UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	**INF 3130/4130: *Algoritmer: Design og effektivitet***
Eksamensdag:	**Fredag 14. desember 2007**
Tid for eksamen:	**Kl. 09.00 til 12.00**
Oppgavesettet er på:	**4 sider**
Vedlegg:	**Ingen**
Tillatte hjelpemidler:	**Alle trykte og skrevne**

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

Les oppgavene nøye, og lykke til!

Oppgave 1 Strengsøk (14%)

a) Tegn et suffix-tre for strengen `GAACAACT`.

b) Beregn shift-verdier for strengen `GAACAACT` slik det gjøres i den forenklede Boyer-Moore-algoritmen (Horspool-algoritmen). Ta utgangspunkt i at alfabetet består av de vanlige norske, store bokstavene (A—Å).

Oppgave 2 A-søk og heuristikk-funksjoner (16%)*

a) Vi er gitt et enkelt brikkespill bestående av n hvite (**H**) og n sorte (**S**) brikker, samt en åpen rute, som vist på figuren under.

1	2	…	n	$n+1$	$n+2$	$n+3$	…	$2n+1$
S	**S**	**…**	**S**		**H**	**H**	**…**	**H**

Målet er å flytte alle de hvite brikkene over på venstre side av brettet og de sorte på høyre side. Hvor den åpne ruten havner er uinteressant, og vi bryr oss heller ikke om den innbyrdes rekkefølgen av de hvite og de sorte brikkene. I slutt-tilstanden skal det altså ikke være noen sorte brikker til venstre for (med lavere indeks enn) noen hvit brikke. Vi kan gjøre to slags flytt:

- Flytte en nabo-brikke av den åpne ruten inn i den åpne ruten. Dette har kostnad 1.
- Hoppe over nøyaktig én brikke, inn i den åpne ruten. Dette har kostnad 2.

Vurder om følgende forslag til heuristikker, angitt for en generell spill-situasjon, er monotone eller ikke:

1. Antall enkelt-flytt (av kostnad 1) som trengs for å flytte alle H-brikkene helt til venstre (om det ikke var noen S-brikker) pluss tilsvarende antall enkeltflytt det tar å flytte alle S brikker helt til høyre.

2. For hver S-brikke teller vi opp antall H-brikker til høyre for denne. Så summerer vi opp disse verdiene for alle S-brikkene.

b) Når vi arbeider med A*-søk og heuristikk-funksjoner, hender det ofte at vi kan bruke forenklede versjoner av vårt opprinnelige problem til å lage heuristikk-funksjonene våre. Professor Max har på denne måten laget tre ulike heuristikk-funksjoner for et problem han arbeider med: h_1, h_2 og h_3. Han har verifisert at alle de tre funksjonene er monotone, men han har ikke klart å avgjøre hvilken han skal bruke. For noen noder ligger heuristikk h_1 nærmest den faktiske kostnaden, for andre h_2 eller h_3. Professorens datter foreslår at han benytter funksjonen

$$h(x) = \max\{h_1(x), h_2(x), h_3(x)\} .$$

Kan han det? Forklar hvorfor/hvorfor ikke.

Oppgave 3 Dynamisk programmering (13%)

En lang tømmerstokk av lengde l skal kappes i mindre deler på et sagbruk. På grunn av kvist, stokkens tykkelse, hva delene skal brukes til ol., er posisjonene hvor stokken skal kappes bestemt på forhånd. Disse er: p_1, p_2, ..., p_{n-1}, målt fra den venstre enden av stokken. I tillegg lar vi p_0=0 være den venstre enden av stokken, og p_n=l den høyre. Etter å ha delt den opprinnelige stokken én gang har vi to nye stokker, etter å ha delt den to ganger har vi tre, osv.

På sagbruket går stokkene på transportbånd, slik at det tar tid t å kappe en stokk som er t lang uansett hvor den kappes. Stokken kan bare kappes én gang per gjennomkjøring, de to nye delene må eventuelt prosesseres på nytt, hver for seg, om de skal kappes ytterligere. Stokker som ikke skal kappes mer kjøres ikke gjennom saga, disse tar altså tid 0.

