NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

AVSLUTTENDE EKSAMEN I

TDT4120/IT1105

ALGORITMER OG DATASTRUKTURER

Fredag 8. desember 2006
Kl. 09.00 – 13.00

Faglig kontakt under eksamen:

Magnus Lie Hetland, tlf. 918 51 949

Hjelpemidler:

Alle trykte og håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Sensurdato:

8. januar. Resultater gjøres kjent på http://studweb.ntnu.no og sensurtelefon 815 48 014.

Viktig:

Les hele oppgavesettet før du begynner, disponér tiden og forbered evt. spørsmål til faglærer kommer til eksamenslokalet. Les oppgaveformuleringene grundig. Det er angitt i prosent hvor mye hver deloppgave teller ved sensur. Gjør antagelser der det er nødvendig. Skriv kort og konsist. Skriv fortrinnsvis i rutene på svarskjemaet (dvs. ikke legg ved egne ark med mindre det er tvingende nødvendig). Bruk gjerne blyant (og viskelær). Lange forklaringer som ikke direkte besvarer oppgaven tillegges liten eller ingen vekt.

Svarskjema

Vennligst oppgi svarene dine i rutene nedenfor. Oppgavetekstene begynner på side 4.

1a (6%):
1b (6%):
1c (6%):
1d (6%):
1e (6%):
2a (10%): $D^{(0)} \ldots D^{(4)}$: $\begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}$
3a (5%):
3b (5%):
3c (5%): $T(n) = \Theta($ $)$.

4a (5%):	
4b (3%): Θ( )	
4c (7%):	
4d (5%): Θ( ).	
5a (5%):	
5b (5%):	
6a (2%):	6b (3%):
6c (10%): Kjøretid: Θ( )	

Oppgave 1 (30%)

Svar så kort og presist som mulig på de følgende spørsmålene.

a) Hvis input-tallene ikke er uniformt fordelt vil ikke BUCKET-SORT nødvendigvis få lineær kjøretid. Hvorfor?

b) Hvis c er tabellen som brukes i LCS-LENGTH(X, Y), hva representerer tallet $c[i, j]$, der $1 \leq i \leq length[X]$ og $1 \leq j \leq length[Y]$?

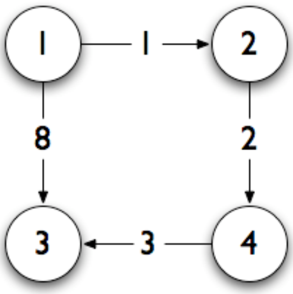
c) Hvorfor er det ikke noe poeng i å bruke en (binær) heap som prioritetskø i DIJKSTRA hvis grafen er komplett, dvs. antall kanter er $\Theta(V^2)$?

d) Anta at du har et lineært program som inneholder en variabel x som kan være negativ. Hva vil du gjøre med denne variabelen hvis det lineære programmet skal konverteres til standardform?

e) Argumentér kort for at HAM-CYCLE kan ses som et spesialtilfelle av TSP.

Oppgave 2 (10%)

Betrakt følgende graf $G = (V, E)$:



Du skal simulere/utføre FLOYD-WARSHALL på grafen G.

a) Fyll ut tabellene $D^{(0)} \dots D^{(4)}$ på svarskjemaet.

Oppgave 3 (15%)

Anta at utsagnet «Algoritme A har kjøretid $O(n^2)$» gir en tett, øvre grense for kjøretiden til A.

a) Innebærer dette at A har en kjøretid på $\Theta(n^2)$? Begrunn svaret.

b) Innebærer dette at A har en *worst-case*-kjøretid på $\Omega(n^2)$? Begrunn svaret.

c) Løs rekurrensen $T(n) = 16T(n/4) + n^2$. Oppgi svaret i Θ-notasjon. Begrunn svaret svært kort.

Oppgave 4 (20%)

Betrakt følgende algoritme:

WHATZIT(A, B):
for $i \leftarrow 1 \dots n$
    $B[i] \leftarrow 0$
for $i \leftarrow 1 \dots n$
    $B[A[i]] \leftarrow B[A[i]] + 1$

Anta at A og B er tabeller av lengde n som kan inneholde heltall i verdiområdet $1 \dots n$. Anta at første element i både A og B har indeks 1.

a) Hva gjør algoritmen WHATZIT? (Dvs. hva er hensikten med koden? En direkte «gjenfortelling» av pseudokoden med dine egne ord gir ikke nødvendigvis noen uttelling.)

b) Hva er kjøretiden til algoritmen WHATZIT? Oppgi svaret i Θ-notasjon.

