UiT Norges arktiske universitet

Fakultet for naturvitenskap og teknologi

Programmering som verktøy for å lære computational thinking

En kvalitativ studie av elevers resonnement i problemløsning med programmering som verktøy.

Markus Løkke Granaas

Mastergradsoppgave i matematikk ved lektorutdanningen trinn 8-13, MAT-3907, juni 2022

Innholdsfortegnelse

Tabelliste

Figurliste

Forord

En blandet følelse av begeistring og tomhet blomstrer frem etter å ha fullført oppgaven som symboliserer slutten på min studietid. Fem innholdsrike og gode år på lektorutdanningen i realfag ved universitetet i Tromsø er over, og nye dører skal åpnes. Etter studieperioden og arbeidet med oppgaven sitter jeg igjen med et smil om munnen og en ryggsekk fylt med kompetanse og fantastiske opplevelser. Det er mange som fortjener en takk for dette.

Først ønsker jeg å takke skolen, elevene og lærerne som har tatt seg tiden til å delta i dette forskningsprosjektet. Samarbeidet har vært upåklagelig.

Videre ønsker jeg å rette en stor takk til min veileder Johan Lie, som gjennom konstruktive tilbakemeldinger, sterke faglige diskusjoner og godt humør har styrket oppgaven. Jeg vil også takke veileder Ragnar Soleng for gode matematiske innspill. Jeg ønsker også å takke mine studiekamerater for å ha vært gode sparringspartnere, og ikke minst for lange kaffe- og lunsjpauser fylt med latter. Dette har gjort studietiden og oppgaveskrivingen til en minnerik opplevelse.

Avslutningsvis ønsker jeg å takke de som står meg nærmest. Jeg ønsker å takke mine brødre, Morten og Magnus, som alltids stiller opp for meg og gir mange gode råd. Videre må jeg takke min samboer, Aili, som gjennom sin utstråling og gode vesen sørger for ingen gråe hverdager. En spesiell stor takk rettes til mine foreldre, Sissel og Annathon, for å ha videreført deres gode verdier og holdninger til meg. Uten deres kjærlige og støttende ord hadde oppgaven aldri blitt til.

Sammendrag

Samfunnets stadige endringer er katalysert av teknologiens utvikling og fremskritt. Samfunnet digitaliseres ytterligere og digitale verktøy er blitt en stor del av menneskets hverdag. Dette har endret kompetansebehovet for fremtiden. For å dekke morgendagens kompetansebehov og holde tritt med samfunnets digitale og teknologiske utvikling har Utdanningsdirektoratet (2020) gjennom *kunnskapsløftet 2020* implementert programmering i matematikkfaget. I lys av dette samt egen erfaring og motivasjon er følgende problemstilling utformet: *Ved hvilke kjennetegn av computational thinking resonnerer elever kreativt i problemløsning med tekstbasert programmering som verktøy?*

Problemstillingen har blitt belyst gjennom en kvalitativ tilnærming med deltakende observasjon som metode. Studiens datamaterialet består av notater og lyd- og skjermopptak av elevers problemløsning med programmering som verktøy. Datamaterialet er analysert ved hjelp av et sammensatt rammeverk bestående av Lithner (2006) rammeverk for analysering av imitativ og kreativ resonnement og Shute et al. (2017) sitt rammeverk for computational thinking.

Analysen viser at Shute et al. (2017) sine komponenter av computational thinking er til stede i elevers resonnement i problemløsning med programmering som verktøy. Komponentene som ble observert flest ganger var algoritmer, feilsøking og dekomposisjon. Videre viser analysen at resonnementene tilknyttet komponentene var både kreative og imitative. Studiens resultater antyder at computational thinking gjennom programmering kan bidra til å øke elevers matematiske kreativitet, og ha en positiv effekt på elevers matematiske kunnskap. Det ble også avdekket et behov for å øke elevenes feilsøkingsevner i programmering.

1 Innledning

Samfunnets stadige endringer er katalysert av teknologiens utvikling og fremskritt. Vi står overfor en fjerde industriell revolusjon som karakteriseres av teknologiens utvikling (Forsström & Kaufmann, 2018, s. 18). Samfunnet digitaliseres ytterligere og digitale verktøy blir en større del av menneskets hverdag. Bruken av datateknologi har påvirket stort sett alle studieretninger og endret måten vi arbeider på (Barr et al., 2011, s. 23). I følge Balanskat og Engelhardt (2015, s. 4) er digital kompetanse og ferdigheter en av de viktigste faktorene for å lykkes i den digitale revolusjonen. Ferdigheter i programmering anses som en viktig kompetanse i det 21.århundret (Forsström & Kaufmann, 2018, s. 18), og computational thinking trekkes frem som 21.århundrets nye literacy, med det menes kyndighet i computational thinking (Grover & Pea, 2013, s. 3; Shute et al., 2017, s. 143; Wing, 2010, s. 3). Dette begrunnes blant annet ved at programmeringsferdigheter bidrar til å forstå dagens digitaliserte samfunn og fremmer viktige ferdigheter for det 21.århundret, slik som problemløsning, kreativitet og logisk tenkning (Balanskat & Engelhardt, 2015, s. 6).

I overordnet del av læreplanverket under *formål med opplæringen* er det påpekt at opplæringen skal åpne dører mot verden og fremtiden (Kunnskapsdepartementet, 2017). Utdanningssektoren må forberede dagens elever på fremtiden ved å tilegne de nødvendig kunnskap for å mestre og forme fremtiden. Lærere, økonomer og politikere verden rundt har anerkjent databehandling og programmering som viktige ferdigheter for dagens elever. Som følge av dette har flere land på ulike måter innført programmering i læreplan (Balanskat & Engelhardt, 2015, s. 6). For å holde tritt med samfunnets digitale og teknologiske utvikling har Norge nylig integrert programmering i matematikkfaget gjennom *kunnskapsløftet 2020*.

1.1 Bakgrunn for valg av tema og problemstilling

Matematisk kreativitet sørger for matematikkfeltets utvikling som helhet (Sriraman, 2008, s. 13). Likevel har kreativiteten fått mindre plass i skoleverket over de siste tiårene (Beghetto & Kaufman, 2010, s. 191). Ved hjelp av forskning og reformer har matematikkdidaktikere og politikere forsøkt i flere tiår å utvikle elever til og bli dyktige problemløsere. Til tross for dette er det fortsatt imitativ resonnement som er dominerende blant elever (Lithner, 2006, s. 2). I følge Lithner (2003, s. 54) skyldes dette en undervisningskultur med sterkt fokus på prestasjon og memorering av instruksjoner. Elevene memorerer instruksene så godt at de utføres med svært få feil. Som resultat av dette behøver ikke elevene å resonnere over riktigheten av instruksjonene.

Min opplevelse av egen skolegang samsvarer med Lithner (2003) skildringer. Jeg opplevde i egen skolegang at undervisningskulturen bar preg av et prestasjonsfokus, og problemløsningen hovedsakelig innebar memorering av algoritmer. Siden den gang er kunnskapsløftet 2020 og ny læreplan for matematikk innført. Under *fagrelevans og sentrale verdier* er det påpekt at matematikkfaget skal legge til rette for kreativitet gjennom å la elevene få nok tid til å tenke, reflektere, resonnere og stille spørsmål (Utdanningsdirektoratet, 2020). Videre under fagets kjerneelement *utforsking og problemløsning* er det poengtert at problemløsning innebærer å utvikle metoder for å løse ukjente problemer. Disse to utdragene fra kunnskapsløftet 2020 er motpoler av min opplevelse av egen skolegang.

Balanskat og Engelhardt (2015, s. 6) peker på programmering som et virkemiddel for å utvikle kreativitet og problemløsning. Dette samsvarer med min erfaring fra programmering- og matematikkemner på universitetsnivå. Etter å ha gjennomført flere programmeringsemner oppleves det slik at min matematiske kreativitet og problemløsningsevner har økt. Dersom dette er tilfellet kan programmering bidra til å utvikle matematikkfaget og resultere i matematisk kreative elever med dyktige problemløsningsevner. Min opplevelse fra universitetet har bidratt til å fremme en interesse og nysgjerrighet for sammenhengen mellom matematisk kreativitet og programmering. Spesielt når denne sammenhengen er situert i en problemløsningskontekst. På bakgrunn av nevnte interesse ønsker jeg å undersøke følgende problemstilling:

Ved hvilken kjennetegn av computational thinking resonnerer elever kreativt i problemløsning med tekstbasert programmering som verktøy?

På bakgrunn av studiens problemstilling er det gjort en kvalitativ studie, der jeg har analysert elevenes kommunikasjon, og gjennom dette undersøkt elevenes resonnement situert i en problemløsningskontekst der computational thinking og programmering har stått i fokus. Formålet med studien var å undersøke elevers bruk av computational thinking og kreative resonnement i problemløsning der programmering blir brukt som verktøy.

1.2 Oppgavens struktur

Oppgaven er bygd opp av fem kapitler. I første kapittel ble studiens tema aktualisert ved å peke på samfunnets digitale og teknologiske utvikling. Videre ble valg av tema og problemstilling begrunnet. Dette ble begrunnet gjennom kunnskapsløftet 2020 og utdanningssektorens behov for utvikling, samt egen interesse og motivasjon for temaet.

I oppgavens andre kapittel er studiens teoretiske grunnlag presentert. Relevante begreper og definisjoner blir redegjort. Videre presenteres studiens teoretiske rammeverk. Dette er satt samme av Lithner (2006) teoretiske perspektiver på imitativ og kreativ resonnement, og Shute et al. (2017) sine komponenter av computational thinking. Teoriene blir presentert hver for seg før de senere blir satt sammen og til slutt utgjør studiens sammensatte rammeverk. Avslutningsvis presenteres tidligere forskning på sammenhengen mellom matematisk kreativitet og computational thinking.

I kapittel tre blir studiens metodiske tilnærming begrunnet og redegjort. Først presenteres studiens vitenskapssyn. Deretter presenteres prosjektets gjennomførelse gjennom beskrivelse av utvalg, utforming av problem og innsamling av data. Valg som er tatt i tilknytning dette vil presenteres og begrunnes. Videre vil analyseverktøy, transkripsjon og koder benyttet i studien forklares. Kapittelet avrundes med en drøfting omkring studiens etiske aspekter.

Det fjerde kapittelet omhandler studiens analyse. Her blir studiens resultater presentert både kvantitativt og kvalitativt. Den kvantitative presentasjon inneholder tabeller og grafer som er utarbeidet etter antall koder i hver komponent av studiens sammensatte rammeverk. Den kvalitative delen inneholder utdrag fra transkripsjonen. Utdragene analyseres med hensyn på studiens sammensatte rammeverk.

I oppgavens siste kapittel diskuterer jeg funnene fra analysen. Funnene blir drøftet med bakgrunn i studiens sammensatte rammeverk og diskutert opp mot tidligere forskning. Avslutningsvis følger en kritisk refleksjon av studien og tanker om videre forskning.

2 Teori

I dette kapittelet presenteres relevant teori for studiens problemstilling. Begrepene *Computational Thinking* (CT), *programmering* og *matematisk kreativitet* vil bli definert og tydeliggjort. Oppgaven vil klargjøre forskjellen og sammenhengen mellom CT og programmering. Videre vil oppgaven ta for seg hvordan man observerer matematisk kreativitet. Det vurderes som nødvendig å presentere tidligere forskning på sammenhengen mellom kreativitet og CT. Avslutningsvis vil studiens sammensatte teoretiske rammeverk presenteres.

2.1 Definisjon av computational thinking

Det har lenge vært problemer og misforståelser rundt begrepet CT. Årsaken bak dette er mangelen på en entydig og felles akseptert definisjon (Barr et al., 2011, s. 20; Barr & Stephenson, 2011, s. 112; Grover & Pea, 2013, s. 1). I Norden har det også vært definisjonsproblemer rundt begrepet (Kilhamn et al., 2021, s. 284; Stenseth et al., 2019, s. 8). I Norge oversettes CT til *algoritmisk tenkning* (Stenseth et al., 2019, s. 8; Utdanningsdirektoratet, 2019), mens i Sverige brukes begrepet *datalogisk tänkeande (Kilhamn et al., 2021, s. 284)*. Den norske oversettelsen er noe problematisk ettersom begrepet kan forveksles med det engelske begrepet *Algorithmic thinking*, som innebærer evnen til å arbeide med algoritmer og lage stegvise instruksjoner for å løse et bestemt problem (Kaufmann & Stenseth, 2021, s. 1030). CT innebærer mer enn kun evnen til å tenke algoritmisk (Shute et al., 2017, s. 146). For å unngå og bidra til flere misforståelser rundt begrepet, vil det engelske begrepet computational thinking brukes i denne oppgaven.

CT stammer tilbake til Seymour Paperts (Papert, 1980) arbeid med programmet Logo (Shute et al., 2017, s. 143), men har gjennomgått en oppstandelse etter Jeannette Wings (Wing, 2006) innflytelsesrike artikkel (Barr et al., 2011, s. 111; Barr & Stephenson, 2011, s. 20; Grover & Pea, 2013, s. 1; Hickmott et al., 2018, s. 48; Lye & Koh, 2014, s. 52; Shute et al., 2017, s. 143). Artikkelen populariserte begrepet CT og skapte en debatt omkring definisjonen. I artikkelen skriver Wing (2006, s. 33) at CT representerer en universell holdning og et ferdighetssett som alle, ikke bare informatikere, burde inneha. Dette støttes av Cuny et al. (2010, s. 3) som skriver at CT er det 21.århundrets nye *literacy*. Også Forsström og Kaufmann (2018, s. 19) påpeker viktigheten av å kunne programmere i det 21.århundret og begrunner dette med samfunnets digitale utvikling og behov for ferdigheter som problemløsning, kreativitet og logisk tenkning. Artikkelen til Wing (2006) manglet en tydelig definisjon på CT. Som svar på spørsmålene omkring en manglende definisjon utarbeidet Wing sammen med Jan Cuny og Larry Snyder en

definisjon på CT (Cuny et al., 2010). Dette er blitt til den mest refererte definisjonen av CT (Shute et al., 2017, s. 143). Cuny et al. (2010) definerer CT som den tankeprosessen som tas i bruk under formulering og løsning av problemer, der løsningen er representert på en måte som effektivt kan utføres av et menneske eller en datamaskin. Med andre ord er CT den mentale aktiviteten som tar sted under formulering og løsning av et problem. Løsningen kan utføres av et menneske eller en datamaskin, eller en kombinasjon av begge.

Utdanningsdirektoratet (2019) bruker begrepet *algoritmisk tenkning* og presiserer at dette er den norske oversettelsen av CT. I likhet med Wing (2006) bruker Utdanningsdirektoratet (2019) strofen *å tenke som en informatiker* i deres beskrivelse av CT, mer presist beskrives CT som en problemløsningsmetode som innebærer en systematisk tilnærming til både formulering og løsning av problemer. Det utdypes at CT handler om å dekomponere problemer til mindre delproblemer som er mer håndterlige. Informasjon skal analyseres og organiseres på en logisk måte, og ved hjelp av algoritmer komme frem til ønsket løsning. Det benyttes modeller og abstraksjoner av den virkelige verden og unødvendige detaljer fjernes. Løsningen kan generaliseres og benyttes til å løse lignende problemer.

Videre listes det opp 6 nøkkelbegreper i CT. Disse er *logikk, algoritmer, dekomposisjon, mønstre, abstraksjon* og *evaluering*, som vist i Figur 1. I denne figuren har Utdanningsdirektoratet (2019) også inkludert fem arbeidsmåter for CT. Disse er: 1) *fikle*, som innebærer utforskning og eksperimentering, 2) *skape*, altså designe og lage, 3) *feilsøke*, oppdage og rette feil, 4) *holde ut*, som innebærer å prøve igjen og igjen, 5) *samarbeide.*

Figur 1: Den algoritmiske tenkeren (Utdanningsdirektoratet, 2019).

Barr og Stephenson (2011, s. 115) viser til en definisjon av CT utarbeidet av *Computer Science Teachers Association* i samarbeid med *Society for Technology in Education.* Her defineres CT som en tilnærming til problemløsning der implementasjon kan utføres av en datamaskin. Konsepter slik som abstraksjon, rekursjon, iterasjon, skape ekte og virtuelle gjenstander, og prosessering og analysering av data er sentrale i deres definisjon av CT. Videre utdypes det at CT er en problemløsningsmetode som kan automatiseres og generaliseres. Det påpekes at brukere av CT ikke bare bruker verktøy, men også skaper verktøy.

Barr et al. (2011) definerer CT på en operativ måte, som innebærer en beskrivelse av komponenter av CT. Definisjonen beskriver CT som en problemløsningsprosess og inkluderer følgende komponenter: Formulere problemer som muliggjør at datamaskiner eller andre verktøy kan brukes som hjelpemidler i problemløsningen. Logisk organisere og analysere data. Representere data ved hjelp av abstraksjoner, slik som modeller og simuleringer. Automatisere løsningen ved hjelp av algoritmer. Identifisere, analysere og implementere mulige løsninger i et forsøk på å effektivisere algoritmen. Til slutt, generalisere problemløsningsprosessen til en mengde problemer.

Shute et al. (2017) inkluderer både en definisjon og et rammeverk for CT. Her defineres CT som det grunnlaget nødvendig for å løse problemer effektiv, med eller uten datamaskiner, med løsninger som kan anvendes i andre sammenhenger. Videre deles CT inn i seks komponenter: *Dekomposisjon, abstraksjon, algoritmer, feilsøking, iterasjon* og *generalisering*, slik vist i

Figur 2. Dekomposisjon handler om å bryte problemet ned i mindre deler som er mer håndterlige. Abstraksjon innebærer å finne essensen av et system, og deles videre inn i tre underkategorier: a) *datainnsamling og analysering*, som innebærer å samle den mest relevante og viktige informasjon, og undersøke sammenhenger mellom datasett, b) *mønstergjenkjenning*, identifisere underliggende mønstre og regler i datasett, c) *modellering,* lage modeller eller simuleringer som representerer et system. Neste komponent er algoritmer som innebærer å designe instruksjoner som gjengir løsningen til et problem. Instruksjonene kan utføres av både et menneske og en datamaskin. Algoritmer deles inn i fire underkategorier: a) *algoritmisk design*, som innebærer å stegvis beskrive løsningen til et problem, b) *parallellisme* er søken etter å utføre flere av stegene samtidig, c) *effektivitet* handler om å fjerne unødvendige steg, og beskrive løsningen ved færrest mulige steg, d) *automasjon*, dette er å automatisere utførelsen av løsningsalgoritmen når nødvendig for å løse lignende problemer. Den fjerde komponenten av CT er feilsøking. Feilsøking er å oppdage og identifisere feil i løsningen, for så å fikse feilen dersom løsningen ikke fungerer. Neste komponent i rammeverket til Shute et al. (2017, s. 153) er iterasjon, som innebærer å repetere designprosessen av løsningen for å optimalisere løsningen. Siste komponent er generalisering, som handler om å overføre CT ferdighetene til et bredt spekter av lignende situasjoner for å kunne løse problemer effektivt.

