Løsningsforslag for eksamen i fag TDT4120 Algoritmer og datastrukturer Tirsdag 9. desember 2003, kl. 0900–1500

Faglig kontakt under eksamen:
Arne Halaas, tlf. 41661982; Magnus Lie Hetland, tlf. 91851949.
Hjelpemidler: Alle kalkulatortyper tillatt. Alle trykte og håndskrevne hjelpemidler tillatt.

Oppgave 1 (5%)

a. Anta at vi sammenligner implementasjoner av to algoritmer A_1 og A_2 på samme maskin. For input-størrelse n bruker A_1 $9n^2$ steg mens A_2 bruker $81n \log_2 n$ steg. Hvert steg i A_2 krever dobbelt så mye tid som hvert steg i A_1. For hvilke verdier av n bruker A_1 kortere tid enn A_2?

Svar (5%):
$1 < n < 126$. (Oppgaveteksten gir oss ulikheten $\sqrt[18]{2^n} < n$. Vi antar at n er et heltall.)

Oppgave 2 (15%)

Anta at du har tre tabeller, A, B og C, med positive reelle tall. Hver av tabellene har lengde n.

a. Du vil finne et segment $A[i \ldots j]$ slik at $A[i] \times A[i+1] \times \ldots \times A[j]$ blir størst mulig. Hvordan vil du gå fram? Referer gjerne til algoritmer i pensum. Hva blir kjøretiden?

Svar (5%):
Lager en tabell A' der $A'[k] = \log A[k]$ (for $1 \leq k \leq n$) og bruker algoritmen fra øving 5, "aksjespekulanten." Kjøretiden blir $\Theta(n)$.
(Man kan eventuelt modifisere algoritmen så den finner maksimalt produkt heller enn maksimal sum.)

b. Du ønsker å finne ut om det finnes tre tall a, b og c, slik at A inneholder a, B inneholder b og C inneholder c og slik at $a + b + c = x$ for en gitt x. Beskriv kort (enten

med pseudokode eller dine egne ord) en algoritme som løser problemet i $\Theta(n^2 \log n)$ tid, *worst-case*.

Svar (5%):
Sorter C. For alle par a, b finn $x - (a + b)$ i C med binærsøk.

c. Du ønsker å løse samme problem som i oppgave b, men kan nå anta at A, B og C er heltallstabeller, og at heltallene faller i et tallområde fra 1 til M. Beskriv kort (enten med pseudokode eller dine egne ord) en algoritme som løser problemet i $\Theta(n^2)$ tid, *worst-case*. Gi også (i stikkordsform) eventuelle antagelser om M og maskinvaren du bruker for at kjøretiden skal gjelde.

Svar (5%):
Alternativ 1:
Lag en Boolsk tabell T der $T[c] \leftrightarrow c \in C$. For hvert par a, b, slå opp $x - (a + b)$ i T. Antar at T får plass i minnet.
Alternativ 2:
Sorter B og C. For hvert tall a, start et søk etter $x - a$ fra 1 i B og fra n i C. Hvis summen av b og c er for stor, flytt nedover i C; ellers, flytt oppover i B. Denne er litt vanskeligere, men gjør ingen antagelser om M, og krever ikke at tallene er heltall.
Begge alternativer gir full uttelling.

Oppgave 3 (30%)

Du har oppdaget følgende pseudokode i en gammel lærebok i algoritmer. Du er usikker på hvilket språk læreboken er skrevet på, og har litt problemer med å skjønne enkelte av ordene i pseudokoden:

```
BRILLIG(A[1...N]):
if N = 1:
    return A[1], A[1]
slithy ← ⌊N/2⌋
gyre, gimble ← BRILLIG(A[1...slithy])
wabe, mimsy ← BRILLIG(A[slithy + 1...N])
if gyre < wabe:
    borogroves ← gyre
else
    borogroves ← wabe
if gimble < mimsy:
    mome ← mimsy
else
    mome ← gimble
return borogroves, mome
```

a. Hva gjør algoritmen BRILLIG?

