NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

LØSNINGSFORSLAG, AVSLUTTENDE EKSAMEN I

TDT4120

ALGORITMER OG DATASTRUKTURER

Lørdag 12. august 2006
Kl. 09.00 – 13.00

Faglig kontakt under eksamen:

Magnus Lie Hetland, tlf. 918 51 949

Hjelpemidler:

Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Sensurdato:

2. september 2006. Resultater gjøres kjent på http://studweb.ntnu.no og sensurtelefon 815 48 014.

Viktig:

Les hele oppgavesettet før du begynner. Les oppgaveformuleringene grundig. Det er angitt i poeng hvor mye hver deloppgave teller ved sensur. Gjør antagelser der det er nødvendig. Skriv kort og konsist. Skriv fortrinnsvis i rutene på oppgavearket. Lange forklaringer som ikke direkte besvarer oppgaven tillegges liten vekt.

Oppgave 1 (20%)

Hver av de følgende deloppgavene består av 5 utsagn. Ved hvert av disse utsagnene skal du sette kryss ved «ja» eller «nei». Sett kryss ved «ja» hvis du mener utsagnet er sant og ved «nei» hvis du mener at utsagnet er usant eller ikke stemmer (evt. ikke gir mening). Ikke kryss av for «nei» hvis du er *enig* i et utsagn som bruker ordet «ikke». Kryss da av for «ja», som betyr at utsagnet *stemmer*.

Hvert utsagn der krysset er satt riktig gir 1 poeng, et utsagn uten kryss gir 0 poeng, og et utsagn der krysset er satt galt gir –1 poeng. En negativ totalsum for en deloppgave (a–e) rundes opp til 0.

Les oppgavene nøye. Oppgavetekstene er utformet slik at svarene ikke skal være opplagte. Svar bare dersom du er sikker på at du har forstått oppgaven og at du vet svaret.

a) Ta stilling til følgende påstander om sorteringsalgoritmen MERGESORT.

[ ] Ja [x] Nei 1. Den utnytter overlappende delproblemer.

[ ] Ja [x] Nei 2. Den er asymptotisk mer effektiv enn HEAPSORT.

[ ] Ja [x] Nei 3. Den har i verste tilfelle kjøretid $\Theta(n^2)$.

[x] Ja [ ] Nei 4. Den egner seg godt til rekursiv implementasjon.

b) Betrakt rekurrens-ligningen $T(n) = 2T(\lfloor n/2 \rfloor) + n$.

[ ] Ja [x] Nei 1. Rekurrensen er typisk for dynamisk programmering.

[x] Ja [ ] Nei 2. Rekurrensen er typisk for *splitt-og-hersk*-algoritmer.

[x] Ja [ ] Nei 3. $T(n) \in \Omega(n \log n)$.

[x] Ja [ ] Nei 4. Rekurrensen beskriver bl.a. *best-case*-kjøretiden til QUICKSORT.

c) Ta stilling til følgende utsagn om *selection*-problemet.

[x] Ja [ ] Nei 1. Det kan løses i $O(n)$ tid i verste tilfelle.

[ ] Ja [x] Nei 2. Det krever sortering av datasettet.

[ ] Ja [x] Nei 3. RANDOMIZEDSELECT bruker grådighet.

[ ] Ja [x] Nei 4. RANDOMIZEDSELECT bruker memoisering.

d) Betrakt et problem P som kan løses med en grådig algoritme.

[ ] Ja [x] Nei 1. Det vil lønne seg å bruke dynamisk programmering for å løse P.

[x] Ja [ ] Nei 2. Lokalt optimale valg vil gi en optimal løsning for P.

[x] Ja [ ] Nei 3. Problemet P har optimal delstruktur.

[x] Ja [ ] Nei 4. Kjøretiden for å løse P vil kunne bli mer enn lineær.

e) Betrakt problemet VERTEX-COVER — å finne et *vertex cover* med størrelse k på en graf $G = (V, E)$.

[x] Ja [ ] Nei 1. Størrelsen til et *cover* er antall noder det består av.

[x] Ja [ ] Nei 2. Det finnes en kjent reduksjon fra VERTEX-COVER til HAM-CYCLE.

[ ] Ja [x] Nei 3. Det finnes en kjent reduksjon fra VERTEX-COVER til MINIMAL-SPANNING-TREE.

[ ] Ja [x] Nei 4. Parameteren k kan ikke være større enn $|E|/2$.