Facet	Definition
Decomposition	Dissect a complex problem/system into manageable parts. The divided parts are not random pieces, but functional elements that collectively comprise the whole system/problem.
Abstraction	Extract the essence of a (complex) system. Abstraction has three subcategories: (a) *Data collection and analysis*: Collect the most relevant and important information from multiple sources and understand the relationships among multilayered datasets; (b) *Pattern recognition*: Identify patterns/rules underlying the data/information structure; (c) *Modeling*: Build models or simulations to represent how a system operates, and/or how a system will function in the future.
Algorithms	Design logical and ordered instructions for rendering a solution to a problem. The instructions can be carried out by a human or computer. There are four sub-categories: (a) *Algorithm design*: Create a series of ordered steps to solve a problem; (b) *Parallelism*: Carry out a certain number of steps at the same time; (c) *Efficiency*: Design the fewest number of steps to solve a problem, removing redundant and unnecessary steps; (d) *Automation*: Automate the execution of the procedure when required to solve similar problems.
Debugging	Detect and identify errors, and then fix the errors, when a solution does not work as it should.
Iteration	Repeat design processes to refine solutions, until the ideal result is achieved.
Generalization	Transfer CT skills to a wide range of situations/domains to solve problems effectively and efficiently.

Figur 2: Shute et al. (2017) komponenter av computational thinking.

De ovennevnte definisjonene innehar mange likheter. Konsepter som algoritmer, abstraksjon, generalisering og dataanalyse er inkludert i alle definisjonene. I følge Shute et al. (2017, s. 145) er dekomposisjon, abstraksjon, algoritmer og feilsøking de mest brukte komponentene av CT i forskningen. Problemløsning og CT er nært beslektet (Shute et al., 2017, s. 146). I alle de ovennevnte definisjonene er problemløsning et sentralt begrep, og ord som problemløsningsmetode og tankeprosess er gjengangere.

Videre i denne studien er det valgt å bruke definisjon til Shute et al. (2017) og dets sammenhørende rammeverk. Definisjonen vurderes som konkret og tydelig med et tilhørende rammeverk som utdyper definisjonen.

2.2 Sammenhengen mellom programmering og computational thinking

CT blir ofte knyttet til programmering (Shute et al., 2017, s. 143), og i enkelte tilfeller blir begrepene brukt som synonymer. Det anses som nødvendig å definere begrepet programmering, og klargjøre forskjellen og sammenhengen mellom CT og programmering. Forsström og Kaufmann (2018, s. 19) definerer programmering som prosessen knyttet til utvikling og implementering av instruksjoner til et dataprogram, slik at datamaskinen kan løse spesifikke problemer, og støtte menneskelig interaksjon. Balanskat og Engelhardt (2015, s. 7) bruker begrepet *datamaskin programmering* og definerer begrepet som prosessen bak skapning og implementering av stegvise instruksjoner til en datamaskin, slik at datamaskinen kan utføre spesifikke oppgaver og problemer, og støtte menneskelig interaksjon. Hickmott et al. (2018, s. 49) definerer programmering noe mer kortfattet, programmering er å skrive kode som instruerer en datamaskin til å utføre handlinger.

Forsström og Kaufmann (2018) og Balanskat og Engelhardt (2015) definisjon er svært like. Definisjonen til Hickmott et al. (2018) er noe annerledes formulert, men essensen er den samme. De ovennevnte definisjonene beskriver programmering som en handling eller prosess og alle inkluderer skapelse av instruksjoner til en datamaskin. Videre i denne studien er det valgt å bruke definisjonen til Forsström og Kaufmann (2018, s. 19). *Programmering er prosessen knyttet til utvikling og implementering av instruksjoner til et dataprogram, slik at datamaskinen kan løse spesifikke problemer, og støtte menneskelig interaksjon.* Definisjonen vurderes som presis og spesifikk.

Det er viktig å tydeliggjøre forskjellen mellom begrepene CT og programmering, likeså er det viktig å fremheve sammenhengen mellom CT og programmering. Programmeringsferdigheter og CT er nært knyttet (Shute et al., 2017, s. 146). Programmering er en fundamental ferdighet av CT og et viktig verktøy for de kognitive prosessene inkludert i CT (Grover & Pea, 2013, s. 2). Ved å programmere blir man eksponert for CT, og på denne måten kan programmering brukes til å lære CT (Lye & Koh, 2014, s. 52). Det er viktig å poengtere at CT handler om mer enn bare programmering (Shute et al., 2017, s. 143; Wing, 2006, s. 35). Programmering sees på som en ferdighet, mens CT sees på som en tankeprosess eller en problemløsningsmetode. Dette kommer tydelig frem i de ovennevnte definisjonene av CT og programmering. CT er et

bredere og mer omfattende begrep enn programmering. På mange måter er CT et paraplybegrep som blant annet omfatter programmering. En kan også anse programmering som en ferdighet av CT.

Et eksempel på skilnaden mellom CT og programmering er prosjektet *CS unplugged*, som kan oversettes til frakoblet informatikk (https://www.csunplugged.org/en/). I dette prosjektet fra universitetet i Canterburys blir barn undervist i CT helt uten bruk av datamaskiner og programmering.

2.3 Definisjon av matematisk kreativitet

Formålet med denne studien var å undersøke elevenes kreative resonnement i bruken av CT. Dette krever en tydeliggjøring av begrepet *matematisk kreativitet* samt en presentasjon av etablert teori om emnet.

I følge Haylock (1997, s. 68) eksisterer det ingen definisjon av *kreativitet* som er generelt akseptert i forskningen. Spørsmålet om hva kreativitet er, blir ofte ignorert eller besvart på mange ulike måter (Kaufman & Beghetto, 2009, s. 1), som gjør det vanskelig å definere begrepet kreativitet (Sriraman, 2008, s. 14). Ettersom det ikke eksisterer en akseptert definisjon på generell kreativitet, er det ikke overraskende også en mangel på definisjon av matematisk kreativitet (Haylock, 1997, s. 68; Sriraman, 2008, s. 14). Det er riktig nok mange som har forsøkt å definere både generell kreativitet og matematisk kreativitet. Blant annet definerer Sriraman (2008) generell kreativitet som evnen til å produsere nyskapende og originalt arbeid, mens Haylock (1997, s. 68) definerer generell kreativitet som et bredt spekter av kognitive aspekter som kategoriseres etter ytelse og utfall. Sriraman (2008, s. 15) har også forsøkt å definere matematisk kreativitet, der matematisk kreativitet defineres som prosessen som resulterer i originale og innsiktsfulle løsninger på gitte problem, uavhengig av det matematiske nivået.

Kaufman og Beghetto (2009) definerer ikke kreativitet, men introduserer et rammeverk for klassifisering av ulike grad av kreativitet. Rammeverket heter *The Four C Model* og skiller mellom fire typer kreativitet: *mini-c, little-c, pro-c* og *big-c*. Storartet kreativitet faller inn under kategorien big-c. Dette er kreativitet fra mennesker som betraktes som genier, for eksempel kreativiteten som Albert Einstein viste i utviklingen av relativitetsteorien. Kreative verk som anses som big-c er relevant i flere tiår. Arbeid som ikke er revolusjonerende og imponerende nok til å kategoriseres som big-c, men likevel anses som svært kreativt i sitt respektive yrke eller fagfelt, kategoriseres som pro-c. Denne formen for kreativitet stammer fra mennesker som

anses som eksperter i sitt fagfelt eller yrke, som for eksempel en ledende matematikkprofessor. Et viktig aspekt som skiller pro-c fra big-c er tid. En oppdagelse eller oppfinnelse gjort i dag kan anses som revolusjonerende, men vise seg å være uinteressant for historiebøker. For eksempel er det vanskelig å avgjøre om Andre Wiles løsning av Fermats sats tilhører big-c eller pro-c. Det er også mulig at arbeid anses som irrelevant i dag, men som i senere tid viser seg å være banebrytende, for eksempel Nicolaus Copernicus arbeid med det heliosentriske verdensbildet som ikke ble anerkjent før etter hans død.

Neste grad av kreativitet er kreative handlinger utført av ikke-profesjonelle, såkalt hverdagskreativitet, og kalles for little-c. Dette er handlinger i hverdagen som anses som kreative, men som ikke kan kategoriseres som pro-c. Slike handlinger kan være å tilføre en ekstra ingrediens til middagen for og lage en kreativ vri. Siste grad er mini-c, som fokuserer på at handlingen er ny og kreativ for personen som utfører handlingen. Et barn som oppdager at multiplikasjon er kommutativt kan oppleves som svært kreativt av barnet selv, men er ikke en nyhet for folk flest. Uten mini-c kategorien ville slik form for kreativitet blitt neglisjert, ettersom dette ikke ville blitt kategorisert som little-c. Ved hjelp av et slikt nyansert rammeverk kan lærere og foreldre vurdere elevers kreative potensial og identifisere elever med svært høy kreativt potensial samtidig som elever med mindre kreativt potensial får annerkjennelse for deres kreativitet.

Av de ovennevnte definisjonene vurderes Sriraman (2008) definisjon av matematisk kreativitet som mest passende for denne studiene. Denne definisjonen omhandler spesifikt matematikk, og poengterer at matematisk kreativitet er uavhengig av det matematiske nivået. Ved denne definisjon kan en betrakte løsninger på matematikkproblemer i skolen som kreative.

2.4 Observasjon av matematisk kreativitet

Det eksisterer ulike rammeverk for observasjon av matematisk kreativitet i skolen. I følgende delkapittel ønsker jeg å presentere rammeverket til Haylock (1997) og Lithner (2006).

Haylock (1997) skiller mellom to måter å identifisere matematisk kreativitet i skolen. Den første innebærer å studere informantens kognitive prosesser under problemløsningssituasjoner og vurdere om de kognitive prosessene kan anses som kreative. En viktig kognitiv prosess i matematisk problemløsning er å tenke fleksibelt og unngå fiksering på en bestemt løsningsstrategi. Eleven kan for eksempel fiksere på en algoritme som tidligere har fungert, eller på et spesifikt tema i matematikk som eleven begrenser seg til på bakgrunn av problemets overflateegenskaper. En kognitiv prosess der informanten unngår fiksering, anses som kreativ.

Haylock (1997) andre tilnærming til identifisering av matematisk kreativitet handler om å undersøke et produkt laget av eleven, og vurdere ut fra bestemte kriterier om kreativ tenkning har tatt sted i utformingen av produktet. Ved hjelp av *divergerende produksjonsoppgaver* kan produktet vurderes etter kriterier slik som originalitet, fleksibilitet og hvorvidt løsningen er passende. Et eksempel på en divergerende produksjonsoppgave er å lage flest mulige figurer med areal på to kvadratcentimeter i form av å tegne rette linjer mellom punkter på et centimeter rutenett med 9 punkter. Videre vil svarene fra informanten bli vurdert etter hvor mange forskjellige løsninger informanten fant, hvor mange av disse løsningene anses som korrekt og hvor originale løsningene var sammenlignet med resten av informantene.

Lithner (2006) har utviklet et rammeverk for å analysere *kreativ og imitativ resonnering* i problemløsningssituasjoner. Rammeverket er forankret i empirisk data og delvis basert på det ovennevnte rammeverket til Haylock (1997). I rammeverket definerer Lithner (2006, s. 4) resonnering som den tekningen som produserer påstander og konklusjoner. Resonneringen anses som kreativ dersom følgende fire kriterier er oppfylt:

1) *Originalitet*, en ny (for eleven) sekvens av resonnering blir skapt, eller en glemt sekvens blir gjenskapt.
2) *Fleksibilitet*, eleven unngår fiksering.
3) *Plausibilitet*, det eksisterer argumenter for valg og implementasjon av strategi som støtter konklusjonen.
4) *Matematisk fundament*, argumentasjonen tar utgangspunkt i elevens matematiske egenskaper.

Resonnering som ikke kan anses som kreativ, blir kalt *imitativ resonnering*. Mer spesifikt kategoriseres resonneringen som imitativ dersom den tar utgangspunkt i tidligere erfaringer og er blottet for innslag av kreativitet. Resonnering som innebærer å følge eller kopiere et eksempel eller en modell, uten noen form for kreativitet, anses som imitativ. Videre deles imitativ resonnering inn i *memorert resonnering* og *algoritmisk resonnering*. For *memorert resonnering* må følgende kriterier oppfylles:

1) Strategivalget er å memorere et komplett svar.
2) Strategiimplementasjon innebærer kun å reprodusere det memorerte svaret.

Løsningen er memorert i den minste detalj. Denne typen strategivalg er mulig i enkle oppgaver, slik som omgjøring av måleenheter, men straks oppgavene blir mer avansert vil et slikt strategivalg være lite effektivt. Det kan da være mer passende å memorere en løsningsalgoritme. Dette kalles for *algoritmisk resonnering*. Følgende kriterier må være oppfylt for å kategorisere resonneringen som *algoritmisk*:

1) Strategivalget er basert på å huske et sett med regler og/eller instruksjoner som fører til riktig løsning.
2) Strategiimplementasjon innebærer kun å implementere reglene eller utføre instruksjonene.

Den eneste utfordringen i algoritmisk resonnering innebærer å identifisere riktig løsningsalgoritme, resterende resonnering er triviell. Videre deles algoritmisk resonnering inn i tre undergrupper, slik vist i Figur 3. Skilnaden mellom undergruppene skyldes forskjellige strategivalg for å identifisere riktig algoritme. *Kjent algoritmisk/memorert resonnering* kjennetegnes ut fra følgende kriterier:

1) Strategivalget innebærer å identifisere og kategorisere problemet til en bestemt gruppe problemer, og benytte at gruppen har en tilhørende kjent løsningsalgoritme eller et memorert svar.
2) Løsningsalgoritmen implementeres eller det memorerte svaret blir skrevet ned.

Dersom problemet ikke er tilstrekkelig kjent og eleven kjenner for mange algoritmer til å implementere alle, kan strategivalget basere seg på å avgrense antall algoritmer. Dette kalles *avgrenset algoritmisk resonnering* dersom følgende kriterier er oppfylt:

1) Strategivalget innebærer å avgrense antall algoritmer gjennom overflatelikhetene mellom algoritmene og problemet. Deretter blir en av de avgrenset algoritmene valgt.
2) Algoritmen implementeres.

Eleven kan ikke forutsi resultatet av implementasjonen. Dersom resultatet ikke fremstår rimelig for eleven, vil algoritmen bli skrotet uten noen form for evaluering. Vanligvis blir kun én algoritme forsøkt. Dette fordi eleven enten oppnår et akseptabelt resultat ved første algoritme eller fordi antall kjente algoritmer er så begrenset at eleven kun kjenner til én algoritme som kan implementeres.

Dersom hverken kjent eller avgrenset algoritmisk resonnering fører frem, søker eleven veiledning fra en person eller fra tekstlige kilder. Denne formen for strategivalg kategoriseres som *veiledet algoritmisk resonnering*, og deles videre inn i *tekstveiledet* og *personveiledet algoritmisk resonnering*. Resonneringen anses som tekstveiledet dersom følgende kriterier er oppfylt:

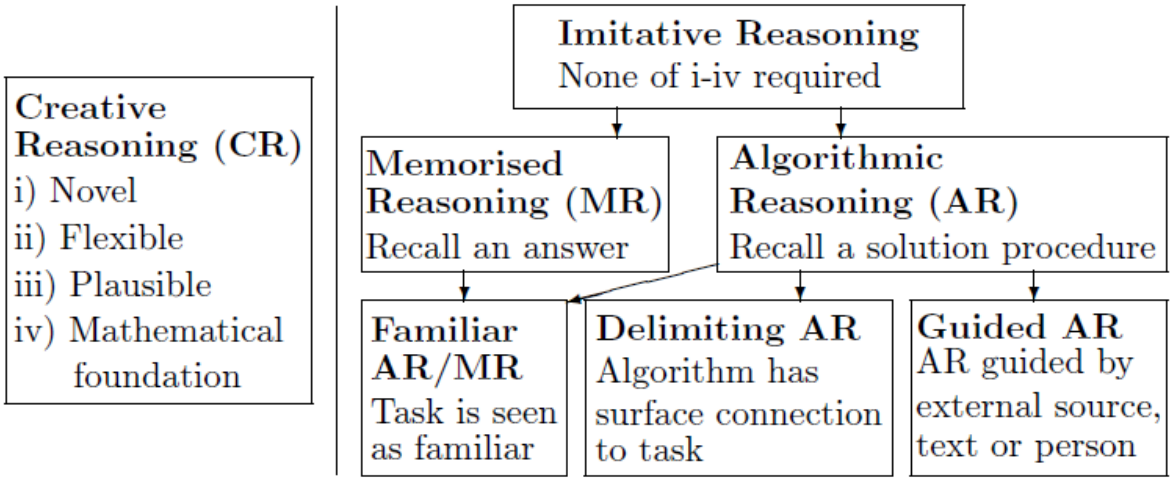
1) Strategivalget baserer seg på å sammenligne overflatelikheter mellom problemet og eksempler, teoremer, definisjoner, regler eller lignende tekstlige kilder.
2) Implementasjon innebærer å kopiere den tekstlige kilden.

Personveiledet algoritmisk resonnering er når:

1) Strategivalgene blir bestemt av en veileder.

2) Implementasjon innebærer å følge veiledning og utføre resterende rutinemessige kalkulasjoner eller operasjoner.

Lithner (2006, s. 21) poengterer at rammeverket kun inkluderer de mest vanlige formene for resonnering, ifølge empiriske undersøkelser.



Figur 3: Lithner (2006) skiller mellom kreativ og imitativ resonnering.

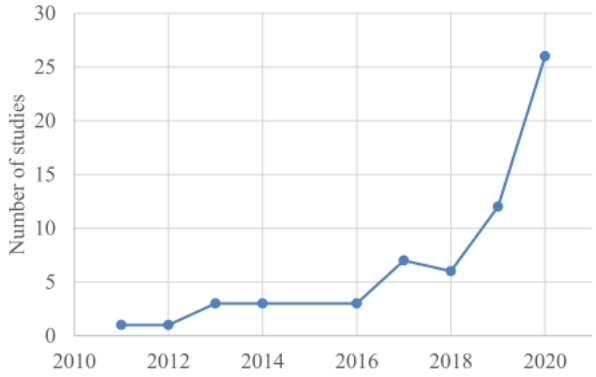
Både Haylock (1997) og Lithner (2006) vurderer matematisk kreativitet som en kognitiv prosess der fleksibelt tenkning er sentralt, og motpolen til dette er fiksering på algoritmer. Divergerende produksjonsoppgaver, som nevnt av Haylock (1997) ble vurdert som vanskelig å gjennomføre i denne studien grunnet mangelen på slike oppgaver med sammenheng til CT. Lithner (2006) sitt rammeverk ble vurdert som mer passende for denne studien, mye grunnet rammeverkets spesifisitet. Videre i denne studien vil rammeverket til Lithner (2006) brukes for å undersøke elevenes kreative resonnement i problemløsningssituasjoner.

