TDT4120 Algoritmer og datastrukturer

Eksamen, 12. desember 2023, 15:00–19:00

Faglig kontakt	Magnus Lie Hetland
Hjelpemiddelkode	E

Løsningsforslag

Løsningsforslagene i rødt nedenfor er *eksempler* på svar som vil gi uttelling. Det vil ofte være helt akseptabelt med mange andre, beslektede svar, spesielt der det bes om en forklaring eller lignende. Om du svarte noe litt annet, betyr ikke det nødvendigvis at du svarte feil!

Merk: Det å forstå hva det spørres etter *er en del av oppgaven*. Enkelte oppgaver kan være konstruert slik at det er nettopp dette som er utfordringen, så om du ikke får dem til, vil de kunne virke uklare eller tvetydige.

1 I de følgende deloppgavene er det meningen at du skal svare *svært kort*.

5 % *a*) Hva er kjøretiden til Insertion-Sort i beste tilfelle?

$\Theta(n)$

$O(n)$ gir også full uttelling. $\Omega(n)$ gir 4 poeng.

Relevant læringsmål: Forstå Insertion-Sort (kjenne kjøretidene under ulike omstendigheter).

5 % *b*) Hva er den største fordelen med en tabell (*array*) fremfor en lenket liste?

Direkte oppslag (og innsetting) i konstant tid.

Her godtas alle formuleringer som får frem hovedpoenget med at tabeller kan indekseres i konstant tid, mens lenkede lister må traverseres fra starten.

Relevant læringsmål: Forstå hvordan *lenkede lister* fungerer.

5 % *c*) I Counting-Sort(A, n, k) er A en tabell (*array*) med n verdier. Hva er k?

Største mulige verdi.

Angir altså verdiområdet, $0, \ldots, k$. Her godtas også andre formuleringer som får frem at k angir eller begrenser hvilke verdier A kan inneholde.

Relevant læringsmål: Forstå COUNTING-SORT.

5 %

d) Hva er topologisk sortering?

Ordning av nodene i en graf, der kantene peker fremover.

Her godtas også andre forklaringer av ensrettingen av kantene, inkl. om man sier at de peker bakover.

Å kun si at det er en ordning eller sortering av nodene i en graf gir 3 poeng.

Relevant læringsmål: Forstå TOPOLOGICAL-SORT.

5 %

e) I en maks-haug (*max-heap)* ligger verdien x i en foreldrenode, mens verdiene y og z ligger i henholdsvis venstre og høyre barnenode. Hvilke krav stilles til forholdet mellom verdiene?

Her er det altså snakk om *verdier*, ikke f.eks. indekser.

$y, z \leqslant x$

Om man bytter på det, og oppgir at y og z skal være større enn x, gir det 1 poeng. Om man svarer f.eks. $y \leqslant x \leqslant z$, gir det ingen uttelling.

Relevant læringsmål: Forstå *haugegenskapen* (*the heap property*).

2 I de følgende deloppgavene er det oppgitt informasjon om funksjonene $f(n)$ og $g(n)$. I hvert tilfelle, uttrykk $f(n) + g(n)$ med asymptotisk notasjon.

Under eksamen ble følgende oppgitt: Med formuleringen «i hvert tilfelle» menes «i hver deloppgave». Hver deloppgave skal besvares med ett uttrykk.

5 %

a) $f(n) = \mathrm{O}(n^2)$, $f(n) = \Omega(n)$, $g(n) = \mathrm{O}(n^2)$, $g(n) = \Omega(n^2)$

$\Theta(n^2)$

$\mathrm{O}(n^2)$ gir 4 poeng. $\Omega(n^2)$ gir 3 poeng.

Her er det altså slik at $f(n)$ er både $\mathrm{O}(n^2)$ og $\Omega(n)$, etc. Om man i stedet har tolket ligningene som ulike definisjoner av $f(n)$ og $g(n)$, og har fått riktig svar (f.eks. $\Omega(n) + \mathrm{O}(n^2) = \Omega(n)$) for ulike kombinasjoner (enten 2 eller 4), gir det 1 poeng.

Relevant læringsmål: Kunne definere og bruke *asymptotisk notasjon*, O, Ω, Θ, o og ω.

5 %

b) $f(n) = \Omega(n^2)$, $f(n) = \omega(n)$, $g(n) = \mathrm{O}(n^2)$, $g(n) = o(n^3)$

$\Omega(n^2)$

Tilsvarende som i *a*, om har tolket ligningene som ulike definisjoner av $f(n)$ og $g(n)$, og har fått riktig svar for ulike kombinasjoner, gir det 1 poeng.

