UNIVERSITETET I TRONDHEIM
NORGES TEKNISKE HØGSKOLE
INSTITUTT FOR DATATEKNIKK OG TELEMATIKK
7034 Trondheim

Eksamen i 45011 Algoritmer og Datastrukturer

Torsdag 12. januar 1995, Kl. 0900-1300.

Løsningsforslag

Oppgave 1

Innsettingssortering og Boblesortering er utelukket p.g.a. kjøretid over 1s. Quicksort er raskest men har "worst case" ytelse som er $O(n^2)$. Quicksort vil dermed ikke kunne garantere at tabellen blir ferdig sortert på 1s. Heapsort bør foretrekkes da denne er er nest raskest og har "worst case" lik $O(n \cdot \log(n))$.

Oppgave 2

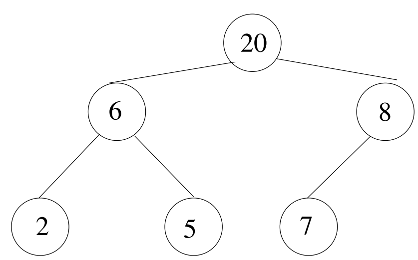
Innsettingssortering er raskere enn Quicksort på små vektorer. Det er derfor lurt å "bytte" til innsettingssortering når Quicksort har fått brutt ned vektorene til små delvektorer. Alternativ A og B er like effektive i form av *O*-notasjon. De to alternativene er nærmest identiske fordi Innsettingssortering i Alternativ B kun vil flytte elementer innen de små delvektorene som QuickSortB ikke ferdigsorterte. Alternativ B bør likevel foretrekkes da man her har langt færre funksjonskall .

Oppgave 3

3a) Det blir viktig å lage trær med lav dybde, mens det spiller mindre rolle om hver node inneholder mye data. Et generelt B-tre med mangle nøkler i administrasjonsnodene er da velegnet.

3b) Det er ikke like viktig at treet har lav dybde. Det viktigste er at den totale datamengden som må undersøkes er minst mulig. En må da ha få nøkler i hver node og et 23-tre vil dermed være velegnet.

Oppgave 4



Oppgave 5

Teoremet som viser at sortering er $\Omega(\ n*\log(n)\ )$ baserer seg på antagelsen at sorteringen skjer ved kun å sammenligne to og to elementer fra vektoren. Tellesortering tilfredsstiller ikke denne antagelsen. Ved tellesortering utnytter man i tillegg kjennskap til elementenes verdiområde og det er derfor mulig å oppnå lineær kjøretid.

Oppgave 6 (6 %)

Formålet med en hash-funksjon er å oppnå O(1) kjøretid i member/insert/delete operasjonene ved å benytte selve nøkkelen til å gjøre et direkte oppslag i datastrukturen gitt ved nøkkelens hash-verdi. Hash-funksjonen må være rask å evaluere og samtidig må nøklene spres så uniformt som mulig mellom de ulike hash-verdiene i hash-tabellen.

Hash tabell etter innsetting:

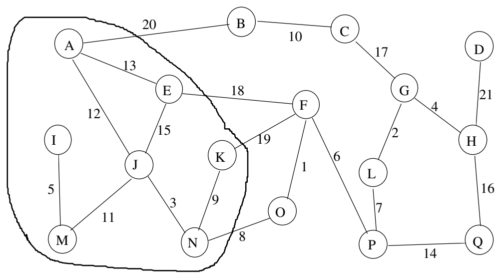
0	1	2	3	4	5	6	7	8	9
			23	44	5	33	37	87	

Oppgave 7

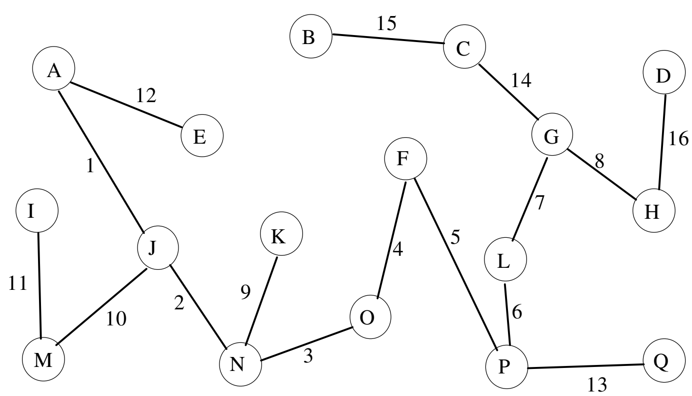
7a) (5%) A BEJ CFNM GKOPI LHQ D (mellomrom mellom de ulike nivåene) [Mange lovlige svar]

7b) (5%) ABCGHDQPFEJMINOKL (valgt først i alfabetet hele tiden) [Mange lovlige svar]

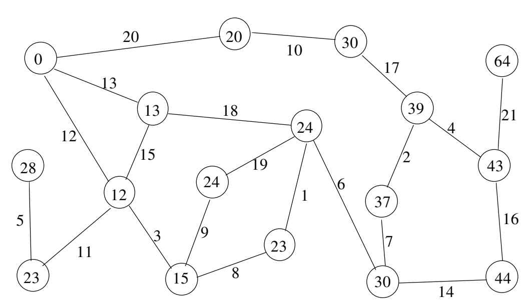
7c) (5%) "N"-"O" kanten er den billigste kanten mellom de innringede og de andre nodene i grafen. Påstand følger derfor ut fra basis teoremet for konstruksjon av minimale spenntrær. Se figur:



7d) (5%) Spenntreet er gitt i figuren under med node "A" som kildenode:



7e) (5%) A=0, B=20, C=30, D=64, E=13, F=24, G=39, H=43, I=28, J=12, K=24, L=37, M=23, N=15, O=23, P=30, Q=44:



7f) (5%) Maskimal flyt = 12, begrenset av ("F"-"P"), ("G"-"L") og ("G"-"H"). Kan løses ved 3 pass i Ford-Fulkerson algoritmen.