2.5 Tidligere forskning på sammenhengen mellom kreativitet og computational thinking

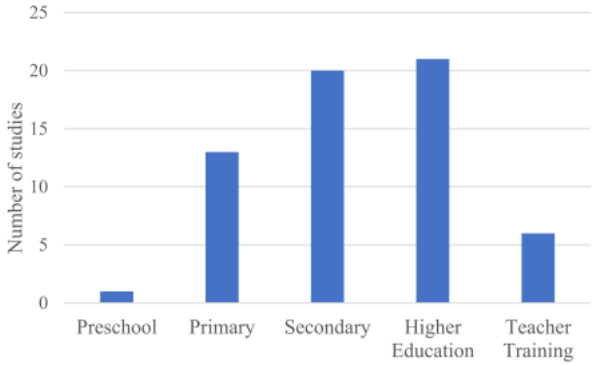
For å svare på studiens problemstilling anses det som relevant å undersøke eksisterende forskning på området. På denne måten dannes et inntrykk av hvilken forskning som er gjort og hvilken kunnskap som er opparbeidet.

Israel-Fishelson og Hershkovitz (2022) har gjennomført en litteraturstudie på 62 empiriske undersøkelser fra 2011-2022 som studerer både CT og kreativitet. Litteraturstudien viser at 61% av publikasjonene er publisert i årene 2019 og 2020, slik vist i Figur 4. Av de 62 undersøkelsene var 53 utført i en kontekst som gjorde det mulig å skille studiene inn etter

fagområder og utdanningsnivå. Så mange som 70% av studiene ble utført i en *STEM*-kontekst. Figur 5 viser distribusjonen av studiene etter utdanningsnivå. Figuren viser tydelig at flesteparten av studiene er utført i ungdomsskole, videregåendeskole og i høyere utdanning. Litteraturstudien viser at forskning på sammenhengen mellom kreativitet og CT er mangelfull, men at forskningen vokser nærmes eksponentiell. Også Israel-Fishelson et al. (2021, s. 92) peker på mangelfull forskning innen dette området. Før 2012 var forskningen hovedsakelig viet til definisjonsproblemet rundt CT og utvikling av programmer for å utvikle CT, f.eks. programmet Scratch (Grover & Pea, 2013, s. 5).



Figur 4: Distribusjon av publikasjoner fra 2011 til 2020 som inneholder både kreativitet og CT (Israel-Fishelson & Hershkovitz, 2022).



Figur 5: Distribusjon av studier om kreativitet og CT etter utdanningsnivå (Israel-Fishelson & Hershkovitz, 2022).

Til tross for begrenset forskning hevder flere at det eksisterer en assosiasjon mellom kreativitet og CT eller programmering (European Commission, 2018; Grover & Pea, 2013, s. 19; Israel-Fishelson & Hershkovitz, 2022, s. 628; Israel-Fishelson et al., 2021, s. 926; Shell et al., 2014, s. 1). Shell et al. (2013) gjennomførte en empirisk undersøkelse i informatikkemner på universitetsnivå, som undersøkte hvordan studenters kreativitet, motivasjon og selvregulering

påvirker karakterer og læring av CT. Funnene tyder på at kreativitet kan øke forståelsen av CT. Denne påstanden ble også bekreftet i et senere studie utført av Shell et al. (2014) som peker videre på viktigheten av å kombinere kreativitet og CT i STEM undervisning.

Hershkovitz et al. (2019) undersøkte sammenhengen mellom kreativitet og CT gjennom en empirisk undersøkelse av 57 elever på barnetrinnet. Studiens funn antyder en assosiasjon mellom CT og kreativitet i CT. Studien fant ingen sammenheng mellom generell kreativitet og CT, men antyder at det i noen tilfeller eksisterer en positiv assosiasjon mellom kreativitet i programmering og generell kreativitet. En studie gjort av Seo og Kim (2016) undersøkte om programmering i par kan forbedre CT og kreativiteten til elever på barnetrinnet. Antall informanter var 162. Resultatene tyder på at kreativitet kan økes ved hjelp av undervisning i CT. Knobelsdorf og Romeike (2008) gjorde en studie der de undersøkte om kreativitet kan være en inngang til informatikk. De analyserte 135 biografier fra informatikkstudenter som skulle beskrive deres interesse for informatikk og hvorfor de valgte utdanningen. Studentene var i startfasen av utdanningen, og resultatene viste at elevene som valgte å studere informatikk hadde flere karakteristiske trekk for kreativitet.

Flere av de ovennevnte studiene påpeker den begrensede mengden forskning gjort på CT (Forsström & Kaufmann, 2018, s. 28; Grover & Pea, 2013, s. 5). Som et resultat av dette eksisterer det også lite forskning på sammenhengen mellom kreativitet og CT (Israel-Fishelson & Hershkovitz, 2022, s. 11; Israel-Fishelson et al., 2021, s. 949). Noe av forskningen som er gjort viser til en positiv assosiasjon mellom CT og kreativitet (European Commission, 2018; Grover & Pea, 2013; Hershkovitz et al., 2019; Israel-Fishelson & Hershkovitz, 2022; Israel-Fishelson et al., 2021; Knobelsdorf & Romeike, 2008; Seo & Kim, 2016; Shell et al., 2014; Shell et al., 2013), men forskningen er mangelfull og på generelt grunnlag kan man ikke trekke noen konklusjoner om sammenhengen mellom CT og kreativitet. Forskningen på dette området er raskt økende (Israel-Fishelson & Hershkovitz, 2022), og det kan forventes mange publikasjoner i fremtiden.

2.6 Studiens sammensatte teoretisk rammeverk

Denne studien ønsker å studere elevers bruk av CT og kreative resonnement i problemløsning med programmering som verktøy. Rammeverket til Shute et al. (2017) tar for seg ulike komponenter av CT, mens rammeverket til Lithner (2006) kategoriserer resonnering som imitativ eller kreativ. Ved å kombinere disse tar studien for seg et sammensatt rammeverk som kan benyttes til å analysere elevers bruk av CT og vurdere hvorvidt resonneringen som førte til

CT var kreativ eller imitativ. Ved hjelp av det sammensatte rammeverket kan man først analysere om CT har vært brukt i problemløsningen og hvilken komponent av CT. Videre kan det analysere om bruken av CT stammer fra et kreativt eller imitativt resonnement. Dersom bruken av CT stammer fra et kreativt resonnement kategoriseres resonnementet som kreativ CT. Dersom dette ikke er tilfellet kategoriseres resonnementet som imitativ CT. Gjennom bruk av rammeverket kan man undersøke hvilke komponenter av CT den kreative resonneringen forekommer oftest.

3 Metode

Dette kapittelet tar for seg studiens metodiske tilnærming. Gjennom kapittelet redegjør jeg for valg og vurderinger som er tatt i forbindelse utforming av metode. Først presenterer jeg studiens vitenskapssyn. Videre vil gjennomføring av prosjektet beskrives. Her vil valg av informanter og matematisk problem begrunnes. I delkapittelet 3.3 vil utførelse av datainnsamling bli beskrevet. Deretter vil transkriberingsprosessen bli presentert, der blant annet bruk av koder vil forklares. Valg tilknyttet studiens etiske aspekter blir beskrevet og begrunnet i delkapittel 3.5. Avslutningsvis vil studiens relabilitet, validitet og generaliserbarhet bli vurdert.

3.1 Metodisk tilnærming

Jeg har i hovedsak valgt en kvalitativ tilnærming for å svare på studiens problemstilling. Jeg ønsker å undersøke elevenes resonnement, og kvalitative metoder egner seg godt for å få kunnskap om hvordan enkeltpersoner resonnerer og reflekterer (Thagaard, 2009, s. 12).

3.1.1 Fenomenologisk og hermeneutisk vitenskapssyn

Denne studien har elementer av både *fenomenologisk* og *hermeneutisk* vitenskapssyn. Fenomenologi handler om subjektets opplevelse og søken om forståelse av en dypere mening i enkeltindividets erfaringer (Thagaard, 2009, s. 38). I fenomenologi er realiteten slik subjektet opplever den. På denne måten blir den ytre omverden satt i skyggen av fenomenverden slik subjektet opplever den. Hermeneutikk har en sentral rolle for samfunnsvitenskapelige tilnærminger der fortolkninger tar sted. Hermeneutikk innebærer å tolke subjektets handlinger ved å søke etter kunnskap om de dypere meningene bak handlingene. Målet er å oppnå en gyldig forståelse av subjektets meninger (Thagaard, 2009, s. 39)

Det er nødvendig å analysere subjektets resonnement for å svare på studiens problemstilling. Ved et fenomenologisk vitenskapssyn kan studiens analyse samsvare med subjektets opplevelse av realiteten. For å oppnå dette må forskeren være åpen for informantenes erfaringer. Denne studien vil også analysere subjektets handlinger i søken etter resonnementet bak handlingen. På denne måten kan jeg utvikle en dypere og gyldig forståelse av meningen bak elevens handlinger. Et slikt vitenskapssyn er å anse som hermeneutisk.

3.1.2 Deltakende observasjon

Deltakende observasjon er en metode som brukes innenfor kvalitativ forskning. Deltakende observasjon er når forskeren befinner seg sammen med informantene og systematisk kommuniserer med informantene (Thagaard, 2009, s. 70). Under deltakende observasjon

forsøker forskeren å få en innsikt i informantenes situasjon, og samhandler med informantene for å få deres tolkning av situasjonen. I en skolesituasjon egner deltakende observasjon seg særlig godt, dette fordi det allerede eksisterer en etablert rolle som forskeren kan gå inn i, nemlig lærerrollen (Thagaard, 2009, s. 71). Jeg valgte å tre inn i lærerrollen ved å først avholde en ordinær undervisningstime før jeg uken etter begynte med datainnsamling. Fordelen ved dette er at lærerrollen er kjent for informantene, og kan derfor være meningsfylt for dem. På denne måten kan man unngå at mitt nærvær skaper ubehag og forvirring. Ved bruk av rolleplassering kan det oppstå konflikt med forskeren eget ønske om å observere (Gleiss & Sæther, 2021, s. 107). I et forsøk på å forhindre dette ble graden av deltakelse grundig vurdert. Det var viktig at graden av deltakelse ikke overgikk graden av observasjon.

Deltakende observasjon kategoriseres etter forskerens grad av deltakelse. I denne studien tok jeg rollen som *observatør-som-deltaker.* Dette er en rolle der forskeren er tilbaketrukken og deltakelsen er mindre omfattende (Cohen et al., 2018, s. 552). Når det gjelder forskerens påvirkning av situasjonen, er det vanskelig å unngå noen grad av påvirkning. Forskerens deltakelse påvirker situasjon gjennom fysisk tilstedeværelse, men situasjon blir også påvirket gjennom forskerens utforming av oppgave. Spørsmålet blir dermed ikke hvorvidt situasjon er påvirket, men om hvordan og til hvilken grad. En kan hevde at den ideelle forskeren påvirker situasjonen minst mulig, men grad av deltakelse avgjør ikke forskningens troverdighet, den skaper tilgang til ulike data (Gleiss & Sæther, 2021, s. 111-114). Dette var noe som ble reflektert over og grundig vurdert før utførelsen av datainnsamlingen.

I deltakende observasjon kan det være strategisk å alliere seg med personer som er sentrale i miljøet som skal observeres. En slik person blir ofte betegnes som *portvakt,* ettersom de representerer en inngang til miljøet. En portvakt kan inneha viktig kompetanse og informasjon om miljøet, og dermed være en stor ressurs for forskeren (Thagaard, 2009, s. 67). Ettersom jeg skulle studerer en klasse var det nødvendig å involvere klassens lærer. Videre i forskningen ble læreren bevisst brukt som en ressurs. Lærerens relasjon til klassen og læreres informasjon om elevenes kunnskapsnivå var viktige ressurser som ble benyttet.

Valget av deltakende observasjon ble sett på som nødvendig ettersom jeg ønsket å ha muligheten til å assistere elevene i problemløsningen samt besvare eventuelle spørsmål. Det ble også vurdert som hensiktsmessig å ha muligheten til å påminne elevene om å diskutere og fortelle underveis hva de tenkte på. På denne måten skapes gode arbeidsvilkår for å studere elevenes resonnement.

3.2 Gjennomføring av prosjektet

3.2.1 Utvalg

Informantene er fra en matematikklasse der faget som ble undervist var matematikk R1. Klassen ble valgt gjennom min relasjon til skolen, dog var dette en klasse jeg ikke hadde kjennskap til fra tidligere. Videre ble klassen delt inn i par, og tre par ble utpekt til å delta i forsøket. Observasjonsstudier er tidskrevende arbeid og produserer store mengder data (Tjora, 2010, s. 34), dermed ble det sett på som nødvendig å begrense antall par til tre. I kvalitativ forskning ønsker en å gå i dybden, noe som er tidskrevende. Dermed er det vanlig med relativt få antall strategisk utvalgte i kvalitativ forskning.

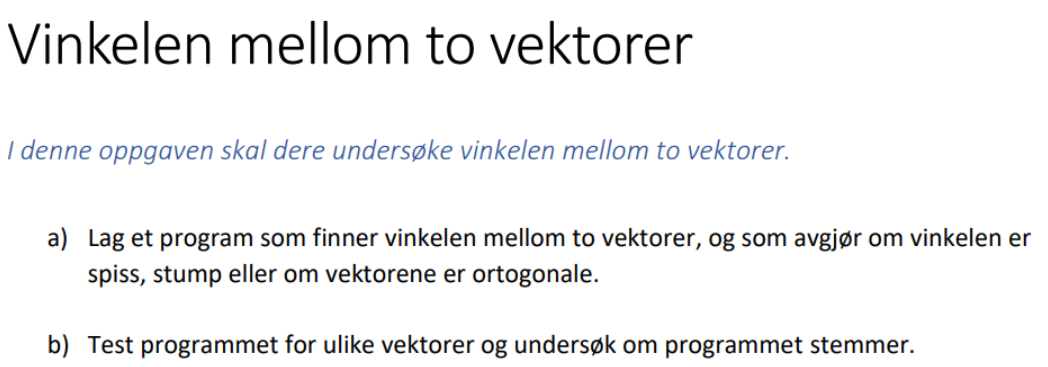
Utvalg der informantene ikke er tilfeldig valgt ut, og forskningens grad av generalisering er lav kalles ofte for ikke-sannsynlighetsutvalg (Gleiss & Sæther, 2021, s. 39). Videre skiller man mellom to måter å begrense utvalget på, nemlig casestudier og kriterieutvalg. I casestudier begrenses utvalget ved å bruke én eller flere caser, mens i kriterieutvalg inviteres informanter til å delta i undersøkelsen på bakgrunn av bestemte kriterier. Kriterieutvalg er egnet der man ønsker å studere informantenes erfaring, problemer, opplevelser og lignende. (Tjora, 2010, s. 34). I denne studien var det nødvendig at informantene hadde ønskede egenskaper eller kvalifikasjoner som var strategisk gunstig for studiens problemstilling og de teoretiske perspektivene som skulle anvendes (Thagaard, 2009, s. 55). Kriterier som var nødvendig for å delta og gjennomføre denne studien var knyttet til elevenes kunnskap i programmering og matematikk. Når det gjelder programmeringskompetanse var elevene nødt til å være kjent med hvordan variabler defineres, hvordan importere og bruke funksjoner fra bibliotek og hvordan man bruker operasjonene addisjon, subtraksjon, multiplikasjon og divisjon. Videre ble kjentskap til if-else, input-funksjonen og løkker sett på som gunstig kompetanse, men var ikke et kriteria. De matematikkfaglige kriteriene må sees i lys av problemet elevene skulle løse. Problemet ble delvis utformet etter pensum i elevenes matematikkfag, som beskrives i detalj senere i oppgaven. På bakgrunn av dette ble det satt et kriteria at elevene måtte være på et matematikkfaglig nivå som tilsvarer bestått i faget Matematikk R1.

Læreren brukt som portvakt og ressurs, og fungerte som en døråpner inn til informantene og gjorde rekrutteringsprosessen smidigere. Lærerens kunnskap og kjennskap til klassemiljøet ble brukt som en ressurs, og parene ble dannet basert på lærerens kjennskap om relasjonene mellom elevene. I en klasse eksisterer det et nettverk av relasjoner (Lyngsnes & Rismark, 1999, s. 114-115). Faktorer som motivasjon for oppgaven eller forventninger om hvordan arbeidet med oppgaven skal foregå har betydning for elevenes arbeidsinnsats og læringsutbytte. For å sikre

best mulig arbeidsinnsats ble det vurdert som hensiktsmessig å søke råd hos læreren prosessen rundt inndeling av par. Forhåpentligvis ville en god relasjon mellom elevene føre til en diskusjon der begge partene var trygge på hverandre.

3.2.2 Utforming av problem

Problemet elevene fikk utdelt er presentert i Figur 6. Elevene ble utfordret til å lage et program som avgjør hvorvidt vinkelen mellom to todimensjonale vektorer er spiss, stump eller rettvinklet. Videre skulle elevene teste programmet for ulike vektorer. Problemet ble utformet på bakgrunn av en rekke parametere som blir presentert i denne oppgaven.


Vinkelen mellom to vektorer

I denne oppgaven skal dere undersøke vinkelen mellom to vektorer.

a) Lag et program som finner vinkelen mellom to vektorer, og som avgjør om vinkelen er spiss, stump eller om vektorene er ortogonale.

b) Test programmet for ulike vektorer og undersøk om programmet stemmer.

Figur 6: Problemet elevene fikk utdelt.

Det ble brukt mye tid på utforming av problem. Det var ønskelig at problemet kunne løses på flere måter og gi rom for kreative resonnementer. Problemet skulle, til en viss grad, være ukjent for elevene. Dette for å hindre at problemløsningen kun skulle innebære implementering av en tillært løsningsalgoritme, likevel skulle det ikke være nødvendig for elevene å bruke matematikk de ikke hadde blitt introdusert for.

Samtidig var det essensielt for studiens problemstilling at oppgaven skulle løses ved hjelp av tekstbasert programmering. I likhet med problemets matematiske side skulle det ikke være nødvendig for eleven å ta i bruk ukjente programmeringsfunksjoner og operasjoner, men det var ønskelig at problemet skulle utfordre elevene til å implementere kjente programmeringsfunksjoner og operasjoner på en ukjent måte. For eksempel at rekkefølgen på utførelsen av funksjonene og operasjonene er av en ukjent karakter.

Det var ønskelig at kun problemets karakter var ukjent for elevene. Dette for å forhindre at det oppstår for mange ukjente momenter for elevene i læringssituasjonen. For mange ukjente momenter kan øke problemets vanskelighetsgrad og derav resultere i mangelfull datagrunnlag. Vanskelighetsgraden på problemet ble vurdert som en avgjørende faktor for kvaliteten på

datainnsamlingen. Et for vanskelig problem kan utfordre elevenes utholdenhet og resultere i elever som gir opp etter en kort stund, noe som fører til en begrenset datamengde. Dersom problemet er for lite utfordrende kan det oppstå lite diskusjon i problemløsningssekvensen, som også vil resulterer i en begrenset datamengde. Med fokus på dette, ble problemets vanskelighetsgrad drøftet med klassens matematikklærer og sammenlignet med lignende problemer i diverse lærebøker.

Tidsrammen elevene fikk på å løse problemet ble sett i sammenheng med vanskelighetsgraden og rådført med matematikklæreren. En tidsramme på 45 minutter ble vurdert som tilstrekkelig for å løse problemet, og nok tid til å unngå et tidspress.

