Avsluttende eksamen i
TDT4120 Algoritmer og datastrukturer

Eksamensdato	5. august 2008
Eksamenstid	0900–1300
Sensurdato	26. august
Språk/målform	Bokmål
Kontakt under eksamen	Magnus Lie Hetland (tlf. 91851949)
Tillatte hjelpemidler	Alle trykte/håndskrevne; bestemt, enkel kalkulator

Vennligst les hele oppgavesettet før du begynner, disponer tiden og forbered evt. spørsmål til faglærer kommer til eksamenslokalet. Gjør antagelser der det er nødvendig. Skriv kort og konsist. Lange forklaringer og utledninger som ikke direkte besvarer oppgaven tillegges liten eller ingen vekt. Ved sensur vektlegges oppgavene som angitt. For hver oppgave teller alle deloppgaver like mye.

Oppgave 1 (25%)

a. Du kjører dybde-først–søk (DFS) i en urettet graf. Hva slags kanter kan du ende opp med i DFS-treet?

Teorem 22.10: *In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.*

Her kan man selvfølgelig si at selve *treet* kun kan inneholde *tree edges* («tre-kanter»). Et slikt svar aksepteres også.

Du kommer over et program som søker i binære søketrær. Du er usikker på om programmet bruker en eller annen form for balansering (som i *red-black*–trær) eller ikke. Du skal likevel forsøke noe å si noe om kjøretiden til ett enkelt søk med dette programmet.

b. Fyll inn de fire ulike kjøretidsbeskrivelsene i tabellen nedenfor.

Best-case	$\Omega(1)$	$O(1)$ / $O(\lg n)$
Worst-case	$\Omega(\lg n)$	$O(n)$

Her skal man gi øvre og nedre grenser for en samling kjøretidsfunksjoner (dvs. både med og uten balansering).

Merk: Grensen $O(\lg n)$ for best-case er basert på antagelsen at vi kan stå overfor et tre der objektene lagres i løvnodene (slik som det gjøres i for eksempel 2-3-trær), mens O(1) er for trær der objektene også lagres i interne noder, sammen med nøklene. Slik oppgaven er formulert ville begge deler kunne være akseptable svar.

Anta at du skal implementere en prioritetskø, og vurderer om du skal bruke en sortert liste, en usortert liste, eller en haug (*heap*).

c. For hver løsning, hva blir kjøretiden for å sette inn ett element, for å ta ut ett element, og for å sette inn og ta ut n elementer? (For det siste tilfellet, anta at operasjonene gjøres for eksempel annenhver gang, så køens størrelse holdes på $\Theta(n)$.) Fyll inn tabellen nedenfor.

Metode	**Én innsetting**	**Ett uttak**	***n* inn + *n* ut**
Sortert liste	$O(n)$	$O(1)$	$O(n^2)$
Usortert liste	$O(1)$	$O(n)$	$O(n^2)$
Haug (heap)	$O(\lg n)$	$O(\lg n)$	$O(n \cdot \lg n)$

d. Løs rekurrensen $T(n) = 5T(n^{0.2}) + \lg n$. Oppgi svaret i Θ-notasjon. Begrunn svaret kort.

Standard variabelskifte ($m = \lg_5 n$ og $S(m) = T(5^m)$) gir $S(m) = 5S(m/5) + \Theta(m)$, dvs. $S(m) \in \Theta(m \cdot \lg m) = \Theta(\lg(n) \cdot \lg(\lg(n)))$.

Oppgave 2 (25%)

a. Forklar kort hvordan hauger (heaps) kan brukes til konstruksjon av Huffman-koder.