Oppgave 2 (35%)

a) Beskriv FLOYD-WARSHALL med pseudokode.

Svar (10%):

```
FLOYD–WARSHALL(W)
n ← rows[W]
D^(0) ← W
for k ← 1 to n
    for i ← 1 to n
        for j ← 1 to n
            D^(k)[i, j] ← min(D^(k–1)[i, j], D^(k–1)[i, k] + D^(k–1)[k, j])
```

b) Betrakt en urettet, sammenhengende, vektet graf G. Anta at alle kantene i G har ulike vekter. Gi en kort begrunnelse for hvorfor den letteste kanten (den med lavest kostnad) i et hvilket som helst snitt (*cut*) i G må være med i et minimalt spenntre over G.

Svar (8%):

Hvis ikke kan den den byttes ut med en lettere kant, som gir et lettere tre, og vi får en selvmotsigelse.

Merk: Her er det ikke nok å vise hva Prims og Kruskals algoritmer ville ha gjort. Man må i så fall påpeke at det kun finnes ett mulig minimalt spenntre (ettersom alle kantene er forskjellige), så enhver algoritme ville ha måttet gjøre de samme valg.

FORD-FULKERSON er en generell metode for å finne maksimal flyt i flytnettverk. Den generelle metoden har ikke garantert polynomisk kjøretid. EDMONDS-KARP er en variant av FORD-FULKERSON som har polynomisk kjøretid på grunn av måten den velger flytforøkende veier/stier på.

c) Hvordan velger EDMONDS-KARP flytforøkende stier? (Svar kort.)

Svar (8%): Den velger de korteste (de med færrest kanter), f.eks. vha. BFS.

Å svare at den velger den kanten som gir størst flytøkning e.l. er galt, og gir ingen uttelling.

Betrakt følgende tabell, *A*: [2, 2, 4, 0, 3, 4, 3]. Denne tabellen skal sorteres med tellesortering (COUNTING-SORT). Anta at du på forhånd vet at verdiene i tabellen kun kan falle i området fra og med 0 til og med 5. Anta at telle-tabellen din heter *C* og resultat-tabellen din heter *B* (som i læreboka).

e) Oppgi innholdet i tabellen *B* og *C* for hvert tall som settes inn i *B* (det vil si, *rett etter* at tallet har blitt satt inn og *C* har blitt oppdatert). Oppgi kun de første 3 trinnene. Sett en strek («–») for eventuelle uinnfylte plasser i tabellene. Merk at et tall alt er fylt inn i *C*.

Svar (9%):

Trinn 1:

B: [, , , , , ,]

C: [, , 3, , ,]

B: [–, –, –, –, 3, –, –], C: [1, 1, 3, 4, 7, 7]

Trinn 2:

B: [, , , , , ,]

C: [, , , , ,]

B: [–, –, –, –, 3, –, 4], C: [1, 1, 3, 4, 6, 7]

Trinn 3:

B: [, , , , , ,]

C: [, , , , ,]

B: [–, –, –, 3, 3, –, 4], C: [1, 1, 3, 3, 6, 7]

Oppgave 3 (20%)

Betrakt følgende pseudokode:

```
PLOFFSKIN(gromboolian, chankly, bore)
if chankly ≥ bore
        return 42
for pelican ← chankly … bore
        print gromboolian[pelican]
jee ← ⌈(chankly – bore) / 4⌉
PLOFFSKIN(gromboolian, chankly+jee, bore–jee)
```

a) Anta at *A* er en 1-indeksert tabell med lengde *n*. Hva blir kjøretiden til PLOFFSKIN(*A*, 1, *n*)? Oppgi kjøretiden i Θ-notasjon. Vis hovedtrekkene i utregningen.

Svar (10%):

Antall **print**-operasjoner er:

$T(n) = T(\lfloor n/2 \rfloor) + n$ (evt. $T(\lfloor n/2 \rfloor) + \Theta(n)$)

Gjetter $T(n) \in O(n)$, som for SELECT. Bruker induksjon på $T(n) \leq c \cdot n$:

$T(n) \leq c \cdot (n/2) + n = (c/2 + 1) \cdot n \leq c \cdot n$ (stemmer hvis $c \geq 2$). Tilsvarende (omvendt) for $\Omega(n)$.

Bruk av andre løsningsmetoder som f.eks. Masterteoremet gir også full score.