Betrakt følgende algoritme:

```
WAZZUP(A, a, b, c):
if c – b < 5
    return b, c
x = b + 5
y = c – 5
if a ≥ A[x]
    return WAZZUP(A, a, x, c)
else
    return WAZZUP(A, a, b, y)
```

Anta at A er en sortert tabell med n heltall, og at a, b og c er heltallsparametre.

c) Hva gjør algoritmen WAZZUP hvis den kalles som WAZZUP(A, a, 1, n)? (Dvs. hva er hensikten med koden? En direkte «gjenfortelling» av pseudokoden med dine egne ord gir ikke nødvendigvis noen uttelling.) Anta at a er et tall som finnes i A.

d) Hva er kjøretiden til algoritmen WAZZUP hvis den kalles som WAZZUP(A, a, 1, n)? Anta at a er et tall som finnes i A. Oppgi svaret i Θ-notasjon. Gi en kort begrunnelse for svaret.

Oppgave 5 (10%)

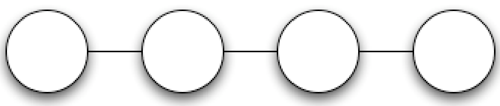
Svar så kort og presist som mulig på de følgende spørsmålene.

a) Professor Znurrebarth har laget en algoritme for å beregne kvadratrøtter (med begrenset presisjon). Kjøretiden for å beregne kvadratroten av heltallet n med algoritmen hans er $\Theta(\lg n)$. Han mener denne kjøretiden er sublineær (asymptotisk raskere enn lineær). Er du enig eller uenig? Begrunn svaret.

b) Du har observert at dataspillet *Slurm Invaders* ligner svært på det NP-komplette problemet VERTEX-COVER, og du akter å bruke denne likheten til å vise at *Slurm Invaders* er NP-komplett. Hvordan vil du gå frem? Du kan anta at du alt har vist at *Slurm Invaders* er i mengden NP. (Skriv kort.)

Oppgave 6 (15%)

En *uavhengig mengde* (*independent set*) av noder i en graf $G = (V, E)$ er et subsett I av V som er slik at det ikke finnes kanter i E mellom noen av nodene i I. Å finne store uavhengige mengder er vanligvis vanskelig (NP-komplett) – men anta her at grafen G har en spesiell struktur: Nodene kan ordnes som en sekvens $[v_1, v_2, \ldots, v_n]$, og det er kun kanter mellom naboer i denne sekvensen, som illustrert ved følgende figur:



Anta at hver node v_i har en vekt w_i. Du ønsker å finne en uavhengig mengde S i grafen G med størst mulig vekt (dvs., slik at summen av nodevektene i den uavhengige mengden blir størst mulig).

En intuitivt tiltalende (grådig) løsning vi kan kalle «Tyngst først» er å først finne den tyngste noden v, legge den til i mengden S, og fjerne v og naboene til v fra G. Dette gjentas til grafen er tom for noder.

a) Gi et eksempel på en graf G av typen beskrevet ovenfor, der algoritmen «Tyngst først» ikke returnerer den tyngste uavhengige mengden. Oppgi svaret som en sekvens av nodevekter.

En annen intuitivt tiltalende løsning vi kan kalle «Annenhver» er å dele nodene i to mengder, S_1 og S_2, der S_1 er mengden av noder på oddetallsposisjoner, og S_2 er mengden av noder på partallsposisjoner. Vi lar så mengden S være den av S_1 og S_2 som har størst vekt.

b) Gi et eksempel på en graf G av typen beskrevet ovenfor, der algoritmen «Annenhver» ikke returnerer den tyngste uavhengige mengden. Oppgi svaret som en sekvens av nodevekter.

c) Beskriv (tekstlig eller med pseudokode) en algoritme som finner vekten til den tyngste uavhengige mengden i en graf G (med vekter w_i) med struktur som beskrevet ovenfor. Algoritmen skal være så effektiv som mulig. Oppgi kjøretiden i Θ-notasjon som en funksjon av antall noder, n. Skriv kort.
Merk: Algoritmen skal kun finne/returnere *den totale vekten* til den tyngste uavhengige mengden – ikke hvilke noder/vekter mengden består av.