

**Løsningsforslag til eksamen i fag  
SIF8010 Algoritmer og Datastrukturer  
Tirsdag 14. Desember 1999, kl 0900-1500**

**Faglig kontakt under eksamen:** Arne Halaas, tlf. 73 593442.

**Hjelpemidler:** Alle kalkulatortyper tillatt. Alle trykte og håndskrevne hjelpemidler tillatt.

**Rubrikksvar:** Alle svar skal avgis i angitte svar-ruter. Ikke legg ved ekstra ark som svar.

**Krav:** Det kreves "bestått" både på de ordinære og på de øvingsrelaterte spørsmål.

**Husk:** Fyll inn rubrikken "Student nr." øverst på alle ark.

**Oppgave 1. (18%)**

- a) Er høyden på et 2-3-tre med  $n$  dataposter  $\Theta(\log_3 n)$ ?
- b) Er Dijkstras algoritme basert på Dynamisk Programmering?
- c) Er Kruskals algoritme en grådighetsalgoritme som alltid gir beste løsning?
- d) Er antallet ordninger ved topologisk sortering av en vilkårlig graf  $G(V,E)$   $O(|V|!)$ ?
- e) Er antallet ordninger ved topologisk sortering av en vilkårlig graf  $G(V,E)$   $O(|E|!)$ ?
- f) Er Quicksort  $\Omega(n \cdot \log n)$  ?
- g) Bør "sortering ved innsetting" brukes i forbindelse med Quicksort ?
- h) Kan maks-flyt-algoritmen brukes til å finne ut om en sammenhengende graf  $G$  har en bro?
- i) Er Huffmans algoritme aktuell i forbindelse med fletting (Merge) av  $m > 2$  sorterte lister?

Svar: (Stryk "Ja" eller "Nei". Begrunnelsen må fylles ut. Hvert delsvar teller 2%)

- |                       |  |
|-----------------------|--|
| a) Ja/ <del>nei</del> | Begrunnelse: Høyden er i intervallet $[\log_3 n, \log_2 n]$ .<br>Idet $\log_2 n = \text{konst} \cdot \log_3 n$ , er høyden $\Theta(\log_3 n) = \Theta(\log_2 n) = \Theta(\log_q n)$              |
| b) Ja/ <del>nei</del> | Begrunnelse: $\{d[i]\}$ forbedres rekursivt.<br>Dijkstras algoritme bruker både dynamisk programmering og grådighet.   |
| c) Ja/ <del>nei</del> | Begrunnelse: Det henvises til bevis i kompendiet, basert på at "nei" gir selvmotsigelse.   |
| d) Ja/ <del>nei</del> | Begrunnelse: Antall mulige ordninger av $ V $ noder er $ V !$ .  |
| e) <del>Ja</del> /nei | Begrunnelse (Moteksempel): I en graf med kun 1 avhengighet vil vi ha $ E ! = 1! = 1$ , men dette tallet begrenser ikke de $ V !$ mulige ordninger.<br>(Antar at $ V $ er mye større enn $ E $ .) |

- |           |   |
|-----------|---|
| f) Ja/nei | Begrunnelse: Støttes av en generell sats: Alle sorteringsalgoritmer basert på sammenligninger er $\Omega(n \cdot \log n)$ .                                     |
| g) Ja/nei | Begrunnelse: Alt bør sorteres ved innsetting når de sorterte intervallene er blitt korte ( $\leq 10\pm$ ).  |
| h) Ja/nei | Begrunnelse: Alle kanter gis flytkapasitet 1, og en kjører alle-til-alle-flytmaks for å dekke alle kilde-sluk-kombinasjoner. Maks flyt=1 $\Leftrightarrow$ bro. |
| i) Ja/nei | Begrunnelse: De til enhver tid 2 korteste listene flettes. Fletterekkefølgen gis av Huffmans algoritme med "listelengde" som vekt.                              |

### Oppgave 2. (20%)

Vi definerer problemet  $P(A,n,k,b)$  slik: *Finn, om mulig, et utvalg av  $k$  ( $>1$ ) verdier i  $A = \{a_1, a_2, \dots, a_n\}$  som er slik at summen av disse  $k$  verdiene er lik  $b$ . Alle verdiene er heltall.*

(a) Skisser en algoritme Q som løser problemet  $P(A,n,k,b)$  i  $O(n^{k-1} \log n)$  tid.

