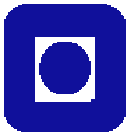
Norges Teknisk-Naturvitenskapelige Universitet
Institutt for Datateknikk og Informasjonsvitenskap



78010 / 45011 Algoritmer og Datastrukturer
Løsningsforslag til eksamen 13.12.97

Oppgave 1

a) P(i,j) er sannsynligheten for at A vinner serien (dvs. n kamper), gitt at
- A må vinne i av de resterende kampene, dvs. n-i kamper er vunnet.
- B må vinne j av de resterende kampene, dvs. n-j kamper er vunnet.

b) $T(1) = c$

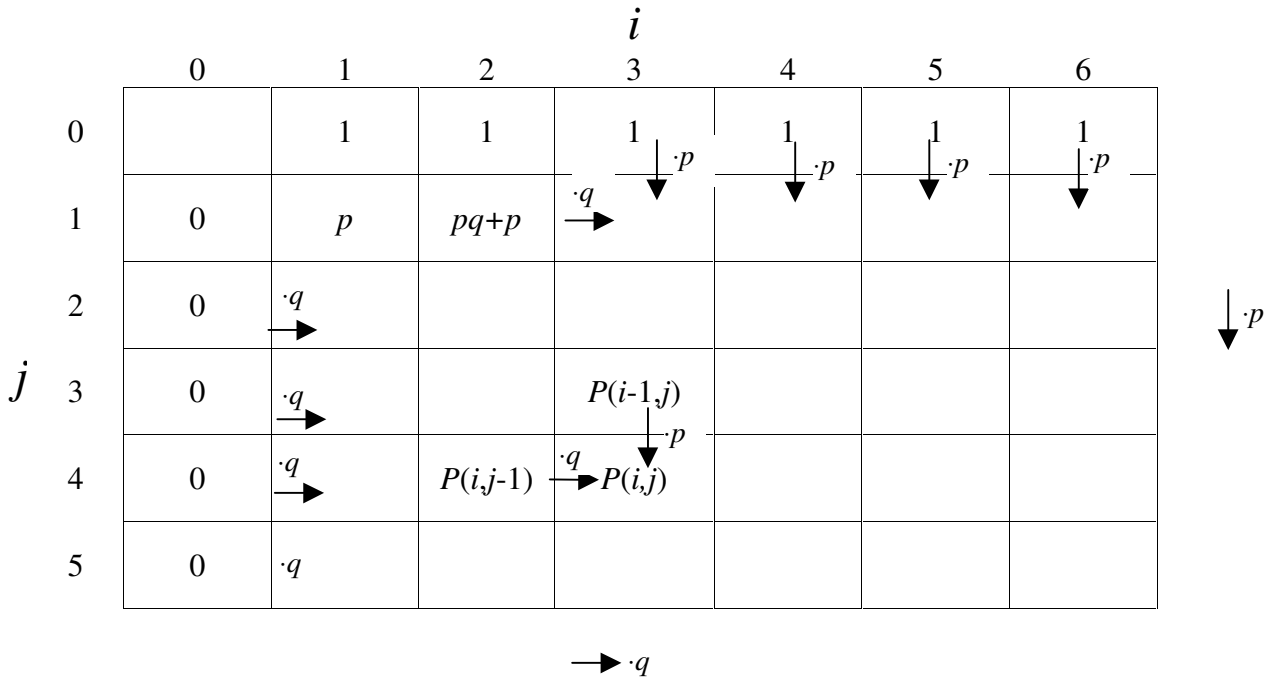
$T(k) \leq 2\,T(k\text{-}1) + d,\ k > 1$
$T(k) \leq n\,T(k\text{-}2) + 2d + d,\ k > 2$
…
$T(k) \leq 2^{k-1}\,T(1) + (2^{k-2} + 2^{k-3} + \ldots + 2 + 1)d$
$= 2^k\ (c/2 + d/2) - d$

$\Rightarrow T = O(2^k) = O(2^{i+j})$,

som er $O(4^n)$ hvis $i = j = n$.

Følgende avsnitt er ikke del av et svar på oppgave, bare et eksempel på hvordan dynamisk programmering skal brukes.

Selvsagt skal dynamisk programmering brukes i praksis. Ser følgende sammenheng i en matrise (i,j) :



Vi kan da lett lage en dynamisk programmeringsalgoritme for å beregne $P(i,j)$:

```
function P(i,j)
      // Initier matrise
      for k = 0 to i do
            M[0,k] = 0
      for k = 0 to j do
            M[k,0] = 1

      for m = 1 to i do
            for n  = 1 to j do
                  M[m,n] = pM[m-1,n] + qM[m,n-1]

      return M[i,j];
```

Denne har kjøretid $O(n^2)$. Se også oppgave 3, eksamen aug. 1997.

Oppgave 2

MERK: Pr. def. kan det kun være 1 kjendis

a) Idé: Ta 2 vilkårlige personer A og $\boldsymbol{B}$**.**
Hvis A kjenner $B \Rightarrow A$ er ikke kjendisen.
Hvis A ikke kjenner $B \Rightarrow B$ er ikke kjendisen.