Relevant læringsmål: Kunne definere og bruke *asymptotisk notasjon*, O, Ω, Θ, o og ω.

3 Løs følgende rekurrenser. Oppgi svaret i Θ-notasjon.

5 %

a) $\mathrm{T}(n) = \mathrm{T}(n-1) + n^2 - (n-1)^2$

$\Theta(n^2)$

Her kan man først regne ut $n^2 - (n-1)^2$ og så bruke iterasjonsmetoden på $\mathrm{T}(n) = \mathrm{T}(n-1) + 2n - 1$, men det er antagelig enklere å bruke metoden direkte, siden vi får en teleskopsum som eliminerer alle ledd unntatt n^2 og grunntilfellet:

$$\begin{aligned}
\mathrm{T}(n) &= n^2 - (n-1)^2 + \mathrm{T}(n-1) \\
&= n^2 - (n-1)^2 + (n-1)^2 - (n-2)^2 + \mathrm{T}(n-2) \\
&= n^2 - (n-2)^2 + \mathrm{T}(n-2) \\
&\vdots \\
&= n^2 - (n-i)^2 + \mathrm{T}(n-i) \\
&= n^2 - (n-n)^2 + \mathrm{T}(n-n) \\
&= n^2 + \mathrm{T}(0) = n^2 + \Theta(1) = \Theta(n^2)
\end{aligned}$$

Relevant læringsmål: Kunne løse rekurrenser med *iterasjonsmetoden*.

5 %

b) $\mathrm{T}(n) = 2\mathrm{T}(n/4) + \sqrt{n}\lg^2 n$

$\Theta(\sqrt{n}\lg^3 n)$

Her gjelder tilfelle 2 av masterteoremet, med $a = 2$, $b = 4$ og $k = 2$.

(Merk at $\sqrt{n} = n^{1/2}$ og $\log_4 2 = 1/2$.)

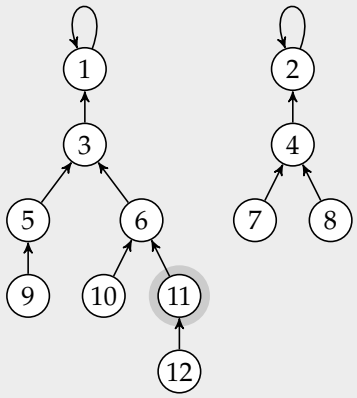
Dette er oppgave 4.5-1c fra læreboka.

Relevant læringsmål: Kunne løse rekurrenser med *masterteoremet*.

5 % **4** Du har oppgitt følgende frekvenser for alfabetet `a`, . . . , `h`:

`a`:1 `b`:1 `c`:2 `d`:3 `e`:5 `f`:8 `g`:13 `h`:21

Figur 1



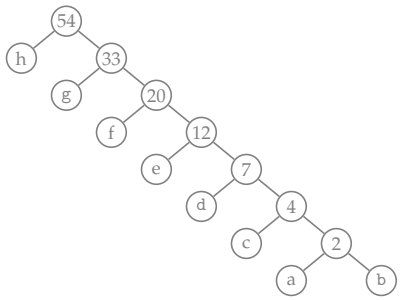
I en Huffman-kode for disse tegnene, hva er antall siffer for tegnet e?

Her er vi altså ute etter antall binære siffer som trengs for å kode én e med Huffman-koden, ikke det totale antall siffer som brukes på alle e-ene i teksten.

4

Oppgaven er basert på oppgave 15.3-3 i læreboka.

Huffman-treet er ikke helt entydig på de nederste nivåene, men dette er et mulig tre:



Relevant læringsmål: Forstå HUFFMAN og *Huffman-koder*.

5 % **5** I figur 1 ser du en disjunkt-mengde-skog (*disjoint-set forest*). Vi kan representere foreldrepekerne med en tabell A, der $A[v] = v.p$:

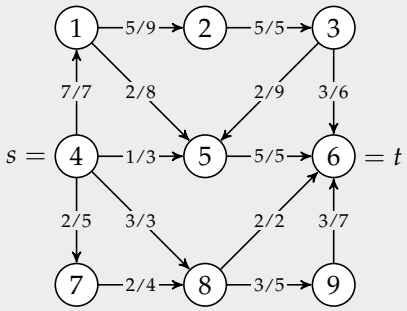
$A = \langle 1, 2, 1, 2, 3, 3, 4, 4, 5, 6, 6, 11 \rangle$

Utfør FIND-SET(11) og oppdater A. Hvordan ser A ut etterpå?

