16

Brugen af håndkørsler og animationer i forelæsninger om algoritmer

Mikkel Abrahamsen

Datalogisk Institut, Københavns Universitet

Indledning

I forelæsninger om algoritmer er der typisk mange studerende, som ikke får ret meget ud af forelæsningerne af forskellige årsager. I den klassiske forelæsning gennemgår forelæseren mundtligt pensum og skriver på tavlen eller viser slides mens de studerende ser passivt til. Her har de studerende ofte så travlt med at skrive noter og kopiere fra tavlen eller slides, så kun få faktisk forstår hvad der bliver forklaret (Rodger, 1995).

Derudover kan det være svært for de studerende at holde sig vågne i et mørkt auditorium hvis de skal sidde inaktivt og overvære forelæsningen. På kurser med mange studerende vil de fleste være anonyme i massen, og det medfører nemt at de bliver uopmærksomme på undervisningen, mister motivationen til at studere og i sidste ende forlader uddannelsen (Kay, 1998).

Der findes mange måder at imødegå problemet på. På Datalogisk Institut har de fleste kurser med mindst 30 studerende også øvelsestimer, hvor de studerende undervises på mindre hold og hvor undervisningen i højere grad er båret af aktiviteter som de studerende gennemfører. Dette er også anbefalet af Kay, 1998, som desuden foreslår mange andre tiltag.

I denne artikel vil jeg fokusere på hvad man kan gøre i selve forelæsningen for at de studerende bedre kan følge med når der undervises i algoritmer. En algoritme er i sig selv en beskrivelse af en aktivitet som en computer kan udføre, svarende til en nedskreven koreografi for en dans eller et partitur for et stykke musik. For at forstå hvordan algoritmen virker er det derfor meget nyttigt at se en visualisering af hvad den foretager sig i stedet for bare at læse om algoritmen eller få den beskrevet mundtligt. En visualisering

kan fx tilvejebringes med animationer eller ved såkaldte "håndkørsler" af algoritmen, hvor et menneske simulerer computeren og udfører algoritmens arbejde når den fodres med en (meget) begrænset mængde input-data. I både animationer og håndkørsler vises de vigtigste objekter som algoritmen arbejder med, ligesom objekternes indbyrdes relationer optegnes. Det illustreres hvordan objekterne og deres relationer forandrer sig mens algoritmen arbejder, således at det ønskede resultat til sidst er blevet produceret. Disse animationer og håndkørsler giver et afbræk i forelæsningen som kan være med til at gøre de studerende mere opmærksomme.

For de fleste mennesker er det nemmere at forstå en algoritme ved at se en animation end ved at læse en tekst som beskriver algoritmen, og i praksis er der gode erfaringer med bruge animationer i undervisningen. De studerende lærer mere og bliver mere engagerede (Kay, 1998; Rodger, 1996; Roßling & Freisleben, 2000). Jeg har ikke kunnet finde litteratur specifikt om håndkørsler af algoritmer, men mine egne erfaringer (til dels dokumenteret i denne artikel) er at de har en lignende positiv indvirken på undervisningen og de studerendes læring, idet en håndkørsel på sin vis bare er en animation som skabes her og nu af et menneske.

Udover i sig selv at bidrage med en ny dimension i undervisningen har animationer den fordel at de nemt kan bruges til at aktivere de studerende. Nogle animationer er dynamiske så man selv kan vælge den data, som algoritmen skal bearbejde. Her kan man bede de studerende vælge data. Bagefter kan man ændre data lidt, så man får illustreret hvordan algoritmens opførsel ændrer sig tilsvarende. Håndkørsler er også fortrinlige til at tilvejebringe denne type pointer. Andre animationer er statiske (fx en video på youtube), hvor man ikke selv har indflydelse på data. Her har man dog mulighed for at pause animationen og spørge de studerende: "hvad vil algoritmens næste skridt være?" Eller man kan pause efter et overraskende skridt og spørge: "hvorfor skete netop dette og ikke noget andet?" Håndkørsler har endvidere den fordel, at man nemt kan ændre algoritmens opførsel en lille smule og illustrere hvorfor den derved ikke længere giver det korrekte resultat. Det giver en bedre forståelse af hvorfor den rigtige algoritme er som den er.

