kuler like tunge og den siste kula var lettare en dei to andre. Elevane si utfording var å finne kva kule som var lettast med færrast mogleg vegingar. Dei har utforma ein plan og deretter utførar planen. Her kjem dei fram til eit svar som dei ikkje er nøgd med. I utsegna under så adapterer Elev 1 den første planen til ein ny løysing. Dei arbeidar igjennom ulike scenario med feilsøking og endrar planen sin ut i frå resultata av feilsøkinga.

Elev 1: Ei gruppe elever får utdelt ei vekt og tre kuler. Ein blå, ein grøn, og ein raud. To av dei er like tunge, medan ein er lettare enn dei to andre. Då må dei vege dei 2 eller 1(gonger)... Ein kan legge to på om gongen.

Elev 1: Men høyr, du legg på to kuler, om den eine er lettare enn den andre så veit du kva som er den lette. Om den andre er tyngre enn den eine så veit du kva som er lettast, og om du legg to på og dei er like tunge så veit du at den som ligg igjen er lettast. Då finn du uansett ut kva som er lettast på ei veging.

4.1.2 Ei testande og feilsøkjande tilnærming

Denne tilnærminga til løysingsmetode var den eg observerte mest av i observasjonen av elevane. Den var svært tydeleg i samband med programmeringskonteksten, men kom og fram under oppgåveløysinga på papir. Då undersøkte elevane oppgåvene fleire gonger for å sikra seg at svaret var rett. Her testar elevane programmet dei skriv etter kvar oppgåve og endra dei etter svaret på feilsøkinga. Det var ulik grad kor nøye dei prøvde ut korleis programmet fungerte, men gruppene hadde oftast ei fullstendig testing, der dei prøvde ut fleire ulike variablar, for å sjekke at svara var korrekte i alle tilfelle. Elevane brukte resultatet frå feilsøkinga til å enten anerkjenna at dei hadde løyst oppgåva rett, eller for å finne eller oppdage feil i logikken. Gruppe 1 gjennomfører i utdraget under ei rekke testar for oppgåve 2. Her prøver dei ut ulike variablar for å sjå om alle aspektane ved programmet fungerer. I utdraget under har elevane laga eit program for å teste oppgåve 2 i programmeringsoppgåvene. Her skal dei skrive eit program som spør etter to variablar, alder og om du har tatt lappen. Gut 1 og 2 arbeider her samen for å teste ut ulike kombinasjonar som programmet kan ta inn.

Gut 1: Då skriver du for eksempel 22.
Gut 2: Ja me prøver med det, har du tatt lappen? Nei

Gut 2: Svar: Du kan ikkje kjøra.
Gut 1: Prøv da same berre du seier ja.
Gut 2: Har du tatt lappen. Ja da du kan kjøre.
Gut 1: Prøver med 11, for ung til å kjøre. Alt funger.

Her prøver eleven ut ulike verdiar som skal gi ulike svar. Då dei har prøvd ut dei ulike mogelegheitene, konkludere dei med at alt fungere. Ved å prøve ut dei ulike verdiane, kan eleven vite med større sikkerheit at programmet faktisk fungerer, og ved å gjere denne feilsøking, viser elevane ein testande og feilsøkjande tilnærming til oppgåveløysinga. Feilsøkinga gir og eleven ein tryggleik og sjølvtillit der dei får bekrefta at deira program fungerer.

Brennan og Resnick(2012)skilder ein rekke ulike strategi som kan bruker til feilsøking og testing:

1. *Identify (the source of) the problem*
2. *Read through your scripts*
3. *Experiment with scripts*
4. *Try writing scripts again*
5. *Find example scripts that work*
6. *Tell/ask someone else about the problem*
7. *Take a break*

I dømet over illustrere dei to ulike teknikkar for feilsøking og feilretting (debugging). Dei bruker både ulike testar på programmet din funksjon, men dei søker og anerkjenning av partneren sin. Dette kunne observerast igjen i begge gruppene. Andre strategiar som elevane bruker er å identifisere problemet, lese igjennom, utprøving av koder og snakke med andre.

Andre strategiar som Brennan og Resnick(2012)skildrar er å ta ein pause i oppgåveløysinga, alternativt å sjå etter andre liknande koder som fungere. På grunn av skulesituasjonen observasjonen fann stad i, hadde ikkje elevane moglegheitene til å ta noko særleg med pausar, eit alternativ til pausar var at dei kunne ha gått vidare på andre oppgåver, men ingen valte å gjere det. Elevane valte ikkje å søke etter svar på internett, som er ein vanleg praksis innanfor programmering. Dette kan komme av at dei ikkje er vane med å gjera det, eller dei kan sjå på den type informasjonsinnsamling som juks eller plagiat. Dersom dei sammelinka eller brukte oppatt kode dei allereie hadde skrevet, har eg valt å kategorisere det som ei gjenbrukende tilnærming.

Då eleven brukte den feilsøkende tilnærminga ved oppgåveløysinga på papir, var det svært ulikt det dei gjorde i ein programmeringskontekst. I forbindelse med programmering tok dei ulike oppgåvene og «feilsøkte» dei ein om gangen, medan då dei løyste oppgåver på papir testa dei fleire oppgåver samtidig. Her i denne analysen har eg kategorisert tilfellet då elever går tilbake på oppgåvene for diskutere dei meir som feilsøkende og testande tilnærming. Her observerte eg at elevane kommuniserte meir enn første gangen dei gjennomførte oppgåva og dette gjorde at dei kunne finne fleire feil. I døme under har Gruppe 2 diskutere dei oppgåvene igjen etter dei har svar på. Dette er andre gangen dei diskutere dette utsegna. I denne andre diskusjonen vel dei å

framme sine argument meir tydeleg. Her ytra eleven sin meining og dei andre på gruppa var einige.

> *Jente: Stemme den denne veiene, visst han bur i Karasjok så bur han i Norge, men visst han bur i Norge så treng han ikkje å bu i Karasjok.*

4.1.3 Ei gjenbrukende tilnærming

Denne tilnærminga vert ikkje så ofte brukt av elevane som dei to første tilnærmingane, både i programmering og oppgåveløysinga på papir kontekstene. Ei gjenbrukende tilnærming vert samansett av å bruke oppatt kunnskap, mønster, algoritmar og andre deler av informasjon som skal til for å løyse oppgåva. Her observerast tilfelle der elevane tydeleg gjenkjenner at dei kan bruke oppatt informasjon, og den vert kategorisert som gjenbrukende tilnærming. I tilfelle 1 så identifiserte gut 1 at oppgåva dei arbeida med var lik den førre oppgåva. Dette gjorde at dei på at dei kunne bruke oppatt oppgåve 1 sin løysingsmetode i oppgåve 2. Det andre tilfellet var å bruke oppatt deler av ei kode dei hadde allereie skreve, igjen i den same oppgåva. I begge gruppene kunne elevane kjenne igjen då det var nyttig å bruke denne tilnærminga.

> *Tilfelle 1*
> *Gut 1: Litt som i sta?*
> *Gut 2: Litt som i sta ja*
>
> *Tilfelle 2*
> *Gut 1: Går det ann å kopiere den eller noko sånt?(Elevane viser til ein del av koden som kan gjenbrukas i ein annan del av programmet)*
> *Gut 1: Ja, både den og den.*
> *Gut 2: Tar den og berre kopiere den då.*
> *Gut 1: Sånn ja, satt du den tilbake. Du treng å gjere det ein gang til fordi me skal jo bruke den.*
> *Gut 1: Navn og den der. (Elevane bruker igjen den blokka der Sprite spør om alder og berre endra på spørsmålet)*

Utdraget over illustrerer dei to ulike tilfelle som peikte seg ut som gjenbrukende tilnærminga. Der det første utdraget omhandlar då elevane viser til heile oppgåva og det andre viser til då elevane bruker delar av løysinga oppatt i den same oppgåva. I begge tilfella oppdaga elevane dette svært raskt. Dersom dei ikkje brukte denne tilnærminga i starten av oppgåva eller i byrjinga av det neste steget på løysinga, var det ofte at elevane ikkje brukte den gjenbrukende tilnærminga.

Då elevane løyste oppgåvene på papir var den gjenbrukende tilnærminga nesten ikkje til stades i den eine gruppa. Gruppe 1 brukte den gjenbrukende tilnærminga, då dei diskuterte oppgåve 3 der Gut 1 viste til dømet som var brukt tidlegare i ein introduksjon til timen. Eleven kjenner igjen oppgåva og peikar på at det er berre tala som vert endra, slik at dei kan løyse oppgåva likt som vert gjort i introduksjonen.

> *Gut 1: Stemmer ikkje, da var sånn på PowerPoint, dei har berre bytta ut tala.*

Medan i Gruppe 2 brukte dei denne tilnærminga aktivt i oppgåveløysinga på papir. Då kom dette fram både då det vart gjenkjente at oppgåvene likna på kvarande, men og som eit verktøy då dei satt fast i ei oppgåve. Då ytra dei at dei skulle attende og sjå på dømet, for å skape klarheit i oppgåva og om dei kunne kjenne igjen nokre av mønstera dei skulle bruke.

4.1.4 Ein abstrakt og modulariserande tilnærming

Denne kategorien tilnærming har eg fokusert på deler der elevane har valt å dele opp oppgåva og fokusere på trinna til oppgåva i staden for å takle heile oppgåva. I programmeringskonteksten arbeider elevane ofte med å takle kvar if blokk kvar for seg. Dette gav ein naturleg oppdeling og stegvis oppgåveløysing for elevane. Vidare brukte dei oppgåve oppsettinga som ei inndeling for å skape moduler. Det vert observert at i oppgåver som består av fleire komponentar der programmet skulle spørje om ulike kriteria var oppfylt, valte ofte elevane å dele opp oppgåva i dei delane og sette dei saman programmet til slutt. I dømet under både kategoriserer og prioriterer elevane kva dei skal gjere. Her arbeider dei med den siste oppgåva på programmeringsdelen av oppgåvene, som består av fleire enkelt delar og dei har diskuterte litt kva ulike elementa som må til for å løyse oppgåva. Her trengte dei fleire ulike variablar som alder, namn og tal timar. Deretter måtte dei lagre desse variablane.

> *Gut 1: Først må den spørja om alderen. Kor gammal er du?*
> *Gut 2: Kvifor har du «if» blokk der då? Du skal jo berre spør om alderen.*
> *Gut 1: Først må me spør kor gammal dei er før me finner ut kor mykje dei tener.*
> *……..*
> *Gut 1: Først. Spør om du kan han heiter, er han skal berre huske på namnet. Skal me få da til?*
> *Gut 2: Sikkert den here då?*
> *Gut 1: Då setter me namn til svaret. Så neste tingen blir alderen då?*
> *Gut 2: Spør først om kor gamle dei er, og så må me ha sånn «if».*

I utdraget setter elevane opp ein prioritering over både kva dei skal gjere, men og ein rekkefølgje. Dei veler å diskutere dei ulike modulane. I det første utsegna diskuterte elevane den første delen, deretter kjem dei tilbake til den andre delen. Dette kan tyde på at dei har delt opp oppgåva. Desse delane blir deretter til modular som gjer at elevane kan handtere større oppgåver ved å dele dei opp og løyse dei forskjellige segmenta individuelt. Elevane vel å komme tilbake til modulen som handlar om alder seinare, dette kan tyde på at dei tenker på ulike deler av oppgåva som meir separate og kan arbeidast på til ulik tid.

I konteksten der elevane løyste oppgåvene på papir var det litt meir utfordrande å kategorisere då elevane brukte den abstrakte og modulariserande tilnærminga. Det som gjorde seg klart etter analysen var at då elevane skulle diskutere oppgåvene for andre gang valte dei ofte å dele oppgåve meir opp. Denne tilnærminga blei og ofte observert saman med feilsøkinga ved oppgåveløysing på papir. I utdraget under diskuterer elevane i gruppe 1 eit utsegn. Dei etabler at utsegna er sant då A utsegna følger B utsegna. Elevane diskutere om dersom B følger A vil ha same utfallet.

Gut 2: Kari bur i Karasjok, Kari bur i Norge, den stemmer, ikkje sant?
Gut 1: Stemmer ikkje, visst ho bur i Norge, så trenger ho ikkje å bu i Karasjok.

Dette utdraget viser at Gut 2 etablerer at første del av oppgåva stemmer, medan Gut 1 sitt vidare arbeid viser at andre del av oppgåva ikkje stemmer.

4.2 Kva spor kan me sjå av algoritmisk tenking i elevane sitt arbeid i både ein programmeringskontekst og ved oppgåveløysing på papir?

Denne delen av analysen vil utforske og undersøke kva spor av algoritmisk tenking me kan sjå hos dei forskjellige elevgruppene då dei løyser oppgåver på papir og i ein programmeringskontekst. Her vil eg undersøke dei ulike tilnærmingane til elevane, samt prøve å skildre både likskapar og ulikskapar mellom oppgåveløysingane hos elevane.

4.2.1 Gruppe 1

For å sjå etter algoritmisk tenking i denne gruppa har eg undersøkt elevane si bruk av tilnærmingar. Elevane har i tydelege trekk brukt tilnærmingane, men dei presenter seg ulikt mellom programmeringskontekst og oppgåveløysing på papir. Dette vil eg utdjupe og diskutere vidare i avsnittet under. I oppgåveløysinga kunne me sjå at elevane brukte ulike tilnærmingar for å navigere oppgåva. Gruppe 1 hadde ein tydeleg bruk av feilsøking i begge observasjons moglegheitene.