Svar ved å liste opp tallene i A. Du trenger ikke skrive $A = \langle \ldots \rangle$.

1, 2, 1, 2, 3, 1, 4, 4, 5, 6, 1, 11

Figur 2



Her søker vi altså fra 11 og oppover til 1, og sørger for at alle noder vi treffer på (11, 6 og 3) ender med å peke på 1 (*path compression*). Det er altså bare A[11] og A[6] som endrer seg.

Relevant læringsmål: Forstå *skog*-implementasjonen av *disjunkte mengder*.

5 % **6** Hvilke forøkende stier vil Edmonds-Karp finne i flytnettet i figur 2?

Det er altså meningen at du skal utføre Edmonds-Karp på flytnettet, men *uten* initialiseringen $(u, v).f = 0$. Oppgi stiene som sekvenser av noder. Skriv én sti per linje, i den rekkefølgen de finnes. For eksempel:

1, 2, 3, 4, 5
7, 6, 5, 4, 3, 2, 1
4, 5, 6, 7, 8, 9

(Dette er kun et eksempel på formatet, *ikke* et faktisk gyldig sett med stier.)

4, 5, 3, 6
4, 7, 8, 9, 6

Om man bytter om på stiene (og altså ikke spesifikt har brukt BFS, men fortsatt har brukt FORD-FULKERSON), gir det 4 poeng.

Om man kun oppgir den siste stien (og altså ikke har fått med seg flytopphevingen fra 5 til 3) gir det 2 poeng.

Relevant læringsmål: Forstå *Edmonds-Karp*-algoritmen (FORD-FULKERSON med BFS).

5 % **7** Hvis det er uavgjort mellom noen kandidater i noen av rangeringene i stabil matching (*the stable-marriage problem*), kan vi fortsatt garantert finne en stabil matching? Forklar kort.

Algoritme 1

```
UNTITLED(A, p, r, k)
1  if p < r
2      q = RANDOMIZED-PARTITION(A, p, r)
3      UNTITLED(A, p, q − 1, k)
4      if q < k
5          UNTITLED(A, q + 1, r, k)
```

Ja. Likeverdige kandidater kan ordnes vilkårlig i listene. Om vi så bruker GALE-SHAPLEY, får vi ingen blokkerende par med denne rangeringen, og ingen kan oppstå om vi igjen tar hensyn til at det er uavgjort mellom noen.

Det vil si, om et umatchet par w, m ikke foretrekker hverandre fremfor sine partnere, så vil de heller ikke plutselig gjøre det om det blir uavgjort mellom enkelte. Det verste som kan skje er at de ikke foretrekker sine partnere fremfor hverandre.

(I litteraturen kalles dette en *svakt stabil*, eller *weakly stable*, matching.)

Her godtas også andre korrekte argumenter for hvorfor det fortsatt må eksistere en stabil matching, enten de bruker GALE-SHAPLEY eller ikke.

Relevante læringsmål: Forstå hva en stabil matching (*stable matching*) og et blokkerende par (*blocking pair*) er; forstå GALE-SHAPLEY; kunne analysere algoritmers korrekthet; kunne konstruere nye algoritmer.

8 Betrakt algoritmen UNTITLED (algoritme 1), der $A[1:n]$ er en heltallstabell og algoritmen startes med kallet $\text{UNTITLED}(A, 1, n, k)$.

5 %

a) Hva gjør algoritmen?

Her er vi ute etter resultatet av å kjøre algoritmen, eller hvilket problem den løser, ikke hvordan den oppfører seg, trinn for trinn.

Sorterer de k minste elementene i A, så de havner i $A[1:k]$.

Om man forklarer noe om at den faktisk kan ende med å finne og sortere flere enn k minste elementene (dvs., de r minste elementene, i $A[1:r]$, så snart $q < k$) så gir det også full uttelling.

Om man kun sier at algoritmen finner de k minste elementene, at den finner det k-ende minste elementet, sier den er ekvivalent med RANDOMIZED-SELECT eller at den plasserer de k minste elementene først, gir det 2 poeng.

Om man sier at algoritmen sorterer hele tabellen, eller mener den er ekvivalent med RANDOMIZED-QUICKSORT, gir det 1 poeng.

Relevante læringsmål: Forstå RANDOMIZED-QUICKSORT; forstå RANDOM-

IZED-SELECT; kunne analysere algoritmers korrekthet; kunne konstruere nye algoritmer.

5 %

b) Hva er den forventede kjøretiden, som funksjon av n og k? Oppgi svaret i Θ-notasjon.