Vi kan altså gå sekvensielt gjennom par (A,B) av kandidater, for så å eliminere enten $\boldsymbol{A}$ **eller** $\boldsymbol{B}$. Den eliminerte erstattes med neste kandidat. Når alle n personene har inngått i en partest står en igjen med den eneste mulige kandidaten. **Må** sjekke om denne faktisk er en kjendis.

b) Algoritme, skrevet i Java:

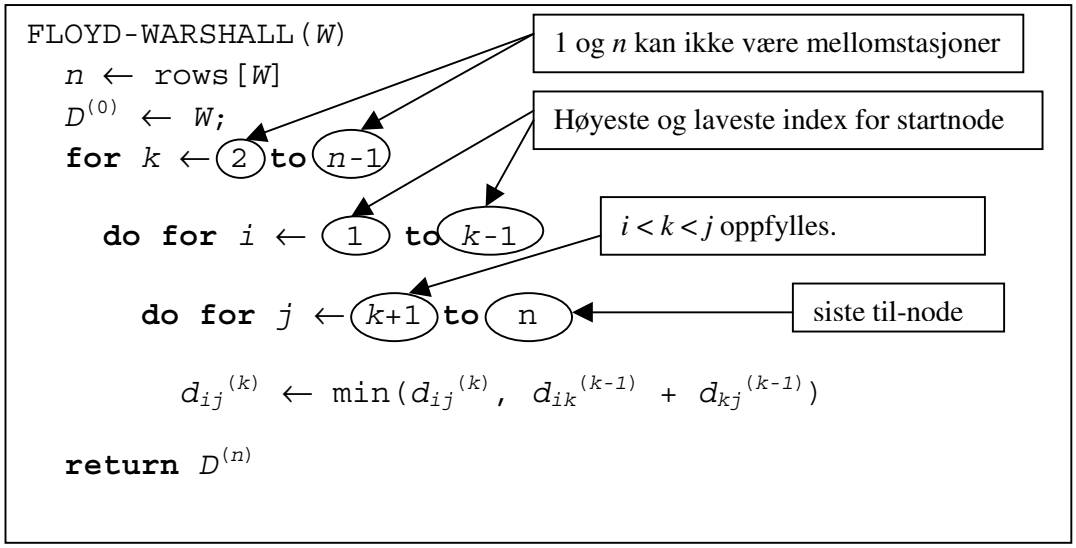
```
int Kjendis(Bool K[n, n]) {
    // Returnerer kjendisnummeret (p) dersom kjendis
    // finnes, ellers 0
    int   p, i = 1, j = 2, neste = 3;

    // Finn mulig kandidat
    while (neste <= n+1) do {
      if (K[i,j]) i = neste;
      else j = neste;
      neste++;
    }
    if (i == n+1) p = j;
    else  p = i;

    // Må sjekke om p virkelig er kjendisen
    int k = 1;
    Bool muligkjendis = true;
    K[p,p] = false;      // For å lette test-arbeidet
    while (muligkjendis && k <= n) do {
      if (K[p,k])   muligkjendis = false;
      if (!K[k,p])
        if (k != p) muligkjendis = false;
      k++;
    }
    if (muligkjendis)
      return p;
    return 0;  // ingen kjendis
}
```

C-variant:

```
int kjendis = 1;
int funnet = true;

for (test=2; test<=n; test++)
  if K(kjendis, test) kjendis = test; // Evt. kjendis
                                      // viloverleve

// Sjekker at alle kjenner kjendis, kjendis ingen
for (test=1; test<=n; test++)
     if (kjendis != test) && (K(kjendis,test) >=
          (K(test, kjendis)) return 0;

return (kjendis);
```

Oppgave 3

a) Bruker Floyd-Warshall som kjent algoritme. Fordi det ikke er mulig å padle motstrøms, trenger vi bare beregne halve *D* matrisen i algoritmen. (Superindeks *k* kan fjernes i en implementasjon.)



b) Floyd-Warshalls algoritme er bedre enn $|V|$ ganger Dijkstra. De modifikasjoner som er gjort ovenfor halverer minst konstanten i kjøretiden for Floyd-Warshall. (Fortsatt er asymptotisk kjøretid $O(|V|^3)$).

Alternativ løsning:

Bruk DAG-SHORTEST-PATH $|V|$ ganger. Denne algoritmen har $O(|V| + |E|)$ kjøretid etter at nodene er topologisk sortert. Vi antar at vi kan lage grafen slik at denne er topologisk sortert i utgangspunktet.

Antall kanter i grafen blir da : $V + V\text{-}1 + V\text{-}2 + \ldots + 2 + 1 = O(|V|^2)$. Kjøretiden for DAG-SHORTEST-PATH blir da $O(|V| + |V|^2) = O(|V|^2)$. Denne må kjøres $|V|$ ganger, noe som gir total kjøretid $O(|V|^3)$. Siden DAG-SHORTEST-PATH vil evaluere færre kanter enn DIJKSTRA vil denne algoritmen ha en konstant i noenlunde samme størrelse som den modifiserte Floyd-Warshall algoritmen.

Oppgave 4

a) $n\ \theta(q) = \theta(q\ n)$. Enkel innenfor/utenfor algoritme som i kompendiet.

b)

Sorter på y-verdier:	$\theta(q \log q)$
n stk binære søk	$n\ \theta(\log q)$
n stk Behandling av 0,1 eller 2 skjæringer	$n\ \theta(1)$
Totalt	$\theta((q+n) \log q) + \theta(n)$ $= \theta((q+n) \log q)$

c)

Sortere på y-verdier	$\theta(q \log q)$
n stk binære søk	$n\ \theta(\log q)$
n Behandling av inntil $\sqrt{q}$ skjæringer	$n\ O(\sqrt{q})$
Totalt	$\theta(q \log q) + n(\theta(\log q) + O(\sqrt{q}))$ $= O((q+n) \log q + n\sqrt{q})$