Forskjellen mellom elevane si tilnærming då dei gjennomførte oppgåvene i dei ulike kontekstane var tydlege. Her kunne me sjå korleis dei valte og kor ofte dei valte å gjennomføre tilnærmingane.

Likskap og skilnader innan Gruppe 1 då dei gjennomførte ein adaptiv og iterativ tilnærming

Då elevane brukte den adaptive tilnærminga i programmeringskonteksten, var det meir retta mot planlegging av kva ulike verktøy dei trengte og utvikingen av denne planen. Dette kunne gi uttrykk for ein diskusjon om dei ønska å bruke if blokker og kor dei trong variablar. Medan i oppgåveløysing på papir var denne tilnærminga svært lite til stede. Her brukte elevane mest opplesinga av oppgåva i planlegginga for å identifisere moglege problem eller element som dei skulle handtere, deretter prøvde dei seg fram og endra det etter kvart. Ein slik måte å bruke denne tilnærminga på var likt hos begge gruppene som var observert. Dette kan komme av at evnene til å identifisera problema ikkje er ekskludere til oppgåveløysing i ein programmeringskontekst, medan å diskutere kva verktøy ein treng er meir nødvendig i ein programeringskontest for elevane. Dette kan komme av at dei ikkje har like mykje trening på å skrive programmer som dei har å løyse oppgåver på papir. I døme under diskutere elevane kva som skal til for å løyse oppgåva, og vel å fokusere først på om å lage kode for å spør om ein har tatt lappen. Dette gjer dei ved å modulere oppgåva og ta for seg ein av delane først.

Gut 1: Men korleis skal eg få da til?
Gut 2: Nå du ikkje berre spør han då.
Gut 2: Då må du spør om han har tatt lappen

Likskap og skilnader innan Gruppe 1 då dei gjennomførte ein testande og feilsøkende tilnærming

Ein testande og feilsøkjande tilnærming er den tilnærminga som vert brukt mest over begge observasjonen for både Gruppe 1 og 2. Det som sto ut som ein tydeleg forskjell var at elevane var mykje meir inkonsekvent med feilsøkinga si då dei gjennomførte oppgåver på papir. Eit av tilfella der dei avgjorde å gjenta feilsøkinga i eit sett med oppgåver skjedde etter dei hadde fått tilbakemelding og informasjon om oppgåva. Deretter var feilsøkinga framleis rotete. I motsetting til feilsøkinga då dei løyste oppgåver ved hjelp av programmering. Her gjennomførte dei feilsøkinga systematisk og testa for fleire ulike variablar. Dette er illustrert i to dømer under. Det første døme så feilsøkte elevane oppgåve 3 på programmeringsoppgåvesettet. Her har dei laga ein kode og skal no prøve om den fungere.
Døme på programmering:

Gut 2: Skal vi prøve, sjå om den funkar?
(Prøver ut koden med talet 14)
Gut 1: Den går ikkje.
Gut 2: Ikkje så bra.
Elevane bytter ut ei blokk og snur ulikhets teiknet slik at den delen av programmet fungere.
Gut 2: Sånn der funka da. Kor mange timer jobber han i uka? Så seier eg 10 timer i uka kor gammal er han? 14 skriver eg. Då får eg opp, 450.
Gut 1: Koffor sa den ikkje navnet?
Elevane har ikkje laga ein blokk for å gi ut variabelen namnet, berre for å lagre variabelen namnet.
Gut 2: Vent den seier at svaret er 0?
Elevane har ikkje lagra variabelen alder.
Gut 1: Skal den ikkje sei namnet ditt då? Berre ta sei navn og tar bort dei pluss greinene(blokka) og ein sei alder og så tar du sei svar.
Gut 1: Du må ha en sei i blokk.
Gut 2: Skal vi prøve med det same? 14.

I dømet kjem det tydeleg fram at elevane arbeider med eit mål og dei registrerer feil då det kjem opp. Her identifisere dei to feil, der den første er at namnet kjem i feil rekkefølge og at programmet ikkje registrerte alderen. Deretter gjer dei ei prioritering over kva som er lett å rette. Her retter dei først namnet og deretter alderen.

Døme på oppgåveløysing på papir ligg under. Her diskutere dei ulike framgangsmåtar for svaret dei har kome fram til. Det er ikkje nokre tydlege rekkefølge eller plan på feilsøkinga.

Gut 1: Tenk som ein programleder og så du skrev med at han gir bort visst du har meir bananar og så har ikkje gulerota noko med da å gjøre.

Jente: Kunne ha gitt bort bananar men i dette her tilfelle l i utgangspunktet så veit ikkje eg kan.
Jente: Men skal me berre skrive at han ikkje gir den bort.
Gut 2: Han ikkje vil gi bort bananen sin.
Gut 2: Ikkje bananen.

I utdraget gjettar elevane på tilfeldige svar. Dei arbeider ikkje med ein plan eller system. Det er ganske stort skilje mellom denne tilnærminga frå då dei arbeide med programmering og då dei arbeide med oppgåver på papir.

Likskapar og forskjellar innan Gruppe 1 då dei gjennomførte ei gjenbrukende tilnærming

Den gjenbrukende tilnærminga er vert observert i ulik grad av elevane under observasjonane. Her kom den gjenbrukane tilnærminga tydeleg fram under programmeringssituasjonen, men meir tvilande under oppgåveløysinga på papir. Her var det hovudsakeleg eit tilfelle der dei refererte til oppgåver med lik struktur. Som døme viser under gjenkjenner dei då oppgåvene liknar og dei klarer å gjenbruka programmet dei allereie hadde skreve. Elevane i Gruppe 1 har her gjennomført oppgåve 1 i programmerings oppgåvesettet, og så kjenne att strukturen på den neste oppgåva. Her ser dei at dei kan bruke oppatt deler av oppgåva

Gut 2: (Leser opp oppgåve 2)
Gut 1: Litt som i sta?
Gut 2: Litt som i sta ja.
Gut 1: Så visst du har mindre enn 18 så seier den...

Likskap og skilnader innan Gruppe 1 då dei gjennomførte den abstrakte og modulariserande tilnærminga

Den abstrakte og modulariserande tilnærminga vert også brukt ein del i denne gruppa. Elevane brukte den ofte for å isolere deler av oppgåva og løyse den først delen av oppgåva, deretter arbeide med andre deler av oppgåva og sette oppgåva saman. Elevane brukte desse tilnærmingane i ulike oppgåver, men innafor oppgåveløysing i ein programmeringskontekst var denne tilnærminga mest brukt ved oppgåve nummer 3. Dette var den mest komplekse oppgåva som hadde flest deler elevane skulle sette saman. Her brukte elevane tid på diskutere dei ulike delane og kva dei trengte i dei ulike delane av oppgåva. I døme under diskuterer elevane om dei treng den eine variabelen, i dette tilfellet var variabelen alder.

Gut 1: Først må den spør om alderen. Kor gammal er du?
Gut 2: Kvifor har du «if» blokk der då? Du skal jo berre spør om alderen.
Gut 1: Først må me spør kor gammal dei er før me finner ut kor mykje dei tener.
Gut 2: Å ja visst du er over 20 så, okei du kan starte med kor gammal er di og så, nei. Altså visst det var over 20 så gang her du berre med 45 eller 55.
Gut 2: Nei no skjønte eg ikkje...
Gut 1: Først. Spør om du kan han heiter, er. Han skal berre huske på navet. Skal me få da til?
Gut 2: Sikkert den her då?

Gut 1: Så setter me namn til svaret. Så neste tingen blir alderen då?
Gut 2: Spør først om kor gamle dei er, og så må me ha sånn «if».

I dette tilfellet deler elevane opp oppgåvene og veljar å starte med alderen. Deretter diskuterer dei litt om denne variabelen. På slutten av dette utdraget trekker gut 1 fram eit nytt element, men vert minna på om at dei først skal gjere variabelen alder. Denne diskusjonen ved element var nesten eksklusiv til programmeringskonteksten. Då elevane i denne gruppa brukte ein abstrakt og modulariserande tilnærming ved oppgåveløysing på papir, så var det på oppgåver som ikkje var samansette eller består av fleire deler. Dei oppgåvene bestod av to trinn og her brukte dei tilnærminga til å undersøke trinna kvar for seg.

4.2.2 Gruppe 2

Denne gruppa brukte dei den algoritmiske tilnærminga meir enn den andre gruppa. Dei vert meir aktivt brukt i begge observasjonane, men i likskap til Gruppe 1 er det tydelege forskjellar mellom korleis elevane brukte tilnærmingane då dei løyste oppgåver på papir samanlikna med då dei programmerte. Ei av dei tydelege forskjellane mellom gruppene var då denne gruppa satt fast på ei oppgåve, brukte dei feilsøking for å diskutere dei ulike moglegheitene. Ei anna tydeleg forskjell mellom gruppene var at Gruppe 2 hadde uttalt meir aktiv bruk av gjenbrukende tilnærming ved å identifisere element som kunne brukast oppatt. Her brukte elevane oppatt deler av tidlegare oppgåver i problemstillingane og nye oppgåver, slik at dei kunne bygge på tidlegare erfaringar.

Likskap og skilnader innan Gruppe 2 då dei gjennomførte ein adaptiv og iterativ tilnærming

Elevane på Gruppe 2 arbeide aktivt med denne tilnærminga under både oppgåveløysing på papir og ved programmeringsoppgåvene. Det som skilte seg ut frå den andre gruppa, illustrerte at elevane i denne gruppa laga planer for nesten alle oppgåvene då dei løyste oppgåvene på pair. Dette gjorde at dei kunne gjennomføra ei feilsøking på løysingsmetodane sine, og adaptere oppgåveløysinga si ut i frå svaret frå feilsøkinga. Då eg kategoriserte og analyserte denne observasjonen blei hendingar då elevane diskuterte kva element og kriteria som måtte til for å løyse oppgåva kategorisert som adaptiv og iterativ tilnærming. Elevane trakk fram desse kriteria på ulike måtar, enten dei ramsa dei opp, gjentok dei eller i likskap med Gruppe 1 gjentok oppgåveteksten, men med sine eigne ord og med fokus på kriteria. Måten elevane kommuniserte dei ulike kriteria for å løyse oppgåva var ofte avhengig av om dei løyste oppgåva på papir eller ved hjelp av programmering. Då dei lista opp kriteria for ei oppgåve på papir var det ofte meir tilfeldig og oppramsing av alle kriteria til oppgåva, ikkje berre dei nødvendige. Medan då dei fann fram kriteria for programmering var det meir strukturert og kritisk til kva dei trengte. Det første døme er frå oppgåveløysing på papir der eleven ramser opp dei spesifikke kriteria for oppgåva. Oppgåva går ut på om Helle (ho) har nokk eple til å lage eplekake.

Elev 2: Tre eller fleire så kan ho lage eplekake?

Her har eleven identifisert vilkåra og kommuniserer dei vidare til resten av gruppa. Det neste utsegna er døme på då eleven gjenfortel oppgåva. Her har han først lest opp ordrett frå oppgåva og deretter før eleven samanfattar informasjonen til dei andre

elevane. Dette var også noko Gruppe 1 gjorde ein del og desse gjenfortellingane liknar på kvarandre.

> *Elev 1: Ein gruppe elever får utdelt ein vekt og tre kuler. Ein blå, ein grøn, og ein rød. To av dei er like tunge, mens ein er lettare enn dei to andre. Då må dei vege dei... 2 eller 1 gangar. Kan legge to på om gangen.*

Her leser eleven opp oppgåva samtidig som han begynner å prosessere kva som må til for å løyse oppgåva. Deretter fortel han dette ut til gruppa. Etter ei stund må eleven endre si plan for korleis oppgåva kan løysast. Her har han inngått i ein adaptiv prosess der han endrar si originale plan for å tilpasse seg til oppgåva og få rett løysing.

> *Elev 1: Men høyr så du legger på to kuler, hvis den eine er lettare enn den andre så veit du kva som er den lette, hvis den andre er tynger enn den eine så veit du kva som er lettare, og hvis du legge to på og dei er like tunge så veit du at den som ligg igjen er lett. Då finner du uansett kva som er lettare på ein veging.*

Då Gruppe 2 skulle programmere var dei veldig tydlege på kva dei ville gjere. Dette kan komme av at dei var tre elever på gruppa eller at dei ønska å kommunisera det både til kvarandre og til meg sidan eg var til stades under observasjonen. Her var det meir tydelege dømer som illustrerer den adaptive og iterative tilnærminga slik Brennan og Resnick. I utdraget under diskutere elevane kva dei treng å gjere for å løyse oppgåve 3.

> *Elev 2: Her må vi bruke variablar som alder og som navn, men å ta startar først, «evens so when then it is clicked» og så meg ha «ask», som er sikkert på «sensing». Nei, ende til kor gammal er du.*
> *Elev 1: Og så må jeg ha «if», ligg på kontrollen, og så ein variabel... Eg er med på det må bruke denne etter på. (vel variabel namnet, diskutere om dei treng det minimalt.)*

Her diskutere dei kva dei treng for å løyse oppgåva. Dei vel i slik som Gruppe 1 å bruke Scratch sine blokker til å kommunisere dei ulike konsepta dei treng. Etter å ha identifisert og fått oversikt over kva funksjoner som var nødvendige for å løysa oppgåva, gjekk dei direkte over til gjennomføringa.