I dette projekt vil jeg undersøge hvordan studerende opfatter at deres læring påvirkes af brugen af animationer og håndkørsler af algoritmer. Mit undersøgelsesspørgsmål er altså: Hvordan oplever studerende at brugen af animationer og håndkørsler af algoritmer i forelæsninger påvirker deres læring?

Undervisningssammenhæng

Jeg havde anledning til at eksperimentere med brugen af animationer og håndkørsler i to forskellige kurser:

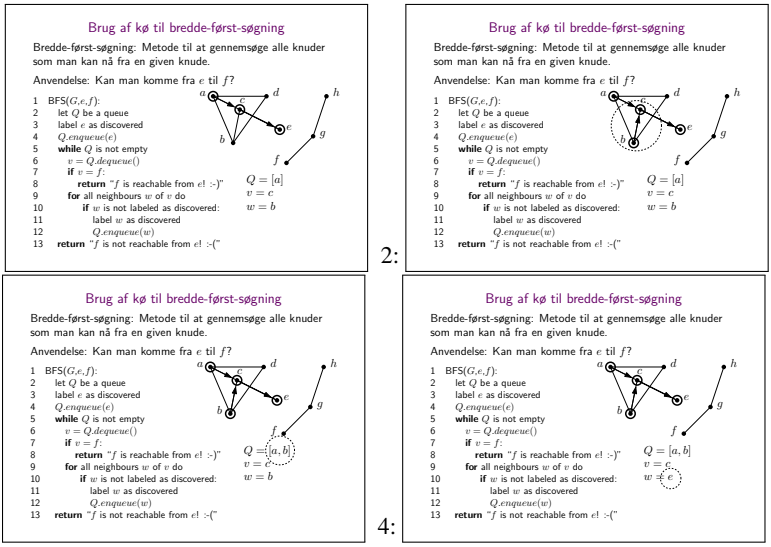
- Diskret Matematik og Algoritmer (DMA), obligatorisk kursus på første år af bachelordelen af datalogiuddannelsen, blok 1 og 2. Der var 211 tilmeldte studerende. Der er to enkeltforelæsninger og en dobbeltforelæsning om ugen, seks øvelsestimer, og to timer til lektiecafé. Jeg gav forelæsningerne i to af ugerne.
- Advanced Algorithms and Data Structures (AADS), obligatorisk kursus på første år af kandidatdelen af datalogiuddannelsen, blok 1. Der var 149 tilmeldte studerende. Der er to dobbeltforelæsninger og fire øvelsestimer om ugen. Jeg gav tre af dobbeltforelæsningerne.

Min brug af animationer og håndkørsler

DMA

I kurset DMA var jeg så at sige blot vikar, idet jeg havde overtaget de to undervisningsuger fra en kollega. Jeg fik udleveret slides og ugeplaner og det lå helt fast hvilket pensum, der skulle gennemgås. De udleverede slides var ret kedelige og lagde ikke meget op til hverken animationer eller håndkørsler. Pensum i de to uger, hvor jeg skulle undervise, handlede om nogle meget basale datastrukturer og en teknik til at analysere sådanne (såkaldt amortiseret køretidsanalyse). En *datastruktur* er en måde at organisere dataen i en computer på, som gør det muligt at tilgå og modificere dataen effektivt. For at en algoritme kan udføre et stykke arbejde på en mængde data, skal dataen være lagret på en måde så algoritmen kan læse den og foretage de nødvendige udregninger og modifikationer. Derfor fungerer algoritmer i et tæt samspil med datastrukturer, og effektive datastrukturer er en forudsætning for effektive algoritmer, men der er ikke så meget algoritmisk over datastrukturen i sig selv – det kommer først når den bliver brugt af en algoritme.

