

Villedning som virkemiddel til læring og kritisk holdning, observasjoner fra et undervisningseksperiment

Kjell Ellingsen
Handelshøgskolen i Bodø
Høgskolen i Bodø
kjell.ellingsen@hibo.no

Sammendrag

Artikkelen beskriver et undervisningseksperiment i forbindelse med et kurs i Algoritmer og datastrukturer. Under veiledning på en innleveringsoppgave ble studentene flere ganger bevisst ledet inn mot løsninger som viste seg å være blindspor. Målene var å aktivisere studentene samt gi dem repetisjon av større pensumdelar. Artikkelen beskriver de ulike trinn mot en løsning og relaterer kunnskapsbehovet ved hvert trinn mot en taksonomi for kunnskapsnivå. Til slutt reflekteres det over mulige årsaker til at studentene viser liten kritisk evne under eksperimentet.

Innledning

Jeg går først gjennom en del problem knyttet til algoritmekurset og de tiltak vi har prøvd ut tidligere. Deretter beskriver jeg en taksonomi for kunnskap før jeg går over til selve undervisningseksperimentet. Her beskrives de enkelte trinn ganske detaljert og det gjøres løpende en kort vurdering av studentenes handlingsmønster.

Problem

De fleste informatikkstudier har minst ett kurs i algoritmiske metoder og datastrukturer. Studentene opplever disse kursene som meget utfordrende og de har ofte en høyere strykeprosent enn andre kurs. Hva kan være årsak til dette? Jeg har konstatert at slike kurs fordrer gode forkunnskap i matematikk og programmering. De introduserer et sett av til dels komplekse datastrukturer samt mer eller mindre intrikate algoritmer. Og læringskurven er bratt ettersom teoristoff fra et kapittel gjerne er grunnleggende for forståelsen av, og brukes aktivt i det neste.

Tidligere tiltak

For å møte disse utfordringene hadde vi ved Høgskolen i Bodø allerede innført krav om at studenten måtte ha bestått kurs i programmering og diskret matematikk før de får ta algoritmekurset. Vi hadde også økt læringsperioden ved å strekke kurset over to semestre. For å sikre jevnere innsats hadde vi videre innført to gruppeoppgaver underveis og latt disse telle med i endelig karaktersetting.

Endret undervisningsopplegg og utvidelse av problemstilling

Tradisjonelt har undervisningen i algoritmekurset vårt bestått i ordinære forelesninger, ukentlige lab-øvinger samt to mer omfattende innleveringsoppgaver hvor studentene jobber gruppevis og får veiledning kun på forespørsel.

Etter ønske fra studentene endret vi litt på undervisningsopplegget i det aktuelle semesteret – forelesningsdelen ble redusert noe til fordel for problemløsning i plenum.

Alt under de første problemdiskusjonene, la jeg tidlig merke til at studentene hadde en lite kritisk holdning til de forslagene jeg kom med. Denne observasjonen førte til at jeg parallelt med problemløsningen, ville observere og analysere studentenes adferd nærmere.

Innledende analyse av problemet

Under problembeskrivelsen har vi vært inne på at kurset krever kunnskaper og innsikt på ulike nivå. På den ene side er veien fra å *forstå* en algoritme i form av å innse at den har den ønskede effekt på en gitt inndatamengde til å kunne *anvende* den på en kritisk måte og *modifisere* eller selv *konstruere* den, er temmelig lang. Her er rekursjon i seg selv et godt eksempel. Vi husker alle vår egen utvikling fra den innledende tvil om at en slik algoritme faktisk gjorde det den var ment til, via et sett av overbevisende eksempler hvor rekursjonstrinnene ble ekspandert og fram til en helhetlig innsikt om at en basis samt et rekursjonstrinn er tilstrekkelig for å forstå rekursjonens hemmelighet.

På den annen side ser vi at selv om algoritmekurset er teoretisk orientert, er det de færreste studenter som klarer å oppnå forståelse av pensum kun gjennom litteraturlitnærming. Pensumlitteratur og teoriforelesninger må suppleres med praktiske, programmeringsorienterte øvinger. Ulempen med dette er imidlertid at studentene må mestre flere abstraksjonstrinn samtidig, fra konkrete detaljer i programmeringsspråket til det å kombinere løsninger fra ulike teorikapitler. I tillegg til disse utfordringene måtte studentene i mitt eksperiment forholde seg til (egentlig vurdere) forslag som ble spilt inn fra faglærer. Også andre forskere (Woodley, 2007) har påpekt at informatikk som fagområde har spesielle utfordringer i og med at studentene skal gis både en teoretisk bakgrunn samtidig som de skal oppøve utøvende ferdigheter.