Svar: 4%

- Sorter A
- For hver av i alt  $O(n^{k-1})$  utvalg av  $k-1$   $a$ -verdier:  
Finn den  $k$ -te verdien ved binærøøk i sortert A

Totalt  $O(n \log n) + O(n^{k-1}) \cdot O(\log n) = \underline{O(n^{k-1} \log n)}$

(b) Skisser en mer effektiv algoritme R som løser problemet  $P(A,n,k,b)$  i  $O(n^{k-1})$  tid når  $k > 2$ .

Svar: 4%

- Sorter A
- For hver av i alt  $O(n^{k-2})$  utvalg av elementer av  $k-2$   $a$ -verdier:  
Finn de 2 siste verdiene ved å søke igjennom A lineært fra hver side (tidligere Eksamensoppgave) for, om mulig, å finne "restverdien" som en sum av disse to Verdiene.  
(Lineærøøk: Hvis summen er for stor – drop høyeste element; for liten – dropp minste.)

(Merknad: Total kompleksitet er  $O(n \log n) + O(n^{k-2}) \cdot O(n)$ . For  $k \leq 2$  blir dette  $O(n \log n)$ .)

(c) Hvordan vil du løse problemet  $P(A,n,k,b)$  når  $n > 100$  og  $k = n-3$  ?

Svar: 5%

Beregn  $S = \sum a_i$ , og løs, dersom  $b < S$ , problemet  $P(A,n,3,S-b)$ . Svaret blir da de andre verdiene (det vil si  $A-P$ ). Dette problemet har (naturligvis) samme tidskompleksitet som  $P(A,n,3,b)$ .

(d) For hvilke verdier av  $k$  vil du anta at  $P(A,n,k,b)$  krever mest tid for å bli løst? (Begrunn.)

Svar: 4%

$k = n/2$ . Det henger sammen med at binomialkoeffisienten er "størst midt på," dvs. at " $n$  over  $m$ " er størst når  $m = \lceil n/2 \rceil$ .

Vi definerer nå problemet  $P'(A,n,b)$  slik: Finn, om mulig, et utvalg av inntil  $n$  verdier i  $A = \{a_1, a_2, \dots, a_n\}$  som er slik at summen av disse verdiene er lik  $b$ . Alle verdiene er heltall.

(e) Hvilken metode vil du foreslå for å løse problemet  $P'$ ? Angi metodens tidskompleksitet.

Svar: 3%

Dette er ryggsekkproblemet. Løst med dynamisk programmering er kjøretiden  $O(nb)$ , ved utfylling av en  $n \times b$ -tabell.

### Oppgave 3. (8%)

Student Lurvik hevder å ha utviklet en ny datastruktur for prioritetskøer som støtter operasjonene  $\text{Insert}(\text{Queue}, \text{element})$ ,  $\text{FindMaximum}(\text{Queue})$  og  $\text{DeleteMaximum}(\text{Queue})$ . Lurvik påstår at alle disse 3 operasjonene kun krever  $O(1)$  tid.

(a) Det er ingen grunn til å tro på Lurvik. Hvorfor ikke?

Svar: 8%

$n$  stk.  $\text{Insert}$ -operasjoner, etterfulgt av  $n$  stk. ( $\text{FindMaximum}$ ,  $\text{DeleteMaximum}$ ) ville gitt en generell  $O(n)$ -algoritme for sortering, i strid både med  $\Omega(n \cdot \log n)$ -grensen og alle kjente metoder.