Likskap og skilnader innan Gruppe 2 då dei gjennomførte ein testande og feilsøkende tilnærming

Den testande og feilsøkende tilnærminga dominerte denne gruppa si oppgåveløysing både på papir og ved hjelp av programmering. Dei brukte denne tilnærminga aktivt då dei skulle finne svar som dei var usikker på var korrekte då dei løyste oppgåver på papir, og då dei skulle sikre om det var rett program dei hadde laga for å løyse oppgåva.
Då elevane arbeidde med oppgåveløysinga i programmeringskonteksten, var dei veldige systematiske i korleis dei gjennomførte feilsøkinga og testinga av funksjonane i programmet dei hadde laga. I dømet under arbeidde dei seg igjennom ein feil, som dei hadde identifisert i oppgåve 1. Her brukte dei ulike testar til å identifisere kor feilen låg og deretter undersøkte dei koden som høyrte til den delen av programmet.

> *Gut 1: Jo me er ferdig.*

Gut 3: Skal me prøv den då?
Gut 1: Korleis prøve me den?
Gut 2: Du skriv 12 ja, men gå på skolen. Og så prøver me på nytt, så skriv over «meir enn 19» kanskje 20 eller 21 går på skulen.
Gut 1: Me rota da til. (Elevane fekk feil svar med talet 20)
Gut 2: Større enn 19 går på jobb. (Elevane refererer til oppgåva)
Gut 3: Treng vi dei greiene der? Eller kan vi berre ta sei gå på jobb.
Gut 1: Me har ikkje blokka if / then, me kan berre sei else (blokka).
Gut 3: Han går jo berre på jobb uansett då?
Gut 2: If alder, eg veit ikkje hvis variabelen er større eller mindre fordi han seier «how old are you?». Så if alder så program veit ikkje kva alder er !! Så me treng å fortele eller lære den kva alder er først...
Gut 1: Korleis gjer me da? (Elevane får instruksjon på kva ulike blokker som kan fungere for dette problemet).
Gut 2: Me treng ikkje å ha alder då, me kan berre ha nummer. Hvis den ikkje veit kva det er, kvar er grunne for å bruke det?

Då elevane gjennomførte feilsøking og testing av sine oppgåveløysing på papir gjekk Gruppe 2 meir strategisk til verks enn Gruppe 1. Det var spesielt i oppgåve 6 der dei skulle finne det kontrapositive utsegna for eit utsegn. Her hadde Elev 2 ein del problemet med å avgjere kva som var rett svar og løyste dette ved å ramse opp dei ulike kombinasjonane som han kunne finne. Tilfeldig gjetting er ein av strategiane som Brennan og Resnick(2012) viste til når dei undersøkte denne tilnærminga og elevane i Gruppe 2 brukte denne strategien etterkvart i denne oppgåva.

Elev 2: Hør hvis X ikkje er mindre enn 1000 då er det ikkje mindre enn 100. då er det over 1000 det er rart. Eller hvis x ikkje er mindre enn 1000 så er ikkje mindre enn 100? hvis x er mindre enn 100 så er x ikkje mindre enn 1000
...
Elev 2: Ja fordi det er 100, det går visst du sette det opp, nei vent det er tre måter du kan sette det opp på, nei fire måter å sette dei opp på og to av dei er feil.
...
Elev1: Skal me berre gjette på ein?
Elev 2: Berre ta ein tilfeldig ein. Skulle hatt fleire dømer.

Likskap og skilnader innan Gruppe 2 då dei gjennomførte ei gjenbrukende tilnærming

Elevane i Gruppe 2 identifiserte fleire stader der dei kunne bruke denne tilnærminga enn Gruppe 1. Dette var spesielt tydeleg i oppgåveløysing på papir. Då arbeidde elevane med å samanlikne med oppgåver døme og anerkjenner løysingsmetoden som kan brukast oppatt i den neste oppgåva. I oppgåveløysinga ved hjelp av programmering var det færre tilfelle av denne tilnærminga. hovudsakeleg påpeikte dei gjenbruken av løysingsmetode frå oppgåve 1 til oppgåve 2 ved programmering.

Utdraget under er frå oppgåveløysinga ved hjelp av programmering, der elevane arbeida med oppgåve 2. Denne oppgåva hadde ein liknader oppsett til oppgåve 1 og elevane kjente igjen dette. Deretter identifiserte dei kva som er likt frå den førre oppgåva og kva

som måte endrast. Her viser elevane at dei kan bruke denne tilnærminga ved å bruke oppatt deler av blokker og identifiserer kva dei måtte endre for at desse funksjonsblokkene skulle kunne fungere i det nye programmet. Elevane diskuterer her kva som må endrast for å kunne løyse oppgåva og dette gjer at dei slepp å skrive opp att programmet, men kan bruke det dei har alt har skrive i førre oppgåve. Idet dei identifiserte koden som kunne bukast oppatt og veit kor dei skal bruke det på ein hensiktsmessig måte, så viser det at dei har utvikla den gjenbrukende tilnærminga til oppgåveløysing.

> *Gut 1: Da er det same som den forgje men me må berre erstatte teksten.*
> *Gut 2: Og tala er ikkje 19 no da er 18, da er jo akkurat da same men endre litt på variablane.*

Då elevane i Gruppe 2 gjennomførte oppgåver på papir ytra dei tydlegare om oppgåver som likna kvarandre og moglegheita for å bruke oppatt strategiar. Det første utdraget under fortel elevane om at oppgåva er lik som førre oppgåve, og dette gjer at dei kan gjenbruka løysingsmetoden frå førre oppgåve.

> *Elev 3: Det er det same som oppgåve 1*

Det andre døme eg ønsker å trekke fram frå elevane sin oppgåveløysing på papir er då dei støtta seg til gjenbruk for å finne løysingen til eit problem. Elevane brukte lang tid på oppgåve 6 og utrykte at den var utfordrande fordi oppgåva var ukjent og tematikken var noko dei ikkje hadde arbeida med før. Etter ein del diskusjon av korleis dei skulle løyse oppgåva, vende elev tilbake til oppgåveteksten og dømet som vet gitt der. Han kjem fram til at dei skal finne det motsette. Denne kunnskapen bruker deretter elev 2 til å prøve å sjå korleis dømet vert satt opp, for å finne samanhengar mellom rekkefølga på utsegna og då «ikkje» vert brukt.

> *Elev 1: (eleven leser opp dømet) Figuren er eit kvadrat og figuren er ikkje eit kvadrat.*
> *...*
> *Elev 1: Me skal ha det motsette.*
> *Elev 2: Som det er vi flytter litt mellom tusen og har med ikkje heller... Ja, men her har dei bytta, her står det kvadratet etter, ikkje først og her står kvadratet først. (Her diskutera elevane feil løysing på ei oppgåve, men dei brukar dømet aktivt for å atterskape riktig løysing)*

Likskap og skilnader innan Gruppe 2 ved gjennomføring av ei abstrakt og modulariserande tilnærming

Den abstrakte og modulariserande tilnærminga vert mindre brukt i Gruppe 2 enn i Gruppe 1 ved programmeringsarbeid, men meir ved oppgåveløysing på papir. Elevane var ganske komfortable med programmeringsaktiviteten, og dette kan ha påverka deira behov for å dele opp oppgåvene. Oftast valte elevane først å lage ein plan og deretter utføre planen stegvis. I oppgåve 3 som var den største oppgåva var det tydeleg at dei hadde laga ein plan og detter delte dei oppgåva opp, slik at dei kunne utføre kvar del av oppgåve etter kvarandre. Denne planen var tydeleg fordi elevane diskutere den før dei

begynte å plassere blokkene i programmet. I dømet under så lister opp eleven opp kva som er nødvendig for den delen av oppgåva dei arbeider med.

> *Elev 2: Ja og så tar me if og så må me inn på «sensing», men me treng ikkje å bry oss om namnet før om heilt til slutten. Og då må eg ha sett der og if og sett «if svar er meir enn 20».*

Då Gruppe 2 arbeidde med oppgåveløysing på papir, nytta dei liknande metode som Gruppe 1. Her delte dei opp dei enkelte oppgåver vert einige om dei delane. Dette gjorde at dei hadde mindre informasjon å handtere og dei kunne dermed fokusere på den delen dei arbeidde med. Ved å dele opp problemet kunne dei løyse mindre problem og deretter sette dei samen til ein løysing for det store problemet. I dette utdraget har dei to utsegn som kan vera logisk knyta samen. Elevane skal undersøke om utsegn A er avhengig av utsegn B og om utsegn B er avhengig av utsegn A. I dømet under finn først elev 1 ut at utgang A ikkje er avhengig av utsegn B, deretter finn elev 2 at utsegn B ikkje er avhengig av utsegn A.

> *Elev 1: Nei berre fordi Sol har fire bein betyr det ikkje at Sol er ein katt.*
> *Elev 2: Nei men Sol er ein katt, berre for det er ein katt så må dei ikkje ha fire bein.*
> *Elev 1: Her er ingen av veiene riktige.*

5 Drøfting

I drøftinga skal eg først utforske resultata frå analysen basert på mi problemstilling. Den fokuserer på eleven sine kjenneteikn på algoritmisk tenking i forhold til oppgåver basert på vilkår.

- Kva kjenneteiknar 7 elever sitt arbeid med oppgåver basert på vilkår?
- Kva spor av algoritmisk tenking kan ein sjå i elevane sitt arbeid i ein programmeringskontekst og ved oppgåveløysing på papir?

Deretter skal eg sjå på resultata mot kva programmering i skulen kan bety. Her trekk eg fram både kjerneelementa og grunnleggande ferdigheiter som er nemnd i læreplanen for matematikk(Kunnskapsdepartementet, 2019). Føremålet med dette kapittel er å sette resultatane i ein kontekst utanfor det valte rammeverket.

5.1 Kva kan eg sei om kjenneteikna etter denne analysen?

I analysen identifiserte eg alle dei algoritmiske tilnærmingane hos begge gruppene, både då dei løyste oppgåver på papir og då dei arbeide med programmering. Graden av tilnærminga (av dei ulike tilnærminga) varierte mellom gruppene og tilnærmingane presenterte seg svært ulikt mellom arbeida på papir og arbeida med programmering. Oppgåvene baserte seg på Brennan og Resnick(2012) sitt algoritmiske konsept vilkår og dette kan ha påverka omfanget av dei algoritmiske tilnærmingane eg observerte.

5.1.1 Ei adaptiv og iterativ tilnærming (Being incremental and iterative)

I intervjuet med ulike «scratchers» kom Brennan og Resnick(2012)fram til at laging av eit program og prosjekt er ein prosess.

> *«It is an adaptive process, one in which the plan might change in response to approaching a solution in small steps. In conversations with Scratchers, they described iterative cycles of imagining and building – developing a little bit, then trying it out, and then developing further, based on their experiences and new ideas.»(Brennan & Resnick, 2012, s. 7)*

Ein slik prosess kan innehalde planlegging og at planane endrar seg undervegs for både å passe inn med feilsøkinga og nye idear. I analysekapittelet vert slike planer observert, der den adaptive prosessen ofte skjer i samband med andre tilnærmingar som testande og feilsøkjande tilnærmingar. Elevane i Gruppe 2 prøvde ofte ut ein plan og deretter endra den ut i frå kva utfall dei fekk då dei arbeide med oppgåver på papir. I den siste oppgåva var det fleire element dei skulle ta omsyn til og dette førte til ein del utforsking hos elevane. Elev 1 kom fram til denne planen, og etter litt utforsking blei den endra.

> *Elev 1: Ei gruppe elever får utdelt ei vekt og tre kuler. Ein blå, ein grøn, og ein raud. To av dei er like tunge, medan ein er lettare enn dei to andre. Då må dei vege dei 2 eller 1(gonger)... Ein kan legge to på om gongen.*

Elev 1: Men høyr, du legg på to kuler, om den eine er lettare enn den andre så veit du kva som er den lette. Om den andre er tyngre enn den eine så veit du kva som er lettast, og om du legg to på og dei er like tunge så veit du at den som ligg igjen er lettast. Då finn du uansett ut kva som er lettast på ei veging.

Ei slik utvikling av planar kan vera eit kjenneteikn ein kan møte på hos elevar i fleire situasjonar. Dersom elevane arbeider med strategiar som inneheld feilsøking, eller utforsking og planlegging kan denne kombinasjonen bidra til ein slik adaptiv og iterativ prosess som Brennan og Resnick(2012) skildrar.

I analysen kom det mykje betre fram når elevane brukte dei algoritmiske tilnærmingane då dei gjennomførte oppgåver i ein progammeringskontekst. For den adaptive og iterative tilnærminga, var det tydelege kjenneteikn på at tilnærminga var ein prosess som varte gjennom heile oppgåva. Dette samsvarer igjen med Brennan og Resnick(2012) si skildring. Det var fleire individuelle hendingar som viste til denne prosessen, over fleire periodar i oppgåveløysinga og ofte i samband med andre tilnærmingar, og då spesielt den feilsøkjande og testande tilnærminga. Dette kjem nok av at eleven skulle programmere ut i frå oppgåver som ikkje var for opne. Det gjorde at deira strategiar ofte blei retta mot å tilpasse sine program mot det oppgåva sa dei skulle gjere. Her samanlikna elevane svara deira opp mot oppgåveteksten for å sjekke sin eigen plan, deretter retta dei på den. Dersom elevane skulle ha arbeide meir med friare oppgåver, kan det tenkast at denne prosessen kunne ha omfamna andre tilnærmingar.

