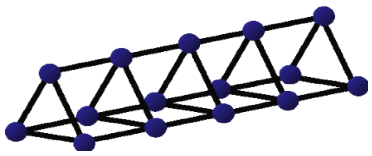
Avsluttende eksamen i
TDT4125 Algoritmekonstruksjon, videregående kurs
Partielt løsningsforslag

Eksamensdato	19. mai 2008
Eksamenstid	1500–1900
Sensurdato	9. juni
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.

Oppgave 1

La grafen G ha en struktur som vist i Figur 1. Vi kan se for oss at nodene er sammenføyninger som skal inspiseres av en robot. Kantene kan tilsvare bærende stål/tre-bjelker. Alle de 27 inntegnede kantene har lengde 1.



Figur 1

Vi skal finne en kortest mulig rundtur (Travelling Sales Person/TSP-tur) gjennom alle de n=15 nodene, uten nødvendigvis kun å følge de inntegnede (fysiske) kantene? (Anta at avstandene mellom nodene måles i rett linje.)

a. Hva er lengden til den korteste rundturen i grafen G? Gi et bevis for at svaret er korrekt.

Svar: Korteste tur = 15. Det er ikke mulig å finne et kortere spenntre enn et med lengde 14, derfor finnes heller ingen kortere rundtur enn en med lengde 15 idet minsteavstanden mellom to vilkårlige noder er 1.

b. Hva er den korteste rundturen APPROX-TSP-TOUR vil kunne finne i grafen G? Vis også at denne algoritmen kan finne en tur med lengde $11 + 4 \cdot \sqrt{2} \approx 16.7$.

Svar: Spenntre som inkluderer én langside og 10 «avstikkere». Du får da 5 «snarveier» langs eksisterende kanter (lengde 1) og 4 diagonaler (lengde $\sqrt{2}$).

c. Hva er den korteste rundturen Christofides' algoritme vil kunne finne i grafen G? Vis også at denne algoritmen kan finne en tur med lengde $14 + \sqrt{17} \approx 18.1$.

Svar: Den vil finne kunne finne en 15-tur, men dersom du er uheldig med MST-valget vil tur-lengden i verste fall bli $14 + \sqrt{(1 + 4^2)}$ – altså alle tre langsider (12), to «koblingskanter» + lang diagonal. Det minimale spenntreet er da denne turen *uten* diagonalen.

d. Konstruer et eksempel, parametrisert på n (og bare n), gjerne beslektet med Figur 1, der Christofides' algoritme vil kunne finne en rundtur som asymptotisk blir 50% lengre enn den korteste rundturen.

Svar: Fjern én av noderekkene i Figur 1. Vi vil da kunne ende opp med en «sikk-sakk»-løsning + en lang diagonal. En optimal løsning har lengde n, mens sikk-sakken har lengde n–1 og diagonalen har lengde som nærmer seg $n/2$ (for store n).

Oppgave 2

Anta at du har to avstandsfunksjoner d og f, der $f(x, y) \geq d(x, y)$ for alle x, y. Anta at d er billigere å beregne enn f.

a. Hvordan kan d brukes til å redusere kostnaden ved et *range query* med f?

Svar: Hvis $d(q, o) > r$ kan o forkastes.

Anta at situasjonen er omvendt: At f er billigere å beregne enn d.

b. Hvordan kan f brukes til å redusere kostnaden ved et *range query* med d?

Svar: Hvis $f(q, o) \leq r$ kan o returneres.

Anta at du setter søkeradien slik at kun et lite mindretall av objektene i datasettet faller innenfor. Anta også at d og f approksimerer hverandre godt.

c. Tror du situasjonen i **a** eller **b** vil føre til størst besparelse? Begrunn svaret.

Svar: For et søk der et mindretall av objekter ligger innenfor søkeradien (også for den approksimerte avstanden) er det større sannsynlighet for at tilfellet i **a** vil slå til.

d. Gjelder svarene i **a** og **b** også for *k nearest neighbor*-søk? Begrunn svaret.

Svar: Nei. Det at en avstand dominerer en annen betyr ikke at den bevarer avstandsrekkefølger.