Af ovenstående grund kan det synes lidt ligegyldigt udelukkende at lære om datastrukturer i en hel uge – man må have blik for de mangfoldige algoritmiske problemer, som datastrukturerne gør det muligt at løse før man kan se en mening med dem. Derfor bestræbte jeg mig på at tilføje eksempler på algoritmer som anvender de datastrukturer, som blev gennemgået. Min



Figur 16.1. Fire på hinanden følgende slides fra min animation af hvordan man afgør om der er en vej mellem to knuder i en graf G med en dybde-først-søgning, som er en algoritme der anvender en kø Q – en af de basale datastrukturer som skulle gennemgås da jeg underviste på kurset DMA. De stiplede cirkler viser hvor der er sket en forandring i forhold til foregående slide og er tilføjet af hensyn til læseren af denne artikel. Hele animationen består af 35 slides, og hvis man ikke går i detaljer med hver enkelt slide, kan man løbe dem igennem hurtigt og derved opnå en tegnefilmsagtig effekt.

tanke var at det ville sætte pensum i en kontekst, som kunne gøre det lettere for de studerende at forstå vigtigheden af datastrukturerne. Udfordringen var at jeg måtte tage forskud på pensum som hørte til i senere uger af kurset eller senere kurser på studiet. Derfor skulle min gennemgang af disse algoritmer være kursorisk og ikke tilbundsgående. De studerende blev gjort opmærksom på at de ikke forventedes at forstå alle detaljer, men at det blot var ment som en motivation for og illustration af at det gennemgåede pensum var vigtigt stof.

I det følgende giver jeg et eksempel på noget af det ekstra undervisningsmateriale, som jeg udviklede til en dobbeltforelæsning i kurset. En af de datastrukturer der skulle gennemgås er en såkaldt *kø*, som kan holde styr

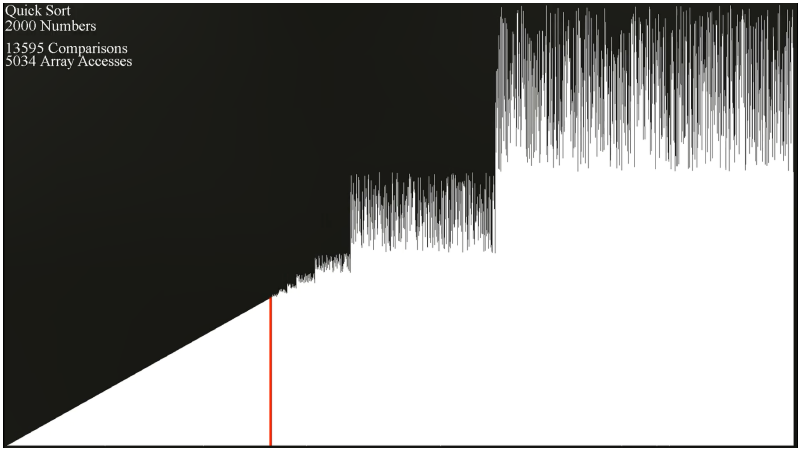
på en mængde af data som venter på at blive håndteret af en algoritme. Data kan puttes ind i køen og data kan trækkes ud af køen. Den data der kommer ud er altid den, der har "stået i kø" i længst tid, deraf navnet "kø". Køer er anvendelige i et hav af forskellige situationer, hvoraf en af de vigtigste er i algoritmer til at finde veje i grafer (en *graf* er den tekniske betegnelse for det man i daglig tale kalder et netværk). Derfor valgte jeg at lave et ekstra sæt slides som indeholder en animation af hvordan en algoritme kaldet *dybde-først-søgning* anvender en kø til at afgøre om der findes en vej mellem to knudepunkter i en graf. Jeg lavede en slide for hver operation som algoritmen udfører så man kan følge alle algoritmens skridt. Fire på hinanden følgende af disse slides kan ses i Figur 16.1. Hver slide viser pseudokoden til algoritmen i venstre side, altså en beskrivelse af hvad algoritmen foretager sig i en form der minder lidt om hvordan algoritmen ville se ud hvis man skrev den ned i et programmeringssprog. I højre side ses den graf som algoritmen analyserer, og indholdet af algoritmens variable – det er altså kun her der sker forandringer. Algoritmen finder de knuder hvorfra man kan nå knuden e. Efterhånden som algoritmen arbejder, markeres de knuder der kan nå e med en ring, og kanterne erstattes med pile som viser hvordan man kommer hen til e. På den måde viser animationen hvor meget af grafen algoritmen har processeret på ethvert tidspunkt mens den arbejder.