**Oppgave 4. (21%)**

Vi skal her se på et problem som skal løses ved hjelp av Dynamisk Programmering:

Problem  $P(S, n)$ : Gitt en sekvens  $S = \langle s_1, s_2, \dots, s_n \rangle$  bestående av  $n$  heltall. Finn lengden  $L_n$  av den lengste subsekvensen  $S^*$  i  $S$  som er slik at verdiene i  $S^*$  er stigende. Verdiene i  $S^*$  må ikke nødvendigvis være naboer i  $S$ .

Merk at det her kun spørres etter lengden  $L_n$  av den (en av de) lengste subsekvensen(e) i  $S$ .

Eksempel ( $n=9$ ):

$S = \langle 9, 5, 2, 8, 7, 3, 1, 6, 4 \rangle$ . Her er  $L_n = 3$ . Subsekvensen  $S^*$  består da av enten  $\langle 2, 3, 4 \rangle$  eller  $\langle 2, 3, 6 \rangle$ .

**(a)** Beskriv kort hvordan vi kan finne  $L_n$  ved dynamisk programmering.

Svar: 5%

$L_i = \max\{L_j\} + 1$  hvis og bare hvis  $s_j < s_i$ , hvor  $0 < j < i$ .

Dvs.:

$L_i$  er løsningen på  $P(S_i, i)$ , hvor  $S_i$  er et prefiks av  $S$  med lengde  $i$ .

**(b)** Finn tidskompleksiteten til metoden foreslått i (a)

Svar: 4% Dobbelt løkke, dvs.  $O(n^2)$

**(c)** Forklar kort hvordan du kan finne selve sekvensen  $S^*$  ved å føye ekstra informasjon til løsningen foreslått i (a). Bruk gjerne det oppgitte eksempelet for å illustrere ideene.

Svar: 4%

Føy til  $\{P_i\}$  der  $P_i$  peker til nærmeste lavere  $S$ -indeks i den lengste sekvensen som ender i  $s_i$ . Følg så kjeden tilbake fra  $s_n$ .

**(d)** Hva blir tidskompleksiteten i (c)?

Svar: 4% Ingen forandring, dvs.  $O(n^2)$

**(e)** Foreslå en praktisk sammenheng der problemet  $P(S, n)$  er av interesse.

Svar: 4% Dukker opp i forbindelse med mønstergjenkjenning i permutasjoner. (Andre gode forslag premieres).

**Oppgave 5. (8%)**

Vi skal i denne oppgaven se på et praktisk problem knyttet til et rettet nettverk  $G=(V,E)$ . Kantene i  $E$  representerer vannførende kanaler, hver med en spesifisert kapasitet  $c$  kubikkmeter pr. sekund. Kanalene møtes i noder som ikke har noen kapasitetsbeskranking. Kantene har i tillegg en parameter  $InTown$  som har verdien  $True$  dersom kanten ligger i tettbebyggelsen,  $False$  ellers.

(Vi antar at en eventuell oversvømmelse bare vil forekomme i én kanal, dvs. vi ser bare på hvor oversvømmelsen starter.)

**(a)** Hvilken metode vil du bruke for å finne ut om en det er mulig at en oversvømmelse rammer en av kanalene i tettbygd strøk?

Svar: 4% (Retning kan være helning – naturlig å anta DAG.) Bruk Maks-flyt-algoritmen (Ford Fulkerson). Kanter med uendelig kapasitet fra en virtuell kilde  $S$  til noder med utgående åpne kanaler (tilsig, nedbør). Tilsvarende fra noder med fritt avløp til et virtuelt sluk  $T$  (hav etc.). Maks-flyt fra  $S$  til  $T$  gir et *snitt*. Mulig oversvømmelse hvis man har minst én  $InTown$  i dette snittet. Snittet trenger ikke være entydig – det er naturlig å anta at man velger det som er nærmest  $S$ .