5.1.2 Ein testande og feilsøkjande tilnærming (Testing and debugging)

Denne tilnærminga hadde svært tydelege kjenneteikn som gjorde det naturleg å kategorisere den. Under observasjonen der elevar arbeide med programmering, tok alle aktive val for å undersøke deira program og for å teste dei. I Brennan og Resnick (2012) si forsking på unge "Scratchers" fann dei at denne tilnærminga hadde utvikla seg over tid:

"In interviews, Scratchers described their various testing and debugging practices, which were developed through trial and error, transfer from other activities, or support from knowledgeable others(Brennan & Resnick, 2012, s. 8)."

I observasjonen og samtaler med lærarane fekk eg inntrykk av at dette stemte også for elevane som var med på mi undersøking då dei løyste oppgåver ved hjelp av programmering. Her arbeide dei ofte igjennom ein rekke testar for å sjekke ulike delar av koden. Eit slikt testregime er noko som kan kjennast igjen som eit av kjenneteikna for den algoritmiske tilnærminga og tenkinga. Dette arbeidet for å bygge opp ei vane for testing og feilsøking er noko læraren aktivt kan legge til rette for i undervisinga. Noko av denne typen aktivitet som utforsking gjennom testing og feilsøking kjem naturleg for elevane og kan kjennast igjen relativt fort. Til dømes viste gruppe to dette under oppgåveløysinga på papir, der dei fann dei ut at dei kunne teste ut tilfeldige rekkefølger.

Elev 2: Hør viss X ikkje er mindre enn 1000 då er det ikkje mindre enn 100. Då er det over 1000 det er så rart. Eller viss X ikkje er mindre enn 1000 så er ikkje mindre enn 100? Om X er mindre enn 100 så er X ikkje mindre enn 1000
...

Elev 2: Ja fordi det er 100, det går om du set det opp, nei vent det er tre måtar du kan sette det opp på, nei fire måtar å sette dei opp på og to av dei er feil.
...
Elev1: Skal me berre gjette på ein?
Elev 2: Berre ta ein tilfeldig ein.

I siste delen av elevane sin kommunikasjon over byrjar dei å trekke parallellar mellom gjenbruk og feilsøking. Dette kan vera bygginga på ein strategi. Brennan og Resnick (2012) skilder ein rekke ulike strategiar som kan brukast til feilsøking og testing:

1. Identify (the source of) the problem
2. Read through your scripts
3. Experiment with scripts
4. Try writing scripts again
5. Find example scripts that work
6. Tell/ask someone else about the problem
7. Take a break

Dei strategiane elevane brukar er å identifisere problemet, lese igjennom, utprøving av koder og snakke med andre. Nokre av strategiane var ikkje mogeleg å sjå under observasjonen, det kjem nok av rammeverket og situasjonen elevane var i. Døme på dette er ta ein pause og i nokre tilfella finne dømer frå liknande problem. I observasjonen kan me kjenne igjen strategiar som å «finne dømer som fungerer» og «eksperimenter med programmet», som er strategiar elevane over byrjar å danne seg.

Desse strategiane var som nemnd tidlegare mykje meir innøvd då elevane arbeide med programmering. Det kan komme av fleire ulike grunner. Ein av dei er at strategiane som Brennan og Resnick(2012)skilder er basert på elevar sitt arbeid med Scratch og dermed ikkje er så enkelt overførbart mellom programmering og oppgåveløysing på papir (2012). Dette gjer gjenkjenninga av desse strategiane meir utydelege. Det eg opplevde under analysen var at tilnærminga til algoritmisk tenking var tydeleg til stades, men den kom best fram under programmeringssituasjonen. Elevane brukte feilsøking under begge observasjonane, men det var meir tydeleg under programmeringa.

5.1.3 Ei gjenbrukende tilnærming (Reusing and remixing)

Å bruke oppatt andre sin kode er ein vanleg praksis innafor programmering som Brennan og Resnick(2012)fortel om under. Denne praksisen med gjenbruking av kode lever på at utviklaren har ein kultur for å dele ulike løysingar og algoritmar for fremma utviklinga innafor teknologi og programmering. Det er vanlig innafor forsking ved programmering å dele både resultat og program slik at fleire kan utforske dei.

Building on other people's work has been a longstanding practice in programming and has only been amplified by network technologies that provide access to a wide range of other people's work to reuse and remix. …. Reusing and remixing support the development of critical code-reading capacities and provoke important questions about ownership and authorship (Brennan & Resnick, 2012, s.7).

Denne kulturen tenar på at ulike prosjekt er Open source. Open source viser til ein rekke verdiar som Perens et al. kallar «the open source way». Denne tenkinga omfattar produktet, ulike prosjekter eller ideen. Her blir verdiar som utveksling av idear, samarbeid og deltaking nemnd under.

> *«The term originated in the context of software development to designate a specific approach to creating computer programs. Today, however, "open source" designates a broader set of values—what we call "the open source way." Open source projects, products, or initiatives embrace and celebrate principles of open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community-oriented development(Perens et al., 1998).»*

Dersom læraren etabler ein delings kultur eller «open source way» kan det hjelpe elevane til å utvikle både deira algoritmiske tilnærming og samtidig bli meir vane med deling.

Sidan denne praksisen er tydeleg etablert hos ulike yrkesgrupper, kan gjenkjenninga av dette kjenneteiknet vera tydeleg. Ut i frå analysen kom elevane sin bruk av den gjenbrukende tilnærminga tydeleg fram både under programmeringa og oppgåveløysinga på papir. Her refererte dei ofte til anten at oppgåvene var like, eller at dei kunne gjenbruka deler av programmet dei hadde skrive før. Under observasjonen brukte ingen av elevane nokon program frå andre kjelder enn si gruppe. Dette ser eg på som eit vidare steg som er viktig for elevane si utviking av dette kjenneteiknet.

> *Gut 1: Litt som i stad?*
> *Gut 2: Litt som i stad ja.*
> *Elevane viser til deler av oppgåva som kan brukast oppatt i den neste oppgåva.*
> *……*
> *Gut 1: Går det ann å kopiere den eller noko sånt?(Elevane viser til ein del av koden som kan gjenbrukas i ein annan del av programmet)*
> *Gut 1: Ja, både den og den.*
> *Gut 2: Tar den og berre kopiere den då.*
> *Gut 1: Sånn ja, satt du den tilbake. Du treng å gjere det ein gang til fordi me skal jo bruke den.*
> *Gut 1: Navn og den der. (Elevane bruker igjen den blokka der Sprite spør om alder og berre endra på spørsmålet*

I dømet over viser det at eleven aktivt nyttar både kommunikasjon og samarbeid som Perens et al. (1998) trakk fram som gode kvalitetar frå «the open soures way». Dei fell også innafor kategorien til Brennan og Resnick (2012) ved gjenkjenning og bruk av det. Ved å kombinere desse to konsepta kan ein fort kjenne att elevane sine strategiar då det gjelder gjenbruk. Mykje av både den gjenbrukende tilnærminga og «open source way» har glidande overgangar til andre algoritmiske tilnærmingar. Dette kan tyde på at dei ofte er knyta saman og kan vera eit kjenneteikn på algoritmisk tenking.

5.1.4 Ei abstrakt og modulariserande tilnærming (Abstracting and modularizing)

Denne tilnærmingane var litt vanskeleg å kjenne igjen, det kan komme frå oppgåvelaget og kor komplekse dei er. I følge Brennan og Resnick(2012)kjem denne tilnærminga meir til syne nå vi skal bygge noko større.

> *Abstracting and modularizing, which we characterize as building something large by putting together collections of smaller parts, is an important practice for all design and problemsolving (Brennan & Resnick, 2012, s. 8).*

Ved å dele opp større prosjekt har elever moglegheit til å takle mindre deler om gongen. Då kan eit stort prosjekt bli meir overkommeleg. Brennan og Resnick (2012) seier at dette ikkje berre er viktig i programmering men òg i problemløysing. Kanoknitanunt et al. presiserer at problemløysing er for alle nivå, noko som gjer at abstrahering og modulariserande tilnærming også er ei tilnærming alle nivå kan arbeide med (Kanoknitanunt et al., 2021b).

Under analysen observerte eg at det var oppgåveløysing som gav mest moglegheiter til denne tilnærminga. Her kunne me tydeleg sjå på større oppgåver (oppgåve 3) at elevane delte opp oppgåva, undersøkte og løyste dei ulike delane. Dette spelte også inn i feilsøkinga der dei arbeide med å feilsøke dei ulike delane. Med det meina eg at dei la opp testar for å teste ulike deler og prøvde å bruke ulike kombinasjonar av desse delane.

I utdraget under ser me tydeleg at elevane deler opp oppgåva og legg opp til ei prioritering av oppgåver dei må gjennomføre. Det ser me ved at elevane påpeiker at dei skal først finne alderen og deretter endre prioriteten til å finne variabelen namn og deretter alderen. Dette er vera eit tydeleg kjenneteikn på denne tilnærminga:

> *Gut 1: Først må den spør om alderen. Kor gammal er du?*
> *Gut 2: Kvifor har du «if» blokk der då? Du skal jo berre spør om alderen.*
> *Gut 1: Først må me spør kor gammal dei er før me finner ut kor mykje dei tener.*
> *Gut 2: Å ja visst du er over 20 så, okei du kan starte med kor gammal er di og så, nei. Altså visst det var over 20 så gang her du berre med 45 eller 55.*
> *Gut 2: Nei no skjønte eg ikkje...*
> *Gut 1: Først. Spør om du kan han heiter, er. Han skal berre huske på navet. Skal me få da til?*
> *Gut 2: Sikkert den her då?*
> *Gut 1: Så setter me namn til svaret. Så neste tingen blir alderen då?*
> *Gut 2: Spør først om kor gamle dei er, og så må me ha sånn «if».*

5.2 Kva er forskjell mellom algoritmisk tenking på papir og ved programmering?

Wing (2006) viser til i si skildring av algoritmisk tenking, at det kan utførast både av eit menneske eller av ein maskin

> *«Algoritmisk tenking bygger på kraften og begrensningene til dataprosesser, enten de utføres av et menneske eller av en maskin» (Wing 2006, s 33).*

Den vanlege tolkinga av dette blir å forstå det som eit samarbeid mellom maskin og menneske (Bocconi et al., 2018; Brennan & Resnick, 2012; Cui & Ng, 2021; Tran, 2019). Eg ønskte å undersøke dette litt nærmaste og sjå kva spor av algoritmisk tenking eg kunne observere når elevane løyste oppgåver på papir og deretter samanlikne det mot oppgåver løyst ved programmering. Desse elevane arbeide med likande oppgåver som baserte seg på Brennan og Resnick(2012) sine algoritmiske konseptvilkår, og vert kategorisert ut i frå dei algoritmiske tilnærmingane. Under analysen såg eg ein tydeleg forskjell mellom tilnærming til oppgåveløysing på papir eller ved hjelp av programmering.

5.2.1 Ei adaptiv og iterativ tilnærming (Being incremental and iterative)

Den adaptive og iterative tilnærminga presenterte seg ulikt både mellom gruppene og mellom oppgåveløysinga på papir og ved programmering. Dei to ulike gruppene brukte denne tilnærminga ulik då dei skulle løyse oppgåver på papir, men hadde ein meir lik tilnærming då det kom til programmering. Dette kom til syne under analysen der dei delte kjenneteikn då det kom til denne tilnærminga, som å endre planen etter feilsøking og fokusere på variablar og prioriteringa av desse.

Det var få tilfelle der eg observerte denne tilnærminga hos Gruppe 1 under oppgåveløysinga på papir. Elevane brukte ofte tid i byrjinga av oppgåveløysinga både på papir og i ein programmeringskontekst til å formulere ein plan. Desse planane var berre i få tilfelle adaptert til å handtere utfordringar som ofte vert oppdaga i samband med feilsøking.

Gruppe 2 hadde ei liknande tilnærming til den adaptive og iterative tilnærminga då dei arbeide med programmering. Dei presenterte same aktivitet då ny informasjon kom fram med feilsøkinga. Dette er ein av grunnane til at ein må endre planen sin i følgje Brennan og Resnick(2012). Dei arbeide meir adaptivt under oppgåveløysinga med papir. Her var det tydeleg at dei endra planen etter noko diskusjon og meir analyse av dømet.

5.2.2 Ei testande og feilsøkjande tilnærming (Testing and debugging)

Denne tilnærminga var nok den mest interessante å observere, sidan den hadde mange ulike måtar å presentere seg på. I artikkelen nemner Brennan og Resnick (2012) 6 ulike strategiar for feilsøking. Dei nemner at desse strategiane vert utvikla gjennom prøving og feiling, overføring frå andre aktivitetar, eller støtte frå kunnskapsrike andre. Då elevane arbeide med problemløysing på papir, var det oftare desse tre framgangsmåtane presenterte seg.