Svar (5%):
Den finner minimum og maksimum i A.

b. Anta at $N = 256$. Hvor mange sammenligninger av typen *gyre* $<$ *wabe* og *gimble* $<$ *mimsy* utføres totalt? (Vi er kun ute etter ett tall.)

Svar (5%):
510

c. Sett opp en eksakt rekurrens som uttrykker antall sammenligninger som en funksjon $C(N)$. Anta at $N = 2^M$ for et heltall M.

Svar (5%):
$C(1) = 0$, $C(N) = 2 \cdot C(\frac{N}{2}) + 2$ for $N > 1$.

d. Løs rekurrensen i oppgave c. Uttrykk svaret eksakt, uten bruk av asymptotisk notasjon.

Svar (5%):
$C(N) = \sum_{i=1}^{\log_2 N} 2^i = 2N - 2$

Du bestemmer deg for å optimalisere algoritmen. Du endrer utsagnet

if $N = 1$:
    return $A[1], A[1]$

til det følgende:

if $N = 2$:
    if $A[1] < A[2]$:
        return $A[1], A[2]$
    return $A[2], A[1]$

e. Sett opp en eksakt rekurrens som uttrykker antall sammenligninger som en funksjon $C(N)$. Anta at $N = 2^M$ for et heltall M. Rekurrensen skal også telle sammenligninger av typen $A[1] < A[2]$.

Svar (5%):
$C(2) = 1$, $C(N) = 2C(\frac{N}{2}) + 2$ for $N > 2$
(Antar at $M \geq 1$.)

f. Løs rekurrensen i oppgave e. Uttrykk svaret eksakt, uten bruk av asymptotisk notasjon.

Svar (5%):
$C(N) = \frac{N}{2} + \sum_{i=1}^{\log_2 N-1} 2^i = \frac{3}{2}N - 2$

Oppgave 4 (30%)

a. Anta at du har en urettet graf. Du vet at hver node har maksimalt 3 naboer. Argumenter svært kort for at det er mulig å finne en to-farging av grafen som er slik at hver node maksimalt har 1 konflikt (nabo med samme farge).
Hint: Bruk det totale antall konflikter i argumentasjonen.

Svar (5%):
Start med vilkårlig tofarging. Hvis en node har mer enn 1 konflikt, skift farge. Dette reduserer det totale antallet konflikter (med 1 eller 3). Gjenta til antall konflikter ikke kan reduseres.

Anta at du har oppgitt et flytnettverk definert ved følgende kapasitetsmatrise:

$$C = \begin{bmatrix} 0 & 7 & 6 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Her kan vi for eksempel se at kapasiteten mellom node 4 og node 5 er $C[4,5] = 5$. Anta at node 1 er kilden og at node 7 er sluket.

b. Hvor mange mulige snitt finnes det mellom kilde og sluk?

Svar (5%):
$2^{(7-2)} = 32$
En alternativ tolkning (som ikke er i tråd med bokens definisjon, og derfor ikke gir full score) er at vi her kun er ute etter snitt som kan oppstå under kjøring av FORD-FULKERSON, det vil si alle snitt der nodene i S (den "venstre" halvdelen) kan nås fra s via andre noder i S. Det er 21 slike snitt.

c. Hvordan kan man bruke FORD-FULKERSON til å finne et minimalt snitt? Skriv kort.

Svar (5%):
Nodene som besøkes i siste iterasjon havner "til venstre" for snittet.

d. Finn et minimalt snitt i flytnettverket. Beskriv snittet ved å oppgi alle nodene som befinner seg på samme side som kilden.