3.3 Utførelse av datainnsamlingen

Datainnsamlingen ble utført over tre dager der hver gruppe utførte prosjektet på forskjellige dager. Det ble kommunisert til elevene at problemet ikke skulle diskuteres med grupper som ikke hadde arbeidet med problemet. Problemløsningsprosessen og resonnementet starter i det øyeblikket elevene mottar informasjon om problemet. Dersom elevene hadde blitt informert om problemet på forhånd av datainnsamlingen ville datamaterialet vært foruten startfasen av elevenes resonnement. En tillitbasert tilnærming til dette var mest naturlig, og jeg har ingen indikasjoner som tyder på at elevene har snakket om problemet med andre grupper.

For å samle inn datamateriale ble det valgt en kombinasjon mellom notater, lydopptak og skjermopptak. Ved lydopptak blir alt som sies bevart. Dette muliggjør et større fokus på veiledning. I tillegg til at dette gir et godt datamateriale, er fremgangsmåten oversiktlig og enkel å håndtere i etterkant. En kan kritisere lydopptak for å påvirke informantene ettersom alt de sier blir tatt opp, men ifølge Thagaard (2009, s. 102) tenderer denne distraksjon å forsvinne under problemløsningsprosesser. Dette er også noe som oppgavens data indikerer.

Lydopptak egner seg dårlig til å samle informasjon om informantenes kroppslige bevegelser. For å inkludere dette i datamaterialet ble det valgt å inkludere notater som datainnsamlingsmetode. Ettersom lydopptak fanget elevenes diskusjon ble det ansett som kun nødvendig å notere elevenes fysiske handlinger, slik som bruk av lærebok, skrivebok, mobil og så videre. På samme måte som for lydopptak kan en hevde at notatføringen virker forstyrrende på informantene (Thagaard, 2009, s. 102). For å minimere forstyrrelsen valgte jeg å plassere meg bak elevene, utenfor deres synsfelt, men jeg var likevel tilgjengelig for veiledning. I tillegg ble forstyrrelsen redusert ettersom noteringen var redusert som følge av lydopptaket. På grunn av redusert notering kunne jeg også øke min personlige kontakt og sosiale interaksjon med

informantene, noe som tillot meg i større grad å fokusere på veiledningen. En svakhet med notater som innsamlingsmetode er at forskeren er nødt til å analysere underveis i noteringen (Thagaard, 2009, s. 102). Analyse i den forstand at forskeren må vurdere hva som er hensiktsmessig å notere.

Ettersom problemet skulle løses ved hjelp av tekstbasert programmering på datamaskin ble det vurdert som nødvendig å inkludere skjermopptak av elevenes programmering. Ved å inkludere skjermopptak ble elevens syntaksfeil, input, output, feilkoder og så videre tilgjengeliggjort for analysering. Dette komplimenterer dataen fra lydopptak og notater, og muliggjør en mer dynamisk og nyansert analyse ettersom en kan følge elevenes tankeprosess mens de programmerer.

For å forsikre at datamaterialet ble så detaljert som mulig ga jeg elevene, i forkant av problemløsningen, følgende monolog anbefalt av Ericsson og Simon (1993, s. 378):

> *Fortell meg ALT dere tenker på fra første øyekast på problemet til dere har funnet en løsning. Jeg ønsker at dere snakker høyt HELE TIDEN. Jeg vil IKKE at dere skal forsøke å planlegge hva dere skal si eller prøver å forklare meg hva dere mener. Bare lat som dere sitter helt alene. Det er viktig at dere heletiden snakker høyt. Hvis dere blir stille for en lengere periode vil jeg be dere snakke.*

I tillegg til monologen forsøkte jeg å predikere hvilken spørsmål elevene kom til å spørre og hvilke utfordringer som kunne oppstå. Jeg forberedte svar på de predikerte spørsmålene og reflekterte over hvor omfattende veiledningen skulle være i diverse utfordrende situasjoner. Dette for å forsikre at veiledningen i minst mulig grad skulle være styrt av impulser, og resultere i for mye eller for lite veiledning, noe som ville påvirket datamaterialet.

For å skrive kode benyttet elevene programmeringsspråket *Python (Dvergsdal, 2019)* og et vitenskapelige utviklingsmiljø *Spyder (https://www.spyder-ide.org/),* som blant annet inneholder editor og analyse- og feilsøkingsfunksjoner laget for Python. Dette ble valgt fordi elevene til vanlig benytter dette oppsettet for programmering i undervisningen. Lydopptaket og skjermopptak ble gjennomført i programmet *Zoom*. Zoom er et webkonferanseverktøy der brukeren kan avholde og delta i videokonferanser, nettmøter og lignende ("Zoom", 2020). For å ivareta elevenes anonymitet utførte de programmering på min datamaskin. Jeg startet et Zoom-møte på min datamaskin og brukte Zooms innebygde funksjon til å dele skjerm. Dette gjør at skjermen på min datamaskin vises i Zoom-møtet. Til slutt brukte jeg Zooms

opptaksfunksjon til å ta opp møtet. Da ble både skjermen og lyden i møtet tatt opp. Dette resulterte i studiens lyd- og skjermopptak.

3.4 Transkribering og analyse

3.4.1 Transkribering

Datamaterialet dannet fra lydopptakene ble transkribert ved hjelp av programvarene *ELAN* og *Microsoft Office Word*. ELAN er en programvare for å transkribere og annotere lyd- og videoklipp (Max Planck Institute for Psycholinguistics, u.å). Microsoft Office Word er en programvare for å skrive og behandle tekst (Rossen, 2020). Det transkriberte datamaterialet inneholder skjermbilder fra skjermopptakene gjort i Zoom. Skjermbildene er inkludert i transkripsjonen når det ble vurdert som nødvendig.

Ved transkribering av intervju risikerer en å tape visuelle ledetråder og dynamikken i intervjuet (Tjora, 2010, s. 144). For å best mulig bevare dynamikken og de visuelle ledetrådene valgte jeg å utføre transkriberingen selv. Da er risikoen mindre for å miste viktig informasjon i *oversettelsesprosessen (Tjora, 2010, s. 144).* Transkriberingen ble *normalisert*. Dette innebærer å transkribere på bokmål selv om informantene kommuniserte på dialekt (Tjora, 2010, s. 144). Det ble vurdert som lite hensiktsmessig for prosjektets formål å transkribere på dialekt. Pauser i elevenes diskusjon og interjeksjoner slik som «hm» ble transkribert slik at differansen mellom transkriberingen og den autentiske diskusjonen og kommunikasjon ble minst mulig. For å markere steder der dette er tilfelle er det brukt tegn. Beskrivelse av tegnene er gitt i Tabell 1. For å anonymisere elevenes identitet ble elevene gitt fiktive navn.

Tabell 1: Betydning av tegn brukt i transkriberingen.

Tegn	Betydning
[-]	Pause med varighet på ett eller to sekunder
[- -]	Pause med varighet på tre til ti sekunder
[- - -]	Pause med varighet på lengere enn ti sekunder
[…]	Utydelig setning som ikke lar seg transkribere
/	Når en elev avbryter en annen og overtar praten
[ ]	Beskrivelse av situasjon eller en hendelse

Studiens funn blir presentert ved representative og relevante utdrag av transkripsjonen. På denne måten blir datamaterialet presentert på en oversiktlig og hensiktsmessig måte.

3.4.2 Analyse og koding

Analyse er en prosess hvor en søker informasjon og mening gjennom å dele opp datasett etter fellestrekk (Gleiss & Sæther, 2021, s. 170). I denne studien har jeg benyttet meg av en *abduktiv* analysemåte med hovedvekt på *deduktiv* analysemåte. Abduktiv analysemåte er en blanding av induktiv og deduktiv analysemåte (Gleiss & Sæther, 2021, s. 171). Deduktiv analysemåte innebærer å benytte tidligere forhåndsetablerte koder fra forskningslitteraturen, mens i induktiv analysemåte utarbeides egne koder ut fra datamaterialet (Tjora, 2012, s. 193-194). *Koding* er navnet på prosessen der data brytes ned i mindre segmenter for deretter undersøke, sammenligne og kategorisere segmentene (Cohen et al., 2018, s. 668). Videre kan man tilskrive segmentene spesifikke merkelapper, disse merkelappene kalles for koder. Gjennom analysering og koding kan forskeren oppdage mønstre i datasettet.

I denne studien ble det benyttet forhåndsetablerte koder fra Lithner (2006) og Shute et al. (2017), men gjennom analyse av datamaterialet kombinert de forhånsetablerte kodene til nye koder. På denne måten er studiens analysemåte abduktiv.

Tabell 2 viser kodene som ble brukt for rammeverket til Shute et al. (2017). Hver komponent av CT har fått en kode som består av de to første bokstavene til den respektive komponenten. I Lithner (2006) sitt rammeverk har kreativ resonnering fått koden **CR** og imitativ resonnering **IR**. I studiens sammensatte rammeverk er disse kodene slått sammen. For eksempel er kreativ resonnerende dekomposisjon kodet som **CR**De, mens imitativ resonnerende dekomposisjon er

IRDe. Lithner (2006) sitt rammeverk er kodet med fet skrift for å tydelig skille de to rammeverkene.

Tabell 2: Forklarelse av kodene som ble brukt for rammeverket til Shute et al. (2017).

Begrep	**Kode**
Dekomposisjon	De
Abstraksjon	Ab
Algoritmer	Al
Feilsøking	Fe
Iterasjon	It
Generalisering	Ge

3.5 Etikk

I kvalitative forskningsmetoder kommer forskeren ofte tett innpå informantene gjennom observasjonsstudier eller dybdeintervju, derfor er det viktig å vurdere de etiske betraktningene (Tjora, 2010, s. 40). I følge Thagaard (2009, s. 25-30) er det tre etiske grunnprinsipper som må oppfylles for at en forskningspraksis skal anses som etisk forsvarlig. Disse er:

- *Informert samtykke.*
- *Konfidensialitet.*
- K*onsekvenser av å delta i forskningsprosjektet.*

Prinsippet om informert samtykke innebærer at informanten skal informeres om prosjektets formål, metode, frivillighet og at samtykket om å delta i prosjektet kan trekkes tilbake til enhver tid (Thagaard, 2009, s. 26). For å ivareta de etiske betraktningene rundt grunnprinsippet informert samtykke ble prosjektet meldt inn til *Norsk senter for datahåndtering* (NSD). Prosjektet ble detaljert beskrevet overfor NSD, og et skjema for å innhente samtykke fra informantene ble utarbeidet. NSD godkjente prosjektet og tilhørende samtykkeskjema. Samtykkeskjemaet inneholdt informasjon om prosjektets formål, frivillighetsgrad og informantenes rettigheter og personvern, se vedlegg A. Det ble presisert skriftlig gjennom samtykkeskjemaet og muntlig at samtykket kunne til enhver tid trekkes tilbake. For å gi elevene god betenkningstid ble samtykkeskjemaet levert ut til gjennomlesing én uke før utførelse av

datainnsamlingen. Elevene som deltok i studien var over 15 år og kunne dermed selv samtykke til deltakelse.

Prinsippet om konfidensialitet innebærer at deltakere i forskningen har krav på at all informasjon de gir blir behandlet konfidensielt (Thagaard, 2009, s. 27). For å ivareta elevenes anonymitet ble lyd- og skjermopptakene kun lagret lokalt på min datamaskin, og ble oppbevart utilgjengelig for uvedkommende. I tillegg ble elevene anonymisert i transkriberingsprosessen.

Det tredje grunnprinsippet omhandler hvilke konsekvenser forskningen kan ha for de som deltar (Thagaard, 2009, s. 28). Det skal vurderes hvorvidt forskningen kan utføre skade for deltakere eller føre til andre belastninger. I denne studien kunne slike belastninger vært at eleven går glipp av ordinær undervisning. For å unngå dette utarbeidet jeg og klassens lærer et opplegg som elevene skulle gjennomføre mens de andre deltok i forskningsprosjektet. På denne måten var alle forskningsdeltakere a jour med den ordinære undervisningen etter endt forskningsperiode.

På bakgrunn av ovennevnte etiske betraktninger vurdere jeg at studien følger de tre grunnprinsippene for etisk forsvarlig forskningspraksis.

3.6 Reliabilitet, validitet og generaliserbarhet.

Kvalitet på forskning blir ofte vurdert på bakgrunn av tre faktorer: *reliabilitet*, *validitet* og *generaliserbarhet* (Tjora, 2012, s. 202).

3.6.1 Reliabilitet

Reliabilitet handler om i hvor stor grad forskningen er repliserbar, om hvorvidt en annen forsker som gjennomfører tilsvarende metode ville fått de samme resultatene (Thagaard, 2009, s. 198). For å vurdere forskningens relabilitet stiller man vanligvis to spørsmål (Gleiss & Sæther, 2021, s. 202):

1) Til hvilken grad har datamaterialet blitt påvirket av metoden?
2) Er forskningsresultatene reproduserbare?

Ettersom det er blitt brukt deltakende observasjon vil datamaterialet være noe påvirket av meg som forsker. Jeg forsøkte å være så objektiv og lite deltakende som mulig, og det blir tydelig poengtert i analysedelen når datamaterialet kan være påvirket av min veiledning. Bruken av lydopptak er noe som styrker oppgavens reliabilitet. Lyd- og skjermopptak skaper data som er mer objektiv enn forskerens egne notater (Thagaard, 2009, s. 199). Forskerens egne notater er rekonstruksjoner og kan bære preg av forskerens oppfatninger og tolkning. Basert på dette ble

det vurdert til at studiens metode er nok detaljert beskrevet og har hatt relativt lite påvirkning på datamaterialet.

Hvorvidt forskningsresultatene er reproduserbare må sees i sammenheng med den fenomenologiske hermeneutisk tilnærmingen. Det er sannsynlig at en annen forsker som gjennomfører tilsvarende studie med samme metode hadde fått andre resultater. Dette skyldes den fenomenologisk hermeneutiske tilnærmingen, og det faktum at studien ble gjennomført på kun seks informanter. På den andre siden er det vel så sannsynlig at forskeren kan oppnå samme resultat. Det er uenighet hvorvidt reliabilitet er relevant for kvalitativ forskning (Thagaard, 2009, s. 198). I forskningslogikk basert på konstruktivistisk ståsted er ikke reliabilitet relevant. Likevel vurderes denne studien med til dels stor grad av reliabilitet.

3.6.2 Validitet

Validitet referer til tolkning av data og gyldigheten av forskerens tolkning (Thagaard, 2009, s. 201). I kvalitative studier vil en positivistisk tradisjon vurdere validiteten ut fra hvorvidt studien har kommet frem til en sannferdig kunnskap om virkeligheten (Gleiss & Sæther, 2021, s. 205)

I denne studien ble Lithner (2006) sitt rammeverk for analysering av kreativt og imitativt resonnement og Shute et al. (2017) sitt rammeverk for CT brukt som grunnlag for mine fortolkninger. Rammeverket til Lithner (2006) bygger på empiri og beskriver detaljert skilnaden mellom kreativ og imitativ resonnement. Rammeverket for CT fra Shute et al. (2017) er detaljert beskrevet og skilnaden mellom komponentene av CT er tydelig. Studiens fortolkninger er tydelig redegjort ut fra ovennevnte teori, og styrker studiens validitet (Thagaard, 2009, s. 201)

Tolkning av data blir godt beskrevet og begrunnet i kapittel 4 Analyse. Basert på ovennevnte kriteria anses derfor studiens validitet som god.

3.6.3 Generaliserbarhet

Generaliserbarhet handler om hvorvidt studiens funn er generaliserbar fra én kontekst til en annen (Gleiss & Sæther, 2021, s. 207). Det er en diskusjon om hvorvidt generaliserbarhet er nødvendig for kvalitative studier. I kvantitative studier tenker man ofte statistisk når det gjelder generalisering. I Kvalitative studier derimot, der det ikke er mulig med en statistisk fremgangsmåte, og derfor, skiller man mellom tre former for generalisering: *Naturalistisk, moderat og konseptuell.*

I naturalistisk generalisering redegjør forskeren for detaljene i studien og leseren selv vurderer om funnene er gyldig for leserens egen forskning. I moderat generalisering må forskeren selv beskrive i hvilke situasjoner resultatene kan anses som gyldige. I Konseptuell generalisering utvikler man konsepter eller teorier som vil kunne ha relevans for andre situasjoner enn de respektive situasjonene som ble studert (Tjora, 2010, s. 208-209).

Jeg velger å vurdere studiens generaliserbarhet i form av moderat generalisering. Dermed vurderes studien til å være generaliserbar innenfor den norske utdanningssektoren. Som tidligere nevnt ble studien kun utført på seks informanter, men disse informantene anses å være til en viss grad representativ for elever i den norske videregående skolen.

4 Analyse

I dette kapittelet analyseres studiens innsamlet datamateriale gjennom problemstillingen:

Ved hvilke kjennetegn av computational thinking resonnerer elever kreativt i problemløsning med tekstbasert programmering som verktøy?

Datamaterialet som analyseres er dannet fra notater og lyd- og skjermopptak av elevers arbeid med problemløsning ved hjelp av programmering.

Kapittelet er todelt. Første del er en kvantitativ presentasjon av transkripsjon. Her ønskes det å presentere antall registrerte koder i hver komponent av CT og forholdet mellom antall kreative og imitative resonnement i komponentene. Dette gjøres for å danne et inntrykk av elevenes resonnering og bruk av CT. Videre ble det foretatt en dypere analyse ved å se på utdrag fra elevenes diskusjon og skjermbilder fra elevenes arbeid. Utdragene er sortert etter komponentene i rammeverket til Shute et al. (2017) og er analysert i henhold til studiens sammensatte rammeverk fra delkapittel 2.6.

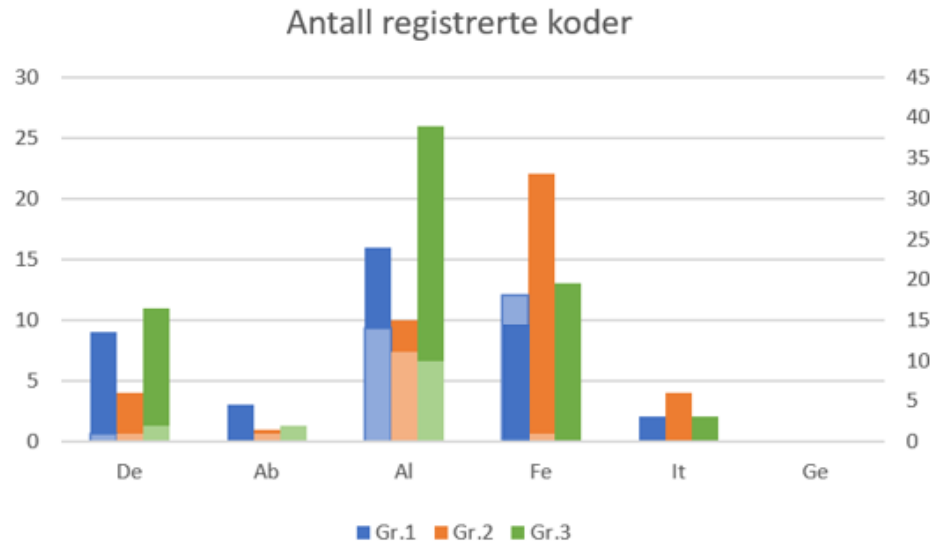
4.1 Kvantitativ presentasjon av transkripsjon

Gjennom en kvantitativ presentasjon av transkripsjon blir det presentert en oversikt over størrelsesforholdet mellom imitativ og kreativ resonnering, og hvilke komponenter av CT som ble observert mest. Dette gir også et innblikk i sammenhengen mellom komponentene av CT og elevenes resonnement. Tabell 3 viser antall registrerte koder i hver kategori hos hver gruppe. Det er viktig å poengtere at den kvantitative analyse er basert på et begrenset datasett, og er et resultat av min tolkning av elevenes diskusjon. Figur 7 viser et søylediagram over antall registrerte koder i hver komponent hos hver gruppe og størrelsesforholdet mellom imitative og kreative resonnement. Antall imitative resonnement er presentert i mørke farger, mens kreative resonnement er presentert i lyse farger.