Gruppe 1 demonstrerte ein framgangsmåte der dei overførte kunnskap frå andre aktivitetar til oppgåva dei ønskte å løyse på papir. Då elevane arbeide med feilsøking var det mindre utvikla strategiar enn då dei arbeide med programmering. Under programmeringa var det tydeleg at dei har ein plan på kva dei ønsker å feilsøke, medan i oppgåveløysinga på papir var det meir prøving og feiling utan noko plan. Det kan tenke seg at dette viser ein progresjon i den algoritmiske tenkinga hos elevane, at den er meir passande ved programmering og dermed kunne dei vidareføre framgangsmåtane til strategiane Brennan og Resnick (2012) nemnde.

Gruppe 2 arbeide meir systematisk med feilsøking og den testande tilnærminga då dei arbeide med problemløysing, både på papir og ved programmering. Her hadde me tilfelle der dei søkte opp informasjon og refererte til dømet ved problemløysinga på papir. Dei handlingane kan kategoriserast som strategiar som Brennan og Resnick (2012) viste til.

5.2.3 Ei gjenbrukende tilnærming (Reusing and remixing)

Funna i analysen tyder på at det var minst forskjellar mellom problemløysing på papir og ved hjelp av programmering med denne tilnærminga. Her var det tydeleg då elevane valde å gjenbruka deler av andre oppgåver, og andre deler av den same oppgåva. Ein av dei viktigaste evnene ved denne tilnærminga, er evna til å kjenne igjen kva du kan gjenbruka og korleis det kan passe inn (Brennan & Resnick, 2012). Elevane ytra tydeleg då dei kjente igjen deler av oppgåver som kunne brukast oppatt og gjorde det.

5.2.4 Ei abstrakt og modulariserande tilnærming (Abstracting and modularizing)

Abstrakt og modulerande tilnærming er ifølge Brennan og Resnick (2012)viktig i større prosjekter. Dette er nok grunnen til berre ein delvis observasjon av denne tilnærminga hos gruppene. Dette kan kome av at oppgåvene ikkje var komplekse nok, eller at dei ikkje hadde lenger tid til å arbeide med oppgåvene. Når det er sagt var det eit par observasjonar av modulerande tilnærming både ved oppgåveløysing på papir og ved programmering. Her var det tydeleg at den største oppgåva som hadde fleire steg og var meir samansett kravde fleire blokker i programmeringa. Dette kan tyde på at elevane trengte oppgåver som var meir utfordrande for å føle at dei trengte å bruke denne tilnærminga. I analysen under delkapittel 3.1.4 kom det også fram at då elevane arbeide med oppgåver med fleire faktorar så modulerte dei ofte denne oppgåva. Dette kan peike på ein likskap mellom gruppene samt arbeidet med oppgåver på papir og oppgåver i ein programmeringskontekst.

5.3 Grunnleggande ferdigheiter og algoritmisk tenking

I analysen såg eg at algoritmisk tenking kunne knytast til alle dei sentrale trekka ved dei grunnleggande ferdigheitene. Elevane demonstrerte desse ferdigheitene då dei argumenterte for kva som var rett då dei gjennomførte feilsøkinga eller illustrere matematiske ferdigheiter ved å løyse ulikskapane i oppgåvene. Wing definerer algoritmisk tenking som eit sett med problemløysningsmetodar som involverer å uttrykke problem og deira løysningar som en datamaskin også kan utføre. Det inneberer

automatisering av prosesser, men også bruk av databehandling for å utforske, analysere og forstå prosesser (naturlege og kunstige) (Wing, 2008).

5.3.1 Munnlege ferdigheiter

Dei munnlege ferdigheitene fokuserer på kommunikasjon og evne til å diskutere matematiske idear. Dette er ikkje direkte knyta til algoritmisk tenking sidan det er noko eit individ gjer (Wing, 2008), men dei gir ein rekke verktøy som kan bidra til utvikling av denne kommunikasjonen.

> *Muntlige ferdigheter i matematikk innebærer å skape mening gjennom å samtale i og om matematikk. Det vil si å kommunisere ideer og drøfte matematiske problemer, strategier og løsninger med andre. (Utdanningsdirektoratet, 2020, s.4)*

Elevane fekk nytta dei algoritmisk konsepta då dei skulle skildre kva oppgåva trengte (Brennan & Resnick, 2012). Dei algoritmiske konsepta gav elevane eit språk til å drøfte idea, strategiar og løysingar til oppgåver.

Elevane kommuniserte og nytta deira strategiar då dei løyste oppgåvene, anten for å formidle til andre elever kva som var planen, eller for å starte ein diskusjon for å vidareføre løysinga. Eit spesielt tilfelle var under feilsøkings tilnærminga der elevane samarbeida om kva som førte til ein feil.

> *Gut 2: Skal vi prøve, sjå om den funkar?*
> *(Prøver ut koden med talet 14)*
> *Gut 1: Den går ikkje.*
> *Gut 2: Ikkje så bra.*
> *Elevane bytter ut ei blokk og snur ulikhets teiknet slik at den delen av programmet fungere.*
> *Gut 2: Sånn der funka da. Kor mange timer jobber han i uka? Så seier eg 10 timer i uka kor gammal er han? 14 skriver eg. Då får eg opp, 450.*
> *Gut 1: Koffor sa den ikkje navnet?*
> *Elevane har ikkje laga ein blokk for å gi ut variabelen namnet, berre for å lagre variabelen namnet.*
> *Gut 2: Vent den seier at svaret er 0?*
> *Elevane har ikkje lagra variabelen alder.*
> *Gut 1: Skal den ikkje sei namnet ditt då? Berre ta sei navn og tar bort dei pluss greinene(blokka) og ein sei alder og så tar du sei svar.*
> *Gut 1: Du må ha en sei i blokk.*
> *Gut 2: Skal vi prøve med det same? 14.*

Her ser me tydleg at elevane arbeider med diskusjon av både idear og løysingar. Dei identifiserte ulike feil og kommuniserte det til kvarandre ved hjelp av den testande og feilsøkjande tilnærminga.

5.3.2 Å kunne skrive

Ved programmering og algoritmisk tenking må elevane arbeide med å omsetja oppgåva til algoritmar som maskiner forstår (Wing, 2008). Her kan dei arbeide med denne grunnleggande ferdigheita ved å legge til rette for det.

> *Å kunne skrive i matematikk innebærer å beskrive og forklare sammenhenger, oppdagelser og ideer ved hjelp av hensiktsmessige representasjoner. Å kunne skrive i matematikk er et redskap for å utvikle egne tanker og egen læring (Kunnskapsdepartementet, 2019).*

Ved at elevane får arbeida med å skildre og forklare samanhengar, oppdagingar og idear med tanke på programmering og algoritmiske konsept, kan ein skape ei større forståing og hjelpe dei med å utvika dei algoritmiske tilnærmingane. I analysen såg me mindre bruk av denne grunnleggande ferdigheten. Dette kan godt komme av at datamaterialet mitt var fokusert på elevane sin resonnering og tankegong, og dermed var det lite produksjon av tekst.

5.3.3 Å kunne lese

Å kunne lese er eit kjend element sentralt i den norske skulen. Trening på denne ferdigheita blir ofte meir og meir tydeleg opp gjennom trinna når elevane sine leseferdigheiter aukar. Kunnskapsdepartementet skildrar denne grunnleggande ferdigheiten som:

> *Å kunne lese i matematikk vil si å sortere informasjon, analysere og vurdere form og innhold og sammenfatte informasjon i sammensatte tekster. (Kunnskapsdepartementet, 2019, s. 5)*

For elevane er denne ferdigheita svært viktig under programmeringsoppgåvene, fordi dei skal tolke informasjonen dei får og omsetja dei til ein algoritme eller program maskinen forstår. Ofte er oppgåver i program ikkje skrive i programmeringsspråk eller ved blokker. Dei må derfor analysere og sortere informasjonen dei får, for å skrive eit program som fungerer. Gjennom analysen såg me ofte då denne ferdigheita vart brukt og øvd på. Dette kunne komme fram då dei leste opp oppgåver for å trekke ut informasjon, under gjenbrukingstilnærminga eller under feilsøking og testing. Her brukte elevane sine ferdigheiter for å lese og forstå det dei hadde produsert saman med oppgåva som sto framfor dei.

5.3.4 Å kunne rekne

Kunnskapsdepartementet set kjerneelement å kunne rekne inn i alle fag (Kunnskapsdepartementet, 2019). I matematikkfaget er dette ein naturleg del av faget, men kjerneelementet «å kunne rekne» omfatter meir og kan implementerast i fleire tema og skulefag. Kunnskapsdepartementet skildrar kjereelemeter slik:

> *Å kunne regne i matematikk vil si å bruke matematiske representasjoner, begreper og framgangsmåter til å gjøre utregninger og vurdere om løsninger er gyldige. (Kunnskapsdepartementet, 2019, s. 5)*

Algoritmisk tenking og programmering vert knyta til denne grunnleggande ferdigheita. Elevane skal kunne vurdere om løysingar er gyldige. Dette kan dei bruke den testande og feilsøkjande tilnærminga til. Analysen viste at denne tilnærminga også fungerte ved problemløysing på papir. Algoritmisk tenking ligg naturleg i den grunnleggande ferdigheita for å kunne rekne. Det kan vere nyttig å fokusere på det under opplæringa, ved å kjenne igjen kjenneteikna til denne type tenking.

5.3.5 Digitale ferdigheiter

Algoritmisk tenking er naturleg knyta til maskiner og dermed digitale verktøy (Wing, 2008). Dei digitale ferdigheitene er noko som blir meir og meir viktig, og er derfor gjort om til eit kjerneelement av Kunnskapsdepartementet. Dei skildrar digitale ferdigheiter som:

> *Digitale ferdigheter i matematikk innebærer å kunne bruke graftegner, regneark, CAS, dynamisk geometriprogram og programmering til å utforske og løse matematiske problemer. Videre innebærer det å finne, analysere, behandle og presentere informasjon ved hjelp av digitale verktøy. (Kunnskapsdepartementet, 2019, s. 5)*

Algoritmisk tenking skal bygge på algoritmar som både elever og maskiner kan bruke, og dette kan ein sjå igjen i analysen. Her arbeider elevane med programmeringsverktøy og bruker dei ulike tilnærmingane som verktøy for å lage dei ulike programma for å løyse oppgåvene. Analysen viste at elevane brukte algoritmisk tenking svært aktivt under aktivitetar saman med digitale verktøy. Her bidrog dei algoritmiske tilnærmingane til dei digitale verktøya for å løyse oppgåva. Eit slik samarbeid kan vera med på å auke eleven sine digitale ferdigheiter.

5.4 Kjerneelementa i matematikk faget.

Kjerneelementa i matematikkfaget går gjennom alle trinna, og skal følge eleven gjennom 10 år med skule (Hemnes, 2018). Ved å finne igjen kjenneteikna på algoritmisk tenking i elevane, kan me implementere algoritmisk tenking i desse kjerneelementa der dei passer inn. Alle kjerneelementa er komplekse og samansette, det gjer at algoritmisk tenking kan vera med å bidra til utforsking og læring innafor desse elementa.

5.4.1 Utforsking og problemløysing

Innafor kjerneelementa er utforsking og problemløysing svært nært knyta til mi problemstilling.

> *Utforsking i matematikk handler om at elevene leter etter mønstre, finner sammenhenger og diskuterer seg fram til en felles forståelse (Kunnskapsdepartementet, 2019, s. 5).*

Under dette kjerneelementet nemner dei korleis dei ønsker å bruke algoritmisk tenking. I utdraget under trekk dei fram at algoritmisk tenking kan utvikle strategiar for å løyse problemløysingsoppgåver.

> *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 (Kunnskapsdepartementet, 2019, s. 5)*

Kunnskapsdepartementet trekk her fram at algoritmisk tenking kan hjelpe elevane med problemløysing. Dei avgrensar ikkje dette til problemløysing i programmeringskontekst. I analysen såg me fleire gongar at eleven brukte desse strategiane til å løyse problemløysingsoppgåver ved hjelp av strategiar frå algoritmisk tenking.

5.4.2 Resonnering og argumentasjon

Kjerneelementet resonnering og argumentasjon er nært knyta til dei grunnleggande munnlege og skriftlege ferdigheitene. Kunnskapsdepartementet skildrar kjerneelementet resonnering og argumentasjon som:

> *Resonnering i matematikk handler om å kunne følge, vurdere og forstå matematiske tankerekker. Det innebærer at elevene skal forstå at matematiske regler og resultater ikke er tilfeldige, men har klare begrunnelser... Argumentasjon i matematikk handler om at elevene begrunner framgangsmåter, resonnementer og løsninger og beviser at desse er gyldige.(Kunnskapsdepartementet, 2019, s. 5).*

Då elevane arbeide med den feilsøkjande og testande tilnærminga argumenterte dei for ulike resultat. Her fekk elevane oppleve korleis deira program ville endre seg dersom dei endra enkelte element ved deira program. Ved å arbeide med denne tilnærminga, kan elevane utvikle si evne til å resonnere, sidan dei må knytte relasjon mellom deira program og endringane som skjer, samt trekke konklusjonar ut i frå det.