Svar (5%):
1 og 2
(Evt. kan man oppgi bare 2)

e. Et sett med stier i en graf $G = (V, E)$ er kant-disjunkte hvis ingen kant i E inngår i mer enn en av stiene i settet. Beskriv en algoritme som finner (det maksimale) antall kant-disjunkte stier mellom to gitte noder s og t i en urettet graf.

Svar (5%):
Gi alle kanter kapasitet 1 og finn maksimal flyt fra s til t.

f. Du ønsker å øke den maksimale flyten i et flyt-nettverk så mye som mulig, men du får bare lov til å endre kapasiteten på én kant. Hvordan finner du en slik kant? (Bruk pseudo-kode eller egne ord. Anta at du har algoritmer tilgjengelig for å finne maks-flyt og et minimalt snitt.) Hva blir kjøretiden (*worst-case*, i Θ-notasjon)? Vil det alltid være mulig å finne en slik kant? (Begrunn svaret.)

Svar (5%):
Finn et minimalt snitt. For hver av kantene i snittet, sett kapasiteten på kanten til ∞ og finn deretter maksimal flyt. Velg kanten som gir størst økning.
Ikke alltid mulig å øke maks-flyten: Det kan være flere (disjunkte) minimale snitt.
Med EDMONDS-KARP blir kjøretiden $\Theta(VE^3)$.

Oppgave 5 (5%)

a. Følgende problem ble gitt på eksamen i fjor:

> Du har oppgitt et sett S bestående av N reelle tall, samt et reelt tall T og et heltall $K \leq N$. Finnes det en delmenge Q av S med K elementer, der summen av elementene i Q er maksimalt lik T?

Det finnes en algoritme som løser problemet i $\Theta(N)$ tid. Er det rimelig å tro at vi kan finne en like effektiv løsning på følgende problem? Begrunn svaret.

> Du har oppgitt et sett S bestående av N reelle tall, samt et reelt tall T og et heltall $K \leq N$. Finnes det en delmenge Q av S med maksimalt K elementer, der summen av elementene i Q er lik T?

Svar (5%):
En løsning på problemet kan (for $K = N$) brukes til å løse SUBSET-SUM-problemet. Dermed er problemet vårt NP-komplett, og det er ikke rimelig å tro at vi kan finne en $\Theta(N)$-løsning.

Oppgave 6 (15%)

```
SUM(N)
    top ← 1; S[top] ← N; S[0] ← 2; stacksum ← N
    WRITE('N = ')
    while top > 0
        for i in 1 ... top − 1
            WRITE(S[i], ' + ')
        WRITELINE(S[top])
        while S[top] = 1
            top ← top − 1
            stacksum ← stacksum − 1
        if top > 0:
            S[top] ← S[top] − 1
            stacksum ← stacksum − 1
            while stacksum < N
                top ← top + 1
                if N − stacksum ≤ S[top − 1]
                    S[top] ← N − stacksum
                    stacksum ← N
                else
                    S[top] ← S[top − 1]
                    stacksum ← stacksum + S[top]
            WRITE(' = ')
```

a. Hva skriver funksjonen SUM ut hvis $N = 6$? Anta at S er en tabell med så mye plass som er nødvendig. Anta at funksjonen WRITELINE skriver ut argumentene sine (uten mellomrom imellom) og starter en ny linje, mens WRITE skriver ut argumentene sine (uten mellomrom imellom) uten å starte en ny linje.

Svar (15%):

```
N = 6
  = 5 + 1
  = 4 + 2
  = 4 + 1 + 1
  = 3 + 3
  = 3 + 2 + 1
  = 3 + 1 + 1 + 1
  = 2 + 2 + 2
  = 2 + 2 + 1 + 1
  = 2 + 1 + 1 + 1 + 1
  = 1 + 1 + 1 + 1 + 1 + 1
```

Merk: Hvis man tolker **for**-løkken slik at den i første iterasjonen setter *i* lik 1 og 0 blir første linje i koden slik:

```
N = 6 + 2 + 6
```

Det trekkes ikke for dette.