

Midtsemestertest

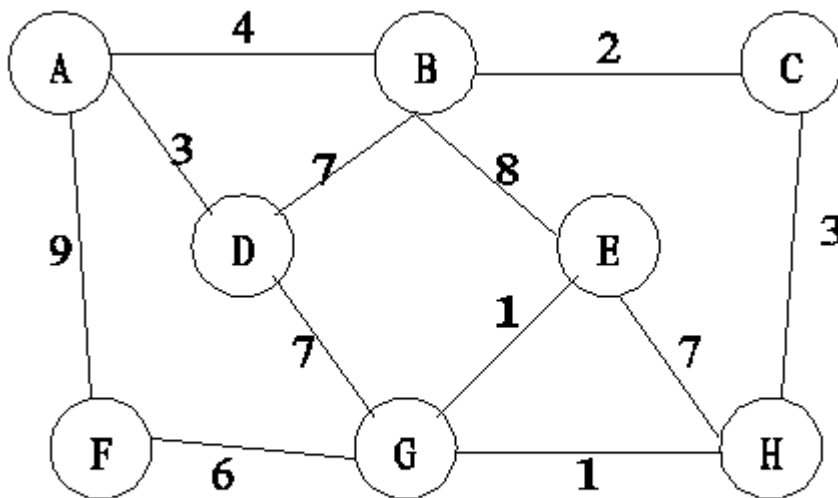
TDT4120 Algoritmer og Datastrukturer

Onsdag 1. oktober 2003

1. Algiator

- a) Hva betyr Stor-Theta notasjon, som i $\Theta(n)$? (5 %)
- b) Hva er worst case kjøretiden til quicksort? (5 %)
- c) Hva menes med at en sorteringsalgoritme er *stabil*? (5 %)
- d) Har et binærtre med høyde n alltid 2^n løvnoder? Hvorfor/hvorfor ikke? (5 %)

2. Spenntre



- a) Bruk en minimal spenntre-algoritme til å finn det minste spenntreet til denne grafen. (10 %)
- b) Anta at vi nå multipliserer alle vektene i denne grafen med $\log(n)$ hvor n er antall noder som finnes i grafen. Hvilke kanter utgjør det minimale spenntreet da? (5 %)

3. Towers of Hanoi

Vi skal nå se på et kjent spill som heter Towers of Hanoi. Mange av dere har sikkert vært borti det før. Her ser dere et bilde av hvordan spillet ser ut i virkeligheten:

Spillet går ut på å flytte alle ringene fra en pinne til en annen. Dette kan kanskje høres lett ut, men her kommer noen regler du må følge:



- Du kan bare flytte en ring av gangen.
- Hvis ring A er større enn ring B, kan ikke ring A ligge over ring B ved noe tidspunkt.

Det viser seg at man kan utforme en kort rekursiv metode som gir deg instruksjoner til alle enkeltstegene du må utføre for å løse oppgaven. Det er det vi skal gjøre nå.

- a) Her følger en kodesnutt i Java som skal løse Towers of Hanoi. Din oppgave er å (10 %) fylle inn linjene A og B slik at den virker. I denne metoden er `antall` antall ringer som skal flyttes, `fra` er hvilken pinne de skal flyttes fra, osv.

```
public static void flytt(int antall, int fra, int til, int via){
    if(antall>1){
        A
    }
    System.out.println("Flytt ring fra "+fra+" til "+til);
    if(antall>1){
        B
    }
}
```

- b) Sett opp rekurrensen for metoden du laget i a). La n betegne antall ringer du skal flytte. Du kan anta at det går i konstant tid å flytte en ring fra en pinne til en annen. (5 %)
- c) Løs rekurrensen i b) for å finne, i Θ -notasjon, hvor mange ganger du må flytte en enkelt ring for å løse Towers of Hanoi. (10 %)

4. Ulykkesnummeret

Trener Roger Mykoddé kjører et hardt opplegg på treningene, men hemmeligheten bak alle seirene i eliteserien er at ingen av fotballspillerne hans har ulykkesnummeret hans, 16, på ryggen. Ikke bare må han unngå å se nummeret for å konsentrere seg om seierstaktikk, men hans relativt avanserte matematiske kunnskaper (til fotballtrener å være) gjør at han heller ikke takler å se at x tall ved siden av hverandre summeres opp til å bli ulykkestallet hans. Laget står alltid oppstilt i samme rekkefølge (en rekkefølge han ikke kan påvirke).

Siden Rogers aversjoner mot tallet 16 foregår på et underbevisst plan, trenger han en algoritme for å komme problemet i forkjøpet, slik at han kan gjenkjenne en slik sum før en kamp.

Fetteren til Roger, som er oberst i sivilforsvaret med ansvar for tunge unødvendige oppgaver, har et tilsvarende problem. Han klarer ikke å se tallet 345678, verken alene eller som en sum av x tall etter hverandre. Han er veldig interessert i å få fatt i Rogers algoritme slik at han også kan bruke den, men det er viktig at kjøretiden ikke er for sakte.

- a) Finn en løsning på problemet gitt ovenfor. Gitt en rekke med positive tall og et ulykkestall, finn ut om rekken inneholder en subsekvens uten hull som summerer til å bli ulykkestallet. Skisser ideen din. (15 %)
- b) Lag pseudokode for ideen din. Du kan også benytte deg av Java-kode hvis du vil. (10 %)

Metodesignaturen er:

for pseudokode:

```
harUlykkesTall(A, ulykkestall)
```

for Java:

```
public static boolean harUlykkesTall(int[] rekke, int ulykkestall)
```

- c) Finn den asymptotiske kjøretiden til programmet ditt i Θ -notasjon. (2,5 %)
- d) Er algoritmen din optimal (ja/nei)? (2,5 %)

5. Masterteoremet

Løs følgende rekurrenser ved å bruke masterteoremet:

- a) $T(n) = 16T(n/2) + n^3 * \log(n)$ (5 %)
- b) $T(n) = 25 * T(n/5) + n^2$ (5 %)

6. Bonusoppgave

Vil du bli en virkelig Algiator? Da bør du benytte sjansen til å knekke Bonusoppgaven vår.

Du har gitt n punkter i x - y -planet, $P_1 : (x_1, y_1)$, $P_2 : (x_2, y_2)$, \dots , $P_n : (x_n, y_n)$. Du skal finne en beste vei fra P_1 til P_n , via et utvalg av de øvrige $(n-2)$ punktene, som er slik at den lengste avstanden mellom nabopunkter på veien er så kort som mulig. (Tenk deg at du skal hoppe fra stein til stein over en elv og at det lengste hoppet skal være så kort som mulig.)

- a) Skisser en så effektiv som mulig algoritme som løser dette beste-vei-problemet ved å bruke Dijkstras algoritme (uendret) som subrutine. (0 %)
- b) Finn en mer effektiv algoritme for å løse vårt beste-vei-problem ved å modifisere koden i Dijkstras algoritme. Vis modifikasjonene i detalj. (0 %)