5.4.3 Representasjon og kommunikasjon

Kommunikasjon og representasjon er eit av kjerneelementa i matematikkfaget som skal arbeidast med og utviklast gjennom heile matematikkutdanninga til elevane. Kunnskapsdepartementet viser til representasjon og kommunikasjon som:

> *Representasjoner i matematikk er måter å uttrykke matematiske begreper, sammenhenger og problemer på. Representasjoner kan være konkrete, kontekstuelle, visuelle, verbale og symbolske. Kommunikasjon i matematikk handler om at elevene bruker matematisk språk i samtaler, argumentasjon og resonnementer (Kunnskapsdepartementet, 2019, s. 5)*

I arbeidet med algoritmisk tenking gjennom denne oppgåva, fekk elevane arbeida med sin kommunikasjon for å ytre sine strategiar og algoritmiske tilnærmingar. Denne

kommunikasjonen var noko eg ønskte å undersøke for å avdekke om dei viste kjenneteikn til algoritmisk tenking. Med dette meiner eg at oppgåvene og metoden for denne oppgåva var lagt opp til at elevane kommuniserte. Ut i frå analysen som bygger på elevane sin kommunikasjon mellom kvarandre, kan ein tenke seg at algoritmiske tilnærmingar gir eleven eit verktøy til å bygge kommunikasjonen på. Brennan og Resnick(2012) trekk fram dei algoritmiske perspektiva og perspektivet om samarbeid, der samarbeid er avhengig av kommunikasjon. Her trekk fram dei at kreativitet og læring er ein sosial aktivitet, der laging av ulike program for programmering er avhengig av interaksjonen mellom personar (Brennan & Resnick, 2012). I analysen såg me at elevane var aktive i å kommunisere samanhengar og problem under observasjonen. Elevane kommuniserte ulike idear om korleis dei kunne løyse eit problem, og formulerte planer som hadde rom for endring ifølge den adaptive prosessen.

Kunnskapsdepartementet(2020, s 5)skriv i si skildring av representasjonar at dei kan vera visuelle eller verbale, men skal utrykke matematiske samanhengar, omgrep eller problem. Ut i frå denne definisjonen til Utdanningsdirektoratet, kan program ein skriv vera ein representasjon av det matematiske problemet ein får presentert. Dette vil seie at programmeringsoppgåver vil føre til at elevane får utvikla deira evne til å arbeide med representasjonar. Programmeringsblokkene til Scratch er i seg sjølv representasjonen. Til dømes if/visst blokka som repersenterar det algoritmiske konseptet vilkår. I dette arbeidet har elevane brukt visuelle, verbale og symbolske representasjonar.

5.4.4 Abstraksjon og generalisering

I matematikkfaget vert abstraksjon skildra som ein overføring av læring mellom det konkrete nivået til det abstrakte tankestruktur, der matematikken føregår som reine tankeobservasjonar. Eit mål for vellykka læring er overføring av kunnskap og informasjon, og evne til å bruke tileigna kunnskap utanfor den lærde situasjonen. Dette er noko som må aktivt følgast opp. Spontanoverføring er vanskeleg å oppnå, sjølv for relativt enkel kunnskap (Detterman, i Sloustky, Kaminski og Heckler, 2005).

> *Abstraksjon i matematikk innebærer at elevene gradvis utvikler en formalisering av tanker, strategier og matematisk språk. (Kunnskapsdepartementet, 2019, s. 5)*

Eit av kjerneelementa til matematikkfaget inneber abstraksjon og generalisering som utdraget over skildrar. Dette kjerneelementet kan sjåast i samanheng med Detterman, i Sloustky, Kaminski og Heckler (2005) sine utsegn om overføring og abstraksjon i samband med læring og kor viktig det er. Den algoritmiske tilnærminga abstraksjon og modularisering overlappar med dette kjerneelementet der elevane som oftast må overføre informasjonen frå oppgåver til generaliserte algoritmar som maskiner kan forstå. I analysen viste det seg at denne tilnærminga ofte var meir tydeleg då elevane løyste oppgåver ved hjelp av programmering. Dette kjem nok grunna elevane har meir trening i å overføre oppgåvene til programmering. Det kom fram at dei brukte ulike strategiar som å identifisere ulike variablar som var nødvendige. I analysen såg me at då elevane løyste den siste oppgåva under programmeringssituasjonen, identifiserte dei kva variablar dei trengte for å løyse oppgåva. Desse strategiane kom nok ikkje spontan sidan Detterman, i Sloustky, Kaminski og Heckler (2005) fortel at det er usannsynleg. Ein kan tenke seg at slike strategiar også kan overførast inn i andre element i matematikkfaget, som kjerneelementa er laga for.

5.4.5 Matematiske kunnskapsområde

Det matematiske kunnskapskjerneelementet kan gjerast sentralt innafor programmeringsundervising og dette kan gjenspeglast i korleis ein legg til rette undervisinga. Sneider et al. viser til fleire områder som overlappar mellom matematisk tenking og algoritmisk tenking.

> *Kunnskapsområdene danner grunnlaget som elevene trenger for å utvikle matematisk forståelse ved å utforske sammenhenger innenfor og mellom kunnskapsområdene (Kunnskapsdepartementet, 2019, s. 5).*

I utdraget nemner Kunnskapsdepartementet at samanhengar mellom kunnskapsområde er viktig for å utforske, og kan skape matematisk forståing. Denne utforskinga kan godt skje i ein programmeringskontekst. Rammer for kva elever utforskar når dei arbeider med programmering er satt av eleven sine ferdigheiter, og oppgåvene dei arbeider med. Desse oppgåvene kan bruke dei algoritmiske konsepta som byggesteinar, men det er svært få avgrensingar når ein arbeider med programmering. I analysen arbeider eleven med ulike matematiske kunnskapsområde som ekvivalens og relasjonar mellom ulike størrelsar.

5.4.6 Modellering og anvendeleg

Eit av felleselementa mellom matematisk tenking og algoritmisk tenking er abstraksjon og modellering (Sneider et al., 2014). Dette fellestemaet går vidare inn i algoritmiske tilnærmingar, der ein av dei er modularisering og abstrahering (Brennan & Resnick, 2012). Kunnskapsdepartementet skildrar modellering som å lage modeller som skildrar dagleglivet, arbeidslivet og samfunnet elles. Slike modeller kan og vera programmer ein skriv for å skildre og løyse oppgåver.

> *Elevene skal ha innsikt i hvordan modeller i matematikk brukes for å beskrive dagliglivet, arbeidslivet og samfunnet ellers. Modellering i matematikk handler om å lage slike modeller(Kunnskapsdepartementet, 2019, s. 5).*

I analysen kom det fram at elevane arbeida ulikt med omsetjinga av oppgåva frå tekst til algoritme, eller eit program. Her var det eit skile både mellom elevane og gruppene når dei arbeide på papir, og i oppgåveløysing ved hjelp av programmering. Her arbeide elevane med å trekke ut variablar frå programmeringsoppgåvene som gjorde at dei fekk arbeide med å identifisere kva dei kunne lage ein modell av.

6 Avslutting

I denne oppgåva har eg drøfta og undersøkt kva som kjenneteikne elevar sitt arbeid med oppgåver basert på det algoritmisk konseptet vilkår. I denne oppgåva har eg brukt dei algoritmiske tilnærmingane som ei ramme for å gi meir innsyn og struktur i analysen min, og utifrå desse rammene kom eg fram til to problemstillingar:

- Kva kjenneteiknar fem elevar sitt arbeid med oppgåver basert på vilkår?
- Kva spor av algoritmisk tenking kan ein sjå i elevane sitt arbeid i ein programmeringskontekst og ved oppgåveløysing på papir?

Drøftinga er baserte på det teoretiske rammeverket frå Brennan og Resnick (2012), tidlegare studiar, teoriar om algoritmisk tenking og innsamla data i form av observasjonar av elevarbeid. Med dette grunnlaget som bakgrunn har eg diskutert elevane sine kjenneteikn på algoritmisk tilnærming i arbeidet med oppgåver basert på vilkår. Etterpå har eg drøfta likskapar og ulikskapar mellom desse kjenneteikna i arbeidet med oppgåver på papir og i ein programmeringskontekst. Vidare har eg drøfta korleis programmering og algoritmisk tenking passer inn i matematikkfaget. Avslutningskapittelet består av nokre oppsummerande refleksjonar knyta til oppgåveproblemstillinga og mogeleg vidare forsking.

6.1 Kva kjenneteiknar elevane sitt arbeid med oppgåver basert på vilkår?

I denne oppgåva har eg brukt Brennan og Resnick(2012) sine algoritmiske tilnærmingar for å finne kjenneteikna til algoritmisk tenking i elevane sitt arbeid med oppgåver basert på vilkår. Ved å undersøke datamaterialet frå observasjonen kom det fram ulike kjenneteikn. Elevane sin bruk av algoritmiske tilnærmingar varierte både basert på kor utfordrande oppgåva var og kor kjente dei var med konseptet oppgåva viste. Elevane sin presentasjon av algoritmisk tilnærmingar endra seg frå då dei arbeide med å løyse oppgåver på papir til då dei løyste oppgåva ved hjelp av programmering. Dette vert utdjupa i kapittel 6.2. Dei ulike tilnærmingane som eg observerte hos elevane var:

- Ei adaptiv og iterativ tilnærming (Being incremental and iterative)
- Ei testande og feilsøkjande tilnærming (Testing and debugging)
- Ei gjenbrukende tilnærming (Reusing and remixing)
- Ei abstrakt og modulariserande tilnærming (Abstracting and modularizing)

Då elevane brukte den adaptive og iterative tilnærminga var det tydleg at dei la ein plan for å identifisere kva variablar og kva type operasjonar dei trengte for å gjennomføre oppgåva. Ofte endra dei planen sin meir etter ein runde med feilsøking eller då dei oppdaga at strategiane dei valde ikkje gav noko tydeleg resultat. Eit kjenneteikn med adaptive og iterative tilnærminga var at den vert brukt periodevis undervegs i oppgåveløysing, og ofte i samband med andre tilnærmingar.

Den testande og feilsøkende tilnærminga var den tydlegaste og mest brukte tilnærminga då elevane arbeide med oppgåvene. Denne tilnærminga innebar å leite etter feil i eit løysingsforslag ved å gjennomføre testar. Ein kan kjenne igjen denne tilnærminga då

elevane diskuterte og prøvde ut svara dei fekk. Dette er eit tydeleg kjenneteikn som ein kan bruke og vidareutvikle i undervisingen vidare.

Ein gjenbrukende tilnærming saman med den abstrakt og modulariserande tilnærminga er meir avhengig av oppgåvene sin karakter, men også situasjonen rundt spelar inn for å kunne kjenne att tilnærmingane. Den gjenbrukende tilnærminga vart observert eit par gonger under oppgåveløysing på papir og ein kan tenke seg at det er gjenkjenning i fleire situasjonar som ikkje vart observert. Denne tilnærminga var meir tydeleg under problemløysing ved hjelp av programmering, og kan tenke seg at dette gir eit større kjenneteikn på algoritmisk tenking når dei arbeider med programmering. Her brukte elevane oppatt deler av koden frå tidlegare oppgåver, for å gjenbruka ei løysing som allereie er testa.

Under observasjonane av elevane var det den siste oppgåva i oppgåvesettet som viste seg å vere mest utfordrande for elevane. Det var synleg både under oppgåveløysing på papir og ved programmering. Desse oppgåvene var det som framkalla flest steg for å kunne løysast. Under løysinga av dei større oppgåvene var det den abstrakte og modulariserande tilnærminga kom best fram. Dette kan komme frå at eit behov for å kommunisere innan i gruppa for å løyse oppgåva. Brennan og Resnick(2012) forklarte i sin artikkelen at ein abstrakt og modulariserande tilnærming er nytting når ein skal gjennomføre større prosjekter og det kunne me sjå igjen i analysen min. Dei viste til at store prosjekter blir ofte laga i separate delar, der elevane løyser eit og eit problem av gangen. Dette vil då vera eit kjenneteikn å sjå etter dersom elever skal arbeide med større prosjekt for å forsikre seg at algoritmisk tenking vert brukt og blir utvikla vidare.

6.2 Kva spor av algoritmisk tenking kan ein sjå i arbeidet med oppgåver både på papir og i ein programmeringskontekst?

Forskjellane mellom bruken av dei algoritmiske tilnærmingane ved oppgåveløysing på papir versus oppgåveløysing i ein programmeringskontekst kom tydleg fram under observasjonane og den etterfølgande analysen. Det viste seg at då elevane arbeidar med oppgåver i ein programmeringskontekst nytta dei tilnærmingane mykje meir aktivt og det var tydlegare å kjenne igjen dei ulike variantane slik som Brennan og Resnick(2012) skildra dei.

Den adaptive og iterative tilnærminga hadde mindre skilnader samanlikna med dei andre tilnærmingane. Forskjellane var basert på kva type element dei trong. Ved programmering diskuterte elevane kva type Scratch blokker det var behov for. Medan i oppgåveløysing på papir diskuterte dei meir stegvis kva type matematikk operasjonar dei trengte å gjere for å løyse oppgåva.

Den testande og feilsøkjande tilnærminga presenterte seg meir likt i oppgåveløysing på papir og ved hjelp av programmering enn dei andre tilnærminga. Her gjennomfører elevane ofte ulike testar for å sjå om løysinga deira er rett. Ei trend i oppgåveløysing på papir er ofte at elevane testar og feilsøkar på fleire av oppgåvene samtidig. Medan då dei arbeidar med programmeringsoppgåver testa og feilsøka dei ei og ei oppgåve av gangen.