Dybde-først-søgning er en del af pensum på kurset, men ifølge planen først mange uger efter de studerende skal lære om køer.

AADS

I den ene dobbeltforelæsning skulle jeg gennemgå en sorteringsalgoritme som hedder *quick-sort*. Efter at jeg havde beskrevet algoritmen vha. pseudokode gennemførte jeg en håndkørsel på tavlen. Derefter viste jeg en bid af en video fra youtube med en animation af hvordan den sorterer 2000 tal[1]. Et billede fra videoen kan ses i Figur 16.2. Til sidst gennemførte jeg en matematisk analyse af algoritmens forventede køretid.

[1] Videoen tog jeg fra https://www.youtube.com/watch?v=xoR-1KwQh2k&t=413s fra tid 6:53 frem til 7:07, og viste den i langsom gengivelse (kvart hastighed).



Figur 16.2. Billede fra den video jeg viste da jeg forelæste om quick-sort. Videoen er en animation af hvordan algoritmen sorterer de 2000 tal $\{1, 2, \ldots, 2000\}$. Før sorteringen begynder er tallene blevet blandet i en tilfældig "roddet" rækkefølge. Hvert tal $x \in \{1, 2, \ldots, 2000\}$ er repræsenteret med et lodret hvidt linjestykke som er x pixels langt og har det nederste endepunkt i bunden af billedet. Når algoritmen arbejder, bytter den tallene rundt, så de til sidst former en hvid trekant begrænset fra oven af den stigende diagonal i billedet. Lidt simplificeret kan man sige at tallene til venstre for den røde lodrette linje allerede er blevet sorteret, mens tallene til højre er blevet sorteret groft i grupper sådan at ethvert tal i én gruppe er mindre end ethvert tal i gruppen til højre for.

I samme forelæsning skulle jeg gennemgå en algoritme til at finde et minimum cut (forkortes min. cut) i en graf, dvs. en mængde af så få kanter som muligt der gør grafen usammenhængende. Efter at have beskrevet algoritmen med pseudokode, gennemførte jeg en håndkørsel på tavlen. Algoritmen er randomiseret, dvs. at dens opførsel afhænger af tilfældige tal som den skal forsynes med. Til min håndkørsel fik jeg ægte tilfældige tal fra siden https://www.random.org/, idet dette gav anledning til en god dialog med de studerende om hvordan man skaffer tilfældige tal når man benytter sig af randomiserede algoritmer. Computere er jo deterministiske af natur (dvs. medmindre de er i stykker opfører de sig altid ens i to givne ens situationer), så tilfældige tal kan kun komme fra en ekstern kilde.

I de andre dobbeltforelæsning jeg gav i AADS, fokuserede jeg ligeledes på at supplere de gennemgåede algoritmer med illustrative håndkørsler. Alle håndkørslerne lavede jeg ved aktivt at inddrage de studerende. Således stillede jeg ofte de studerende spørgsmål som: "hvad er algoritmens næste skridt?", eller: "hvorfor sker der dette og ikke dette?" Andre gange bad jeg også de studerende om at foreslå det input, som vi skulle håndkøre algoritmen på. Min tanke var at de ville være mere interesserede når de fik medbestemmelse over forelæsningen.

I alle disse tilfælde afsluttedes gennemgangen af den pågældende algoritme med en matematisk analyse af algoritmens køretid. Dette er den mest langhårede del af forelæsningen og oftest den del, der tager længst tid at gennemføre, og hvor man som forelæser kan have den største frygt for at mange studerende falder fra. Det er her man kan håbe på at animationer og håndkørsler kan støtte de studerendes forståelse godt nok til at de kan følge den matematiske analyse.