Oppgave 8 (6 %)

Alternativer: n=5•10^8 og m=35

1. "Naive string matching": O(n^2) i worst-case, men i dette tilfellet i praksis lineær: O(n).
2. "Rabin-Karp": O(n) men med relativt høy konstant
3. "Andre algortimer i boka": Ikke pensum og dermed ikke kandidater (heller ikke det beste valget)
4. "Boyer-Moore": Å anbefale ved lange strenger og relativt store alfabet. Kan være O(n/m) og vil i dette tilfellet måtte forventes å være bedre enn lineær.

Konklusjon: Man bør velge Boyer-Moore algoritmen.

Oppgave 9 (28 %)

9a) (14%) Nok med pseudo-kode eller klart formulert algoritmeide**.**

Løsning A: (Rekursiv)

```
FUNCTION svar( start, slutt, starttank ) : Real;
CONST FullTank= 40;
begin
  if (slutt>start) AND ( (Avstand[slutt]-Avstand[start])>starttank ) then
  begin
        billigst := indeks til billigste bensinstasjon av stasjon nr. start, start+1,..,slutt-1

        fylling := Avstand[slutt] - Avstand[billigst];
        if fylling>FullTank then fylling := FullTank;

        resttank := starttank - ( Avstand[billigst] - Avstand[start] );
        if resttank<0 then resttank := 0;

        fylling := fylling - resttank;

        tankpåneste := FullTank - (Avstand[billigst+1] - Avstand[billigst] );
        if tankpåneste<0 then ERROR;

        svar := svar(start,billigst,starttank) + fylling*Pris[billigst] + svar(billigst+1,slutt,tankpåneste);
  end;
end;
```

svar(1,N,0) vil da gi løsningen i O(N*logN) ved heldig splitting, men worst-case er O(N^2) analogt med analyse av QuickSort.

Løsning B: (Dynamisk programmering)

```
FUNCTION svar : Real;
CONST FullTank= 40;
VAR
        mp : Array[1..N,0..FullTank] Of Real; { mp[i,j]=minste totalpris på stasjon "i" med "j" liter på tanken. }
begin
        for j:=0 to FullTank do mp[1,j] := j*Pris[1];

        for i:=2 to N do
        for j:=0 to FullTank do
        begin
                mp[i,j]:= UENDELIG;
                jforrige := j + (Avstand[i] - Avstand[i-1]);
                if jforrige<=FullTank then mp[i,j]:=mp[i-1,jforrige];
                for k:=0 to j-1 do
                begin
                        påfyll := (j-k) * Pris[i];
                        if mp[i,k]+påfyll<mp[i,j] then mp[i,j] := mp[i,k]+påfyll;
                end;
        end;

        svar := mp[N,0];
end;
```

Algoritmen vil kreve O(N*40*40) i kjøretid i tillegg til ekstra plassbehov for mp[].

Løsning C: ("Finn første billigere enn deg selv innen 40 mil")

```
FUNCTION svar : Real;
CONST FullTank= 40; { Setter/antar Pris[N]=0 for å unngå en del testing }
begin
        i:=1;
        while (i<N) do
        begin
                if "det finnes billigere innen 40 mil" then
                begin
                        j := indeks til første stasjon billigere enn nr. i;
                        "Fyll tanken med (Avstand[j] - Avstand[i] ) liter";
                        i := j;
                end else
                        "Fyll full tank og inkrementer i";
        end;
end;
```

Algoritmen vil kreve O(N*40) i worst case kjøretid hvis vi antar at det plass til maksimalt 40 bensinstasjoner på 40 mil. Uten en slik antagelse er algoritmen O(N^2).

9b) (14%) *Lag følgende graf*:
For hver bensinstasjon lager man 41 noder som representerer tilstanden i form av antall liter bensin på tanken. Totalt får man da n=41*N noder i grafen. Kall disse nodene n(i,j) der i indekserer bensinstasjonene: 1..N og j indekserer "bensintanken": 0..40. Innfør en kant fra n(i1,j1) til n(i2,j2) med kostnad lik 0 hvis det er en direkte vei mellom bensinstasjon i1 og i2 med avstanden j1-j2. Innfør i tillegg kanter mellom alle par n(i,j1) og n(i,j2) med kostnad (j2-j1)*Pris[i] for j2>j1. Totalt har man maksimalt e=40*"Antall kanter i opprinnelig graf"+40*40 kanter. Dijkstras algoritme på denne grafen fra n(1,0) til n(N,0) vil løse det gitte problemet.