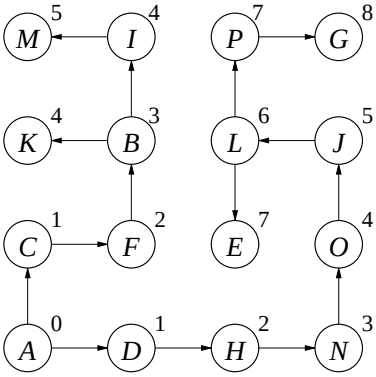
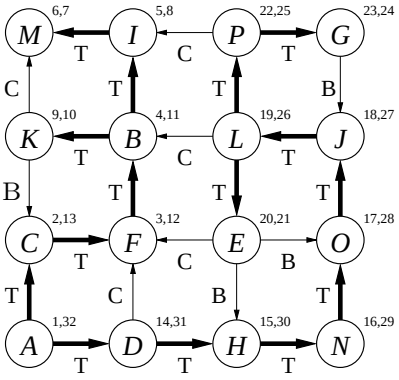
1a



Indsættelser i Q: $A, C, D, F, H, B, N, I, K, O, M, J, L, E, P, Q$

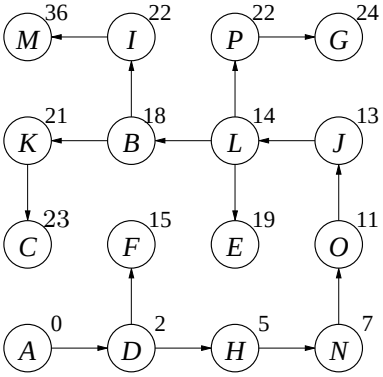
1b



1c

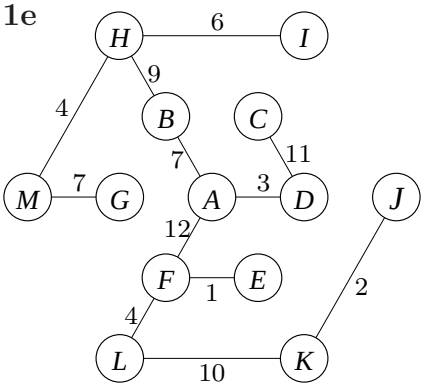
$\{E, G, H, J, L, N, O, P\}, \{B, C, F, K\}, \{A\}, \{D\}, \{M\}, \{I\}$

1d



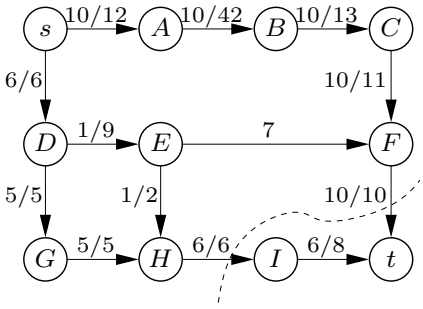
Fjernelser fra Q: $A, D, H, N, O, J, L, F, B, E, K, I, P, C, G, M$

1e



Fjernelser fra Q: $A, D, B, H, M, I, G, C, F, E, L, K, J$

2a



Maximal strømning = 16.
Snit med kapacitet 16: $(\{s, A, B, C, D, E, F, G, H\}, \{I, t\})$

2b

Forbedring	Sti
6	$sDEFt$
4	$sABCFt$
2	$sABCFEHIt$
4	$sABCFEDGHIt$

3a

Find de stærke sammenhængskomponenter i tid $O(n + m)$, og returner alle vigtige knuder i samme stærke sammenhængskomponent som s. Total tid $O(n + m)$.

3b

Kør Dijkstra's korteste veje algoritme på G for at finde den korteste afstand $d_G(s, v)$ fra s til alle vigtige knuder $v \in U$. Lad $\bar{G}$ være G hvor orienteringen på alle kanter er vendt. Kør Dijkstra's algoritme på $\bar{G}$ med start i s for at finde de korteste veje fra alle knuder i G til s. Lad $\delta(v) = d_G(s, v) + d_{\bar{G}}(s, v)$. Total tid $O((m + n) \log n)$. Alternativt kør en APSP algoritme (Floyd-Warshall) på G i tid $O(n^3)$, og lad $\delta_s(v) = d(s, v) + d(v, s)$ for alle $v \in U$.

3c

Kør 3b i tid $O((n + m) \log n)$, og sorter de fundne omkostninger for at besøge de vigtige knuder i tid $O(n \log n)$. Vælg grådigt knuder efter stigende omkostning så

længe den totale omkostning $\leq \Delta$. Total tid $O((n+m)\log n)$.

3d

Kør 3c for hver mulig startknude $s \in V \setminus U$, og returner knuden hvor flest vigtige knuder kan nås inden for omkostning Δ. Total tid $O(n(n+m)\log n)$.

4a

$B(j,i)$	0	1	2	3	4	5	6	7	8	9	10	11	12
0	T												
1	T		T										
2	T		T		T								
3	T		T	T	T	T		T					
4	T		T	T	T	T		T	T	T	T		T

4b

```
for j=0 to k
  for i=0 to n
    if i=0 then B[j,i] = TRUE
    else if j=0 then B[j,i] = FALSE
    else if c[j]>i then B[j,i] = B[j-1,i]
    else B[j,i] = B[j-1,i] OR B[j-1,i-c[j]]
return B[k,n]
```

Tid $O(nk)$.

4c

```
code from 4b)
if B[k,n]=FALSE then print "Ingen løsning"
else report(k,n)

proc report(j,i)
  if i=0 then return
  if c[j]<=i AND B(j-1,i-c[j]) then
    print c[j]
    report(j-1,i-c[j])
  else report(j-1,i)
```

Tid $O(nk)$.

4d

Lad $U(j,i)$ angive om løsningen er entydig og $L(j,i)$ være den største mønt i en entydig løsning for værdien i med mønterne $c_1, \ldots, c_j$.

```
for j=0 to k
  for i=0 to n
    if i=0 then B[j,i] = TRUE, U[j,i] = TRUE, L[j,i] = 0
    else if j=0 then B[j,i] = FALSE
    else if c[j]>i OR B[j-1,i-c[j]]=FALSE then
      B[j,i] = B[j-1,i], U[j,i]=U[j-1,i], L[j,i]=L[j-1,i]
    else
      B[j,i] = TRUE
      if (B[j-1,i]=FALSE) OR (B[j-1,i] AND U[j-1,i] AND L[j-1,i]=c[j]) then
         U[j,i] = U[j-1,i-c[j]], L[j,i] = c[j]
      else U[j,i] = FALSE
return U[k,n]
```

Tid $O(nk)$.

5a

Byg suffikstræet for $T\$$ i $O(n)$ tid. I et DFS gennemløb i $O(n)$ tid, find knuderne V i træet, således at stierne fra roden ned til knuderne indeholder $\geq k$ tegn, og stien ned til fædrerne i træet har $< k$ tegn. For hver knude $v \in V$ lad L_v være listen af start indexerne for suffixene svarende til bladene under v. Alle L_v beregnes i $O(n)$ tid ved DFS gennemløb startende i knuder $v \in V$. Sorter alle L_v listerne, og scan hver liste igennem for at finde det tættetste nabo par i, j. Da alle L_v listerne tilsammen indeholder n heltal mellem 0 og $n-1$, kan disse lister sortes ved én udførsel af radix sort i $O(n)$ tid. Total tid $O(n)$.