UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	**INF 3130/4130: *Algoritmer: Design og effektivitet***
Eksamensdag:	**Torsdag 15. desember 2005**
Tid for eksamen:	**Kl. 14.30 til 17.30**
Oppgavesettet er på 4 sider	
Vedlegg:	**Ingen**
Tillatte hjelpemidler:	**Alle trykte og skrevne**

Kontroller at oppgavesettet er komplett
før du begynner å besvare spørsmålene.

Les oppgavene nøye, og lykke til!

Oppgave 1 (ca 14%)

Hvilke av språkene under er avgjørbare? Gi korte bevis.

a) $L_1 = \{010\}$ (L_1 modellerer problemet: Aksepter bare hvis input-strengen er ’010’)

b) $L_2 = \{M \mid M$ er en Turingmaskin som avgjør $L_1\}$

c) $L_3 = \{M \mid M$ er en Turingmaskin som avgjør $L_2\}$

Oppgave 2 (ca 14%)

Svar JA eller NEI og gi en kort begrunnelse.

a) Optimaliserings-versjonen av noen NP-komplette problemer har effektive approksimasjonsalgoritmer.

b) Det finnes NP-komplette problemer der optimaliserings-versjonen *ikke* har effektive approksimasjonsalgoritmer med mindre P er lik NP.

c) En parallell-maskin med en million prosessorer kan løse *Hamiltonbarhet* (”Hamiltonicity”, for urettede grafer) i polynomisk tid.

Oppgave 3 (ca 12%)

Bruk dynamisk programmering til å finne korteste editerings-avstand (”edit distance”) mellom

”abaka” og ”bakea”

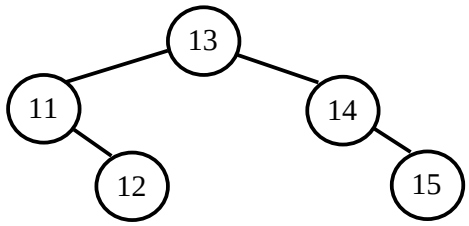
Tegn opp hele tabellen (med utfylling) som du bruker til å finne svaret.

Oppgave 4 (ca 14%)

Vi skal se på det å finne optimale søketrær i det tilfellet at sannsynligheten for å spørre etter andre verdier enn de som er i treet, er null. Vi har gitt alle p-verdiene (altså, frekvensene for forespørsler av de forskjellige verdiene), og alle q-verdier er altså 0. Vi kunne da forsøke oss med en grådighets-orientert algoritme for å bygge treet: Vi sorterer nodne på søke-verdien og bygger søke-treet fra roten og nedover slik: Som rot velger vi den med største p-verdi, som røtter for venstre og høyre subtrær velger vi de med størst p-verdi på hhv. venstre og høyre siden av roten, osv. Vi antar for enkelhets-skyld at alle p-verdiene er forskjellige.

Eksempel: Søke-verdiene vi vil sette inn i treet er 11, 12, 13, 14, 15, og de tilhørende p-verdier som angitt i tabellen under. Algoritmen over vil da først velge verdien 13 (med p=8, som er størst) som hovedroten, og for venstre og høyre subtre vil den velge hhv. rotverdiene 11 (med p= 5, størst på venstresiden) og 14 (med p=6, størst på høyresiden). Treet vil derved bli seende slik ut:

Verdier det skal søkes etter:	11	12	13	14	15
Tilhørende p-verdi (frekvens):	5	4	8	6	3



a) Vis at om det bare er to noder/verdier så vil denne algoritmen gi det optimale treet

b) Vis ved et eksempel at om vi har tre noder/verdier så vil algoritmen *ikke* generelt gi riktig svar.

Oppgave 5 (ca 18%)

Under kommer vi med noen påstander omkring trær og prioritetskøer. Angi for hver av dem om påstanden er riktig, og gi en kort begrunnelse.

1. Binomialtreet B_k kan lages fra treet B_{k-1} ved å legge ett nytt barn til hver node.

2. Dersom vi implementerer prioritetskøer ved AVL-trær (med prioriteten som innsettingsnøkkel), så vil elementer med samme prioritet alltid komme ut (ved kall på deleteMin) i den rekkefølge de er satt inn. (For at det skal være håp om at dette går bra, antar vi at når et element skal settes

inn i treet og treffer på en node med samme prioritet, så går man videre i *høyre* subtre for å finne plass til det).

3. Desom vi implementerer prioritetskøer som "leftiest heaps", så vil elementer med samme prioritet alltid komme ut (ved kall på deleteMin) i den rekkefølge de er satt inn.

4. Høyden av det høyeste treet i en Fibonacci-heap med n elementer ligger i intervallet fra og med $\lfloor \log_2 n/2 \rfloor + 1$ til og med $\lfloor \log_2 n \rfloor$.

5. Antall barn en node kan ha i en Fibonacci-heap på n elementer ligger i intervallet fra og med 0 til og med $\lfloor \log_2 n \rfloor$.