Det følgende er en variant av Dijkstras algoritme som er litt forskjellig fra algoritmen slik den er presentert i læreboka. Anta at køen Q er implementert med en uordnet, lenket liste.

```
DIJKSTRA(G, w, s)
1   INITIALIZE-SINGLE-SOURCE(G, s)
2   S ← ∅
3   Q ← V[G]
4   while Q ≠ ∅
5       d_min ← ∞
6       for each vertex v ∈ Q
7           if d[v] < d_min
8               d_min ← d[v]
9               u ← v
10      Q ← Q − {u}
11      S ← S ∪ {u}
12      for each vertex v ∈ Adj[u]
13          RELAX(u, v, w)
```

b) Hva er kjøretiden til algoritmen ovenfor? Oppgi kjøretiden i Θ-notasjon. Gi en kort begrunnelse for svaret.

Svar (10%): $\Theta(|V|^2)$

Begrunnelse: I iterasjon k av linje 4–13 gjennomfører løkken på linje 6–9 $|V|-k+1$ iterasjoner. Kostnaden blir altså $|V| + (|V| - 1) + \ldots + 1$ som er $\Theta(|V|^2)$.

Enklere forklaringer som er inne på rett spor gir også full score hvis kjøretiden er riktig.

Oppgave 4 (25%)

Butikkjeden «8–12» skal bygge butikker langs hovedgata, som går mellom øst og vest gjennom Metropolis sentrum. De har fått tilbud om å kjøpe flere bygninger, med posisjoner $x[1] \ldots x[n]$ (i meter fra starten av gata, i stigende rekkefølge fra vest-enden). Kostnaden ved å kjøpe disse bygningene ser man bort fra; det er her kun langsiktig fortjeneste som skal vurderes. Den årlige fortjenesten for en butikk kommer an på plasseringen, og fortjenesten for posisjon $x[i]$ er gitt ved $f[i] > 0$.

Bystyret har satt ned foten, og bestemt at det må minst være 500 meter mellom butikkene. Du har fått i oppdrag å beregne det lovlige utvalget av butikkposisjoner som gir størst total årlig fortjeneste.

Innledende analyser fra butikkjedens egne datafolk tyder på at man bør beregne en tabell $y[1]\ldots y[n]$, der $x[y[i]]$ er posisjonen til det østligste huset vest for $x[i]$ som har en lovlig avstand til $x[i]$. Hvis $y[i] = 0$, betyr det at det ikke finnes noen lovlige posisjoner vest for $x[i]$.

a) Beskriv kort en algoritme som beregner tabellen y så effektivt som mulig. Hva blir kjøretiden? (Oppgi svaret i Θ-notasjon.)

Svar (10%):

```
j ← 1
for i ← 1 to n
    while x[j] < x[i] – 500
        j ← j + 1
    y[i] ← j – 1
```

Kjøretid: $\Theta(n)$

Evt. kan man beregne $x[i] – 500$ og bruke binærsøk for å finne dette, med kjøretid $\Theta(n \log n)$. Dette vil ikke gi full uttelling, men er atskillig bedre enn en løsning med kvadratisk kjøretid.

b) Anta at tabellen y fra deloppgave a) er tilgjengelig (det vil si, at den alt har blitt beregnet). Beskriv kort en algoritme som så effektivt som mulig finner den optimale fortjenesten med lovlig plasserte butikker. Hva blir kjøretiden? (Oppgi svaret i Θ-notasjon.)

Svar (15%):

Bruker dynamisk programmering. Optimale delproblemer $F[i]$ angir beste mulige fortjeneste fra vest-enden til og med hus nummer i. Den optimale løsningen inkluderer enten $f[i]$ eller ikke. Hvis hus i inkluderes, blir det neste mulige huset (bakover) nummer $y[i]$, og resten av fortjenesten blir $F[y[i]]$. Hvis huset ikke tas med, blir den optimale løsningen rett og slett $F[i–1]$. Algoritmen blir:

```
F[0] ← 0
F[1] ← f[1]
for i ← 2 to n
        F[i] ← max{f[i] + F[y[i]], F[i–1]}
return F[n]
```

Merk: En grundig forklaring kreves ikke. Forklaringer som beskriver dette som DAG-Longest-Path (dvs. omvendt av DAG-Shortest-Path) er fullgode.

Kjøretid: $\Theta(n)$