Taksonomi

Under tidligere arbeid med studieplaner, har jeg avdekket behov for et mer nyansert begrepsapparat i forbindelse med kunnskap og læring. En taksonomi for et område gir en ordliste, ei klassifisering eller ei ordening av de mest sentrale begrepene innen feltet. Den best kjente taksonomi er sannsynligvis Carl von Linnés system for klassifisering av organismer innen biologifeltet.

Bakgrunnen for å benytte en taksonomi i undervisningssammenheng er både for å sikre bevissthet om valg av kunnskapsnivå ved målformulering for kurs og studier, for lettere å kunne kommunisere disse til faglærere og studenter og for å danne grunnlag for konkrete undervisningsopplegg i det/de aktuelle kursene. Vi har benyttet Blooms taksonomi i denne sammenheng.

Blooms taksonomi

I følge (Valdermo, 1997) er denne taksonomien den mest som benyttes mest innen fagområdet pedagogikk. Så tidlig som i 1956 utviklet ei gruppe ledet av Benjamin Bloom en hierarkisk klassifikasjon som deler kunnskap inn i seks nivå. Den framstilles gjerne ved ei kunnskapstrapp hvor hvert trinn baseres på det underliggende trinnet. Framstillingen nedenfor er basert på (NTNU, 2007).

De enkelte trinnene betegnes og kjennetegnes som følger: 1: *Hukommelse* – en kan gjenfortelle det som står i læreboka med lærebokas ord og uttrykk; 2: *Forståelse* – kandidaten kan gjengi lærestoffet med egne ord, trekke sammenbindende linjer mellom ulike deler av fagstoffet, påpeke likheter og forskjeller; 3: *Anvendelse* – kjennetegnes av at innlært stoff kan brukes til å løse oppgaver på en standardisert måte; 4: *Analyse* – Innlært stoff brukes til å dele opp komplekse oppgaver, skille disse fra hverandre og løse delproblemene hver for seg; 5: *Syntese* – Koble sammen innlært stoff fra ulike deler av faget til å løse komplekse oppgaver på en original og situasjonstilpasset måte; 6: *Vurdering* – Gi uttrykk for egne refleksjoner omkring kunnskapens validitet og nytteverdi.

Casen

Oppgaven er knyttet opp mot et prosjektplanleggingsproblem. Utgangspunktet er et sett av aktiviteter med angitt tidsforbruk og ressursbehov samt avhengigheter mellom aktivitetspar. Ut fra disse opplysningene skal en først beregne tidligste start/tidligste slutt, seneste start/seneste slutt, fri- og totalslakk for hver av aktivitetene uten ressursbeskranking.

Hensikt

Resultatmålet for oppgaven var altså et kjørbart program som gitt en konkret input skulle generere aktivitetsorienterte tidsplaner.

Opprinnelig var det primære delmålet at studentene gjennom praktisk anvendelse skulle få dypere innsikt i grafteoridelen av kurspensum. At de gjennom implementasjonsarbeidet skulle oppøve bedre programmeringsferdigheter var også et delmål. Refleksjoner underveis førte imidlertid til at målsettingen for læringseffekten ble mer nyansert.

Gjennomføring

Etter opprinnelig plan skulle studentene få utlevert en oppgavetekst med tilstrekkelige opplysninger og detaljeringsgrad for å sikre funksjonalitet til sluttproduktet. Denne teksten ga lite direkte veiledning med hensyn til løsning, men gjennom delspørsmål ble studentene ledet til å resonnerer rundt problemstillingen, valg av angrepsvinkel, datastrukturer og algoritmevalg. På bakgrunn av denne oppgaveteksten skulle studentene leverer inn et ferdig arbeide etter to uker.

Da vi besluttet å bruke deler av forelesningstiden til problemløsning, valgte jeg denne oppgaven som utgangspunkt for diskusjonene. Som følge av det ble oppgaven introdusert noe tidligere enn opprinnelig planlagt, noe som igjen førte til at den i sin fulle form krevde innsikt i deler av pensum som ennå ikke var gjennomgått. Undervisningen fikk dermed et islett av problembasert læring, som i følge (Mohus, 2005) kjennetegnes ved at *”læringsoppgaver og kunnskapssøk fordeles til deltakerne i pbl-gruppa som bearbeider disse og i neste runde bringer resultatene inn i undervisningen”*.

For å unngå at disse ”problemløsningstimene” ikke skulle ende opp med at jeg som faglærer kun presenterte en ”mønsterløsning”, ble det i de første timene arbeidet etter ”prøve-og-feile”-metoden. Ja, studentene ble bevisst veiledet (egentlig villedet) inn på blindspor i den hensikt å få belyst større deler av pensum. Som det går fram av den trinnvise beskrivelsen fram mot løsning, lot de seg lede inn i samme blindgate flere ganger. Dette vakte ettertanke og ga grunnlag for spørsmålene som reises mot slutten av denne artikkelen.