Tabell 3: Antall registrerte koder i hver kategori.

	Gruppe 1		Gruppe 2		Gruppe 3	
	Imitativ	Kreativ	Imitativ	Kreativ	Imitativ	Kreativ
Dekomposisjon	9	1	4	1	11	2
Abstraksjon	3	0	1	1	0	2
Algoritmer	16	14	10	11	26	10
Feilsøking	9	18	22	1	13	0
Iterasjon	2	0	4	1	2	0
Generalisering	0	0	0	0	0	0

Figur 7 og Tabell 3 viser at flesteparten av elevenes utsagn ble kodet til komponentene algoritmer, feilsøking og dekomposisjon. Et fåtall av utsagnene ble kategorisert til komponentene iterasjon og abstraksjon, mens ingen utsagn var knyttet til komponenten generalisering. Den kreative resonneringen forekom oftest i komponenten algoritmer.



Figur 7:Antall registrerte koder i hver komponent av CT, og størrelsesforholdet mellom imitativ og kreativ resonnement. Imitative resonnement er presentert i mørke farger, kreative resonnement i lyse farger.

Gruppe 1 skiller seg ut i komponenten feilsøking. Denne gruppen hadde flest antall registrerte koder i denne komponenten, og andelen kreative resonnement var vesentlig høyere sammenlignet med de andre gruppene, som hadde svært få eller ingen kreative resonnement i denne komponenten.

Gruppe 2 skiller seg ut i komponenten dekomposisjon. Gruppen har få antall registrerte koder i denne komponenten sammenlignet med de andre gruppene.

Gruppe 3 skiller seg ut i komponenten algoritmer. Sammenlignet med de andre gruppene hadde gruppe 3 få antall kreative resonnement i forhold til antall imitative resonnement i denne komponenten. Antall imitative og kreative resonnement i komponenten algoritmer var tilnærmet lik for gruppe 1 og 2.

4.2 Utdrag fra transkripsjon

For å konkretisere analysen har jeg valgt å inkludere utdrag fra elevens arbeid. Ved å inkludere utdragene ønsker jeg å fremvise hvordan komponentene av CT og imitative og kreative resonnement kommer til uttrykk og blir identifisert i elevenes arbeid.

Utdragene er sortert etter komponentene av CT og valgt på bakgrunn av relevans til studiens sammensatte rammeverk.

Som den kvantitative presentasjon viste, varierer antall registrerte koder i hver komponent. Dermed vil antall utdrag i hver komponent og hos hver gruppe variere. I forkant av utdragene vil konteksten for utdraget og tidspunktet i problemløsningen bli tydeliggjort. Skjermbilder av elevens foreløpig arbeid er inkludert i utdragene når det anses som nødvendig for å forklare konteksten. Skjermbildene refereres etter figurnummer. Linjene i utdragene er markert som «UxLy», der x referer til nummeret på utdraget og y referer til linjenummer i utdraget. For eksempel vil linje 6 i utdrag 2 refereres til som «U2L6» og kan leses som utdrag 2, linje 6. Dette er gjort for å lettere referere til spesifikke linjer i utdragene. Når utdragene er presentert og konteksten er gjengitt er utdragene, analysert og tolket ved hensyn på Studiens sammensatte teoretisk rammeverk 2.6. I de neste delkapitlene presenteres de ulike komponentene av CT, og dermed hvordan man har kommet fram til Tabell 3.

4.2.1 Dekomposisjon

Dekomposisjon innebærer å bryte ned problemet i mindre delproblemer. De mindre delproblemene er mer håndterlig og lar seg lettere løse (Shute et al., 2017). Det ble observert dekomposisjon hos alle gruppene.

Første utdrag er hentet fra gruppe 1. Gruppen har nettopp lest oppgaveteksten for første gang. Utdraget viser hvordan gruppen dekomponerer problemet inn i to faser. Første fase innebærer å finne vinkelen mellom to vektorer, mens andre fase innebærer å avgjøre om vinkelen er spiss, stump eller om vektorene er ortogonale. I U1L3 informerer Tobias om definisjonene av spiss

og stump vinkel, samt hva det vil si at to vektorer er ortogonale. Dette tyder på at eleven har memorert definisjonene. På samme måte i U1L5 har eleven memorert at skalarproduktet til to ortogonale vektorer er null. Eleven har feilmemorert terminologien og bruker feilaktig skalarpunkt istedenfor skalarprodukt. Elevenes resonnement tar utgangspunkt i tidligere erfaringer og innebærer memorering av konsepter. Dette kategoriseres som imitativ resonnering (Lithner, 2006). Ragnas kommentar i U1L8 er også et eksempel på et imitativt resonnement. Ragna kommenterer at det eksisterer en formel for å finne vinkelen mellom to vektorer, men at hun ikke husker denne. Eleven har med andre ord identifisert oppgaven som kjent, med en tilhørende kjent løsningsalgoritme. Dette kjennetegner imitativ resonnering, mer spesifikt underkategorien kjent algoritmisk resonnering (Lithner, 2006). Løsningsalgoritmen er glemt, men er skrevet ned og kan lokaliseres og implementeres. Utdrag 1 kategoriseres som imitativ dekomposisjon. En lignende dekomposisjonsfase ble også observert hos gruppe 2 og 3.

Utdrag 1

U1L1 *Tobias*: [Leser oppgaveteksten].

U1L2 *Ragna*: Ja. Husker du hvordan man regner det?

U1L3 *Tobias*: Okei, spiss er jo at den er under 90 grader. Stump er at den er mer enn 90 grader. Vektorer er ortogonal det betyr at de er [-] det betyr at de er 90 grader.

U1L4 *Ragna*: Ja.

U1L5 *Tobias*: Vi vet i hvert fall at skalarpunktet til to ortogonale vektorer [- -] ja det er vektorer vi har her, det blir jo null.

U1L6 *Ragna*: Ja.

U1L7 *Tobias*: Så det kan vi jo bruke.

U1L8 *Ragna*: Ja, også har vi sikkert også, hvis jeg ikke husker feil, skrevet ned den her formelen for å regne vinkelen mellom vektorer. Jeg husker den ikke.

Like etter gruppe 2 har fått utlevert problemet oppstår diskusjonen gjengitt i utdrag 2. Gruppen dekomponerer problemet og første delproblem innebærer å definere vektorer. Dette kommer frem i U2L3 der Hilde foreslår at de først må spørre etter vektorene. Ettersom problemet spesifikt etterspør vinkelen mellom to vektorer kan denne dekomposisjon ikke anses som kreativ. Dermed vurderes resonnementet som imitativt. I U2L5 og U2L6 diskuterer elevene om variabelnavnet skal være a og b eller vektor1 og vektor2, sistnevnte ble valgt.

Utdrag 2

U2L1 *Hilde*: Kan vi få den til å spørre hva som er vektor greiene?

U2L2 Forsker: Det kan dere.

U2L3 *Hilde*: okei. Okei, skal vi gjøre det da først? Nå husker ikke jeg hvordan man skrev det [-] det var [- -] Okei vi skriver [-] skal vi bare si vektor 1 eller?

U2L4 *Camilla*: ja. Du kan skrive a, b.

U2L5 *Hilde*: A, b, men blir det ikke enklere hvis vi skriver/

U2L6 *Camilla*: /ja vi skriver det.

Samtlige grupper møtte på utfordringer tilknyttet definisjon av vektorer i Python. Utdrag 3 er hentet fra gruppe 3 og skildrer nevnte problem. Elevene har ikke blitt undervist i hvordan vektorer defineres i Python. For å løse problemet valgte gruppen og dekomponere vektoren i x- og y-koordinater og videre definere disse hver for seg. Elevene bryter problemet rundt definisjonen av vektorer ned til to mindre delproblemer. Dette kjennetegner CT komponenten dekomposisjon (Shute et al., 2017). Resonneringen som førte til dekomposisjonen anses som fleksibelt og argumentasjonen omkring strategivalget var plausibelt og basert på elevenes matematiske fundament. På bakgrunn av dette anses resonnementet som kreativt (Lithner, 2006).

Utdrag 3

U3L1 *Hans*: Vi lurer egentlig bare på om vi skal definere u-vektor og v-vektor.

U3L2 *Ole*: Ja, hvordan?

U3L3 *Hans*: ja.

U3L4 *Ole*: La oss si det slik, x1 og y1 er u vektor, kanskje?

U3L5 *Hans*: Mhm.

Også i implementasjon av formelen for skalarprodukt oppsto det dekomposisjon hos gruppe 1 og 2. Istedenfor å skrive formelen i sin helhet på én kodelinje, valgte elevene å kode dette over flere linjer. Begge gruppene valgte å kode absoluttverdiene på egne linjer, slik vist i Figur 8 og Figur 9. Videre dekomposisjon var noe forskjellig, blant annet har gruppe 2 feilaktig byttet om på teller og nevner i formelen. Dersom elevene skulle utført samme operasjonen med penn og papir, er det tenkelig at elevene hadde valgt å gjøre regne formelen i en og samme operasjon. Resonneringssekvensen anses dermed som ny for elevene, og gruppene viser en fleksibel

tilpasning til det gitte problemet. På bakgrunn av dette vurderes resonneringssekvensen som kreativ.

```
11    absu=py.sqrt(x1**2+y1**2)
12    absv=py.sqrt(x2**2+y2**2)
13
14    cosinus_a= (x1*x2+y1*y2)/(absu*absv)
15
16    vinkel=py.arccos(cosinus_a)
17
18    grader=180/py.pi*vinkel
```

Figur 8: Gruppe 1 dekomposisjon av formelen for skalarprodukt.

```
10    v1= sqrt(x1**2 + y1**2)
11    v2= sqrt(x2**2 + y2**2)
12
13    skalar = x1 * x2 + y1 * y2
14
15    x = arccos((v1*v2)/skalar)
16    G = 180/pi * x
```

Figur 9: Gruppe 2 dekomposisjon av formelen for skalarprodukt.

4.2.2 Abstraksjon

I rammeverket til Shute et al. (2017) innebærer abstraksjon å finne essensen av et system. Videre deles abstraksjon inn i tre underkategorier; *datainnsamling og analysering*, *mønstergjenkjenning* og *modellering*. Av disse ble datainnsamling og analysering samt modellering observert.

Datainnsamling og analysering ble observert i diskusjonen til gruppe 1 fra utdrag 1. I utdraget forsøker Tobias å samle relevant informasjon for å løse problemet. I U1L3 samler han informasjon om spiss og stump vinkel. I U1L5 nevner eleven at skalarproduktet til to ortogonale vektorer er null. Videre identifiserer eleven i U1L8 at problemet kan løses ved å bruke en formel kjent fra tidligere. Eleven har forstått essensen av problemet og som følger av dette begynner eleven å samle inn relevant informasjon for å løse problemet. Dette kan kategoriseres som datainnsamling og analysering (Shute et al., 2017). Elevens strategivalg består av memorering av regler og løsningsalgoritmer. Resonneringssekvensen vurderes dermed som imitativ.

Både gruppe 2 og 3 vurderte å først løse problemet ved hjelp av regning for hånd, deretter undersøke hvordan denne løsningen kunne programmeres. Dette kommer frem i utdrag 4 hentet fra gruppe 2. Elevene har i forkant av utdraget definert vektorene og arbeider nå med å finne vinkelen mellom vektorene. I U4L3 foreslår Camilla å først regne for hånd, deretter bruke programmering. Dette kan anses som modellering, i den forstand at eleven ønsker å skape en modell for hvordan systemet opererer. Modellen skapes på papir før den senere implementeres til programmeringen.

Utdrag 4

U4L1 *Camilla*: Det er skalarprodukt.

U4L2 *Hilde*: Det er skalarprodukt. [- -] okei, så vi må få den til å bli [-] Nå finner vi arealet, da må vi finne arealet også, da funker ikke den der. For det der er jo/

U4L3 *Camilla*: /Hva om vi bare regner det ut for hånd først, også bruker programmering, for at det står […]

U4L4 *Hilde*: okei bare gjør det, vær så god. [- -] Okei la oss sjekke.

U4L5 *Camilla*: Okei oppgaven er jo å bare […]

Hos gruppe 3 ble denne formen for modellering også observert. Dette er fremvist i utdrag 5. Denne gruppen hadde ikke definert vektorene i forkant av utdraget, men hadde identifisert at formelen for skalarprodukt kan benyttes til å finne vinkelen mellom to vektorer. I U5L1 kommer Ole med et lignende forslag som Camillas fra U4L3. En slik form for modellering har elevene trolig ikke støtt på tidligere. Elevene viser også en flytende bevegelse mellom to forskjellige tilnærminger til problemet, nemlig å løse oppgaven for hånd eller ved hjelp av programmering. På bakgrunn av dette kan resonneringssekvensen vurderes som kreativ.

Utdrag 5

U5L1 *Ole*: Skal vi løse først for hånd før vi gjør det med programmering?

U5L2 *Hans*: Ja jeg tror vi burde løse [-] Vi burde i hvert fall vite formelen for å løse for hånd, det er jo den [peker på formelen for skalarprodukt].

U5L3 *Ole*: ja.

4.2.3 Algoritmer

Mange observasjoner ble kategorisert til komponenten algoritmer, og resonnementene i tilknytning komponenten var både kreative og imitative. Komponenten algoritmer innebærer å lage instrukser som gjengir løsningen til et problem (Shute et al., 2017). Elevenes bruk av algoritmer var i hovedsak tilknyttet fire delproblemer. Det første problemet handlet om definisjon av vektorer i Python. Neste problem var tilknyttet bruk av input-funksjonen i definisjon av vektorene. Etter dette oppsto det problemer i forbindelse implementasjon av skalarproduktformelen. Siste problem handlet om formulering og bruk av if-else tester.

Elevene hadde ikke tidligere arbeidet med vektorer i Python. Dette førte til utfordringer når de skulle definere vektorer i Python. Utdrag 6 og 7 er hentet fra gruppe 3 og viser nevnte utfordring. I forkant av utdragene har elevene diskutert implementasjon og definisjon av vektorer i Python, og identifisert at skalarproduktformelen kan brukes for å finne vinkelen mellom to vektorer. Tiden mellom utdrag 6 og 7 er kort og ble hovedsakelig brukt til å lete etter informasjon i læreboken.

Utdrag 6

U6L1 Forsker: Hva tenker dere på?

U6L2 *Ole*: Jeg tenker sånn, vi må for eksempel ha en vektor. For eksempel u-vektor, med x og y.

U6L3 *Hans*: Ja.

U6L4 *Ole*: Jeg vet ikke hvordan man skriver det i Python.

Det fremstår fra utdrag 6 at elevene har kunnskap om første- og andrekoordinat hos todimensjonale vektorer, men problemet er å overføre kunnskapen til programkode. Dette kan skyldes at elevene fikserer på måten de skriver vektorer på ved regning for hånd, nemlig på formen $[x_1, y_1]$. En slik fiksering kjennetegner imitativ resonnering (Lithner, 2006). I utdrag 7 kommer Ole med et forslag om å definere vektorkoordinatene hver for seg. Ole resonnerer fleksibelt gjennom å gjøre tilpasninger til problemet basert på hans matematiske fundament. På denne måten unngår Ole den tidligere nevnte fikseringen. Grunnet elevens manglende erfaring med vektorer i Python anses resonneringssekvensen som ny for eleven. På bakgrunn av dette vurderes resonnementet som kreativt (Lithner, 2006).

Utdrag 7

U7L1 *Ole*: Åh, jeg vet! Kan vi gjøre sånn x1, y1 og x2, y2?

U7L2 *Hans*: […]

U7L3 *Ole*: Kan jeg skrive her? [spør forsker om tillatelse til å skrive i notatboken]

U7L4 Forsker: Ja, det kan du.

U7L5 [Elev skriver i notatboken vektorer på formen [x1, y1] og [x2, y2]].

Også gruppe 1 og 3 forsøkte først å definere vektorene som de vanligvis gjør ved regning for hånd, før de senere oppdaget tidligere nevnte metode. Det er verdt å nevne at gruppe 2 og 3 brukte betydeligere lengere tid før de oppdaget nevnte metode, og til tross for å ha oppdaget metoden fortsatte begge gruppene å skrive vektorene som ved regning for hånd, slik vist i Figur 10 og Figur 11. Dette kan ha en sammenheng med at vektormultiplikasjonen i skalarproduktformelen i elevens bok var skrevet som u-vektor multiplisert med v-vektor fremfor på vektorkoordinatform. Elevene forsøker å tilpasse programmet til formelen fremfor å bryte formelen ned i mindre deler slik at formelen tilpasses programmet. Dersom dette er tilfellet anses elevens resonnement som imitativt ettersom elevene fikserer på måten formelen er skrevet på.

```
4   vektor1 = (x1,y1)
5   x1  = float(input(skriv in vektorkoordinatene til x1: ))
6   y1 = float(input(skriv in vektorkoordinatene til vektor u: ))
7
8   vektor2 = (x2,y2)
9   x2  = float(input(skriv in vektorkoordinatene til x1: ))
10  y2 = float(input(skriv in vektorkoordinatene til vektor u: ))
11
12  x = x1 + x2
13  y = y1 + y2
```

Figur 10: Definisjon av vektorer gr. 2.

```
3   x1=float(input("Skriv inn x1:"))
4   y1=float(input("Skriv inn y1: "))
5   x2=float(input("Skriv inn x2: "))
6   y2=float(input("Skriv in y2: "))
7
8   u = x1, y1
9   v = x2, y2
10
```

Figur 11: Definisjon av vektorer gr. 3.

Neste utfordring elevene støtte på var bruk av input-funksjonen i definisjon av vektorene. Alle gruppene ønsket å benytte input-funksjonen, selv om problemet ikke spesifikt nevnte dette. Et eksempel på dette er utdrag 8, og er hentet fra gruppe 1. I U8L3 foreslår Tobias å bruke input-funksjonen og begrunner forslaget med at dette vil gjøre det enklere å teste programmet med flere forskjellige vektorer. Tobias ønsker å bruke input-funksjonen for å effektivisere og automatisere koden, noe som kjennetegner komponenten algoritmer (Shute et al., 2017). Elevene har tidligere arbeidet med input-funksjonen, så selve funksjonen er ikke ny for elevene, men bruken av funksjonen i vektoroppgaver er ny. Det er ikke en nødvendighet å bruke funksjonen for å løse problemet, likevel velger elevene å bruke funksjonen. Resonneringssekvensen anses som ny for elevene og bruken av input-funksjonen anses dermed som et kreativt resonnement.