**(b)** Hvordan vil du finne ut om en oversvømmelse garantert vil ramme et tettbygd strøk?

Svar: 4% Samme modell som i (a) gir *garantert* oversvømmelse dersom samtlige kanter i snittet også er  $InTown$ .

**Oppgave 6, Øvingsrelaterte oppgaver. (25%)**

**(a)** Kjør Partition på tallene : 13, 7, 11, 4, 6, 2, 0, 32, 29.

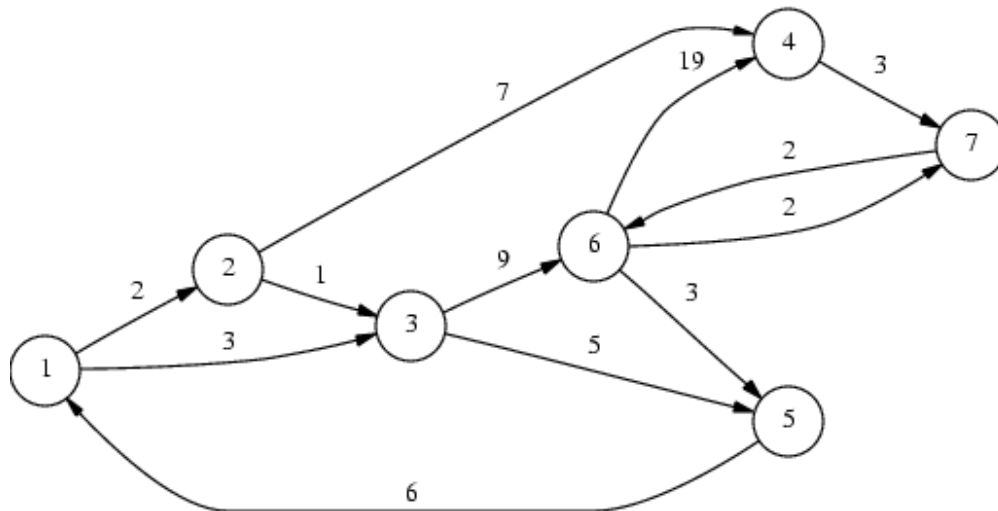
Bruk tallet 13 som pivot-element. Vis høyre og venstre partisjoneringsindeks per steg (i,j).

Svar: 4% (Partisjonerings-indeksene vises ved understrekning. Tallene er allerede partisjonerte – derfor ingen “swapping”)

|   |  |
|---|--|
| 1 2 3 4 5 6 7 8 9                                   |  |
| 1: [ <u>13</u> , 7, 11, 4, 6, 2, 0, 32, <u>29</u> ] | 5: [0, 7, <u>11</u> , 4, 6, 2, <u>13</u> , 32, 29] |
| 2: [ <u>13</u> , 7, 11, 4, 6, 2, 0, <u>32</u> , 29] | 6: [0, 7, 11, <u>4</u> , 6, 2, <u>13</u> , 32, 29] |
| 3: [ <u>13</u> , 7, 11, 4, 6, 2, <u>0</u> , 32, 29] | 7: [0, 7, 11, 4, <u>6</u> , 2, <u>13</u> , 32, 29] |
| 4: [0, <u>7</u> , 11, 4, 6, 2, <u>13</u> , 32, 29]  | 8: [0, 7, 11, 4, 6, <u>2</u> , <u>13</u> , 32, 29] |

