Kontinuasjonseksamen i fag
SIF8010 Algoritmer og Datastrukturer

Mandag 31. juli 2000, kl. 09:00–15:00

Løsningsforslag

Oppgave 1

a) Ja.

$$f(n) = 5n^2 - 64n + 256 \stackrel{?}{=} \Omega(n^2)$$
$$\Uparrow$$
$$5n^2 - 64n + 256 \geq n^2$$
$$\Updownarrow$$
$$4(n-8)^2 \geq 0$$

Altså er $cf(n) \geq n^2$ for $c = 1$ og $n \geq n_0 \geq 1$.

b) Ja. Følger av definisjonen av $O(\cdot)$.

c) Ja. Følger av definisjonen av $O(\cdot)$.

d) Nei. Vi kan ikke overse kostnaden av S_1.

e) Nei. Siden $S_1 = O(f_1(n))$, så vet vi at $f_1(n)$ gir en øvre grense for kjøretiden til S_1, men det er alt vi vet om $f_1(n)$. Vi kan ikke bruke den som noen nedre grense. Det samme gjelder $f_2(n)$ og $f_3(n)$.

f) Nei. Se e).

g) Nei. Se e).

h) Nei. Se e).

i) Nei. Se e).

j) Nei. Vi må blant annet ta hensyn til kjøretiden til S_1, og siden vi ikke har noen nedre grense for noen av kjøretidene kan vi ikke bruke Θ-notasjon.

Oppgave 2

a) Merk at $T(n) = cf(n) + Q(n)$, der $Q(n)$ domineres av $cf(n)$. (Følger direkte av at $T(n) = O(f(n))$.)

Beregn $r(n) = T(n)/f(n)$ for økende n-verdier.

Analyse:

$$\underset{n\to\infty}{r(n)} \begin{cases} \textit{divergerer} & : \ f(n) \textit{ trolig for liten; finn større } f(n). \\ \textit{konvergerer mot } 0 & : \ f(n) \textit{ trolig for stor.} \\ \textit{konvergerer mot } c > 0 & : \ f(n) \textit{ trolig korrekt.} \end{cases}$$

Hvis $f(n)$ viser seg å være "for stor" er det tre mulige grunner:

- $f(n)$ er ikke stram nok. Verste-tilfelle-analysen kan likevel være nyttig.
- Analysen var for verste-tilfelle, men datasettene fanget ikke opp disse.
- Analysen av A er galt utført.

Oppgave 3

a)

- Må anta/finne L og H *åååå* $\in [L, H]$
- Kan bruke diverse RADIX-SORT-varianter med variabelt "boks-antall.' Avhengig av boks-antallet (minst lik 10).
- Objektene "drar med seg" en referanse (peker) til "de øvrige attributtene i hvert personobjekt."

b) $O(k, N)$, der k er "antall runder" (høyst lik 8) av RADIX-SORT.

Oppgave 4

a) Nøkkelobservasjon: Siden det finnes kun én vei fra en node s til en node v så vil enhver metode som finner en sti fra s til v finne den korteste veien. Vi kan altså bruke bredde-først–søk eller dybde-først–søk (som starter i s) for å finne alle de korteste veiene.

For å få tak i sti-vektene i tillegg til selve stiene må algoritmen ta vare på estimater for distansen tilsvarende som i Dijkstras algoritme. Negative sykler finnes enten ved å bruke testen fra slutten av BELLMAN-FORD, eller ved å prøve å forbedre avstanden til en node man allerede har besøkt. Siden man ikke kan ha to "forward"-kanter til samme node i en ensporet graf, må denne forbedringen ha kommet istand vha. en løkke.

b) BFS og DFS har kjøretid $O(V + E)$ (sykelsjekkingen påvirker ikke dette). Når søket foretas i en ensporet graf vil vi kun besøke $O(V)$ kanter:

– Treet ut fra s har $O(V)$ kanter (egenskap ved trær);

– Vi har ingen "forover-kanter" eller "kryss-kanter", siden vi da ville ha to stier fra s til v;

– Vi har maksimalt 1 "bakover-kant" fra hver node v ($O(V)$ totalt).

(Bevis for det siste punktet: Anta at vi hadde to tilbakekanter fra en node w, hhv. til nodene u og v. Disse ligger da på stien fra s til v, og vi antar at u ligger først, sånn at det finnes en sti fra u til v. Da vil det finnes to stier fra w til v: Én via kanten (w, v), og én via kanten (w, u) og så videre via stien fra u til v. Dette bryter med definisjonen for en ensporet graf, og er derfor umulig.)

Total kjøretid blir altså $O(V)$.

Oppgave 5

a) Vi må gå igjennom alle kantene. Siden grafen er komplett, blir kjøretiden

$$\Theta\big(\sum_{i=1}^{n} n\big) = \Theta(n \cdot \frac{n}{2}) = \Theta(n^2),$$

der $n = |V|$.

b) Punkt 1 impliserer at vi har en asyklisk graf. Punkt 2 impliserer at grafen er sammenhengende. Disse to punktene innebærer altså at datastrukturen skal være et *tre*, eller rettere sagt, et *spenntre* for den opprinnelige grafen. Det siste punktet (3) innebærer at det er snakk om et minimalt spenntre.

For å finne det bruker vi en standard-algoritme for å finne spenntrær, enten MST-PRIM eller MST-KRUSKAL.

Siden $|E| = \Theta(|V|^2)$ vil MST-KRUSKAL ha kjøretiden

$$\begin{aligned} O(|E| \lg |E|) &= O(|V|^2 \lg |V|^2) \\ &= O(|V|^2 \lg |V|). \end{aligned}$$

MST-PRIM har kjøretiden

$$\begin{aligned} O(|V| \lg |V| + |E| \lg |V|) &= O(|V| \lg |V| + |V|^2 \lg |V|) \\ &= O(|V|^2 \lg |V|). \end{aligned}$$

Kjøretiden blir altså $O(|V|^2 \lg |V|)$.

c) Siden vi nå arbeider med et tre, så har vi at

$$|E| = |V| - 1 = \Theta(|V|).$$

Dermed blir kjøretiden $\Theta(|V|)$.

d) Den totale kjøretiden for metoden i c) blir

$$O(|V|^2 \lg |V|) + \Theta(|V|) = O(|V|^2 \lg |V|).$$

Siden den opprinnelige kjøretiden var $O(|V|^2)$ er dette ingen forbedring.

e) Hvis vi må se på alle nabokantene til hver kant, vil kjøretiden for en slik "klippe"-prosedyre i et generelt tilfelle være $O(|E|^2)$.

Hvis prosedyren utføres i den opprinnelige strukturen, blir kjøretiden

$$O(|E|^2) = O(|V|^4).$$

I spenn-treet vil kjøretiden derimot bli

$$O(|E|^2) = O(|V|^2).$$

Den totale kjøretiden *uten* omformingen blir altså $O(|V|^4)$, mens kjøretiden *med* omforming blir

$$O(|V|^2 \lg |V|) + O(|V|^2) = O(|V|^2 \lg |V|).$$

Vi får altså en besparelse i total kjøretid ved bruk av omformingen i dette tilfellet.

(Det kan selvfølgelig diskuteres i hvilken grad resultatet vårt blir like korrekt som det ville bli uten omformingen, men det er ikke interessant i denne sammenhengen.