Utdrag 8

U8L1 *Tobias*: Skal vi gjøre slik at man tar å spør etter?

U8L2 *Ragna*: Hva?

U8L3 *Tobias*: Sånn input [-]. Vi må teste med forskjellige, så det blir bare enklere å spørre, blir det ikke det?

U8L4 *Ragna*: Ja, vi kan gjøre det.

Gruppe 2 og 3 fant det utfordrende å implementere input-funksjonen. Gruppe 3 forvekslet print-funksjonen med float-funksjonen, mens gruppe 2 funderte på rekkefølgen på float- og input-funksjonen og hva som er hensikten med å inkludere float-funksjonen. Sistnevnte er vist i utdrag 9. I U9L1 spør Hilde om man skal bruke float eller input først. Camilla svarer ved å stille et nytt spørsmål om hvorvidt det er hensiktsmessig å inkludere float, og kommer med en påstand at float gjør det innskrevet tallet om til et desimaltall. I U9L3 svarer Hilde at det er hensiktsmessig å inkludere float ettersom funksjonen gjør et desimaltall om til et heltall. Camilla beskriver korrekt funksjonen til float og Hilde har rett i at denne funksjonen burde inkluderes. Altså er begge resonnementene delvis korrekt. Elevene diskuterer så rekkefølgen på funksjoner og hva som er hensiktsmessig å inkludere. Dette kan kategoriseres som en diskusjon om algoritmisk design (Shute et al., 2017). Gruppens strategivalg innebærer å memorere den komplette algoritmen for implementasjon av input-funksjonen, noe som kan kategoriseres som et imitativt resonnement (Lithner, 2006). Dersom elevene hadde hatt kunnskap om funksjonen til input og float, hadde implementasjonen trolig vært intuitiv.

Utdrag 9

U9L1 *Hilde*: Vektor 1, så 1 er [-] var det float input eller var det input float? [Elev skriver «vektor1=»].

U9L2 *Camilla*: Hvorfor skrev du float? [-] Da får du et desimaltall.

U9L3 *Hilde*: Nei det blir ikke desimaltall, det er bare for å sikre i tilfelle du skriver desimaltall. Så blir det enklere liksom.

U9L4 *Camilla*: Ja, okei.

U9L5 *Hilde*: Er det input float eller float input? [Elev spør forsker]

Som ved input-funksjonen valgte alle gruppene å benytte seg av if-else tester. Elevenes bruk av if-else tester tyder på at elevene resonnerer over hvilke instrukser som er nødvendige og nyttige for å gjengi løsningen til problemet, noe som kjennetegner komponenten algoritmer (Shute et al., 2017). Problemets ordlegging kan ha påvirket elevene til å inkludere if-else testene, siden problemet etterspør et program som kan avgjøre hvorvidt vinkelen mellom vektorene er stump, spiss eller om vektorene er ortogonal. Dersom et program skal avgjøre noe, kan det fremstå naturlig å inkludere if-else tester. På bakgrunn av dette anses elevenes resonnement som tekstveiledet og kategoriseres som et imitativt resonnement. Utdrag 10 stammer fra gruppe 2, og viser diskusjonen omkring implementasjon av if-else tester. I forkant av utdraget har gruppen definert vektorer, implementert skalarproduktformelen og laget en if-test for å undersøke om vinkelen er mindre enn 90 grader. Videre ønsket gruppen å lage en test for når vinkelen er større enn 90 grader. Gruppen var usikker på forskjellen mellom elif-, if- og else-tester. I utdraget benytter elevene først en else-test før de endrer denne til en mer korrekt elif-test. Diskusjonen omkring else- og elif-test kan anses som en diskusjon om hvilken instruks som fører til korrekt gjengivelse av løsningen, og kan kategoriseres som algoritmisk design.

Utdrag 10

U10L1 *Hilde*: [skriver «else x»] Else, x også mindre [-] hvor fant du det?

U10L2 *Camilla*: Å ja, x er større [Elev skriver «else x>90»].

U10L3 *Hilde*: Også gjør du det samme som du gjorde der [referer til sist skrevet print] [- -]. Nei du må ha de der [referer til at Camilla ikke har skrevet semikolon etter «else x>90»].

U10L4 *Camilla:* Okei sånn.

U10L5 *Hilde*: Du må ha de der to strekene [Referer til at *Camilla* ikke har brukt apostrofe i print-funksjonen]. Okei, også må vi, hvis den er 90 grader da. Så [- -] Skal vi bare skrive

else eller skal vi skrive elif [- -] Der skal det være elif, skal det ikke? [referer til «else x>90»].

U10L6 Forsker: mhm.

U10L7 *Hilde*: [endrer else til elif] Sånn. Også nå er det else [elev skriver «else:»] vinkelen er [- -].

U10L8 *Camilla*: ortogonal.

Ved implementasjon av skalarproduktformelen forsøkte både gruppe 1 og 2 å lage en egen funksjon for dette. I utdrag 11 foreslår Ragna fra gruppe 1 å lage en funksjon for skalarproduktformelen. Gruppen har allerede bestemt seg for å løse problemet med skalarproduktformelen, og diskuterer nå hvordan de skal instruere programmet til å gjengi formelen. Dette kan anses som algoritmisk design (Shute et al., 2017). Definisjon av funksjoner i Python er relativt ukjent for elevene, og ikke nødvendig for å løse problemet. På tross av dette foreslår Ragna å definere en funksjon. Dette anses som en ny resonneringssekvens for eleven, og forslaget vurderes da som originalt, selv om det mangler argumenter som støtter forslaget. I U11L6 blir Ragna spurt om hun ønsker å lage en funksjon. Som svar forteller Ragna at hun tror de må det, ergo fremmer ikke eleven noen argumenter som støtter forslaget. Mangelen på argumenter gjør det utfordrende å vurdere om resonneringen er imitativ eller kreativ. Med tanke på graden av originalitet vurderes dette som kreativ resonnering. Resonneringen til gruppe 2 var av samme karakter. Her kom det også et kreativt forslag om å implementere en funksjon, og i likhet med gruppe 1, manglet også dette forslaget støttende argumenter.

Utdrag 11

U11L1 *Ragna*: Nei, jeg må bare ha tid til å prosessere hva det er du sier [- -] ja, men [- -] Skal vi gjøre som en [-]/

U11L2 *Tobias*: /som en funksjon?

U11L3 *Ragna*: Ja.

U11L4 *Tobias*: Ja, sånn, skal vi definere den?

U11L5 *Ragna*: Ja.

U11L6 *Tobias*: Er det det du vil?

U11L7 *Ragna*: Ja, jeg tenkte vi må kanskje det.

U11L8 *Tobias*: Ja.

U11L9 *Ragna*: Eller ikke […] [-] Ja vi gjør det.

U11L10 *Tobias*: Ja, vi kan gjøre det. [skriver def] Hva skal vi kalle funksjonen?

4.2.4 Feilsøking

Feilsøking innebærer å oppdage og gjenkjenne feil, for så rette opp i feilene (Shute et al., 2017). Dette kan gjøres på mange måter, men det var særlig to former som ble observert i denne studien. Den ene innebærer først å skrive koden, for så undersøke om denne stemmer ved å kjøre programmet. Programmet anses som korrekt dersom output er som ønsket. På denne måten blir programmet hele tiden oppdatert basert på programmererens forståelse. Gruppe 1 var den eneste gruppen som benyttet denne metoden. Den andre metoden som ble observert innebærer at personen som ikke programmerer, studerer koden og kommenterer dersom eleven oppdager noen feil. Ved hjelp av denne strategien unngikk gruppene flere syntaksfeil.

I utdrag 12 gjennomfører gruppe 1 feilsøking ved å først kjøre koden og deretter tolke feilmeldingene. Figur 12 viser elevenes foreløpige progresjon. I U12L3 kommer denne strategien tydelig frem. Elevene har nettopp forsøkt å lage en funksjon for skalarprodukt, og Tobias foreslår å teste funksjonen for å undersøke om denne stemmer. Elevene fikk da opp en feilmelding ettersom funksjonen deres mangler et input-argument. Elevene fokuserer på at pilen fra feilmeldingen i konsollvinduet peker på funksjonsnavnet og kolontegnet, slik vist i Figur 13. På bakgrunn av dette antar elevene at feilen skyldes navnet på funksjonen, og forsøker å endre på dette. Dette resulterer i en ny feilkode som elevene forsøker å tolke og fikse, dette kategoriseres som feilsøking (Shute et al., 2017). Gruppen prøver og feiler. Seansen ender med at gruppen velger å gå bort fra funksjonstilnærmingen. Elevenes feilsøkingsstrategi er vanlig i programmering, men for elevene som har programmert svært lite er ikke denne metoden kjent, dermed kan forslaget fra Tobias anses som originalt. Dette støttes opp av det faktum at ingen av de andre gruppene utførte denne feilsøkingsstrategien. Gruppen viser også en flytende bevegelse av forskjellige tilnærminger for å løse feilmeldingene. På bakgrunn av dette anses resonneringen som kreativ.

Utdrag 12

Figur 12: Feilsøking 1 gr. 1.

U12L1 *Tobias*: Ja.

U12L2 *Ragna*: Og nå, hva da?

U12L3 *Tobias*: Da har vi skrevet inn den funksjonen [henviser til funksjonen u_v], likningene og [- -] Skal vi bare prøve å skrive?

U12L4 *Ragna*: Ja, vi prøver å se hva vi får.

U12L5 *Tobias*: [*Tobias* skriver «print(u_v)»] Er det det?

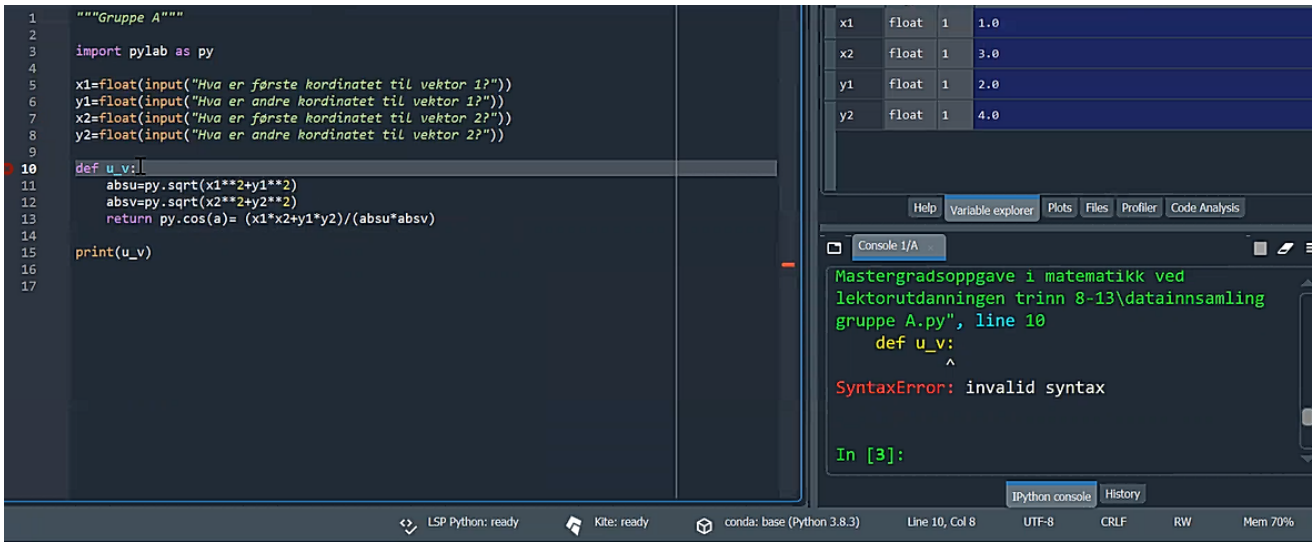
U12L6 *Ragna*: Jeg vet ikke, vi får se.

U12L7 *Tobias*: (…) [Elev kjører programmet]

U12L8 *Ragna*: Da finn vi i hvert fall ut/

U12L9 *Tobias*: [elevene får opp en feilmelding] /okei da fikk vi en feil. Okei, vi må kanskje kalle den noe annet.

U12L10 *Ragna*: Feilen peker på kolon.



Figur 13: Feilsøking 2 gr.1.

U12L11 *Tobias*: Jeg tror ikke vi kan kalle den det der [Elev visker ut «u_v»].

U12L12 *Ragna*: Hvordan kan vi kalle den […]

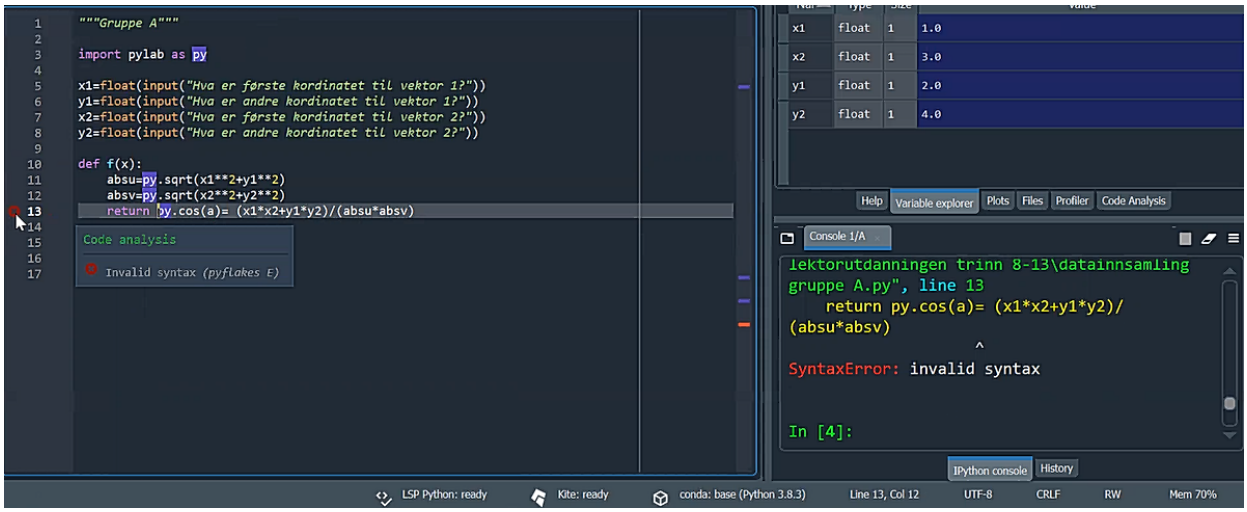
U12L13 *Tobias*: Men så er det [-] trenger vi å [- -] nei bare trenger vi å bruke sånn definisjon av funksjon eller?

U12L14 *Ragna*: Trengs sikkert.

U12L15 *Tobias*: Okei vi kaller den bare for f av x fordi […] [Endrer til «def f(x):»].

U12L16 *Ragna*: Er ikke f av x en graf?

U12L17 *Tobias*: Jo. [Elev kjører koden og får opp en ny feilmelding] Okei det funket ikke. Hva er det vi ikke har brukt nå?



Figur 14: Feilsøking 3 gr.1.

U12L18 *Ragna*: Den peker på er lik.

U12L19 *Tobias*: Den peker på er lik. Okei. Hvor ser du det?

U12L120 *Ragna*: Ser du ikke på den der lille greia [henviser til konsollvinduet].

U12L21 *Tobias*: Okei. Skal det være er lik er lik? [elev prøver å endre til «cos(a) == …», visker raskt ut igjen]. Nei det funket ikke. [- -] Wow okei, skal vi [- -].

Gruppe 2 og 3 benyttet hovedsakelig en feilsøkingsstrategi som innebærer å kommentere på syntaksfeil. Utdrag 10, fra tidligere, er hentet fra gruppe 2 og viser et eksempel på dette. Elevene forsøker å implementere if-else tester. I utdraget koder Camilla, mens Hilde studerer koden og kommenterer dersom hun finner feil. I U10L3 finner Hilde en feil i koden. Else-testen mangler semikolon. En lignende situasjon oppstår i U10L5, her kommenterer Hilde at koden mangler apostrofe i print-funksjonen. Slik form for feilsøking anses som imitativt ettersom feilen gjenkjennes ved hjelp av memorering av syntaks. Også gruppe 1 gjennomførte denne typen feilsøking.

4.2.5 Iterasjon

Iterasjon innebærer å repetere designprosessen for å oppnå den ideelle løsningen (Shute et al., 2017). På grunn av tidsmangel fikk ikke elevene mulighet til å revidere programmet. Det er naturlig å tenke at mesteparten av idealisering av løsningen ville forekommet i denne fasen. Som et resultat av tidsmangel kan datamengden på iterasjon være begrenset. Likevel observerte jeg iterasjon hos både gruppe 1 og 2.

I utdrag 13 har gruppe 1 først definert hver vektorkoordinat og forsøker nå å lage en funksjon for å regne ut vinkelen mellom vektorene. I U13L1 kommenterer Tobias at det sikkert finnes en enklere måte å løse oppgaven på. Det kan tolkes dit hen at eleven ønsker å repetere designprosessen for å oppnå en mer ideell løsning, men på nåværende tidspunkt mangler kunnskap om hvordan dette kan gjøres. Ragnas respons i U13L2 tyder på at eleven også ønsker å idealisere løsningen, men som eleven selv sier, mangler hun ideer for å utføre idealiseringen. Utsagnene kjennetegner komponenten iterasjon og kategoriseres deretter. Ønsket om å idealisere løsningen oppfyller ingen kriterier for et kreativt resonnement, og vurderes dermed som et imitativt resonnement.

Utdrag 13

U13L1 *Tobias*: Ja [- -] det finnes sikkert en mye enklere måte å gjøre dette på, men det går bra.

U13L2 *Ragna*: Jaja, så lenge vi finner det ut så. [- -] Men [-] ja, jeg vet ikke hva vi burde gjøre nå. [- -] Jeg er liksom litt tom for ideer.

Også hos gruppe 2 ble iterasjon observert, dette er vist i utdrag 14. I forkant av dette utdraget har elevene definert vektorkoordinatene, og forsøker nå å utføre vektormultiplikasjon samt regne absoluttverdien av vektorene. I U14L1 forslår Camilla å definere vektorkoordinatene med spesifikke tall fremfor å bruke input-funksjonen. Gruppen bestemmer seg for å gjøre dette, men Hilde ytrer i U14L6 at hun ønsker å bevare det de tidligere har kodet. Ved å bevare dette kan gruppen på et senere tidspunkt sammenligne de forskjellige tilnærmingene og vurdere hvilken som egner seg best. Dette kan anses som iterasjon, i den forstand at gruppen i etterkant kan velge den tilnærmingen som vurderes som mest ideell. Ved å bruke denne metoden vil det bli enklere i etterkant å gjøre justeringer på kodens design. Elevene befinner seg i en til dels ukjent situasjon. Gruppen resonnerer fleksibelt og kommer frem til to tilnærminger til problemet. De velger å bevare begge tilnærmingene. Resonneringen som leder opp til bevarelse av begge tilnærmingene vurderes som kreativ. Dette fordi en situasjon med flere tilnærminger til et problem kan anses som ukjent for elevene.

Utdrag 14

U14L1 *Camilla*: Men jeg skjønner ikke hvorfor vi ikke bare har skrevet et tilfeldig tall.