Trinn og feiltrinn fram mot løsning

I dette avsnittet vil jeg først beskrive prosessen fram til løsning i ganske detaljerte trinn. For hvert av disse trinnene antydes det hvilket kunnskapsnivå som ligger under beslutningen. I den oppsummerende analysen vil jeg vende tilbake til enkelte trinn for å peke på mulige forklaring på handlingsmønster og hvilke spørsmål disse igjen reiser.

Grafproblem

Samlingen av aktiviteter og avhengighetsrelasjoner mellom disse, vakte umiddelbart assosiasjoner til grafer og studentene var samstemte om å velge denne datastrukturen. Vi ser her et eksempel på anvendelse av innlært stoff noe som tilsvarer trinn tre i

Blooms kunnskapstrapp. Slik oppgaven presenterer opplysningene, faller det naturlig å knytte tidsforbruk til aktivitetene. De fleste vil følgelig legge tid som et attributt sammen med aktivitetsidentifikasjon og ressursbehov til en klasse og la denne utgjøre nodene i grafen. Kantene vil da kun representere avhengighetsforholdet mellom aktivitetene. Denne representasjonen tilsvarer det man innen prosjektadministrasjonen betegner et aktivitetsorientert nettverk eller et blokknettverk.

Konvertering til pilnettverk

Problemet med et aktivitetsorientert nett i denne sammenhengen er at avbildningen mot den grafrepresentasjonen studentene har fått i kurset blir noe krevende, man får en modellavstand som gjør at standardalgoritmer må omarbeides før bruk. Den grafrepresentasjon studentene var fortrolig med, fra (Weiss, 1999) er gitt $G(V, E)$, en nodemengde V og en kantmengde E hvor kostnader (veilengde, ressursforbruk og lignende) er knyttet til kantmengden. Dette tilsvarer hendelsesorientert nett eller pilnett som de kjenner fra prosjektadministrasjon.

Konverteringen fra blokk- til pilnett er imidlertid ikke helt triviell. Nodene i pilnettet representerer hendelser (at en aktivitet er avsluttet) og kantene avbilder aktiviteter. For alle aktiviteter med mer enn en forgjenger må man derfor innføre en dummy-aktivitet for å kunne representere hver av forgjengerne eksplisitt. Denne konverteringen behandles også i læreboka og studentene ble også henvist til denne pensumdelen. I og med at konverteringen var ikke-triviell ble studentene bedt om å vurdere merarbeidet opp mot å løse oppgaven basert på et hendelsesorientert nett.

Vurderinger gjort i ettertid, viser at vi her beveger oss opp på trinn 5 og 6 i Blooms kunnskapstrapp i og med at oppgaven krever analyse av en ikke-kjent anvendelse og vurdering av løsningsalternativ. Det var derfor naturlig at vi endte opp i en styrt diskusjon nært knyttet opp mot faktaopplysninger i form av konkrete algoritmebeskrivelser fra pensumlitteraturen.

Etter at studentene hadde lest den aktuelle pensumdelen, diskuterte vi fordeler og ulemper ved de to alternative løsningene i plenum. Drøftingene endte opp med et flertall for å benytte et aktivitetsorientert nettverk hvor en eventuelt måtte modifisere standardalgoritmene før bruk. Også fra et læringsmessig perspektiv blir valget ansett som en fordel i og med at en slik omarbeidelse ville kreve en inngående forståelse av disse algoritmene framfor en mer passiv bruk av dem ("syntese framfor anvendelse").

Sentinel-element

Beskrivelsen som ble gitt for å beregne tidligste start og seneste slutt baserte seg på opplysninger om alle forgjengere og etterfølgere til den aktuelle aktiviteten. I aktivitetsmengden vil det kunne være mer enn ett element som ikke har noen forgjenger og likedan flere som ikke har etterfølgere. At grafen ikke bare har én startnode kompliserer beregningene noe, algoritmen for beregning av tidligste start må da startes om igjen for alle aktiviteter uten forgjengere mens seneste slutt for aktiviteter uten etterfølgere vil være avhengig av den seneste aktiviteten i denne gruppa. Spørsmålet til studentene var: Hva gjør vi for å unngå disse spesialtilfellene? Teknikken med å benytte ekstraelement (sentinel- eller dummyelement) er kjent fra tidligere pensumdelene siden det var benyttet i forbindelse med linelære lister.

Ved å trekke inn ideen om sentinel-element og antyde hvordan disse kunne anvendes i ny sammenheng reduserte jeg problemet fra med to trinn – til anvendelse av kjent kunnskap på en standard måte. Løsningen ble å innføre en start- en sluttaktivitet som henholdsvis forgjenger og etterfølger til dem som ikke hadde en slik fra før.

