UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Prøve-eksamen i:	**INF 3130/4130: *Algoritmer: Design og effektivitet***
Eksamensdag:	**Gjennomgås 30. november, kl 14.15**
Tid for eksamen:	**3 timer**
Oppgavesettet er på:	**4 sider**
Vedlegg:	**Ingen**
Tillatte hjelpemidler:	**Alle trykte og skrevne**

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

Alle syv oppgavene har lik vekt, altså omtrent 14%.
Les oppgavene nøye, og lykke til!

Oppgave 1

L = {(M_1, M_2) | M_1 og M_2 er Turingmaskiner som er ekvivalente, dvs. gir samme output for samme input}

a) Bevis at L er uavgjørbart.

b) Kommenter betydning av øverste resultatet i følgende sammenheng: Det ville vare fint hvis vi kunne teste programmer ved å sammenligne dem automatisk (ved hjelp av et generelt testeprogram) med et kjent korrekt program for samme funksjon. Er dette mulig? Hvorfor?

Oppgave 2

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

Vi skal løse følgende problem: Vi får som input to strenger: S som består bare av bokstaver, og T som også kan inneholde '*'-er. Spørsmålet er om strengene kan gjøres like, om vi får lov til å erstatte hver '*' i T med ingeneting (en tom streng) eller én selvvalgt bokstav. For S="abbcf" og T="a*bc*f", kan dette gjøres slik: T'="a(b)bc()f". For strengene "acb" og "a*c" er det derimot umulig. Beskriv en algoritme som avgjør om man kan oppnå likhet, og som bygger på dynamisk programmering. Skisser et program som implementerer algoritmen.

Hint: Forsøk med et skjema med de to strengene langs hver sin akse, og med boolske verdier. Skill mellom den generelle reglen, og initialiseringen for de ruter der minst en av strengene er tomme.

Oppgave 4

Vi skal søke i strengen "bananananobana" etter patternet "nano", på litt ulike måter.

a) Beregn NEXT[0:3]-arrayet, som Knuth-Morris-Pratt-algoritmen bruker.

b) Beregn SHIFT[a:z]-arrayet som den forenklede Boyer-Moore-alagoritmen (kalt Horspool-algoritmen på forelesningen) bruker; altså algortimen som beregner skift bare ut ifra den siste bokstaven i den aktuelle delen av strengen.

c) I denne deloppgaven skal vi søke i et to-dimensjonalt array, A, etter et to-dimensjonalt pattern, P.

A =

b	a	n	a	n	a	n	o	b	a	n	a
n	a	n	a	n	o	b	a	n	a	b	a
n	a	n	o	b	a	n	a	b	a	n	a
…											

P =

n	a
n	o

Forklar, kort, gjerne med pesudo-kode, men på en slik måte at ideen din kommer klart fram, hvordan man kan kan tilpasse eller sette sammen de søkealgoritmene vi har studert i kurset, på en slik måte at vi kan klare to-dimensjonale søk.

Oppgave 5

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.
   (Hint: Betegnelsen *binomialtre* refererer til strukturen i treet, ikke nødvendigvis måten de bygges opp på.)

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

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, og slik at om r1 og r2 er samme rute (og veien bare består av denne), 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 forklar kort hvorfor eller hvorfor ikke.

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

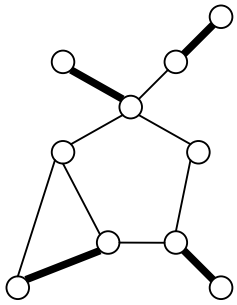
Forslag 2: $\text{minK} * (\min(|i2-i|, |j2-j|) + | |i2-i| - |j2-j| |)$.
Her er minK lik kosten av den ruten med minst kost på hele brettet, og "min(a,b)" er den minimale verdien av tallene a og b, på vanlig måte.

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 Matching

Gitt grafen under, med den angitte matching. Vis hvordan algoritmen for maksimal matching i generelle grafer vil avgjøre om dette er en maksimal matching eller om den kan økes. Gi et eksempel på hvordan nodene kan være merket når algoritmen tar avgjørelsen. Merk blomster som oppstår med en passelig boks/ring rundt dem, og bruk stipling på kanter som hverken er med i noe tre eller i matchingen. Bruk ellers F(irkant)-noder, P(rikk)-noder og S(irkel)-noder, som i kompendiet. Beskriv hva som er avgjørende for at algoritmen tar den avgjørelsen den tar.



(Slutt på oppgavesettet)