Så lenge heapen har mer enn ett element:
Trekk ut de to minste elementene
Lagre de to tilsvarende kodeverdiene
Summér vekten av de to elementene i et sum-element
Sett sum-elementet inn i haugen

Betrakt følgende algoritme, BIDIRECTIONAL-DIJKSTRA, som er ment å finne den korteste avstanden $d(s, t)$ mellom nodene s og t i en urettet graf med positive kantvekter:

> La A og B være to instanser (kjøringer) av Dijkstras algoritme som starter i henholdsvis s og t. A og B alternerer, så etter hver iterasjon i A (dvs. hver node som farges svart) kjøres en iterasjon i B, og omvendt. Når A og B møtes i en node x (dvs. begge har farget x svart) vil $d(s, t) = d(s, x) + d(x, t)$.

e. Er algoritmen BIDIRECTIONAL-DIJKSTRA korrekt? Begrunn svaret.

Nei. Et enkelt moteksempel holder her.

Oppgave 3 (25%)

Det følgende er en del av en kjent algoritme (med endrede variabelnavn):

```
while A ≠ B
    C ← EXTRACT-MIN(A)
    for each D ∈ Adj[C]
        if D ∈ A and w(C, D) < E[D]
            E[D] ← w(C, D)
```

a. Hvilken algoritme er det snakk om?

MST-PRIM.

Merk: Enkelte studenter ble litt forvirret av at det manglet en linje «inne i» koden her (linjen som setter forgjengeren til noden som ble plukket ut). Det burde likevel være klart at dette er et utdrag av Prims algoritme.

Betrakt følgende algoritme, KNOTTY. Anta at første element i sekvensen a er $a[0]$ og at sekvensen inneholder n elementer.

```
1   i ← 1
2   j ← 2
3   while i ≤ n
4       if a[i – 1] ≥ a[i]:
5           i ← j
6           j ← j + 1
7       else
8           a[i] ↔ a[i – 1]
9           i ← i – 1
10          if i = 0
11              i ← 1
```

Algoritmen KNOTTY er ment å sortere sekvensen a i stigende rekkefølge, men 2 av linjene i pseudokoden er gale. Oppgi (i tabellen nedenfor) hvilke linjer det gjelder, og hvordan de kan korrigeres.

(Du kan anta at indenteringen er korrekt, og trenger ikke angi den i svaret ditt.)

Linje	Korrigert versjon
3	while i < n
4	if a[i – 1] ≤ a[i] # Her kan man også bruke <

Oppgave 3 (25%)

Et handelsskip følger en fast rute som går fra havn A til havn Z, og som går innom et antall (potensielt 0) havner på veien (den besøker n havner i alt). I hver havn kan kapteinen kjøpe og selge antikviteter. Prisen for hver antikvitet varierer fra havn til havn, og antikviteten kan naturligvis kun selges etter at den er kjøpt. Skipet har begrenset kapasitet (i kilo).

a. Vis at å finne en plan for kjøp og salg som maksimerer profitten er NP-HARD. (Vis tydelig at du benytter den generelle metoden fra pensum for denne typen bevis.)

1-0-KNAPSACK kan reduseres til dette problemet: Anta at vi bare har to planeter. Antikvitetene er for eksempel gratis på Hesperus og har sin KNAPSACK-pris på Ix. Vekten er gitt av KNAPSACK-vekten.

Merk: Fagets FAQ sier at NP-Hard ikke er pensum – det tas hensyn til dette ved sensur.

Anta at du har oppgitt et sett S med linjestykker i planet, angitt ved koordinatene til deres start- og sluttpunkter. Du har også oppgitt to andre punkter i planet, p_1 og p_2, og ønsker å finne den korteste veien fra p_1 til p_2 uten å krysse noen av linjestykkene i S.

b. Skisser kort en effektiv algoritme som løser problemet. Hva blir kjøretiden? Begrunn svaret.

Her kan man bruke spesialmetoder fra computational geometry (*plane sweep*), men følgende løsning er fullgod: Veien fra p_1 til p_2 består av rette linjestykker, og går innom endepunkter i S. Konstruer en graf med kanter mellom alle endepunkter der disse kantene ikke krysser noen linjestykker, og lengden er geometrisk lengde i planet. Bruk så Dijkstras algoritme. Det tar kubisk tid å generere grafen, som dominerer kjøretiden til Dijkstra. Kjøretid: $\Theta(n^3)$.