Kvalitative interviews

Jeg udførte kvalitative interviews med studerende efter tre dobbeltforelæsninger, én i DMA (den beskrevet i Afsnit 16) og to i AADS (den beskrevet i Afsnit 16 hvor jeg både viste en animation og lavede håndkørsler samt en af de andre hvor jeg kun lavede håndkørsler). Hver dobbeltforelæsning bestod af to forelæsninger på hver 45 minutter adskilt af en pause på 15 minutter. I alle tre tilfælde spurgte jeg i pausen fire eller fem vilkårligt udvalgte studerende, som opholdt sig i auditoriet, om de umiddelbart efter forelæsningen havde tid til svare på nogle få spørgsmål om hvad de lærer meget af og knap så meget af. Jeg valgte at indsamle kvalitative data med gruppeinterviews dels fordi det var nemmere at arrangere end individuelle interviews (jeg skulle ikke aftale individuelle møder med mange studerende) og fordi jeg forestillede mig at de studerende kunne inspirere hinanden med deres synspunkter. Jeg tænkte at forskellige oplevelser og opfattelser blandt de studerende ville komme tydeligere frem i en fælles samtale end ved individuelle møder.

Man kan indvende at der var en risiko for at de studerende ikke svarede ærligt når jeg som deres forelæser interviewede dem – måske ville de ikke føle sig godt tilpas med at fortælle mig hvis der var en del af forelæsningen, de ikke fik noget ud af. Imidlertid havde jeg under vores interviews en

fornemmelse af at de fleste studerende talte helt frit fra leveren og ikke var bange for at sige hvis de syntes at noget ikke fungerede.

Hvert interview foregik på den måde at jeg først skrev et overordnet program for den netop overståede forelæsning op på et whiteboard, sådan at det var nemt at navigere rundt i hvilken del af forelæsningen vi talte om. Her er et eksempel fra en af AADS-forelæsningerne, hvor der blev gennemgået to algoritmer:

Randomized quick sort	**Randomized minimum cut**
Description	Description
Handrunning	Handrunning
Quiz	Analysis
Animation	Quiz
Analysis	Conclusion

Først ville jeg have at vide om de studerende læste før forelæsningen eller brugte forelæsningen som forberedelse til at læse selv bagefter. Jeg syntes det var vigtig information at sammenholde med den studerendes oplevelse af vedkommendes læringsudbytte af forelæsningens forskellige dele. Det var som forventet meget forskelligt hvordan de studerende greb det an. Mange havde læst de relevante kapitler fra lærebogen på forhånd. Der var også nogle som ville læse efterfølgende, og nogle som skimmede teksten før forelæsningen og læste den grundigt efter.

Jeg havde forberedt at vi derefter skulle rundt om de følgende spørgsmål:

- Hvor lærte I mest?
- Hvor lærte I mindst?
- Hvad fik I ud af håndkørslerne?
- Hvad fik I ud af animationen?
- Forstod I pseudokoden bedre efter håndkørslen? – Hvorfor (ikke)?

I praksis behøvede jeg ikke at stille alle disse spørgsmål, for efter det første spørgsmål begyndte i hvert interview en helt naturlig samtale så jeg efterfølgende kun behøvede at stille få opfølgende spørgsmål. I det følgende afsnit samler jeg de vigtigste pointer som jeg fik ud af disse interviews.

De studerendes vigtigste pointer

Jeg har forsøgt at tematisere de kommentarer jeg fik fra de studerende i tre vigtige pointer, som er beskrevet herunder.

Animationer kan motivere datastrukturer uden for pensum

Det materiale jeg udviklede til DMA for at motivere pensum, beskrevet i Afsnit 16, havde den ønskede effekt. Jeg fik følgende feedback fra en studerende: *"Det er virkelig fedt at se de slides med bredde-først-søgning. Det fik jeg meget ud af. De andre slides var sådan lidt kedelige."*

En anden studerende supplerede: *"Det var spændende at se at køer kunne bruges til noget vigtigt."*

Den første studerende tilføjede: *"Jeg synes godt jeg kunne forstå hvordan algoritmen virkede da jeg så animationen, selvom vi ikke lærte alle detaljerne."*

Jeg er altså blevet bekræftet i at det var en god idé at komme med noget ekstra motivation for det obligatoriske pensum, samt at animationer kan give de studerende en anden og mere intuitiv forståelse for algoritmerne end en formel gennemgang og matematisk analyse.

