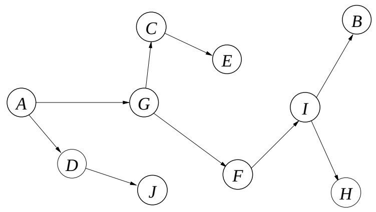
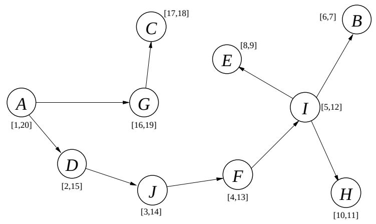
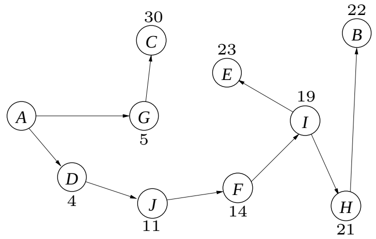
1a



1b



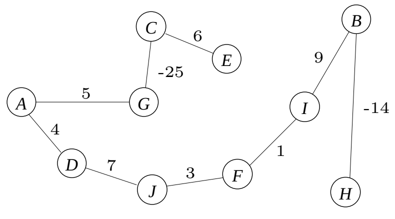
1c



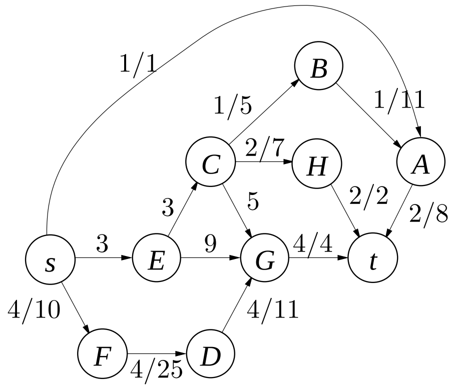
1d

$\{A, C, G\}, \{D\}, \{J\}, \{B, E, F, H, I\}$

1e



2a



Maximal strømning = 8.
Snit med kapacitet 8: $(\{s, D, E, F, G\}, \{A, B, C, H, t\})$

2b

Sti	Forbedring
sAt	1
$sEGt$	3
$sFDGt$	1
$sFDGECHt$	2
$sFDGECBAt$	1

3a

Lav en ny knude s. Forbind s til alle markerede knuder. Kør BFS startende i s. En knude med maximal BFS nummer har størst mulig afstand til den nærmeste markerede knude. Tid $O(g^2)$.

3b

Lav en orienteret graf indeholdende de g^2 knuder fra gittergrafen, plus to knuder s og t. Tilføj kanter (s, v) hvor v er markeret A og (u, t) hvor u er markeret B i gittergrafen. For hver kant $\{u, v\}$ i gittergrafen tilføj orienterede kanter (u, v) og (v, u). Kør Edmonds-Karp algoritmen på den resulterende graf, hvor alle kanter antages at have kapacitet 1. Der findes k kantdisjunkte stier mellem knuder markeret A og B hvis og kun hvis den maksimale strømning er mindst k. Tid $O(k \cdot g^2)$.

3c

Lav en orienteret graf indeholdende to knuder v_{in} og v_{out} for hver af de g^2 knuder v fra gittergrafen, plus to knuder s og t. Tilføj kanter (s, v_{in}) hvor

v er markeret A og (u_{out}, t) hvor u er markeret B. For hver knude v i gittergrafen tilføj kanten $(v_{\text{in}}, v_{\text{out}})$, og for hver kant $\{u, v\}$ i gittergrafen, tilføj orienterede kanter $(u_{\text{out}}, v_{\text{in}})$ og $(v_{\text{out}}, u_{\text{in}})$. Kør Edmonds-Karp algoritmen på den resulterende graf, hvor alle kanter antages at have kapacitet 1. Der findes k knudedisjunkte stier mellem knuder markeret A og B hvis og kun hvis den maksimale strømning er mindst k. Tid $O(k \cdot g^2)$.

4a

$LP(i,j)$	1	2	3	4	5
1	1	1	1	1	3
2	0	1	1	1	1
3	0	0	1	1	1
4	0	0	0	1	1
5	0	0	0	0	1

4b

```
for i=n downto 1
  for j=1 to n
    if (i>j) then LP[i,j]=0
    else if (i==j) then LP[i,j]=1
    else if (x_i==x_j) then LP[i,j]=2+LP[i+1,j-1]
    else LP[i,j]=MAX(LP[i+1,j],LP[i,j-1])
return LP[1,n]
```

Tid $O(n^2)$.

4c

```
<<beregn LP tabellen som i 4b>>
report(1,n)

proc report(i,j)
  if (j<i) return
  else if (i==j) then print x_i
  else if (x_i==x_j)
          print x_i
          report(i+1,j-1)
          print x_j
  else if (LP[i+1,j]<LP[i,j-1]) report(i,j-1)
  else report(i+1,j)
```

Tid $O(n^2)$.

5a

$U_1 = \mathtt{A\,B\,A}$

$U_2 = \mathtt{A\,C\,C\,A}$

$U_3 = \mathtt{C\,B}$

$U_4 = \mathtt{C\,C\,C}$

5b

Konstruer suffixtræet for $S_1\#S_2\#\cdots\#S_k\$$ i tid $O(kn)$.
Marker for hver knude v, om stien fra roden til v er en delstreng U_v, der forekommer i flere S_i'er. Hvis U_v kun forekommer i en streng, gemmes ydermere hvilken det drejer sig om. Dette markeres nede fra og opefter i suffix træet i et rekursivt gennemløb.
Hvis U_v kun forekommer i S_i, og kanten til faderen u har label $\alpha_1\alpha_2\cdots\alpha_l$, så forekommer $U_u\alpha_1$ også kun i S_i. Vi finder nu for hver streng S_i en tilsvarende streng $U_u\alpha_1$ der er kortest mulig. Dette kan gøres i et rekursivt gennemløb af suffix træet og hele tiden huske for alle $S_i$er den korteste delstreng fundet indtil videre der kun forekommer i S_i.
Tid $O(kn)$.