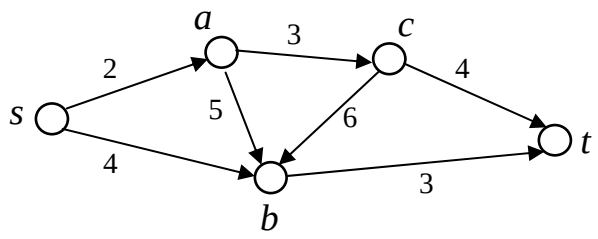
Eksempel: En 10 meter lang stokk skal kappes ved p_1=2 meter, p_2=5 meter, p_3=8 meter. Kapper vi først ved p_3, så ved p_2, og så ved p_1, vil dette ta tid 10+8+5=23. Kapper vi først ved p_2, og deretter p_1 og p_3, vil det ta tid 10+5+5=20, altså noe raskere.

Oppgaven er: Definér en formel (en rekurrensrelasjon) som kan brukes i dynamisk programmering for å finne raskeste måte å kappe den opprinnelige stokken på, og vis hva initialbetingelsene vil være. Skissér gjerne et program, om du syns det er enklere enn å definere en rent matematisk formel.

Oppgave 4 Flyt i nettverk (21%)

Vi skal arbeide med flyt i (rettede) nettverk av den typen som er beskrevet i kap 14.2. Tallene på kantene, som der brukes som kapasiteter (øvre grense for flyt) skal vi her tolke som *nedre* grense for flyt i den kanten, og for kanten *K* skal vi kalle denne *minflyt(K)*. Dette kan f.eks. være aktuelt for rør-systemer i bakken, der det må være en viss flyt så de ikke fryser. Vi ønsker imidlertid at den totale flyten fra *s* til *t* skal være så lav som mulig. En flyt som ikke er mindre enn *minflyt(K)* for noen kant *K* sies å være *lovlig*.

Verdien *minflyt(K)* er for alle *K* et heltall som er null eller positivt. Vi har ingen øvre grense for hvor stor flyten kan være i en kant. De nettverkene vi arbeider med her har ikke rettede løkker, og alle kanter ligger på en rettet vei fra *s* til *t*. Eksempel på nettverk, der *minflyt*-verdiene er satt på kantene:



a) Vi vil lage en algoritme som finner minimal flyt, slik den er angitt over. Algoritmen får ved starten oppgitt en lovlig flyt fra *s* til *t* (som altså tilfredsstiller minflyt-kravet på hver kant, men kan være unødvendig stor). Forklar hvordan vi kan forandre stegene i FordFulkerson-algoritmen så vi kan arbeide oss mot en mindre og mindre, men lovlig, flyt. Det er helt greit å bruke ord til å forklare forandringene, men en programskisse er også OK.

b) Vi vil se på situasjonen når algoritmen fra oppgave a) stopper. Vi får da en situasjon med min-flyt/maks-kutt tilsvarende max-flyt/min-kutt for FordFulkerson. Hvordan vil et maks-kutt generelt se ut, og hvordan kan man finne et slikt når algoritmen stopper?

c) Som angitt over trenger algoritmen fra oppgave a) en lovlig startflyt. Forklar hvordan man kan finne en slik. Du behøver ikke legge vekt på at denne startflyten skal være spesielt lav.

Oppgave 5 Uavgjørbarhet (20%)

Hvilke av følgende språk er avgjørbare? Skissér kort et bevis for hvert av svarene.

a) $L_1 = \{M, x$: Med input x stopper maskin M etter ikke mer enn $|x|$ skritt $\}$

b) $L_2 = \{M$: Maskin M stopper for enhver input med lengde høyst 10 $\}$

c) $L_3 = \{M, x$: Maskin M stopper ikke med input $x\}$

d) $L_4 = \{M$: Maskin M flytter aldri lesehodet til venstre når den startes med blank input $\}$

e) $L_5 = \{M$: Maskin M aksepterer eksakt en streng $\}$

Oppgave 6 Kompleksitet (16%)

Med en vellykket kombinasjon av nanoteknologi, genemanipulering, lim og alkohol, klarte Gustav Hamilton ved Thinking Machines International å skape en spesialisert maskin som løser Hamiltonbarhet (Hamiltonicity) i polynomisk tid. Vurder og argumenter for/imot følgende som mulige konsekvenser av Gustavs oppfinnelse:

a) Tilfredstillbarhet (SAT) kan løses i polynomisk tid.

b) Ikke-tilfredstillbarhet (NON-SAT) kan løses i polynomisk tid.

c) Alle umedgjørlige (intractable) problemer kan løses i polynomisk tid.

d) Optimaliseringsversjonen av Handelsreisendesproblem (TSP-optimization) kan løses i polynomisk tid.

(Slutt på oppgavesettet)