Brugen af håndkørsel og animation til quick sort

Om forelæsningen om quick sort, beskrevet i Afsnit 16, herskede der mere uenighed om nytten af animationer og håndkørsler. Dette bundede til dels i, at mange af de studerende allerede var bekendte med quick sort-algoritmen. Det var jeg ikke klar over inden forelæsningen. De havde dog ikke i tidligere kurser set en gennemgang af den matematiske analyse for køretiden, så jeg burde nok ikke have brugt lige så meget tid på at forklare hvordan algoritmen virker med håndkørsel og animation, og i stedet brugt mere tid på analysen.

"Quick sort har man set så mange gange før."

Desuden fik jeg denne kommentar om den konkrete type af animation:

"Det er lidt en blæreanimation hvor det går helt vildt hurtigt. Det er godt at se hvordan den splitter op, men jeg er mere til animationer hvor man ser hvert step. Jeg får mere ud af mindre eksempler (dvs. mindre input) og langsommere steps."

Jeg foretog både en håndkørsel og viste en animation af quick sort, og om dette fik jeg denne kommentar:

"Det er dobbeltkonfekt at se animationen når man har set en håndkørsel."

Af disse kommentarer har jeg lært at det måske er overflødigt *både* at lave en håndkørsel og vise en animation, samt at animationer af hvordan algoritmen kan arbejde på store input måske ser flotte ud, men ikke nødvendigvis giver de studerende så meget. Derudover skal man være opmærksom på

hvor godt de studerende kender til algoritmerne i forvejen, og give størst opmærksomhed til de dele som ikke har været pensum på tidligere kurser.

Der var dog også en studerende, som sagde:

"Det er sjovt, jeg googlede faktisk i morges sådan en online visualisering af quick sort. Fordi man læser hvordan det virker, men lige sådan hvordan det ser ud med stort input, det er svært at forestille sig."

Det tyder på at den type animationer alligevel kan være til gavn for visse studerende (jeg er ikke i stand til at karakterisere hvilke studerende dette gælder for, men den pågældende studerende forekom mig at være talentfuld, arbejdsom og særligt interesseret).

Nogle, men ikke alle, studerende er glade for håndkørsler

Det virker som om mange studerende er glade for at se håndkørsler af algoritmerne, hvilket jeg underbygger med disse kommetarer:

"Jeg var rigtig glad for håndkørslen af min. cut. Jeg havde svært ved at fange det fra læsestoffet hvad der foregik." (*)

"Det er rart med en håndkørsel inden analysen, fordi man kan sammenholde analysen med håndkørslen, så man forstår hvad de forskellige definitioner i analysen betyder helt konkret fra et eksempel. Når man laver håndkørslen kan man definere de ting som bliver vigtige i analysen. Så er det nemmere at forstå end en formel definition."

"I min. cut fangede jeg først rigtig hvad det var, der foregik, efter at jeg havde set håndkørslen."

"Pseudokoden var ikke så klar før man havde det visuelt."

De studerende, der læser efter forelæsningen er glade for håndkørslerne:

"If you hear about the algorithm for the first time, the examples[2] *are, I think, the most important part to the understanding. But when you did the reading before, you can't get more from it. I like to read after the lecture, so that I know what I should focus on. So the examples are very important for me. That's when I understand the pseudocode. Before the examples, I sometimes can't understand the pseudocode."*

Derimod får nogle af de studerende, der læser før forelæsningen, ikke meget ud af håndkørslerne:

"Jeg fik ikke rigtig noget ud af at få gennemgået algoritmerne med eksempler. Jeg havde læst på forhånd og gennemgået eksempler i bogen og var i gang med at lave opgaverne, så der havde jeg fået forståelsen. Det jeg

[2] Et af mine interviews var på engelsk, hvor vi brugte ordet *example* om håndkørsel.

fik mest ud af var gennemgangen af analysen. Der sad jeg fast i opgaverne, men nu forstår jeg hvordan man laver et bevis for køretiden."