**(b)** Hva ville vært optimalt pivot-element generelt sett når Partition blir brukt i Quicksort?

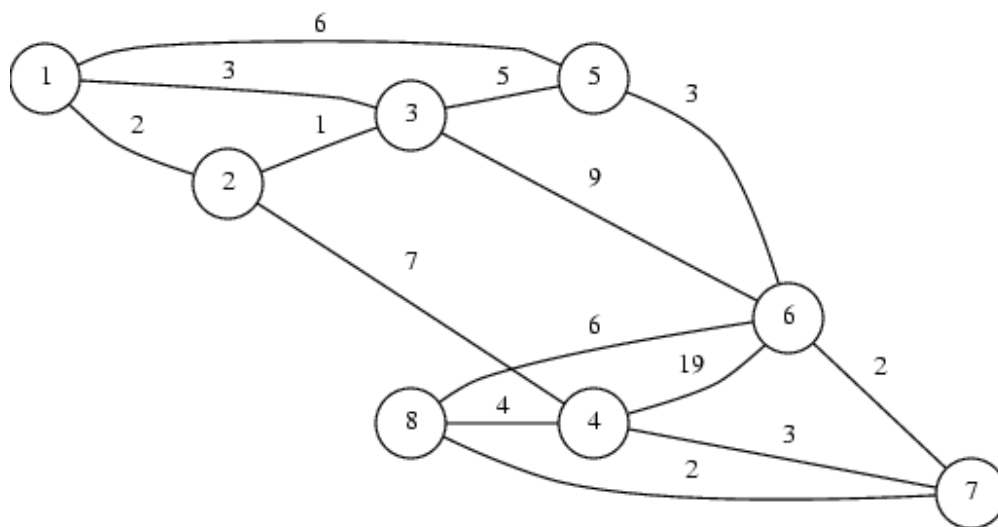
Svar: 2% Medianen



(c) Bruk Dijkstras algoritme til å finne korteste vei fra node 1 til de andre nodene i grafen over. Fyll inn verdier for avstandsfunksjonen  $d$  (for hvert steg i algoritmen) i tabellen under:

Svar: (8%)

| Steg | Node 1 | Node 2 | Node 3 | Node 4   | Node 5   | Node 6   | Node 7   |
|------|--------|--------|--------|----------|----------|----------|----------|
| 1    | 0      | 2      | 3      | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 2    | 0      | 2      | 3      | 9        | $\infty$ | $\infty$ | $\infty$ |
| 3    | 0      | 2      | 3      | 9        | 8        | 12       | $\infty$ |
| 4    | 0      | 2      | 3      | 9        | 8        | 12       | $\infty$ |
| 5    | 0      | 2      | 3      | 9        | 8        | 12       | 12       |
| 6    | 0      | 2      | 3      | 9        | 8        | 12       | 12       |
| 7    | 0      | 2      | 3      | 9        | 8        | 12       | 12       |



(d) Finn minste spenntre i grafen over, ved bruk av Prims algoritme. Fyll inn nodepar (fra-node – til-node) for kantene du legger til i hvert steg i tabellen under:

Svar: 6%

| Steg | Fra-node | Til-node |
|------|----------|----------|
| 1    | 1        | 2        |
| 2    | 2        | 3        |
| 3    | 3        | 5        |
| 4    | 5        | 6        |
| 5    | 6        | 7        |
| 6    | 7        | 8        |
| 7    | 7        | 4        |
| 8    |          |          |

(e) Hva blir summen av kantene i det minimale spenntreet?

Svar: 2% 18

(f) Kan man ha et største spenntre? Hvordan vil du evt. finne det, og hva blir tidskompleksiteten?

Svar: 3%

Å finne et største spenntre er ekvivalent med å finne et minste spenntre. Hvis  $k$  er den største kant-vekten, kan alle kant-vektene  $k_i$  settes til  $k-k_i$  og man kan så kjøre for eksempel Prims algoritme på den omformede grafen. (Eventuelt kan man gjøre om algoritmen til å plukke ut den *største* kanten hver gang istedenfor den *minste*). Kjøretiden er som for å finne minste spenntre:

$$O(|E| \log |V|)$$

Det er generelt bedre å endre datasettet enn å endre algoritmen.