Algoritmebygging

Grunnen skulle nå være beredt for å gjenbruke/tilpasse eksisterende grafalgoritmer eller konstruere genuint nye algoritmer for å beregne de angitte mål: Tidligste start/ tidligste ferdig (TS/TF), prosjektid og seneste start/seneste ferdig (SS/SF).

Initiering: TS og TF for startaktiviteten settes til 1. For de øvrige aktiviteter a , er $TS_a = \max(TF_f + 1)$ hvor $(f, a) \in E$ og $TF_a = TS_a + tid_a$. Prosjekttida blir da TF for sluttaktiviteten. Med dette utgangspunktet blir oppgaven med å bestemme TS og TF det samme som å finne lengste vei gjennom grafen fra start- til sluttaktiviteten. Seneste start/seneste ferdig (SS/SF) blir en analog beregning, men i forgjengerrekkefølge og med utgangspunkt i sluttaktiviteten.

Spørsmål: Hva ligner dette på? Hvilke standardalgoritmer/angrepsvinkler kan det synes fornuftig å ta utgangspunkt i? Gjenkjennesspørsmålet kan vurderes som å være nivå 1 og 2, mens omarbeiding av algoritmen vil karakteriseres som å kreve kunnskap på trinn 5.

Det var i denne delen det ikke ble gjort noe forsøk på å avskjære blindveier. Selv om vi havnet på et blindspor, hadde forsøket en nytteeffekt – vi fikk repetert det aktuelle pensumstoffet og vi fikk gjort en vurdering og analysert hvorfor den valgte løsningsvei ikke førte fram til målet.

Modifisert Dijkstra: Dijkstras algoritme er som kjent, en grådig algoritme som finner korteste vei fra en gitt node til samtlige andre noder i en graf. Spørsmål: hvordan modifisere denne slik at den i stedet bestemmer lengste vei fra startnoden til sluttnoden?

Studentene støtte snart på problemer knyttet til selve kjernen i den ”grådige” tilnærmingen. Det som falt naturlig for de fleste - å velge den noden blant de ubehandlede som hadde høyeste kantkostnad som den neste, ettersom vi nå var ute etter lengste vei - førte ikke fram. Det enkle faktum at summen av mange små bidrag vil kunne overstige kostnaden av et stort enkeltbidrag, kom inn som et nytt moment i denne konteksten. Det som skulle være anvendelse av kjent stoff på et nytt område, krevde nå islag av vurdering – med andre ord to trinn opp i kunnskapstrappa. Dessuten hadde jo alle utgående kanter fra en node samme vekt i og med at denne skulle representere tidsforbruket til noden.

Bredde først: Neste forsøk var å finne løsningen ved en ren bredde-først-traversering av grafen. Etter noe tid kom en av studentene opp med et moteksempel. Et tilfelle hvor en node er avhengig av en annen av høyere rang eller av samme rang som er ubehandlet og at denne forgjengeren inngår i en lengre vei.

Det er interessant å ta med dette beviset ved moteksempel utløste en diskusjon om bevisformer og spesielt induksjonsbevis og derfra var veien over til repetisjon av rekursjon naturlig nok kort.

Dybde først: For dybde-først-traversering har en to muligheter: pre- og postorder. Helt bevisst valgte jeg først å teste en postorder-løsning.

Postorder: Ved en ren postorder-traversering hvor nodene markeres som behandlet etter hvert som de ble sett for første gang, kom det relativt raskt opp moteksempler som utelukket denne løsningsveien. Med utgangspunkt i algoritmen for å beregne maksimal flyt gjennom en graf fra tidligere gjennomgått pensum, ble studentene penset inn på å la algoritmen angre seg. Det vil si at algoritmen unnlater å merke noden som behandlet, men oppdaterer TS og TF hver gang en når noden med en større verdi enn allerede registrert. Med disse råd på veien ble studentene permittert for dagen for å jobbe videre med løsningen.

Alt neste dag ble jeg oppsøkt av to studenter som har problem med å få algoritmen til å fungere på en av de tidligere gitte eksempelgrafene. Symptomatisk nok, lurte de på

om dette skyldes at de har misforstått bruken av angrefunksjonen ettersom den ble noe forskjellig fra den som inngikk i beregning av maks flyt.

Problemet lå selvsagt i løsningsforslaget, i og med at oppdatering av attributtverdien for noder som følge av angret verdi ikke forplanter seg langs hele den opprinnelige kallstien. I den senere diskusjon tar jeg opp spørsmålet om hvorfor ikke studentene innså problemet umiddelbart ettersom det kun krever kunnskap tilsvarende første og andre trinn i Blooms trapp.