"Usually, I already read the description and went through examples before attending the lecture, so I don't get much out of it. That's the easy part of the curriculum. The hard part is the analysis, and that's where I need the lecture. The examples are more like a validation that I got it right. The analysis is usually also where it gets a bit rushed because it's at the end, so I would prefer to have more time for that and less for examples. The ability to go up to you in the break and ask questions, that's where I get the most out of it."

Jeg tror de foregående to kommentarer kom fra ret dygtige og selvkørende studerende, som forstår det meste når de læser det på egen hånd i bogen. Andre studerende har glæde af håndkørsler, selvom de har læst på forhånd, (gen)se fx kommentaren markeret med (*).

Konklusion

Jeg er blevet bekræftet i at animationer kan bruges til at give de studerende en intuitiv forståelse for algoritmer. Animationer kan derfor bruges til at styrke de studerendes forståelse. Desuden kan animationer af algoritmer, som ligger uden for pensum, bruges som et farverigt indspark til at motivere pensum så det bliver mere relevant. Her kan animationen stå næsten alene – de formelle detaljer kan gemmes til senere på uddannelsen.

Jeg er blevet gjort opmærksom på at nogle studerende foretrækker animationer hvor algoritmen kører på input af lille størrelse, så man kan gennemgå algoritmens skridt ét ad gangen. I det tilfælde er der ikke stor forskel på en animation og en håndkørsel.

Man skal være omhyggelig med at undersøge hvor godt de studerende kender til algoritmerne fra tidligere kurser og fokusere på det, der er nyt. Hvis de allerede kender en algoritme, kan man med fordel hurtigt skitsere hvordan den virker, og så fokusere på det, der er nyt (fx den dybdegående analyse som i tilfældet med quick sort). Desuden synes nogle studerende det er overflødigt med både håndkørsler og animationer af samme algoritme.

Nogle studerende læser grundigt før forelæsningen og har derfor ikke brug for håndkørslerne, men har mest glæde af den efterfølgendende analyse. Andre har dog glæde af håndkørslerne selvom de læste før, og de studerende som læser efter forelæsningen er meget glade for håndkørslerne. Derfor er min overordnede konklusion, at det er en fin balancegang hvor mange

animationer og håndkørsler man skal medtage i forelæsningerne. Det skal afvejes efter hvor godt de studerende kender til den pågældende algoritme (eller lignende algoritmer) på forhånd, hvor kompliceret den er at forstå ud fra lærebogens beskrivelse, og hvor lang tid der skal sættes af til at gennemgå den efterfølgende analyse, som ofte er dét, de studerende har de største vanskeligheder med at tilegne sig.

Perspektivering

Det er nemt at variere og tilpasse den måde, man gennemgår håndkørsler på, fordi man bare kan tegne og skrive på tavlen på en anden måde hvis man finder det nødvendigt. Derimod kan det være endog meget tidskrævende at udvikle nye animationer. I dette projekt har jeg kun udviklet én animation, beskrevet i Afsnit 16, hvorimod jeg i et andet tilfælde har brugt en animation fra youtube. I flere af de andre algoritmer, jeg gennemgik på AADS-kurset kan jeg forestille mig at man kunne lave meget lærerige animationer af algoritmerne, men jeg har ikke haft tid til at udvikle dem og har ikke fundet tilfredsstillende videoer eller andet materiale jeg kunne bruge. Derfor er der et stort potentiale for at udvikle flere animationer og undersøge deres gavnlige effekt.

Litteratur

Kay, D. G. (1998). Large introductory computer science classes: Strategies for effective course management. *ACM SIGCSE Bulletin*, *30*(1), 131–134.

Rodger, S. H. (1995). An interactive lecture approach to teaching computer science. *ACM SIGCSE Bulletin*, *27*(1), 278–282.

Rodger, S. H. (1996). Integrating animations into courses. *ACM SIGCSE Bulletin*, *28*(SI), 72–74.

Roßling, G. & Freisleben, B. (2000). Experiences in using animations in introductory computer science lectures. *ACM SIGCSE Bulletin*, *32*(1), 134–138.