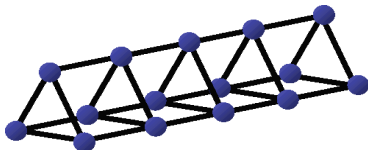
Avsluttende eksamen i
TDT4125 Algoritmekonstruksjon, videregående kurs

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. Ved sensur vektlegges oppgavene som angitt. For hver oppgave teller alle deloppgaver like mye.

Oppgave 1 (20%)

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.

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 $11 + 4 \cdot \sqrt{2} \approx 16.7$.

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 $14 + \sqrt{17} \approx 18.1$.

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 (optimale) rundturen.

Oppgave 2 (20%)

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?

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?

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.

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

Det er allment akseptert at det er lettere å indeksere avstander hvis avstands-fordelingen har høyt standardavvik.

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

Oppgave 3 (15%)

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

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

Oppgave 4 (10%)

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 (10%)

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 (25%)

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 delmengder. 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.

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

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.

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