Forskjellen mellom elevane sin bruk av den gjenbrukende tilnærminga ved oppgåveløysing på papir og i ein programmerings kontekst er større enn dei andre tilnærmingane. Då elevane brukte den tilnærminga ved oppgåveløysing på papir, anerkjente dei berre at oppgåvene var like, men vidareførte ikkje den informasjonen til den neste oppgåva. I programmeringskonteksten var gjenbruk av både deler av oppgåva og heile oppgåva meir vanleg. Dette kan ein og kjenne igjen i ei rekke andre programmeringskulturar og studiar på elevar sitt arbeid med programmering (Cui & Ng, 2021; Perens et al., 1998).

I tilfella med den abstrakte og modulariserande tilnærminga var det nok fleire likskapar mellom oppgåveløysing på papir og oppgåveløysing med hjelp av programmering enn for dei andre tilnærmingane til oppgåveløysing. Denne tilnærminga var mest tydeleg då oppgåvene krev fleire steg for å gjennomføre og her arbeidde elevane med å isolere desse stega for deretter å løyse dei enkeltvis. Den største forskjellen var at den abstrakte og modulariserande tilnærminga var til stades på mindre oppgåver då elevane løyste dei ved hjelp av programmering, men kom berre fram då det var større oppgåver då dei løyste dei på papir.

6.3 Vidare forsking

Cui og Ng fortel i sin artikkel at vidare forsking på algoritmisk tenking bør utforske korleis man best kan orientere elevanes læringsløp for å forbetre matematikkundervisninga med algoritmisk tenking(Cui & Ng, 2021). Etter arbeidet med denne oppgåva har nokre tankar angåande vidare forsking på dette temaet kome fram. Eg stiller meg spørsmål om korleis arbeidet med grupper påverka elevane si algoritmisk tenking, korleis hadde andre algoritmiske konsepteter påverka eit slikt studie og korleis hadde eit større elevprosjekt påverka studiet.

6.3.1 Programmering i grupper eller åleine

Eit av dei elementa som har påverka dette prosjektet er at elevane har arbeidd i grupper. Dette har gjort at eg har fått innsikt i deira kommunikasjon utan å måtte spørje dei ut eller at dei var i ein uvant situasjon, som ein observasjonssetting ofte er. Vidare spørsmål som ein kan undersøke djupare er tankemåten deira, dersom elevane får arbeide aleine og ein kan spørje dei ut om val og strategiar. Dersom elevane hadde hatt individuelle oppgåver måtte ein hatt grundig og informasjonsrik kommunikasjon som gir innsikt i eleven sin tankegang og val av tilnærmingar. Det å oppnå ein slik kommunikasjon kan væra utfordrande i løpet av eit kortare tidsrom.

6.3.2 Andre algoritmiske konsept

Ramma for dette prosjektet er svært avhengig av det algoritmiske konseptet vilkår. Det har avgjort kva type oppgåver som vert løyst og dermed løysingsmetodane elevane brukte. Vidare forsking kunne vore å endra på denne ramma og velja eit anna konsept å utforske. Hadde dei fått same type resultat som meg då?

6.3.3 Større prosjekter

Eit av resultata frå analysen var at den algoritmisk tilnærminga kom meir fram dersom elevane arbeidde med større og meir komplekse oppgåver både på papir og ved hjelp av programmering. Dette hadde vore interessant å undersøke nærmare. Kan desse større prosjekta vera med på utvikle elevane sin bruk av algoritmisk tilnærming? Eit slik prosjekt kunne vera med å informere om korleis ein skal planlegge og utføre undervising med tanke på algoritmisk tenking.

7 Litteraturliste

Adler, P. A., & Adler, P. (1987). *Membership roles in field research*. Sage Publications.

Angrosino, M., & Rosenberg, J. (2011). Observations on Observation: Continuities and Challenges. I N. K. Denzin & UTFORDRANDE. S. Lincoln, *The Sage handbook of qualitative research* (s. 467–478).

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *THE NORDIC APPROACH TO INTRODUCING COMPUTATIONAL THINKING AND PROGRAMMING IN COMPULSORY EDUCATION*. CNR Edizioni. https://doi.org/10.17471/54007

Borg, A., Fahlgren, M., & Ruthven, K. (2020). *Programming as a mathematical instrument: The implementation of an analytic framework*. 10.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. 25.

Brinkmann, S., & Tanggaard, L. (2012). *Kvalitative metoder empiri og teoriutvikling*. Gyldendal akademisk.

Buteau, C., Gueudet, G., Muller, E., Mgombelo, J., & Sacristán, A. I. (2020). University students turning computer programming into an instrument for 'authentic' mathematical work. *International Journal of Mathematical Education in Science and Technology*, *51*(7), 1020–1041. https://doi.org/10.1080/0020739X.2019.1648892

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

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

de Vreese, A. (1995). Euthanasia and the ICU. *Lancet (London, England)*, *346*(8983), 1164. https://doi.org/10.1016/s0140-6736(95)91837-x

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016

Gold, R. L. (1958). Roles in Sociological Field Observations. *Social Forces*, *36*(3), 217–223. https://doi.org/10.2307/2573808

Hemnes, C. (2018). *Programmering og problemløsning i småskolen Programming and problem-solving in early primary school (K-4)*. 107.

Horstmann, C. S., & Necaise, R. D. (2019). *Python for everyone* (3/e, EMEA edition). Wiley.

Israel, M., & Lash, T. (2020). From classroom lessons to exploratory learning progressions: Mathematics + computational thinking. *Interactive Learning Environments*, *28*(3), 362–382. https://doi.org/10.1080/10494820.2019.1674879

Johannessen, L. E. F., & Rafoss, T. UTFORDRANDE., Rasmussen, Erik B??rve. (2018). *Hvordan bruke teori?: Nyttige verktøy i kvalitativ analyse*. Universitetsforlag.

Kanoknitanunt, P., Nilsook, P., & Wannapiroon, P. (2021). Imagineering Learning With Logical Problem Solving. *Journal of education and learning*, *10*(3), 112. https://doi.org/10.5539/jel.v10n3p112

Kong, S.-C. (2019). Components and Methods of Evaluating Computational Thinking for Fostering Creative Problem-Solvers in Senior Primary School Education. I S.-C. Kong & UTFORDRANDE. Abelson (Red.), *Computational Thinking Education* (s. 119–141). Springer Singapore. https://doi.org/10.1007/978-981-13-6528-7_8

Kording i skolen. (2022). *Vilkår i Python*. https://koding.verket.me/grunnleggende-programmering/grunnleggende-python/vilkar-i-python/

Kunnskapsdepartementet. (2019). *Læreplan i matematikk 1.–10. Trinn*. 15.

Kvarv, S. (2014). *Vitenskapsteori tradisjoner, posisjoner og diskusjoner*. Novus.

Lee, I., & Malyn-Smith, J. (2020). Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective. *Journal of Science Education and Technology*, *29*(1), 9–18. https://doi.org/10.1007/s10956-019-09802-x

Levin, O. (2019). *Discrete mathematics: An open introduction*.

Lockwood, E., Lockwood, E., & Asay, A. (2016). ALGORITHMIC THINKING: AN INITIAL CHARACTERIZATION OF COMPUTATIONAL THINKING IN MATHEMATICS. *J. A.*, 8.

Lonchamp, J. (2012). An instrumental perspective on CSCL systems. *International Journal of Computer-Supported Collaborative Learning*, *7*(2), 211–237. https://doi.org/10.1007/s11412-012-9141-4

Ludvigsen, S. (2015). *Fremtidens skole: Fornyelse av fag og kompetanser : utredning fra et utvalg oppnevnt ved kongelig resolusjon 21. juni 2013 : avgitt til Kunnskapsdepartementet 15. juni 2015*. Departementenes sikkerhets- og serviceorganisasjon, Informasjonsforvaltning.

Lye, S. UTFORDRANDE., & Koh, J. UTFORDRANDE. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Matsumoto, P. S., & Cao, J. (2017). The Development of Computational Thinking in a High School Chemistry Course. *Journal of Chemical Education*, *94*(9), 1217–1224. https://doi.org/10.1021/acs.jchemed.6b00973

Mertens, D. M. (2010). Transformative Mixed Methods Research. *Qualitative Inquiry*, *16*(6), 469–474. https://doi.org/10.1177/1077800410364612

NESH. (2021). *Forskningsetiske retningslinjer for samfunnsvitenskap, humaniora, juss og teologi.* https://utfordrande.etikkom.no/forskningsetiske-retningslinjer/Samfunnsvitenskap-jus-og-_humaniora/

NOU, & 2015:8. (2015). *Fremtidens skole: Fornyelse av fag og kompetanser : utredning fra et utvalg oppnevnt ved kongelig resolusjon 21. juni 2013 : avgitt til Kunnskapsdepartementet 15. juni 2015*. Departementenes sikkerhets- og serviceorganisasjon, Informasjonsforvaltning.

Observasjonsmetoder. (2012). I S. Brinkmann, L. Tanggaard, & P. Rautaskoski, *Kvalitative metoder empiri og teoriutvikling*. Gyldendal akademisk.

Pei, C. (Yu), Weintrop, D., & Wilensky, U. (2018). Cultivating Computational Thinking Practices and Mathematical Habits of Mind in Lattice Land. *Mathematical Thinking and Learning*, *20*(1), 75–89. https://doi.org/10.1080/10986065.2018.1403543

Perens, B., Sroka, M., & Stu, M. (1998). *The Open Source Definition*. 9.

Postholm, M. B., & Jacobsen, D. I. (2011). *Læreren med forskerblikk: Innføring i vitenskapelig metode for lærerstudenter*. UTFORDRANDE??yskoleforlaget.

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

Rossen, E. (2022). Informatikk. I *Store norske leksikon*. http://snl.no/informatikk

Sannes, UTFORDRANDE. S. (2020). «Vi kan jo prøv gjentablokka?»: En kvalitativ casestudie om algoritmisk tenkning og løkker i en programmeringskontekst [Masteroppgave]. *NTNU*.

Sloutsky, V. M., Kaminski, J. A., & Heckler, A. F. (2005). The advantage of simple symbols for learning and transfer. *Psychonomic Bulletin & Review*, *12*(3), 508–513. https://doi.org/10.3758/BF03193796

Smestad, B. (2021). *Tall og tanke: Aktivitetsbok: Utforsking og argumentasjon* (1. utgave.). Gyldendal.

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Teacher's Toolkit: Exploring the Science Framework and NGSS: Computational Thinking in the Science Classroom. *Science Scope*, *038*(03). https://doi.org/10.2505/4/ss14_038_03_10

Solem, I. UTFORDRANDE. (2017). *Tall og tanke 2 matematikkundervisning p?? 5. Til 7. Trinn*. Gyldendal akademisk.

Stake, R. E. (1995). *The art of case study research*. Sage Publications.

Su, A. UTFORDRANDE. S., Yang, S. J. UTFORDRANDE., Hwang, UTFORDRANDE.-UTFORDRANDE., & Zhang, J. (2010). A Web 2.0-based collaborative annotation system for enhancing knowledge sharing in collaborative learning environments. *Computers & Education*, *55*(2), 752–766. https://doi.org/10.1016/j.compedu.2010.03.008

The National University of Malaysia (UKM), Chongo, S., Osman, K., Nayan, N. A., The National University of Malaysia (UKM), & The National University of Malaysia (UKM). (2020). Level of Computational ThinkingLevel of Computational Thinking Skills among Secondary Science Student: Variation across Gender and Mathematics Achievement Skills among Secondary Science Student: Variation across Gender and Mathematics Achievement. *Science Education International*, *31*(2), 159–163. https://doi.org/10.33828/sei.v31.i2.4

Torkildsen, UTFORDRANDE. A. (2019). ILU, NTNU oistein.gjovik@ntnu.no. *UTFORDRANDE. A.*, 7.

Tran, UTFORDRANDE. (2019). Computational Thinking Equity in Elementary Classrooms: What Third-Grade Students Know and Can Do. *Journal of Educational Computing Research*, *57*(1), 3–31. https://doi.org/10.1177/0735633117743918

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

8 Vedlegg

8.1 Vedlegg 1 Oppgåvesett

Oppgavesett

Oppgave 1

Hvis Helle har tre eller flere epler, så kan hun lage en eplekake.
Helle har fem epler og et gresskar. Kan hun lage en eplekake?
Ja eller Nei

Oppgave 2

Under følger et sett med utsang der de står på samme linje, hører sammen. Dere skal avgjør om det er noen sammeheng mellom desse to utsangene.
Stemmer det hvis det ene utsanget er sant, så er det andre utsnaget sant?
Og omvnet?
Her kjem et eksempel

Ellens tall er større enn 20.	Ellens tall er større enn 16.
Hvis Ellens tall er større enn 20, *så* er Ellens tall større enn 16?	Det stemmer.
Hvis Ellens tall er større enn 16, *så* er Ellens tall større enn 20?	Det stemmer ikke, Ellens tall kan f.eks. være 18.

a)	Tromsø er en katt.	Tromsø er en gråstripete katt.
b)	Figuren har fire hjørner.	Figuren er et kvadrat.
c)	Sol har fire ben.	Sol er en katt.
d)	Sekken veier mindre enn 20 kg.	Sekken veier mer enn 10 kg.
e)	$x^2 = 25$.	$x = 5$.
f)	Sarakka bor i Karasjok.	Sarakka bor i Norge.