U14L2 *Hilde*: Skal vi gjøre det da? Blir det enklere?

U14L3 *Camilla*: Ja.

U14L4 *Hilde*: Okei, da gjør vi det.

U14L5 *Camilla*: Da regner vi jo med tall. Jeg skjønner det er lurt å lage/

U14L6 *Hilde*: /okei vet du hva, vi bare lar det være der også skrive vi tallene, så har vi det i tilfelle, okei?

4.2.6 Generalisering

Generalisering innebærer å bruke ferdighetene fra CT for å løse problemer i nye situasjoner (Shute et al., 2017). Det ble ikke gjort noen observasjoner av generalisering. Dette skyldes trolig at studiens metode egner seg dårlig for observering av komponenten generalisering. Det kan tenkes at elevene på et senere tidspunkt har benyttet CT ferdighetene de opparbeidet fra denne

studien til å løse et annet problem, men dette lar seg ikke observeres grunnet studiens tidsramme.

4.2.7 Oppsummering av analyse

Analysen viser at for utenom generalisering er alle komponentene av CT observert i elevenes arbeid med problemet. Videre ble det observert både kreative og imitative resonnement i alle observerte komponenter.

I dekomposisjon ble det gjort flere observasjoner i hver gruppe. Det ble observert både imitative og kreative resonnement, riktig nok ble mesteparten ansett som imitative. Gruppene dekomponerte hovedsakelig problemet inn i to faser. Første fase handlet om å finne vinkelen, mens i andre fase måtte man avgjøre om vinkelen var stump, spiss eller rettvinklet. Det ble også observert dekomposisjon når elevene skulle definere vektorer og implementere skalarproduktformelen.

Det var få observerte tilfeller av komponenten abstraksjon. Likevel ble abstraksjon observert i alle gruppene. Underkategoriene *datainnsamling og analysering* og *modellering* ble observert. Modellering ble observert i den forstand at gruppene ønsket å bruke penn og papir for å lage en modell som de senere kunne implementere til programmet. Resonnementene tilknyttet denne modelleringen ble vurdert som kreative.

Det ble gjort flest observasjoner i komponenten algoritmer. Gruppene forsøkte flere forskjellige algoritmiske design. Observasjonene på algoritmer var blant annet tilknyttet definisjon av vektorer, bruk av input-funksjonen, implementasjon av skalarproduktformelen og bruk av if-else tester. I disse prosessene ble det observert både imitative og kreative resonnement. Størrelsesforholdet mellom resonnementtypene varierte mellom gruppene. Gruppe 1 hadde tilnærmet likt antall kreative og imitative resonnement, mens hos de andre gruppene var imitative resonnement dominerende.

For komponenten feilsøking ble det gjort flere observasjoner i hver gruppe. De fleste resonnementene ble vurdert som imitative og tilknyttet syntaksfeilsøking, men hos gruppe 1 ble det observert flere kreative enn imitative resonnement. Gruppe 1 var også den eneste gruppen som gjennomførte en feilsøkingsstrategi der de kjørte programmet for å undersøke om programmet fungerte, en prøve og feile strategi.

Komponenten iterasjon ble kun observert hos to av gruppene, og antall observasjoner var få. Observasjonene var tilknyttet et ønske om å løse oppgaven på en enklere måte, men ble ikke forsøkt utført grunnet begrenset kompetanse. Til tross for få antall observasjoner i komponenten ble både imitative og kreative resonnement observert.

Det ble ikke gjort noen observasjoner av komponenten generalisering. Dette må sees i sammenheng med problemets utforming og prosjekts tidsramme. Problemet utfordrer eleven til å løse én spesifikk oppgave, innenfor et begrenset matematisk område, på begrenset tid. Dette kan være årsaken til nevnte resultat.

5 Diskusjon

Studiens problemstilling lyder som følger: *«Ved hvilke kjennetegn av computational thinking resonnerer elever kreativt i problemløsning med tekstbasert programmering som verktøy?»* Problemstillingen er belyst ved å undersøke elevers resonnement, og bruk av CT i problemløsning med tekstbasert programmering som verktøy. I dette kapittelet drøftes funnene fra analysen i lys av tidligere presentert teori. Diskusjonen tar for seg tre hovedfunn fra analysen. Avslutningsvis følger en drøfting av studiens metodiske styrker og svakheter, samt forslag til videre forskning.

5.1 Definere vektorer i tekstbasert programmering

Fra analysen kom det frem at elevene fant det utfordrende å definere vektorer i Python. Dette er ikke overraskende ettersom elevene ikke har jobbet med vektorer i Python tidligere. Elevene forsøkte å definere vektorene slik de vanligvis gjør ved vektorregning for hånd og i programmet Geogebra. For dem er det nok ingen klar skillelinje mellom objektet vektor, og notasjonen av en vektor. De bruker notasjonen de tidligere har brukt. Denne notasjonen inkluderer parenteser, og en todimensjonal vektor defineres som $v = [x, y]$. I skoleverket er dette en sterk notasjon, og oftest den eneste elevene lærer seg. Det er dermed ikke overraskende at elevene ønsker å benytte seg av denne notasjonen. Dessuten kan man definere vektorer i Python på en lignende måte som elevenes forsøk. Ved hjelp av numpy arrays kan vektorer defineres ved å skrive $v = np.array([x, y])$, altså ikke så ulikt elevens forsøk. Det kan tenkes at elevene har sett lignende notasjon også i Python. På grunn av syntaksfeil oppdager elevene at deres forsøk ikke fungerer. Elevene baserer sitt resonnement på kjent vektornotasjon, og resonnementet vurderes dermed som imitativt. Når elevenes imitative resonnement ikke lykkes, må elevene forsøke andre, mer ukjente tilnærminger.

Etter en kreativ resonneringssekvens oppdager elevene at todimensjonale vektorene kan dekomponeres til x- og y-koordinater, som videre kan defineres hver for seg. Dette er med på å gjøre problemet mer håndterlig for elevene. Elevene dekomponerer vektordefinisjonsproblemet i tråd med Shute et al. (2017) definisjon av dekomposisjon. Elevene uttrykte en forståelse for at todimensjonale vektorer består av to komponenter, og at en vektor kan dekomponeres til disse komponentene. Den matematiske dekomposisjonen av vektorene anses dermed som kjent for elevene. Det som derimot var nytt for elevene var å benytte denne kunnskapen til og definere vektorer. Elevene viser en fleksibel tilnærming til definisjon av vektorer som baserer seg på deres matematiske kunnskap om vektorer. Gjennom dette skapes

en ny sekvens av resonnering. Dette i tråd med definisjon av kreativ resonnering (Lithner, 2006). Ut fra studiens sammensatte rammeverk kan denne observasjonen kategoriseres som kreativ dekomposisjon.

Selv etter å ha definert vektorene ved hjelp av dekomposisjon, velger to av gruppene å definere vektorene i sin helhet ved å sette sammen de dekomponerte elementene. Gjennom å gjøre dette sørger elevene for at vektordefinisjonen i Python samsvarer med deres kjente notasjon. Denne operasjonen tyder på at elevene fikserer på deres kjente vektornotasjon. I følge Haylock (1997) gjennomgår elevene en kognitiv prosess der de fikserer på den kjente notasjonen ettersom dette er en løsningsstrategi som tidligere har fungert. En forklaring på elevenes fiksering er vektornotasjonen sterke posisjon i skoleverket. Elevene har tidligere kun definert vektorer på denne måten og antar dermed det som nødvendig å holde seg til denne definisjonen. En annen forklaring kan skyldes måten skalarprodukt er definert på i elevenes lærebok. I bokens definisjon av skalarprodukt er vektormultiplikasjonen mellom vektor u og v skrevet som $u * v$, mens i absoluttverdiene er vektorene dekomponert til x- og y-koordinater. I søken etter en algoritme for å løse problemet vender elevene seg til læreboken. Basert på overflatelikheter mellom problemet og eksempler med skalarproduktformelen velger elevene å benytte seg av skalarproduktformelen. I følge Lithner (2006) kategoriseres dette som tekstveiledet algoritmisk resonnement. Strategiimplementasjonen innebærer å kopiere skalarproduktformelen. Dette kan ha medført at elevene fikserer på måten definisjonen er skrevet på og anser det som nødvendig å definere vektorene i sin helhet, slik som i læreboken. Denne forklaringen støttes av det faktum at gruppene forsøkte å kode skalarproduktformelen identisk med lærebokens definisjon. I vektormultiplikasjonen benyttet elevene multiplikasjonstegnet, mens i absoluttverdiene benyttet de vektorkoordinatene. Som følge av syntaksfeil i definisjon av vektorene fungerte ikke elevens forsøk på vektormultiplikasjon. Én av gruppene oppdaget dette på egen hånd og endret til vektorkoordinater også i vektormultiplikasjonen. Det er tydelig at gruppens instruksjoner for å gjengi løsningen til problemet baserer seg på et imitativt resonnement. I tråd med det sammensatte rammeverket i oppgaven kategoriseres denne observasjonen som imitative algoritmer.

Den gruppen som gjennom hele problemløsningsfasen forholdt seg til vektorkoordinater valgte også å kopiere skalarproduktformelen fra boken. Ved hjelp av feilsøking oppdaget gruppen raskt at dette ikke ville fungere ettersom de ikke hadde definert vektor u og v i sin helhet. I motsetning til de to andre gruppene valgte denne gruppen å omgjøre vektormultiplikasjonen

fremfor å forsøke og definere vektorene i sin helhet. Gruppen baserer strategivalget sitt på et imitativt resonnement, men ved hjelp av feilsøking endrer gruppen algoritmen sin på en kreativ måte. Elevene omdanner sine imitative algoritmer til kreative algoritmer.

Problemene tilknyttet definisjon av vektorene viser hvordan elevene fikserer på kjente notasjoner og definisjoner. Denne fikseringen hindrer eleven fra å resonnere fleksibelt. Fikseringen på notasjonen er så sterk at selv når elevene kreativt resonnerer frem en annen løsning, blir denne løsningen delvis forkastet grunnet fikseringen. Dette samsvarer med tidligere forskning gjort av Lithner (2006) og Haylock (1997).

Programmeringsdelen av problemet utfordrer elevens imitative resonnering i den forstand at eleven befinner seg i en ukjent situasjon. Den ukjente situasjonen fører til at elevene til dels ikke kan benytte tidligere memorerte algoritmer. Eleven må resonnere fleksibelt og unngå fiksering, noe som videre fører til kreativ resonnering. Gjennom CT må eleven kreativt resonnere seg frem til en ny algoritme. Observasjon av dekomponering av vektorer til vektorkoordinater viser hvordan elevene, i søken etter en ny algoritme, er nødt til å anvende matematisk kunnskap som sannsynligvis hadde blitt utelatt dersom programmeringen ikke inngikk i problemet. Dermed kan CT gjennom programmering bidra til å tvinge frem elevenes kreative resonnement samt inkludere et større spekter av elevenes matematiske kunnskap. Dette samsvarer med tidligere forskning på kreativitet og CT utført av Seo og Kim (2016). I tillegg støttes dette videre med Lye og Koh (2014) sin forskning på sammenhengen mellom CT og programmering. Gjennom bruk av programmering blir eleven eksponert for CT og på denne måten kan programmering utnyttes til å lære CT.

5.2 Bruk av Python-funksjoner

I implementasjon av skalarproduktformelen forsøkte to av gruppene å lage en funksjon for formelen i Python. Funksjonsbegrepet i Python er til en viss grad overlappende med det som vanligvis legges i begrepet funksjoner i matematikk, men uten at det er noe en-til-en korrespondanse mellom begrepene. Ved å lage en funksjon kan gruppene ved en senere anledning enkelt benytte funksjonen til å regne skalarprodukt. En funksjon vil dermed bidra til å effektivisere og automatisere koden. Dersom dette er forklaringen bak elevenes ønske om å lage en funksjon, er dette i tråd med Shute et al. (2017) rammeverk om CT. Gjennom algoritmisk effektivitet, design og automasjon skaper elevene en algoritme som kan generaliseres til å løse lignende problemer. Sett i kontekst med den matematiske notasjonen av

vektorer, så kan det å definere funksjoner i Python bidra til å bygge broer mellom notasjon i Python og mer konvensjonell notasjon i matematikken.

En annen forklaring på elevenes ønske om å bruke Python-funksjoner er tilknyttet elevens nylige gjennomgang av funksjoner i Python. Det er tenkelig at elevenes ønske om å lage en funksjon baseres på deres nylige gjennomgang av pensum i undervisningen. Gjennomgangen kan ha ført til at elevene tenker at det er nødvendig å benytte funksjoner for å løse problemet. Dette argumentet støttes av elevens manglende argumenter for å lage en funksjon. Argumenter som «Ja jeg tenkte vi må kanskje det» fra utdrag 11 demonstrerer dette. Dette kan være et resultat av elevenes ordinære undervisning. For eksempel undervisning som innebærer at læreren først foreleser om et bestemt tema for så delegere oppgaver til elevene i samme tema. Dette kan føre til en arbeidspraksis og problemløsningsstrategi som skaper et behov for å anvende sist tillærte kunnskap. Dette kan tolkes både positivt og negativt, alt etter hvilken kontekst elevene er i. Det kan for eksempel tolkes som meget positivt dersom elevene klarer å ta med seg essensiell informasjon om programmering ved og ta i bruk eksempler de har blitt eksponert for tidligere. På en annen side kan det tenkes at elevene kommer inn i et feil tankemønster der lærerens handlinger er så viktige at de blir bundet av dette.

Uavhengig av forklaringen bak implementasjon av funksjoner vil elevens implementasjon som kategoriseres som kreativ på bakgrunn av situasjonen elevene befinner seg i. Elevene har ikke tidligere implementert funksjoner i en situasjon som denne, og resonneringssekvensen som leder opp til valget av implementasjonen kan dermed anses som kreativt. Samtidig viser elevene en fleksibilitet omkring tilnærmingen til problemet. Dette er i tråd med definisjon av kreativ resonnering (Lithner, 2006). Sett i lys av studiens sammensatte rammeverk utarbeider elevene kreative instrukser for å gjengi løsningen til problemet, noe som kan kategoriseres som kreative algoritmer.

Under implementasjonen oppstår det flere interessante diskusjoner. Blant annet omkring navnet på funksjonen. En av gruppene forsøker å navngi funksjonen *u_v*, men får en feilmelding fra programmet. Elevene konkludere at dette skyldes navnet på funksjonen og endrer dermed dette. Elevene endrer navnet til *f(x)*, noe som viser seg å fungere. Årsaken til feilmeldingen skyldtes mangel på innverdi i funksjonen. Når elevene endrer navnet til *f(x)* oppretter de en variabel *x* som fungerer som funksjonens innverdi. Gruppen retter dermed opp feilmeldingen, men fremstår uvitende om årsaken. Under diskusjonen omkring navneskiftet stiller en elev seg kritisk til det nye navnet ettersom *f(x)* er en graf, ifølge eleven. Dette kan indikere en noe

begrenset forståelse for funksjonsbegrepet, men det kan også skyldes at eleven ikke vil lage forvirring med hensyn på hva funksjonens rolle er i akkurat denne problemløsningsøkten. Den andre eleven er enig i medelevens utsagn, men koden viser seg å fungere og elevene sier seg fornøyd med det. Diskusjonen ovenfor gir oss en innsikt i elevenes forståelse av funksjonsbegrepet. Det fremstår tydelig at elevene først og fremst forbinder funksjoner med grafer, derav graf utsagnet. Dette fører videre til at elevene ikke evner å trekke koblingen mellom funksjoner og inn- og utverdi. Dette kommer til syne gjennom elevenes mangel på input i funksjonen og elevenes mangel på bruk av innverdivariabelen x etter å ha rettet opp feilmeldingen. Denne observasjonen viser hvordan man gjennom programmering og CT kan øke elevenes matematiske kunnskap. Dette samsvarer med forskning presentert i litteraturstudien til Forsström og Kaufmann (2018). Dersom man skal få et sammensatt program til å fungere, er en nødt til å ha en god forståelse for de ulike komponentene og datastrukturene som er underliggende i programmet. Ved å arbeide med funksjoner i tekstbasert programmering kan elevenes forståelse av funksjonsbegrepet utvides. Elevene vil oppdage at funksjoner innebærer mer enn bare grafer. For eksempel kan elever gjennom CT og programmering introduseres for multivariable funksjoner og vektor funksjoner. Dette kan utføres ved å la elever utforske bevegelseslikninger med konstant akselerasjon. For å finne sluttfarten til et legeme med konstant akselerasjon kan elevene lage en funksjon med startfart, tid og akselerasjon som innverdi, og sluttfarten som utverdi.

5.3 Prøve og feile

Det var kun én gruppe som benyttet en feilsøkingsstrategi som innebar å kjøre programmet for og undersøke om programmet var korrekt. Gjennom denne metoden kunne gruppen prøve og feile helt til programmet var korrekt. Metoden gjorde at gruppen fortløpende oppdaget feil i koden. I tillegg oppdaget gruppen flere feil som de andre gruppene overså. Elevene har tidligere arbeidet lite med programmering. Det er dermed grunnlag for å påstå at prøve og feile metoden i programmeringssammenheng er ukjent for elevene. Det er dermed nødvendig å skape en ny sekvens av resonnering. På bakgrunn av dette anses gruppens feilsøking som kreativt. Dette argumentet støttes også av det faktum at de resterende gruppene ikke benyttet denne metoden.

Grunnen til at kun én gruppe benytte prøve og feile metoden kan skyldes elevenes manglende erfaring i Python. På grunn av manglende erfaring i Python kan elevene ha vært usikre på hvordan man kjører programmet og derav tester koden. Et argument som støtter denne påstanden er at gruppene ignorerte advarsler fra programmet som følger av syntaksfeil. Disse

advarslene er markert med et rødt kryss til venstre for koden i editoren. Gruppene retter riktig nok på syntaksfeil, men disse syntaksfeilene blir oppdaget av elevene selv og ikke som følge av advarsler fra programmet. Det kan være ulike årsaker til at elevene ikke oppdager at de automatisk kan få hjelp til å oppdage syntaksfeil i editoren, men det fremstår her som et potensiale for læring.

En annen forklaring på manglende bruk av prøve og feile metoden kan være at elevene resonnerer imitativt. Elevene har for vane å benytte algoritmiske resonnementer i oppgaveløsning (Lithner, 2006, s. 2). Som følger av dette gjør elevene et strategivalg som baserer seg på å følge en bestemt algoritme med en tilhørende kjent strategiimplementasjon som er triviell å gjennomføre for elevene. Dette fører til at elevene ikke har behov for å resonnere under strategiimplementasjon, og dermed heller ikke behov for å undersøke om implementasjonen er utført korrekt. Det kan tenkes at elevene ikke anser det som nødvendig å prøve koden ettersom de følger skalarproduktformelen. Det kan tolkes som at elevene har et potensiale når det gjelder å forstå sin egen læringsprosess innen programmering.