Pre-order: Studentene, som nå var opplagt mer kritisk til forslag fra faglærer, fant straks moteksempler mot en ren preorder-løsning. Denne algoritmen syntes derimot å fungere hvis vi innførte en angrefunksjon slik at TS/TF kunne bli oppdatert når det ble behov for det. Nok en gang går studentene fra timen, glade og fornøyde i den tro at de nå endelig har løst problemet. Neste gang møter flere med en implementasjon som har taklet alle moteksempelene som har vært presentert så langt.

Diskusjonen blir livlig når det såes tvil om at løsningen holder. Men de blir overbevist så snart vi i fellesskap gjør et overslag over asymptotisk tidskompleksitet og finner at vekstraten gitt ved antall kanter er av eksponentiell orden ($T(|E|) = O(2^{|E|})$). (Lim, 2007) beskriver på samme måte hvordan han i et spillbasert undervisningsopplegg om grafer, benytter samme observasjon som innledning til andre vanskelige grafproblemer som Hamilton sykler, fargeleggingsproblemet og uavhengige mengder.

Kombinasjonen av at problemstillingen ligger på middels kunnskapsnivå og sett i lys av et ensidig fokus på korrekthet så langt i diskusjonen, kan langt på vei forklare at den ikke ble avdekket på tross av at estimering av tidsforbruk står sentralt i kurspensum.

Post-order – igjen: Etter innledende mislykkede forsøk med postorder- og preorder-traversering, ble studentene utfordret til å gå dypere inn i problemstillingen. Igjen valgte jeg å spore inn på en omvei til endelig løsning. Problemet ble koblet opp mot gjennomgangen av pensumstoffet i tilknytning til å finne sterke komponenter i en retta graf. Den aktuelle algoritmen består av to trinn: først en postorder-nummerering av nodene samtidig som grafen reverseres, etterfulgt av en preorder traversering av den reverserte grafen i nummerert rekkefølge. Vi skal merke oss at det er meget krevende å forstå denne algoritmen, selv om den kan klassifiseres til nivå 2 – å utvikle den vil derimot knyttes til kunnskap på nivå 7.

Studentene ble satt på sporet om å reversere grafen og naturlig nok da, foreta postorder traversering med utgangspunkt i sluttningen. Konklusjon etter en engasjerende diskusjon var at TS/TF kunne beregnes ved metoden $dfPostOrder(G_r, slutt)$. Når TS/TF var bestemt kunne SS/SF tilsvarende beregnes ved en metode $dfPostOrder(G, start)$.

Det som nå gjensto var å vurdere tids- og plassbruken til algoritmene. I og med at vi ved innføring a sentinel-element har sikret at grafen kun har én komponent vil tidsforbruket både for reversering og traversering være av $O(|E|)$. Hvis en i utgangspunktet hadde valgt en naboliste-representasjon av grafen, vil den reverserte grafen kreve ekstra plass tilsvarende $|E|$. Ved bruk av nabomatrise trengs ikke ekstra plass.

Opgaven til neste gang var å avgjøre om denne løsningen "holdt vann" og i motsatt fall komme opp med moteksempler. Selv om denne kontrollopgaven i hovedsak som tidligere påpekt, kun krever kunnskap på lavt nivå, er den både intrikat og arbeidskrevende.

Topologisk sortering – "Du lurer oss vel ikke?": Etter å ha sett at forrige ukes forslag førte fram til løsning, var det tid for etteranalyse og resonering over løsningen. Hva skjer egentlig ved $dfPostOrder(G_r, slutt)$?

Når løsningen ligger på bordet, er det lett å se at nodene faktisk blir behandlet fra start- til sluttnode på en slik måte at alle forgjengernodene er besøkt før en

etterfølgernode oppdateres. Det vil med andre ord si at nodene håndteres i topologisk rekkefølge for å bestemme TS/TF og i omvendt rekkefølge for å finne SS/SF. Selvsagt – et gjenkjennelsens lys går over gruppa – løsningen ligger i topologisk sortering!

Over til læreboka (Weiss, 1999) igjen, nå er det vel bare å kopiere algoritmen og levere løsningen? Nei, i og med at studenten er blitt ”hjulpeløs” fram til løsning, krevde jeg det nå en kritisk vurdering av algoritmen med hensyn på effektivitet. Studentene ble oppfordret til å finne eventuelle forbedringsmuligheter.