Oppgave 3

Hvis Martin har to bananer, så vil han dele med sin venn. Ellers vil Martin beholde bananen.
Martin har en banan og to gulerøtter. Hva skal Martin gjøre?

_____

Oppgave 4

Betingelsen	verdi
Når det ikke er sol på himmelen så regner det.	Sant eller usant
Hvis en hopper i havet så blir en våt.	Sant eller usant
Etter sommer kjem høst.	Sant eller usant
Hvis Olav er 15 så blir Olav 16 ved hans neste bursdag.	Sant eller usant

Oppgave 5

Hvis Ragnhild har egg og sukker så kan hun lage eggedosis. Dersom hun bare har egg så lager hun eggerøre. I dag har Ragnhild bare 3 egg og en pakke med mel. Hva gjør Ragnhild?

Oppgave 6

Diskuter sammenhengen mellom desse to utsagnene.
Hvis figuren er et kvadrat, er alle vinkelen rette. Hvis ikke alle vinkelen er rette, er figuren ikke et kvadrat.
Hva ville vert det motsatte for dette utsagnet?
Hvis x er mindre enn 100, så er x mindre enn 1 000
Hvis fuglen ikke synger, regner det

Kuleproblemet- å finne den som er lettere

En gruppe elever får utdelt en vekt og tre kuler, en rød, en blå og en grønn. De får vite at to av de er like tunge mens en av kulene er lettere en de andre. Hvordan kan de finne det ut? Hvor mange ganger må de veie de ulike kulene for å bli sikre på sitt svar?

Programming

Oppgave 1

Planlegg og skriv et program der Sprite spør om din alder.

Hvis du er 19 eller yngre, svar at du må være på skolen.

Hvis du er eldre enn 19, svar at du må være på jobb.

Oppgave 2

Skriv et program der Sprite spørre noen om alderen deres.

Hvis svaret mindre enn 18 år, få Sprite til å si "Du er for ung til å kjøre", og hvis de er over 18, få Sprite til å spørre om de har tatt lappen.

Oppgave 3

Du jobber i en butikk, og der får man betalt 45 kr per time frem til fylte 20 år. Når du er 20 år eller eldre får du betalt 55 kr per time. Planlegg og skriv et program der Sprite spør om en persons navn, alder og antall timer jobbet i løpet av en uke. Sprite skal da regne ut lønnen og skrive den ut sammen med personens navn og alder.

8.2 Vedlegg 2

Vurdering

Referansenummer
571767

Prosjekttittel
Programmering i skule matematikk og korleis elvene bruker det som eit verktøy i algebra

Behandlingsansvarlig institusjon
Norges teknisk-naturvitenskapelige universitet / Fakultet for samfunns- og utdanningsvitenskap (SU) / Institutt for lærerutdanning

Prosjektperiode
06.09.2021 - 01.09.2022

Meldeskjema

Dato
11.10.2021

Type
Standard

Kommentar
Det er vår vurdering at behandlingen av personopplysninger i prosjektet vil være i samsvar med personvernlovgivningen så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjemaet med vedlegg den 11.10.2021, samt i meldingsdialogen mellom innmelder og NSD. Behandlingen kan starte.

DEL PROSJEKTET MED PROSJEKTANSVARLIG
For studenter er det obligatorisk å dele prosjektet med prosjektansvarlig (veileder). Del ved å trykke på knappen «Del prosjekt» i menylinjen øverst i meldeskjemaet. Prosjektansvarlig bes akseptere invitasjonen innen en uke. Om invitasjonen utløper, må han/hun inviteres på nytt.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til 01.09.2022.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres, og som den registrerte kan trekke tilbake.

Lovlig grunnlag for behandlingen vil dermed være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

PERSONVERNPRINSIPPER
NSD vurderer at den planlagte behandlingen av personopplysninger vil følge prinsippene i personvernforordningen om:

· lovlighet, rettferdighet og åpenhet (art. 5.1 a), ved at de registrerte får tilfredsstillende informasjon om og samtykker til behandlingen
· formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte og berettigede formål, og ikke behandles til nye, uforenlige formål
· dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante og nødvendige for formålet med prosjektet
· lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for å oppfylle formålet

DE REGISTRERTES RETTIGHETER
Så lenge de registrerte kan identifiseres i datamaterialet vil de ha følgende rettigheter: innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18), og dataportabilitet (art. 20).

NSD vurderer at informasjonen om behandlingen som de registrerte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

Vi minner om at hvis en registrert tar kontakt om sine rettigheter, har behandlingsansvarlig institusjon plikt til å svare innen en måned.

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

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

For å forsikre dere om at kravene oppfylles, må dere følge interne retningslinjer og/eller rådføre dere med behandlingsansvarlig institusjon.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde: https://www.nsd.no/personvernтjenester/fylle-ut-meldeskjema-for-personopplysninger/melde-endringer-i-meldeskjema. Du må vente på svar fra NSD før endringen gjennomføres.

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

Kontaktperson hos NSD: Silje Fjelberg Opsvik
Lykke til med prosjektet!

8.3 Vedlegg 2 samtykke erklæring

«Programming og logiske oppgaver»
Dette er et spørsmål til deg, som foreldre/foresatt, om barnet ditt kan delta i et forskningsprosjekt hvor føremålet er å se på elever sine løysinger av logiske oppgaver og hvordan dei programmere da. Dette er et prosjekt som Institutt for lærerutdanning ved NTNU Trondheim er ansvarlig for. I dette skrivet gir vi deg informasjon om prosjektet og hva deltakelse vil innebære for ditt barn.

Føremål
Dette forskningsprosjektet er en del av en masteroppgave som etter planen skal fullføres i løpet av våren 2022. Oppgaven omhandler temaet programmering og logiske settinger. Jeg ser på dette tema som svært dagsaktuelt og viktig for flere grunner. Blant annet er programmering et nytt tema i skolen og har dermed mindre erfaringer og forsking knyttet til det. Prosjektet vil sette søkelys på hvordan eleven løyser oppgaver først på papir deretter oversette de til programmering. Dette er en utfordring som elever møter flere ganger i løpet av skolegangen sin.

Har du spørsmål om prosjektet er det bare å ta kontakt.

Jeg håper du synes dette prosjektet virker interessant og samfunnsnyttig og vil la ditt barn delta i forskningsprosjektet. På forhånd takk!

Med vennlig hilsen

Marit Hystad (masterstudent)
Tlf: +47 45860443
E-post: marithys@stud.ntnu.no

Hermund Torkilsen (veileder)
Tlf: +47 73412778
E-post: hermund.a.torkildsen@ntnu.no

Samtykkeerklæring

Jeg har mottatt og forstått informasjonen om masterprosjektet *Programming og logiske settinger*, og har fått anledning til å stille spørsmål. Jeg samtykker til at prosjektet kan samle inn datamateriale om utfordrande_____
(navn på elev) gjennom (kryss av der det passer):

- ☐ Observasjon av undervisningsopplegg
- ☐ Intervju
- ☐ Innsamling av elevarbeid

_____
_____
(signert av foreldre/foresatt, dato)

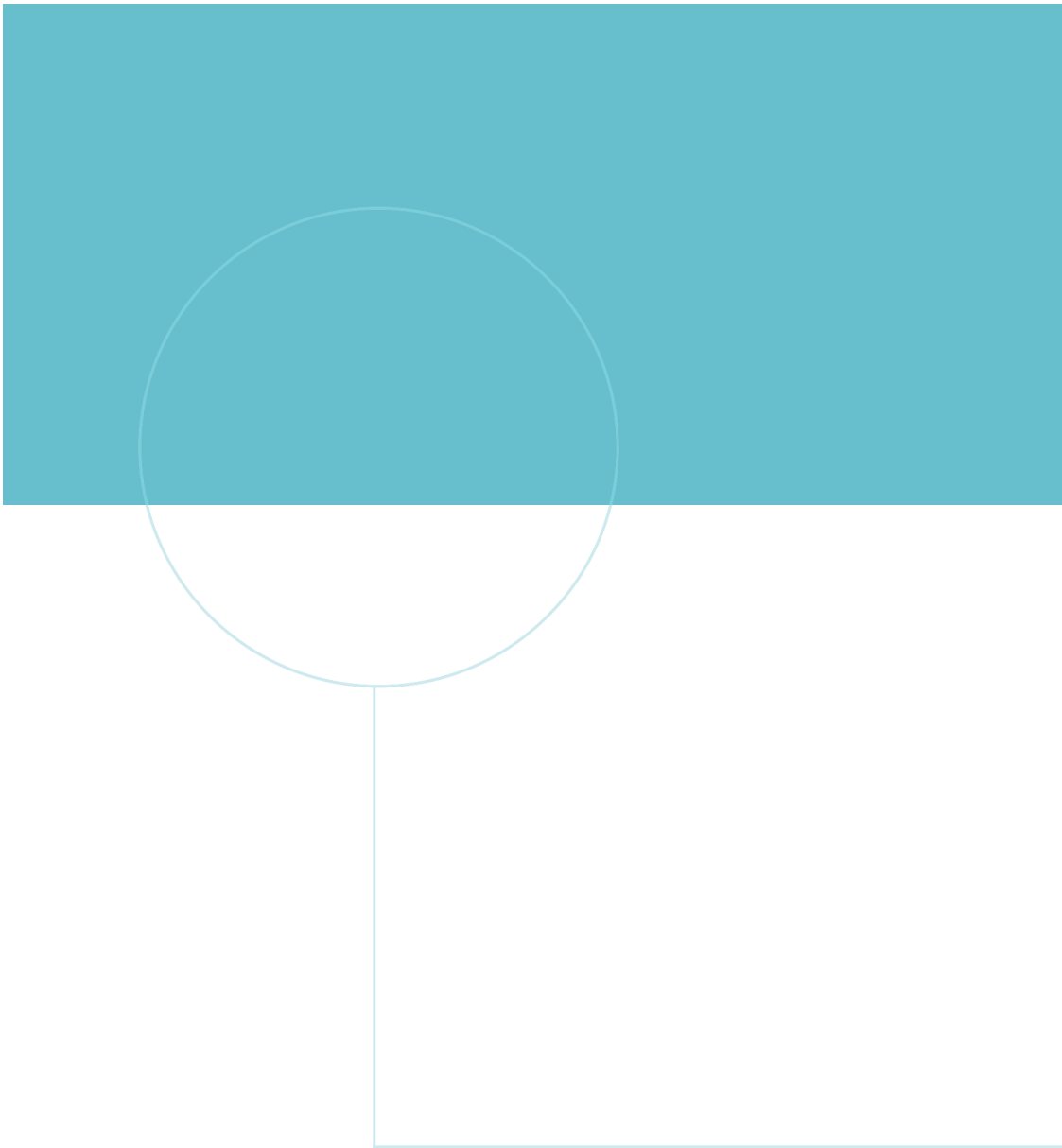
Hva innebærer det for barnet ditt å delta?
Å delta i «Programmering og logiske oppgåver» prosjektet vil innebære at ditt barn er til stede i en undervisningstime som vil bli observert der elevene skal løyse et sett med oppgaver. Det vil også bli gjennomført intervju av et utvalg elever i samråd med lærer for å få bedre innsikt i elevenes tankemåter. Desse intervjuene vil det bli tatt opptak av. Lydmaterialet vil bli anonymisert og lagret på en sikker måte. Hvis du som forelder/foresatt har behov for det, kan dere få se intervjuguide på forhånd ved å ta kontakt med meg.
Dersom du tillater barnet ditt å delta i prosjektet, «begrepsforståelse i statistikk» innebærer det at barnet ditt fyller ut et spørreskjema med ulike oppgaver om statistikk. Oppgavene vil ta ca 30 minutt. Jeg vil observere elevene når de fyller ut spørreskjemaet. I tillegg vil jeg intervjue noen elever, for å få mer utfyllende beskrivelser av oppgavesvara deres. Det vil være spørsmål som «Hvordan vil du beskrive begrepet typetall?». Jeg vil ta lydopptak og videoopptak av intervjuet. Dersom det er ønskelig kan jeg sende spørreskjemaet og intervjuguiden som blir brukt i dette prosjektet. Observasjonen mine, svara til deltakerne på spørreskjema og refleksjonen under intervjua til deltakerne vil bli anonymisert etter datainnsamling.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å la barnet ditt delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle personopplysninger om barnet ditt vil da bli slettet. Det vil ikke ha noen negative konsekvenser for dere hvis du ikke vil at barnet skal delta eller senere velger å trekke barnet ut av prosjektet.

Personvern – hvordan vi oppbevarer og bruker opplysningene
Vi vil bare bruke opplysningene om barnet ditt til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket. De som velger å delta vil ikke kunne gjenkjennes i den ferdige oppgaven. Jeg (Marit, masterstudent) og min veileder ved NTU vil ha tilgang til desse opplysningene. Materialet som vil bli brukt i oppgaven eller presentert for andre vil bli anonymisert slik at deltakerne ikke skal kunne bli gjenkjent.
Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Opplysningene anonymiseres når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen skal skje innen 1. september 2022.
Hva gir oss rett til å behandle personopplysninger om barnet ditt?
På oppdrag fra NTNU (Norges teknisk-naturvitenskapelige universitet) har NSD – Norsk senter for forskningsdata AS vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.
Dine rettigheter

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

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



NTNU

Kunnskap for en bedre verden