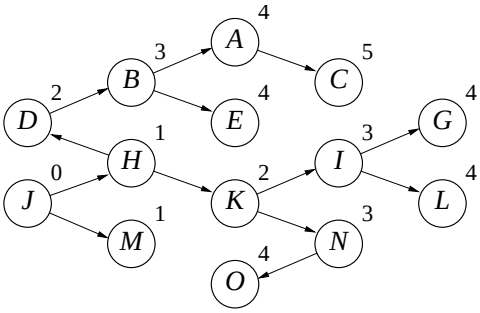
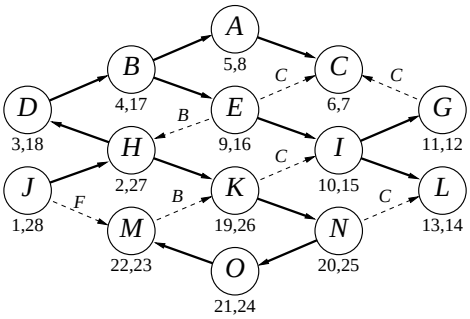
1a



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

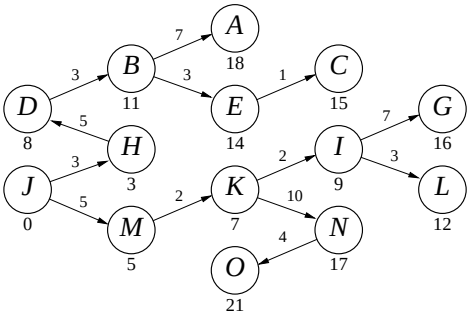
1b



1c

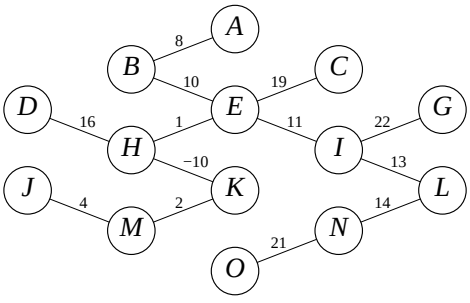
$\{A\}, \{C\}, \{G\}, \{I\}, \{J\}, \{L\}, \{B, D, E, H\}, \{K, M, N, O\}$

1d



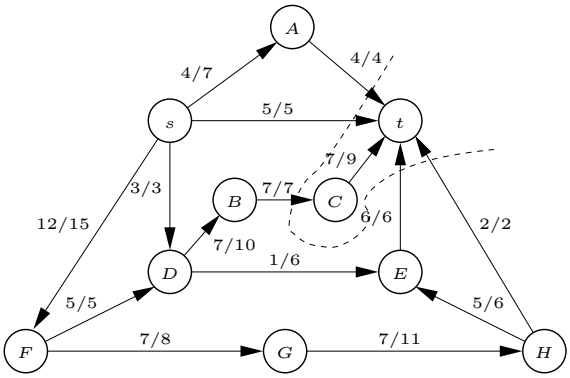
Fjernelser fra $J, H, M, K, D, I, B, L, E, C, G, N, A, O$

1e



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

2a



Maximal strømning = 24.
Snit med kapacitet 24: $(\{s, A, B, D, E, F, G, H\},\{C, t\})$

2b

Forbedring	Sti
5	st
4	sAt
3	$sDEt$
2	$sFGHt$
3	$sFDEt$
2	$sFDBCt$
5	$sFGHEDBCt$

3a

Kør BFS startende i u. Hvis ikke alle knuder kan nås, returner en ubesøgt knude, ellers returner den sidste knude BFS algoritmen finder. Tid $O(n + m)$.

3b

Kør APSP på grafen i $O(n^3)$ tid. For hver knude u find en knude v med maximal afstand fra u, og returner en knude u der har mindst mulig maximal afstand. Tid $O(n^3)$.

3c

Find alle de stærke sammenhængskomponeter i $O(n + m)$ tid. Træk hver sammenhængskomponent sammen til en knude, således at vi får en DAG. Lav en topologisk sortering af DAGen i $O(n + m)$ tid. Grafen G har en broadcast knude hvis og kun hvis alle knuder kan nås fra den første knude i den topologisk sorterede graf, hvilket kan checkes med DFS i $O(n + m)$ tid. Total tid $O(n + m)$.

3d

Hvor hvert par af knuder (u_1, u_2) lav en BFS søgning, hvor man starter samtidige i u_1 og u_2 (dvs. disse får begge BFS afstand 0). Husk det maksimale BFS nummer. For alle de par af knuder (u_1, u_2), hvor BFS gennemløbet besøger alle knuderne i grafen, findes det mindste maksimale BFS nummer, hvilket er den ønske minimale broadcast radius. Total tid $O(n^2(n + m))$.

Mere effektiv løsning: Kør APSP i $O(n^3)$ tid. For hvert par af knuder (u_1, u_2) løb

alle knuder $v \in V$ igennem og find afstanden til den nærmeste u_i vha. de beregnede APSP afstande. Tid $O(n)$ for hvert par af (u_1, u_2). Returner den maksimale afstand til en knude v for det par (u_i, u_j), hvor den maximale afstand til et v er mindst mulig. Total tid $O(n^3)$.

4a

$6, 7, 9, 12, 18, 25, 33$

4b

$\delta(i,j)$	1	2	3	4	5
1	1	2	2	2	2
2		1	2	2	3
3			1	2	3
4				1	3
5					1

4c

```
for i = 1 to n
   A[i,i] = 1
   for j = i + 1 to n
      A[i,j] = 0
      for k = 1 to i
         if x_j − x_i > x_i − x_k then
            ℓ = 1 + A[k,i]
            if ℓ > A[i,j] then
               A[i,j] = ℓ
               K[i,j] = k
r = 1
for i = 2 to n
   if A[i,n] > A[r,n] then
      r = i
return A[r,n]
```

Tid $O(n^3)$.

4d

```
code from 4c)
i = r, j = n
print j
while i ≠ j do
   print i
   k = K[i,j]
   j = i
   i = k
```

Tid $O(n^3)$.

5a

$S =$ `bcababcb`, $i = 2$, $j = 10$

5b

Konstruer suffix-træet for T. Annoter hver knude v med længden $v.\ell$ af strengen fra roden ned til og med v (betegnet $v.S$, men som ikke beregnes), og gem det mindste index $v.i$ og største index $v.j$ i et blad i v's undertræ (dvs. forekomsterne længst til venstre og længst til højre af S i T). For hver knude lad $v.s = \min\{v.\ell, v.j - v.i\}$ være længden af det længste præfix af strengen $v.S$ som har to ikke-overlappende forekomster i T. Lad u være en knude med maximal $v.s$ værdi. Alle disse oplysninger kan beregnes i et DFS gennemløb af suffix træet. Rapporter $u.i$, $u.j$, og rapporter $T[u.i, u.i + u.s - 1]$. Tid $O(n)$.

5c

Som 5b), men beregn for hver knude v også en liste $v.L$ af alle indexerne i v's undertræ, og lad $v.k$ være et index i $v.L$ som er tættest på $(v.i + v.j)/2$, og beregn istedet $v.s = \min\{v.\ell, v.k - v.i, v.j - v.k\}$. For en knude u med maksimal $u.s$ værdi, rapporteres $u.i, u.k, u.j$ og præfikset af $T[u.i, u.i + u.s - 1]$. Tid $O(n^2)$, da hvert $v.L$ har længde $O(n)$, og er konkatenationen af børnenes L-lister.