Dette funnet viser hvordan CT kan benyttes til å utvikle elevenes feilsøkingsstrategier, noe som videre kan ha en positiv effekt på deres problemløsningsevner. Feilsøking fra programmering kan generaliseres til et bredt spekter av lignende situasjoner. På denne måten kan feilsøkingsmetoder bli en del av elevens verktøykasse av problemløsningsstrategier. Dersom dette blir resultatet, er det naturlig å tenke at elevens kreativitet utvikles i takt med elevens møte med problemer som tidligere har vært utenfor utviklingspotensialet. Slik bidrar CT til å åpne nye dører og muligheter som tidligere har blitt vurdert som utilgjengelige for eleven og skoleverket. Ved hjelp av CT er det kun kreativitet og nysgjerrighet som kan avgrense spekteret av problemer Wing (2006, s. 35). Videre er det viktig å inkludere feilsøkingens bruksområde utenfor matematikk og programmering. CT representerer et ferdighetssett og en holdning som alle i samfunnet burde inneha (Wing, 2006, s. 33). For eksempel kan det å følge egne steg i søken etter forsvunnet bilnøkler kategoriseres som feilsøking. Mange programmeringsspråk har svært avanserte feilsøkingsverktøy tilgjengelig for brukeren. Det å ha fokus på hvordan disse verktøyene kan brukes i en kreativ problemløsningsmodus er et åpenbart potensiale for elever som lærer CT.

5.4 Kritisk refleksjon

Ethvert forskningsprosjekt har ulike faktorer som påvirker forskningen, dette prosjektet likeså. I etterkant av utførelsen har jeg reflektert over hvilke faktorer som kan ha påvirket studiens resultater samt hvordan tiltak som kunne blitt gjort for å styrke studiens reliabilitet og validitet.

I forkant av prosjektet var jeg tydelig på at gruppene skulle få tilstrekkelig med tid for å løse problemet. Dette for å unngå å skape et tidspress. Det ble vurdert som tilstrekkelig med 45 minutter per gruppe. Det viste seg likevel at flere av gruppene med fordel kunne hatt lengre tid. Grunnet mangel på tid fikk gruppene en begrenset mulighet til å optimalisere og justere programmet etter endt koding. Dette kan ha ført til den begrensede observasjon av komponenten iterasjon. Gruppene benyttet overraskende mye av tiden på repetisjon av oppgavens matematiske aspekt. Tidsmangelen kunne dermed blitt løst ved å ha valgt et matematisk tema som er mer kjent for elevene, eller ved å utvide gruppenes disponible tid.

Grunnet studiens omfang var det ikke ressurser nok til å inkludere et større utvalg. Utvalgets størrelse gjør at en kan stille spørsmål til studiens grad av generaliserbarhet og validitet. Valg av størrelsen på utvalget er delvis utenfor mitt handlingsrom, derimot er inndeling av parene et valg som jeg måtte ta. Dette valget kan ha påvirket elevenes resonnement. I løpet av analysen kom det frem at gruppedynamikken i de ulike gruppene varierte stort. I to av gruppene var det en elev som var langt mer aktiv enn den andre. Dette kan ha ført til at resonnementene til den passive eleven ble tilsidesatt av resonnementene til den mer aktive eleven. Med to like aktive elever kunne diskusjonen utartet seg annerledes og derav endret studiens resultater. Samtidig kan skjevfordelingen i elevenes aktivitet være et resultat av elevenes kunnskap i dette området. Dersom dette er tilfellet ville ikke nødvendigvis en endring i gruppeinndeling påvirket studien i stor grad.

I alle tilfeller er det viktig å påpeke at jeg gjennom min kvalitativ tilnærming ikke er på jakt etter generelle konklusjoner om sammenhengen mellom CT og matematisk kreativitet, men ønsker å belyse kvalitative aspekter ved elevers arbeid med problemløsning i et prosjekt hvor programmering og CT har stått i fokus.

6 Avslutning

Studiens formål var å undersøke elevers bruk av CT og kreative resonnement i problemløsning der programmering blir brukt som verktøy. Dette ledet til følgende problemstilling.

Ved hvilke kjennetegn av computational thinking resonnerer elever kreativt i problemløsning med tekstbasert programmering som verktøy?

Problemstillingen ble undersøkt ved hjelp av en kvalitativ analyse av elevers resonnement i problemløsning med tekstbasert programmering som verktøy. Analysen bygger på et sammensatt rammeverk bestående av relevant teori.

Studien antyder at Shute et al. (2017) komponenter av CT er til stedet i elevers resonnement i oppgaveløsning med programmering som verktøy. Resonnementene tilknyttet komponentene var både kreative og imitative. Komponentene algoritmer, feilsøking og dekomposisjon ble observert oftest. Det ble observert flest kreative resonnement i komponenten algoritmer.

I studien ble det observert hvordan programmering kan utfordre elevenes imitative resonnement ved å tvinge elevene til å bryte med eksisterende matematisk notasjon for å tilpasse matematikken til programmeringsspråket. På denne måten kan CT gjennom programmering bidra til å øke elevers matematiske kreativitet. På samme måte, for å tilpasse matematikken til programmeringsspråket, ble det observert at elevene måtte benytte matematisk kunnskap som trolig ville blitt utelatt dersom programmeringen ikke var inkludert i problemet. Blant annet kan programmeringens bruk av funksjoner bidra til å utvide og utvikle elevers forståelse for funksjonsbegrepet. En slik observasjon antyder at CT gjennom programmering kan ha en positiv effekt på elevers matematiske kunnskap. Observasjon omkring elevenes feilsøkingsstrategier antyder at det er behov for å forbedre elevenes feilsøkingsevner i programmering. En forbedring av elevens feilsøkingsevner i programmering kan ha en positiv effekt på elevens evner rundt problemløsning, som videre kan føre til at elevenes matematiske kreativitet økes i takt med nye og mer utfordrende problemer. Feilsøking er en ferdighet og kompetanse som kan generaliseres til et bredt spekter av situasjoner innad i matematikken, men også til andre områder. Gjennom feilsøking kan elevene opparbeide en problemløsningsstrategi som innebærer å prøve og feile, og som videre kan utvikle elevenes utholdenhet i møte med problemer.

CT inneholder mer enn bare programmering (Shute et al., 2017, s. 143; Wing, 2006, s. 35). Det er dermed ikke nødvendig med programmering for å utvikle CT, men denne studien antyder at man gjennom programmering blir eksponert for CT, og på denne måten kan programmering benyttes som et verktøy for å lære CT. Det eksisterer flere utfordringer ved å benytte programmering som verktøy for å lære CT. I denne studien ble det observert svært mange syntaksfeil i programmeringen til elevene. Tanken bak elevenes algoritme er god, men implementasjon feiler som følge av syntaksfeil. Dette kan påvirke motivasjonen til elevene i den forstand at stadige tilbakemeldinger fra programmet i form av syntaksfeil kan oppleves som negativt. Et annet negativt aspekt med mange syntaksfeil er at programmeringen kan bli tidskrevende. Dette kan resultere i en utfordring for tidspressede matematikklærer som allerede har mye pensum som skal gjennomgås. En siste nevneverdig utfordring med CT og programmering i matematikkundervisningen er lærerens kompetanse i CT og programmering. Forskning gjort av Kilhamn et al. (2021, s. 304) antyder at lærerens manglende kompetanse i programmering resulterer i undervisning om programmering uten noen tilkobling til matematikk, fremfor å benytte programmering som verktøy for problemløsning og utforskende matematikk.

6.1 Veien videre

CT og programmering er relativt nylig implementert i læreplaner verden rundt (Balanskat & Engelhardt, 2015, s. 6). Som følge av dette er det behov for mer forskning omkring CT og programmering situert i skolesituasjon (Hickmott et al., 2018, s. 64), og spesielt forskning på sammenhengen mellom CT og matematisk kreativitet. Det er i midlertidig positivt at forskningen på dette området er i sterkt vekst og at man kan forvente mange publikasjoner i nær fremtid. (Israel-Fishelson & Hershkovitz, 2022).

Denne studien er en kvalitativ undersøkelse av elevers resonnement i problemløsning med tekstbasert programmering som verktøy. Studien kan ikke si noe generelt om sammenhengen mellom kreativ resonnering og CT. Dette vil kreve større studier med mer ressurser og som gjennomføres over en lengre tid. En slik studie kan være en videreførelse av studien til Miller et al. (2013). Denne studien undersøkte om elevenes CT kan forbedres ved å kombinere CT og kreativ tenkning. Dette ble undersøkt ved å la elevene arbeide med *Computational creativity exercises (CCS),* som hverken inneholdt matematikk eller programmering. Oppgavene hadde kun som formål å forbedre elevenes CT. For eksempel innebar én oppgave å forestille seg at man nettopp hadde oppfunnet en stifter, og videre er nødt til å beskrive objektets fysiske

egenskaper, bruksområdet og hvordan objekter fungerer. På denne måten opparbeider elevene seg CT komponenter slik som algoritmer og abstraksjon.

CCS-oppgavene kan gjøres mer matematikkfokusert og fortsatt uten programmering. Da vil det være svært interessant å la et stort utvalg elever gjennomføre oppgavene over en lengre tidsperiode, og til slutt vurdere hvorvidt elevenes matematiske kreativitet har økt. Elevenes matematiske kreativitet kan vurderes ved hjelp av en standard Torrance test for kreativ tenkning (Torrance, 1974). Elevene gjennomfører testen før og etter perioden med CCS-oppgaver. For å styrke studiens validitet vil det være hensiktsmessig med en kontrollgruppe som ikke gjennomfører CCS-oppgavene, men gjennomfører Torrance testene. En slik kvantitativ studie med et stort utvalg vil kunne antyde en generell sammenheng mellom CT og matematisk kreativitet. Dette vil være et viktig bidrag til forskningen på dette området.

Referanseliste

"Zoom". (2020, 13.05.2022). *Zoom (software)*. I Wikipedia. Hentet 15.05.22 fra https://en.wikipedia.org/wiki/Zoom_(software)

Balanskat, A. & Engelhardt, K. (2015). *Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across Europe*.

Barr, D., Harrison, J. & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20-23. https://files.eric.ed.gov/fulltext/EJ918910.pdf

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads*, *2*. https://doi.org/10.1145/1929887.1929905

Beghetto, R. A. & Kaufman, J. (2010). Broadening conceptions of creativity in the classroom. *Nurturing creativity in the classroom*, 191-205. https://doi.org/10.1017/CBO9780511781629.010

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

Cuny, J., Snyder, L. & Wing, J. M. (2010). *Demystifying computational thinking for non-computer scientist* http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Dvergsdal, H. (2019, 6. november). *Python (programmeringsspråk)*. I Store norske leksikon. https://snl.no/Python_-_programmeringsspr%C3%A5k

Ericsson, K. A. & Simon, H. A. (1993). *Protocol analysis : verbal reports as data* (Rev. utg.). MIT Press.

European Commission. (2018, 13.07.18). *Coding - the 21st century skill*. https://wayback.archive-it.org/12090/20190630043709/https://ec.europa.eu/digital-single-market/en/coding-21st-century-skill

Forsström, S. E. & Kaufmann, O. T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18-32. https://doi.org/10.26803/ijlter.17.12.2

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

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

Haylock, D. (1997). Recognising mathematical creativity in schoolchildren. *ZDM*, *29*(3), 68-74. https://doi.org/10.1007/s11858-997-0002-y

Hershkovitz, A., Sitman, R., Israel-Fishelson, R., Eguíluz, A., Garaizar, P. & Guenaga, M. (2019). Creativity in the acquisition of computational thinking. *Interactive learning environments*, *27*(5-6), 628-644. https://doi.org/10.1080/10494820.2019.1610451

Hickmott, D., Prieto-Rodriguez, E. & Holmes, K. (2018). A Scoping Review of Studies on Computational Thinking in K–12 Mathematics Classrooms. *Digital Experiences in Mathematics Education*, *4*, 1-22. https://doi.org/10.1007/s40751-017-0038-8

Israel-Fishelson, R. & Hershkovitz, A. (2022). Studying interrelations of computational thinking and creativity: A scoping review (2011–2020). *Computers and education*, *176*, 104353. https://doi.org/10.1016/j.compedu.2021.104353

Israel-Fishelson, R., Hershkovitz, A., Eguíluz, A., Garaizar, P. & Guenaga, M. (2021). A Log-Based Analysis of the Associations Between Creativity and Computational Thinking. *Journal of educational computing research*, *59*(5), 926-959. https://doi.org/10.1177/0735633120973429

Kaufman, J. & Beghetto, R. (2009). Beyond Big and Little: The Four C Model of Creativity. *Review of General Psychology - REV GEN PSYCHOL*, *13*. https://doi.org/10.1037/a0013688
Kaufmann, O. T. & Stenseth, B. (2021). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*, *52*(7), 1029-1048. https://doi.org/10.1080/0020739X.2020.1736349
Kilhamn, C., Rolandsson, L. & Bråting, K. (2021). Programmering i svensk skolmatematik: Programming in Swedish school mathematics. *LUMAT: International Journal on Math, Science and Technology Education*, *9*(1), 283–312. https://doi.org/10.31129/LUMAT.9.2.1457
Knobelsdorf, M. & Romeike, R. (2008). Creativity as a pathway to computer science. Annual Joint Conference Integrating Technology into Computer Science Education,
Kunnskapsdepartementet. (2017). *Overodnet del - formål med opplæringen*. Læreplanverket for kunnskapsløftet 2020.
Lithner, J. (2003). Students' mathematical reasoning in university textbook exercises. *Educational Studies in Mathematics*, *52*, 29-55. https://doi.org/10.1023/A:1023683716659
Lithner, J. (2006). *A Framework for Analysing Creative and Imitative Mathematical Reasoning*. Department of Mathematics and Mathemical Statistics, Umeå universitet. https://books.google.no/books?id=QgdgMQAACAAJ
Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Comput. Hum. Behav.*, *41*, 51-61.
Lyngsnes, K. & Rismark, M. (1999). *Didaktisk Arbeid*. Gyldendal Akademisk.
Max Planck Institute for Psycholinguistics. (u.å). *ELAN*. Hentet 15.05.22 fra https://archive.mpi.nl/tla/elan
Miller, L. D., Soh, L.-K., Chiriacescu, V., Ingraham, E., Shell, D., Ramsay, S. & Hazley, M. (2013). *Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses*. https://doi.org/10.1109/FIE.2013.6685067
Papert, S. (1980). *Mindstorms : children, computers, and powerful ideas*. Basic Books.
Rossen, E. (2020, 23. april). *Microsoft*. I Store norske leksikon. https://snl.no/Microsoft
Seo, Y.-H. & Kim, J.-H. (2016). Analyzing the Effects of Coding Education through Pair Programming for the Computational Thinking and Creativity of Elementary School Students. *Indian journal of science and technology*, *9*(46). https://doi.org/10.17485/ijst/2016/v9i46/107837
Shell, D. F., Hazley, M. P., Leen-Kiat, S., Dee Miller, L., Chiriacescu, V. & Ingraham, E. (2014). Improving learning of computational thinking using computational creativity exercises in a college CSI computer science course for engineers.
Shell, D. F., Hazley, M. P., Leen-Kiat, S., Ingraham, E. & Ramsay, S. (2013). Associations of students' creativity, motivation, and self-regulation with learning and achievement in college computer science courses.
Shute, V., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*. https://doi.org/10.1016/j.edurev.2017.09.003
Sriraman, B. (2008). The characteristics of mathematical creativity. *ZDM*, *41*(1), 13. https://doi.org/10.1007/s11858-008-0114-z
Stenseth, B., Kaufmann, O. T. & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten - tidsskrift for matematikkundervisning*, *30*(2), 7-12.
Thagaard, T. (2009). *Systematikk og innlevelse: En innføring i kvalitativ metode*. Fagbokforlaget Vigmostad & Bjerke.

Tjora, A. (2010). *Kvalitative forskningsmetoder: i praksis*. Gyldendal Akademisk.
Tjora, A. H. (2012). *Kvalitative forskningsmetoder i praksis* (2. utg. utg.). Gyldendal akademisk.
Torrance, E. P. (1974). Torrance tests of creative thinking. *Scholastic testing service*.
Utdanningsdirektoratet. (2019, 27.03.2019). *Algoritmisk tenkning*. https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/
Utdanningsdirektoratet. (2020). *Læreplan i matematikk for realfag (matematikk R)* (MAT03-02). https://www.udir.no/lk20/mat03-02
Wing, J. (2006). Computational Thinking. *Communications of the ACM*, *49*, 33-35. https://doi.org/10.1145/1118178.1118215
Wing, J. M. (2010). *Computational thinking: What and why?* https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Vedlegg A

Vil du delta i forskningsprosjektet

Kreativt matematisk resonnement i problemløsning med bruk av tekstbasert programmering

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å undersøke kreativt matematiske resonnement når man arbeider med programmering. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Dette prosjektet er en del av en mastergradsoppgave. Prosjektet ønsker å undersøke om det oppstår kreativt matematisk resonnement når man jobber med oppgaver som inkluderer programmering i skolematematikken. På bakgrunn av dette skal følgende problemstilling analyseres «Er kreativt matematisk resonnement tilstede i problemløsning med bruk av tekstbasert programmering?»

Hvem er ansvarlig for forskningsprosjektet?
Institutt for matematikk og statistikk hos universitetet i Tromsø er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?
Du får spørsmål om å delta ettersom forskningsleder skal gjennomføre praksis ved denne skolen. Videre er du tilfeldig trukket ut til å delta.

Hva innebærer det for deg å delta?
Hvis du velger å delta innebærer det at du samtykker til å bli tatt lydopptak av og at det tas skjermopptak av dataskjermen du jobber på. Elevene blir delt inn i grupper og hver gruppe får utdelt et matematisk problem og en lydopptaker. Jeg vil bruke lydopptakene og skjermopptakene til å analysere diskusjonene i gruppene.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Det er viktig å presisere at deltakelsen i forskningsprosjektet ikke inngår i normal undervisning. Prestasjon i forskningsprosjektet vil ikke bli vurdert og vil dermed ikke ha noen effekt på elevens standpunktskarakter.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

- Det er kun forskningsleder som vil ha tilgang til dine opplysninger.
- Navnet og kontaktopplysningene dine vil jeg erstatte med en kode som lagres på egen navneliste adskilt fra øvrige data.

Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Opplysningene anonymiseres når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen er 15. mai. 2022.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:
- innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra institutt for matematikk og statistikk ved Universitet i Tromsø har NSD – Norsk senter for forskningsdata AS vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Hvor kan jeg finne ut mer?
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:
- Institutt for matematikk og statistikk ved Universitet i Tromsø ved Ragnar Soleng, ragnar.soleng@uit.no, 77644014.
- Vårt personvernombud: Joakim Bakkevold, personvernombud@uit.no, 776 46 322

Hvis du har spørsmål knyttet til NSD sin vurdering av prosjektet, kan du ta kontakt med:
- NSD – Norsk senter for forskningsdata AS på epost (personvernтjenester@nsd.no) eller på telefon: 55 58 21 17.

Med vennlig hilsen

Prosjektansvarlig
Ragnar Soleng

Student
Markus Løkke Granaas

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet *kreativt matematisk resonnement i problemløsning med bruk av tekstbasert programmering*, og har fått anledning til å stille spørsmål. Jeg samtykker til:

☐ å delta i problemløsning i grupper der diskusjonen i gruppene blir tatt opp ved hjelp av lydopptak.

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet

---

(Signert av prosjektdeltaker, dato)