Lærebokas algoritme topSort kan forenklet beskrives med følgende punkt:

- *For hver node tell opp antall forgjengere*
- *Legg noder med null forgjengere inn i en hjelpestruktur*
- *Så lenge hjelpestrukturen er ikke-tom*
 - *ta et element ut av hjelpestrukturen og behandle det*
 - *for alle etterfølgere av denne noden*
 - *tell ned antall forgjenger noder som nå får null forgjengere legges inn i hjelpestrukturen*
- *Sjekk at alle nodene er behandlet (med innføring av sentinelelement blir dette overflødig i vårt tilfelle).*

Som hjelpestruktur for mellomlagring av element som er moden for behandling, kan en velge å bruke en kø eller en stakk. På grunn av likhetstrekk med bredde-først-traversering, vil studentene naturlig nok benytte samme hjelpestruktur som i det tilfellet, nemlig en kø. For å hjelpe studentene inn på en alternativ løsning, introduseres bruken av en stakk. Tanken var at forslaget skulle pense tankene inn mot en rekursiv løsning.

”*Minus minus gir pluss*”: Hintet med stakk var ikke nok. Problemet viste seg å være så intrikat at studentene selv etter eksplisitt opplysning om å bruke rekursjon, hadde vansker med å se en løsningsvei. I innledende forsøk hadde de jo sett moteksempel på bruk av pre-order, og at post-order gjennomløp heller ikke ga ønsket løsning.

Etter noe diskusjon i plenum var det enighet om at formuleringen: ”*En node kan ikke behandles før alle forgjengerne er behandlet*” gir grunnlag for følgende påstand: ”*En node uten ubehandlede etterfølgere, er nå den siste av de gjenværende til å bli behandlet*”. (Utsagnet ligner unektelig på dobbel negasjon av det opprinnelige uttrykket). Sett i lys av denne transformeringen ligger på høyt nivå i Blooms klassifikasjon, er det ikke overraskende at studentene ikke klarer å utlede den ”på direkten”.

Med bakgrunn i det ”reverserte utsagnet”, kom vi lett fram til at en postorder-gjennomgang ville gi en omvendt topologisk rekkefølge. Første tanke er da å snu rekkefølgen via en stakk, men ut fra helhetsvurdering av oppgaven hvor TS/SS beregnes i topologisk rekkefølge og SS/SF i omvendt topologisk orden er det to løsninger som står fram. Den ene er identisk med det som vi tidligere har konstatert med $dfPostOrder(G_r, slutt)$ og $dfPostOrder(G, start)$. Den andre vil være å benytte ei toveisliste til å holde referanser til aktivitetene slik at traversering i en retning gir topologisk rekkefølge og den andre retningen – naturlig nok – omvendt topologisk rekkefølge. Tilsynelatende kan det se ut til at vi her står overfor en klassisk avveining mellom plass- og tidsforbruk.

Et siste hint: Den foreslåtte bruken av toveisliste benytter denne kun til fullstendig gjennomløping forlengs og baklengs. Studentene ble derfor henvist til tidligere øvingsoppgaver hvor vi ved hjelp av en rekursiv algoritme traverserer ei enkeltlenket liste på samme måte.

Hvordan implementere gjennomløping av grafen først i topologisk og så i omvendt topologisk rekkefølge på en mest mulig plass- og tidseffektiv måte, ble den åpne

oppgaven for studentene. Eksperimentet med plenumsveiledning ble nemmelig avrundet her, både fordi fastsatt innleveringsdato for prosjektarbeidet nærmet seg og for å gi rom for å dekke resterende kurspensum på en forsvarlig måte.

Oppsummering og observasjoner

Utgangspunktet for eksperimentet var studentenes ønske om mer aktiv læring – å erstatte deler av forelesningene med problemrettet diskusjon. For å dekke store deler av pensum, valgte jeg å foreslå spor som ikke førte til løsning av det aktuelle problemet, men som hadde relevans i kurssammenheng. Nettopp relevanshensynet ga meg en etisk dekning for forsøket, selv om studentene fikk mye ekstraarbeid hadde de nytte av det for innlæring av det øvrige pensum.

Men som Anton Makarenko (Makarenko, 1977) – uten sammenligning for øvrig – beskriver, kan pedagogiske eksperiment ofte gi uforutsette resultat. I mitt tilfelle var det to slike uventede observasjoner. Den første var at studentene lot seg lede inn på blindspor selv etter gjentatte forsøk. Den andre var mer positiv, studentene viste et stigende engasjement utover forsøksperioden. Disse observasjonene har i ettertid vært utgangspunkt for uformelle faglige diskusjoner og for litteraturgjennomgang. På bakgrunn av disse vil jeg gi en kort refleksjon.

Diskusjon

Ukritiske studenter

Det eneste pedagogiske rammeverket jeg kjente til på den tid forsøket ble gjennomført, var Blooms taksonomi for kunnskap. Gjennom beskrivelsen av eksperimentet har jeg forsøkt å forklare studentenes handling med at enkeltoppgaver har ligget på et høyt nivå i denne kunnskapstrappa. Men vi ser også eksempler på at studentene heller ikke opponerer mot forslag som kun krever lavt kunnskapsnivå for å avsløre.