Som en forenkling, kan du anta at q alltid havner midt mellom p og r.

$\Theta(n + k \lg k)$

$\Theta(n)$ gir 2 poeng. $\Theta(k \lg k)$ og $\Theta(n \lg k)$ gir 1 poeng. $O(n + k \lg k)$ gir 1 poeng (siden oppgaven eksplisitt ber om Θ-notasjon).

Utførelsen har to faser.

Fase 1 ($q \geqslant k$): Tilsvarer RANDOMIZED-SELECT, men stopper når $q < k$. Merk at RANDOMIZED-PARTITION kjøres minst én gang, selv om $q < k$ fra starten av, så lenge $n > 1$. Kjøretid $\Theta(n)$.

Fase 2 ($q < k$): Tilsvarer RANDOMIZED-QUICKSORT på det vi sitter igjen med etter fase 1, nemlig $A[1 : r]$. Siden $q < k$, og q er midt i, har vi færre enn $2k$ elementer. Kjøretid $\Theta(k \lg k)$.

Relevante læringsmål: Forstå RANDOMIZED-QUICKSORT; forstå RANDOMIZED-SELECT; kunne analysere algoritmers effektivitet; kunne løse rekurrenser; kunne bruke asymptotisk notasjon.

9 Du skal lage en spilleliste som er nøyaktig t sekunder lang. Du har n sanger å velge mellom. Du kan anta at alle sangene varer et helt antall sekunder.

5 %

a) Hvordan kan du vise at dette er et vanskelig problem?

F.eks. reduksjon fra SUBSET-SUM, der heltallene blir sekunder.

Om man forklarer at problemet er «ekvivalent med» SUBSET-SUM, på en måte som implisitt innebærer en reduksjon fra SUBSET-SUM (ev. begge veier), gir det 5 poeng. Om man argumenterer for at det «ligner veldig» e.l., gir det 4 poeng.

Om man reduserer *til* SUBSET-SUM (uten også å redusere *fra*), gir det 1 poeng.

Om reduserer fra et annet NP-komplett problem, og forklarer hvordan, gir det også 5 poeng. Om man sier at man vil redusere fra et annet problem, men ikke forklarer hvordan, gir det 1 poeng. Om man sier at man vil redusere *til* et annet problem, gir det 0 poeng.

Relevante læringsmål: Forstå definisjonen av NP-*hardhet* og NP-*kompletthet*; kjenne det NP-komplette problemet SUBSET-SUM; forstå hvordan NP-kompletthet kan bevises ved én reduksjon; være i stand til å konstruere

enkle NP-kompletthetsbevis.

5 % *b*) Hvordan kan du løse problemet?

F.eks. reduksjon til det binære ryggsekkproblemet, der verdi = vekt = tid.

En tilsvarende løsning, der man løser problemet direkte med dynamisk programmering, og forklarer hvordan, vil gi 5 poeng.

Relevante læringsmål: Forstå løsningen på *de binære ryggsekkproblemet*; kunne konstruere nye effektive algoritmer.

5 % **10** Din venn Smartnes leter etter stier i en sammenhengende vektet urettet graf, fra en startnode s til alle andre. Hvis han finner de korteste stiene, vil summen av alle sti-lengdene bli minst mulig, men delene av stiene der de *overlapper* vil da telles med flere ganger. Han vil heller finne et sett med stier som minimerer en tilsvarende sum, der de overlappende delene telles bare én gang. Hvordan kan han gjøre det? Forklar kort.

Han vil minimere kantsummen for en delgraf som kobler s til alle andre noder. Det tilsvarer å finne et minimalt spenntre, som han kan finne med Prims eller Kruskals algoritme.

Her er poenget at beskrivelsen av problemet til Smartnes fremstår som svært uklar, og utfordringen er å klare å forstå at det han er ute etter faktisk tilsvarer minimale spenntrær. Om man får frem det, selv uten å nevne algoritmer for å finne dem, gir det 5 poeng.

Relevante læringsmål: Vite hva *spenntrær* og *minimale spenntrær* er; forstå MST-Kruskal; forstå MST-Prim.

5 % **11** Din venn Klokland er ansvarlig for en konsertserie, men på grunn av budsjettkutt må hun nøye seg med én scene. Flere av konsertene kolliderer tidsmessig, og noen må derfor avlyses. Klokland ønsker å avlyse så få som mulig.

Det eneste hun har tatt vare på av informasjon er starttidspunktene for alle konsertene, samt en urettet graf med konserter som noder, og kanter mellom dem som kolliderer. Hun ønsker å fjerne så få noder som mulig, slik at alle kantene forsvinner.