Oppgave 6 (ca 14%)

Vi har et rutenett, der rutene er nummerert "nedover" med i = 1 til m, og "bortover" med j = 1 til n. Til hver rute er det knyttet en heltallig kost K(i,j) > 0. Vi ønsker å lage en A*-algoritme som får oppgitt to ruter: r1 = (i1,j1) og r2 = (i2, j2) (samt m, n og alle K(i,j)), og som finner "beste" vei (innenfor det gitte rutenettet) fra r1 til r2. Vi vil i den forbindelse se på mulige heuristikk-funksjoner.

En vei består av enkeltskritt, som går fra en rute til en av de 8 naborutene, diagonalt, horisontalt eller vertikalt. Kosten av en vei er summen av kosten av de rutene den er innom, der r2, men ikke r1 regnes med, dog slik at om r2 = r1, så er kosten 0. (Vi kan mer presist si at K(i, j) er det det koster å gå *inn* i ruten (i,j)). Beste vei er den som har minst kost. Under er angitt et rutenett med m=5 og n=8, og med en vei fra r1 = (4, 3) til r2 = (2, 7). K-verdiene er gitt for rutene langs veien, og veien har altså kost = 32.

	j = 1	2	3	4	5	6	7	8
i = 1				2	2			
i = 2			4			7	8	
i = 3		9						
i = 4			3					
i = 5								

a) Under finner du to forslag til to heuristikker for et A*-søk etter beste vei. Vi antar at vi står ved rute r = (i, j), og at målet er r2 = (i2, j2). Angi for hvert forslag om det gir en monoton heuristikk-funksjon, og vis kort hvorfor eller hvorfor ikke.

Forslag 1: $|i2-i| + |j2-j|$.

Forslag 2: $\text{minK} * (\min(|i2-i|, |j2-j|) + | |i2-i| - |j2-j| |)$.
Her er minK lik kosten av den ruten med minst kost på hele brettet, og "min(a,b)" er den minimale verdien av a og b, på vanlig måte.

b) (Vent gjerne med denne til slutt) Angi en heuristikk-funksjon som er monoton, og som, om noen av de angitt i del a) skulle være monotone, er bedre enn denne/disse. Forklar kort.

Oppgave 7 (ca 14%)

Algoritmen skissert under er ment å finne artikulasjonspunkter i en urettet graf (skal angis ved at "artPoint" settes til true hvis og bare hvis det er et artikulasjonspunkt). Vi antar at grafen er ferdig satt opp med node-objekter av klassen under, men går ikke i detalj på hvordan kantene er representert. Grafen vil være sammenhengende, og ha minst to noder. Vi antar at "Gen" er en klasse med en statisk metode "int nextInt( )" som første gang den kalles leverer 1, neste gang 2, osv. Vi regner også at variabelen "preNum" er 0 i alle noder når algoritmen starter. Oppstarten skjer ved at "findArtPoints(from)" kalles utenfra for en eller annen node i grafen, og da med "from == null".

```
1  class Node {
2      int preNum;
3      int minSeen;
4      boolean artPoint = false;
5
6      void findArtPoints(Node from) {              // Det første kallet gjøres med "from == null"
7          int  numCalls = 0;                       // Bare viktig for startnoden (rotnoden)
8
9          preNum = Gen.nextInt( );  minSeen = preNum;
10
11         for Node nb  = <hver av nabonodene til denne noden> do {
12             if  nb.preNum == 0 then {            // Usett node
13                 numCalls++;
14                 nb.findArtPoint(this);
15                 if nb.minSeen >= preNum then { artPoint = true; }
16                 minSeen = min(minSeen, nb.minSeen);
17             } else {
18                 minSeen = min(minSeen, nb.preNum);
19             }
20         }
21         if from == null then {                   // Er i startnoden (rotnoden)
22             if  numCalls > 1 then { artPoint = true; }
23         }
24     }
25 }
```

a) Algoritmen over inneholder to feil: En test mangler inne i for-løkka, og det må gjøres et lite tillegg et sted utenfor for-løkka. Angi feilene, og forklar kort hva som ville gått galt.

b) *(Du får uttelling for å løse denne selv om du ikke har løst pkt a) over.)* Vi definerer en "bro" i en urettet graf som en kant som, hvis den fjernes, vil dele den sammenhengende komponenten den er en del av i to. F.eks., i en graf med to noder og en kant mellom dem, vil kanten være en bro. Oppgaven er å angi forandringer på programmet over slik at det i stedet finner alle broer i grafen. Hver gang en bro er funnet skal programmet gjøre et kall "broFunnet(n1,n2)", hvor n1 og n2 er ende-nodene av bro-kanten. Gi en begrunnelse for at du mener algoritmen du foreslår virker. Det behøver ikke være noe fullverdig bevis, men skal i det minste skissere intuisjonen som ligger under.

(Slutt på oppgavesettet)