Kan noe av forklaringen ligge i undervisningsopplegget og –metodene vi møter disse studentene med? Er studieløpene blitt for velpolerte og tilrettelagte i en slik grad at det går ut over evnen til fagkritisk tenkning? Typiske trekk ved undervisningen er bruken av stadig mer helhetlige og gjennomarbeidede lærebøker med tilhørende pedagogisk oppbygde ressursider. Forelesningene er gjerne basert på "PowerPoint"-baserte presentasjoner som er forbedret i n-te versjon. Dertil hører en øvingsdel som presenterer mønsterløsninger tilpasset og justert overfor flere studentkull. Det er videre blitt mer vanlig å opprette mentorordninger hvor erfarne studenter rettleder og hjelper de yngre gjennom studieløpet. Det kan tyde på at vi i iveren etter å senke læreretskelen ved bare å servere velpolerte sannheter, hindrer utviklingen av fagkritiske evner hos studentene. Behovet for å aktivisere studentene og oppøve kritisk vurderingsevne angis også som et argument for problembasert læring (Mohus, 2005).

Et annet aspekt er at jeg, trass i at jeg innledningsvis beskriver behovet for å kombinere innlæring av teoristoff med praktiske øvinger, kun har benyttet Blooms taksonomi i vurderingen og derved har fokusert bare på den kognitive side av læring. I ettertid har jeg derfor søkt etter andre rammeverk for læring. I denne sammenheng synes en taksonomi for ferdigheter å passe godt. I (Dreyfus, 1988) introduseres en femdel skala for utvikling av ferdigheter innen et felt: 1: *Nybegynner*: Kontekstuavhengige elementer og regler er grunnlag for handling; 2: *Avansert begynner*: Erfaringsbaserte, kontekstavhengige elementer supplerer grunnlaget fra nivå 1; 3: *Kompetent utøver*: Mål og plan velges bevisst og nøye overveid for å redusere kompleksitet. Valget er ikke-objektivt og nødvendig. Utøveren er involvert i resultatet med sin egen person; 4:

Dyktig utøver: Intuitiv problemløsting og intuitivt valg av mål og plan ut fra erfaringsbasert perspektiv. Intuitive valg vurderes analytisk; 5: *Ekspert*: Intuitiv, holistisk og synkron identifisering av problem, mål, plan, beslutning og handling. Flytende, utvungen prestasjon, som ikke avbrytes av analytiske overveielser. Denne ferdighetstrappa er senere blitt utvidet med et sjette trinn, karakterisert ved kreativitet og fornyelse (Flyvbjerg, 1994). Flyvbjerg peker videre på at det er et kvalitativt sprang mellom tredje og fjerde trinn i Dreyfus-modellen. Spranget består i at kontekst og intuisjon erstatter regeltenking som viktigste grunnlag for handling – logikkbasert handling avløses av erfaringsbasert.

Vårt algoritmekurs går over 3. og 4. semester på bachelorstudiet i informatikk. Studentene har på forhånd bestått et programmeringskurs på 20 studiepoeng. På bakgrunn av det, vil jeg plassere dem i kategoriene "Avansert begyner" eller "Kompetent utøver" alt etter individuelle forskjeller. I forhold til algoritmevurdering vil de i de fleste sammenhenger under kurset være "Nybegynnere" ut fra det hensyn at flere tema som bygger på tidligere stoff i kurset og følger for kort tid etter. Det lave ferdighetsnivået kan være forklaringen på at studentene ikke avdekker løsningsforslag som blindspor. Det å se begrensningen med ufullstendig oppdatering i "Dybde-først pre-order" krever typisk erfaringsgrunnlag på høyere nivå.

Utfordringen for en foreleser er å kunne vurdere ferdighetsnivået til studentene og tilpasse undervisningsopplegget etter det.

Engasjerte studenter

Jeg var forberedt på at forsøket kunne føre til frustrasjon hos studentene. En slik frustrasjon kunne enten føre til oppgitt passivitet eller i motsatt fall en økt innsats som en trassreaksjon. Jeg kunne ikke observere tegn til frustrasjon, men studentene viste stor interesse og engasjement fra første dag og denne gløden bare økte utover forsøksperioden.

I en undersøkelse fra universitetet i Wisconsin (Hansen, 2007) hvor studentene rangerte innleveringsoppgavene i programmering langs 10-delte skalaer med hensyn til frustrasjon og engasjement, ble det ikke påvist noen sammenheng mellom frustrasjonsnivå og engasjement. Hensikten med deres eksperiment var å justere oppgavesettene slik at de hadde lavt frustrasjonsnivå samtidig som de var engasjerende.