Det er allmennt akseptert at det er lettere å indeksere avstander hvis avstandsfordeling har høyt standardavvik.

e. Gi et eksempel på en anvendelse der du tror dette standardavviket vil være høyt. Begrunn svaret.

Svar: For eksempel billedsøk. Standardavviket vil være høyt fordi bildene grupperer seg i «klynger», der avstandene er små innad men store utad.

Oppgave 3

Du har et sett med n jobber som skal utføres. Hver jobb i har en *release time*, $r(i)$, og en deadline, $d(i)$. Hver jobb tar 1 time å utføre. En jobb kan ikke utføres før sin *release time*, og den må utføres før sin deadline. To jobber kan ikke overlappe i tid.

a. Beskriv kort en algoritme som finner en lovlig schedule for jobbene, om mulig. Vis at algoritmen er korrekt og oppgi kjøretid i Θ-notasjon.

Svar: Her kan en grådig *earliest deadline first*-algoritme (EDF) brukes (der kjøretiden domineres av sorteringen). Poenget er å vise at dette blir korrekt, for eksempel ved å bruke et såkalt *exchange argument* (forelest som generell bevismetode) og induksjon:

Anta at vi har en optimal, lovlig schedule O. La G være den grådige EDF-schedulen. Anta at de er enige om de $k - 1$ første jobbene. La jobb nummer k være x i G og y i O. Lag nå O' ved å bytte om på x og y i O. Med andre ord: x ender opp tidligere i O' enn i O, og y ender opp senere.

Det er nå klart at G og O' er enige om (minst) de k første jobbene, men er det sikkert at O' er lovlig? Siden O utfører alle jobber før deadline, og x nå har havnet tidligere, så utføres den fortsatt i tide. Siden G utfører jobber etter deres release date vil x også utføres etter sin release time i O'. Siden y nå utføres senere vil den også utføres etter sin release time.

Siden G kjører x før y vet vi at $d(x) \leq d(y)$. Når x og y bytter plass er dette med andre ord også trygt for y med tanke på deadlinen.

Oppgaven er hentet fra eksempel på side 4 her:
http://www.cs.pitt.edu/~kirk/cs1510/notes/greedynotes.pdf

Oppgave 4

a. Beskriv kort hvordan crossover fungerer i genetisk programmering, og hva hensikten med det er.

b. Gi et eksempel på crossover (før og etter) for to trær som representerer regulære uttrykk.

Oppgave 5

a. Konstruer og tegn et felles suffikstre for de to strengene “genetisk” og “nett”.

b. Vis hvordan du kan bruke dette treet til å gjøre et substreng-søk etter strengen “net”.

Oppgave 6

Anta at du har oppgitt en sti som besøker alle nodene i en komplett, vektet graf G nøyaktig én gang (en Hamilton-sti). Nodene skal partisjoneres i to delmengdene. La A og B være stiene i G som består av nodene i hver av disse delmengdene. Du ønsker å minimere $w(A) + w(B)$, altså summen av lengdene til de to stiene.

b. Beskriv en algoritme som finner en slik optimal partisjonering. Oppgi kjøretiden i Θ-notasjon.

Svar: La $s[i, j]$ være den optimale lengden dersom node i tilhører partisjon j (0 eller 1) og la $n[i, j]$ være den tilhørende forrige noden i stien til partisjon j. La $w[i, j]$ være kantvekten mellom node i og j. Løs følgende rekurrens med dynamisk programmering:

$s[i, j] = \min(s[i–1, j] + w[i–1, i], s[i–1, 1–j] + w[n[i–1, 1–j], i])$

$n[i, j]$ settes ut fra hvilket ledd som er minimalt.

Kjøretiden blir $\Theta(n)$.

Anta at det finnes en algoritme som i polynomisk tid kan avgjøre hvorvidt en gitt streng befinner seg i mengden L. La L^* være mengden av strenger som er konkateneringer av strenger fra L.

c. Vis at det da også finnes en algoritme som i polynomisk tid kan avgjøre hvorvidt en gitt streng befinner seg i mengden L^*.

Svar: For alle mulige prefiks av strengen, bruk den opprinnelige algoritmen på prefikset og løs problemet rekursivt for suffikset.