Hun blir litt svett idet hun innser at dette er optimeringsversjonen av VERTEX-COVER, men håper kanskje du har noen gode ideer.

Konstruer og beskriv en algoritme som løser problemet generelt.

Problemet kan reduseres til aktivitetsutvelgelse: Færrest mulige avlysninger tilsvarer flest mulig konserter som *ikke* kolliderer. Vi kan snu tidsaksen: Sorter etter *synkende startid* og velg så neste konsert som ikke kolliderer (med den forrige, om noen), til vi er ferdige.

Eventuelt kan vi finne faktiske intervaller ved å sortere etter starttid og sette sluttid så konserten kolliderer med dem den har en kant til. Vi sorterer så etter sluttid og velger grådig.

Om man sier noe om at dette kan løses grådig, med henvisning til aktivitetsutvelgelse, men uten å faktisk finne intervallene, gir det 3 poeng. Om man finner intervallene, men ikke deretter løser problemet, gir det 2 poeng.

Relevante læringsmål: Forstå eksemplet *aktivitetsutvelgelse*; kunne konstruere nye effektive algoritmer.

5 % **12** Konstruer og beskriv en algoritme som avgjør om en rettet graf har en odde sykel, altså en sykel med et antall kanter som er et oddetall.

Her får du full uttelling med kjøretid $O(V^3)$.

Hint 1: Det kan være nyttig å se på *stier* som en del av løsningen.

Hint 2: Du trenger ikke begrense deg til *enkle* (*simple*) stier og sykler.

Hint 3: Finnes en odde sti fra i til j? Hva med en der antall kanter er et partall?

Kan løses på en lignende måte som transitiv lukning, men med to matriser, A og B, der a_{ij} angir om det finnes en (ikke nødvendigvis enkel) oddetallssti fra i til j og b_{ij} om det finnes en partallssti. For hver i, j og k, oppdateres disse slik:

$a_{ij} = a_{ij} \vee (b_{ik} \wedge a_{kj}) \vee (a_{ik} \wedge b_{kj})$
$b_{ij} = b_{ij} \vee (a_{ik} \wedge a_{kj}) \vee (b_{ik} \wedge b_{kj})$

Sjekk til slutt diagonalen i A, dvs., om $a_{ii} = 1$ for noen i.

Løsningen er en mindre justering av Floyd-Warshall, på samme måte som Transitive-Closure. Om man nevner en av disse, og skisserer hvordan modifikasjonen kan gjøres, uten at det blir helt rett, gir det 4 poeng. Om man *ikke* nevner disse, men løser problemet korrekt med dynamisk programmering, gir det naturligvis 5 poeng.

Løsninger som baserer seg på traversering vil stort sett ikke fungere, men kan likevel gi opptil 2 poeng. (Et unntak er løsningen beskrevet nedenfor, som det ikke forventes at noen finner, men som vil gi 5 poeng.)

Løsninger som oppdager odde *urettede* sykler (f.eks. ved traversering og tofarging) gir 0 poeng.

Relevante læringsmål: Forstå Floyd-Warshall; forstå Transitive-Closure; kunne konstruere nye effektive algoritmer.

For interesserte: Her kan det godt være at stiene underveis ikke er enkle stier, men at de går innom samme noder flere ganger. Det vil i så fall si at vi finner en odde sykel som heller ikke er enkel. Men: En slik sykel vil bestå av flere sykler som tilsammen har et odde antall kanter, og det betyr at minst én av syklene må ha et odde antall kanter. På den måten vil vi her også finne ut om grafen har en odde *enkel* sykel, selv om oppgaven ikke spør etter det.

Dette resonnementet fungerer ikke for partallssykler, og å avgjøre om en graf har en enkel partallssykel er så langt et uløst problem. (Se f.eks. "Pfaffian orientations, 0–1 permanents, and even cycles in directed graphs" av Vazirani og Yannakakis i *Discrete Applied Mathematics*, 1989.)

Problemet med odde sykler *kan faktisk løses i lineær tid*. En mulig løsning, som går utenfor pensum, er å først finne alle sterke komponenter i grafen (f.eks. med Strongly-Connected-Components) og så sjekke om alle er tofargbare (kan avgjøres med traversering, jf. oppg. 34-3a). Dette løser problemet fordi alle sykler må ligge inne i en sterk komponent, og en sterk komponent har en odde sykel hvis og bare hvis den er bipartitt (jf. teorem 1.8.1 i *Digraphs: Theory, Algorithms and Applications* av Bang-Jensen og Gutin, 2002).