Jeg gjennomførte samme spørreundersøkelse overfor mine studenter. Selv om jeg hadde en for begrenset populasjon til å kunne generalisere, ville det gi en indikasjon for det isolerte eksperimentet. Jeg gjennomførte undersøkelsen via e-post, respondentene var selvselektert og utgjorde til sammen en knapp håndfull, så det er all grunn til å knytte stor usikkerhet til resultatet. Tallene var imidlertid så entydige at jeg har tatt de med som et innspill til videre diskusjon. For mens Hansens refererer gjennomsnittsverdier på 7.21 for engasjement og 5.26 for frustrasjon, er gjennomsnittstallene i mitt forsøk henholdsvis 9.3 og 2.7 mens medianene gir ekstermverdiene 10 og 1.

Konklusjon

Som tidligere påpekt, har jeg ikke datagrunnlag for å trekke bastante slutninger i form av generaliseringer ut fra konkrete funn knyttet til eksperimentet. Jeg har imidlertid gjort to observasjoner som kan være av allmenn interesse. Den første er indikasjonene på sammenheng mellom aktivisering av studentene, det å gi utfordringer i undervisningssammenheng og engasjementsnivået hos studentene. Den andre er av mer subjektiv art. Jeg føler at kjennskap til taksonomisettene for kunnskap og ferdighetsnivå har gitt meg et godt rammeverk for å analysere og vurdere undervisningsopplegg.

Denne erfaringen vil jeg gjerne dele med andre slik at vi også kan benytte disse begrepene i samarbeidssituasjoner og diskusjon.

Videre arbeid

Det ville ha vært interessant å gjennomføre et lignende forsøk på en større studentgruppe. Forut for et slikt eksperiment ville jeg anbefale en bredere litteraturanalyse som grunnlag for hypotesedannelse eller oppsett av forskningsmodell samt en mer metodisk gjennomføring med tanke på validitet til resultatene.

Jeg vil selv benytte taksonomi for læring aktivt videre i forbindelse med studie- og kursutvikling samt i ulike faser av undervisningen. Størst nytte av disse forventer jeg å få i forbindelse med evaluering/eksamensarbeid, både ved oppgaveformulering i relasjon til målsettingen for emnet og ved vurdering av kandidatprestasjonene. For på linje med (Anderson, 2007) mener jeg at faglærers rolle har en vesentlig betydning for læringsresultatet i et emne selv etter at kursbeskrivelse og lærebok er fastlagt, og da er det viktig at hun/han har et veltilpasset begrepsapparat å støtte seg til.

Referanser

(Anderson, 2007): Richard Anderson & al: Supporting Active Learning and Example based Instruction with Classroom Technology, *ACM SIGCSE Bulletin, Proceedings of the 38th Technical Symposium on Computer Science Education, March 7-11, 2007*. Pp 69-72.

(Dreyfus, 1988): Hubert L. Dreyfus, Stuart E. Dreyfus: Mind over Machine, The Free Press, 1988, ISBN 0-02-908061-4

(Flyvbjerg, 1994): Bent Flyvbjerg: Rationalitet og Magt, Bind 1: Det konkrete videnskap, Akademisk Forlag, 1994, ISBN 87-500-2993-2

(Hansen, 2007): Stuart Hansen, Erica Eddy: Engagement and Frustration in Programming Projects, *ACM SIGCSE Bulletin, Proceedings of the 38th Technical Symposium on Computer Science Education, March 7-11, 2007*. Pp 271-275

(Lim, 2007): Darren Lim: Taking Students Out for a Ride: Using a Board Game to teach Graph Theory, *ACM SIGCSE Bulletin, Proceedings of the 38th Technical Symposium on Computer Science Education, March 7-11, 2007*. Pp 367-371

(Makarenko, 1977): Anton Makarenko: Veien til livet. Pedagogisk poem 1, Forlaget Ny Dag, 1977, ISBN 82-7009-072-7

(Mohus, 2005): Åge Mohus: Tanker om læring og studiekvalitet, HBo-notat 5/2005, ISBN 82-7314-483-6

(NTNU, 2007): Ordliste (taksonomi) til bruk ved beskrivelse av læringsmål:
http://www.ntnu.no/eksternweb/multimedia/archive/00007/Ordliste_taksonomi_7509a.pdf

(Valdermo, 1997): Vedlegg nr 3 til Åpenbok-rapporten (1997):
<http://uit.no/getfile.php?PageId=1130&FileId=89>

(Weiss, 1999): Mark Allen Weiss: Data Structures & Algorithm Analysis in Java, Addison Wesley, 1999, ISBN 0-201-35754-2.

(Woodley, 2007): Michael Woodley, Samuel N. Kamin: Programming Studio: A Course for Improving Programming Skills in Undergraduates, *ACM SIGCSE Bulletin, Proceedings of the 38th Technical Symposium on Computer Science Education, March 7-11, 2007. Pp 531-535.*