DATALOGISK INSTITUT, AARHUS UNIVERSITET

| |
|---|
| Det Naturvidenskabelige Fakultet |
| EKSAMEN |
| Grundkurser i Datalogi |
| **Algoritmer og Datastrukturer 2 (2003-ordning)** |
| Antal sider i opgavesættet (incl. forsiden): 6 (seks) |
| Eksamensdag: Fredag den 25. juni 2010, kl. 9.00-13.00 |
| Eksamenslokale: Skøjtehallen, Gøteborg Allé 9, Århus N |
| Tilladte medbragte hjælpemidler:<br>Alle sædvanlige hjælpemidler (lærebøger, notater, lommeregner).<br>Computer må ikke medbringes. |
| Materiale der udleveres til eksaminanden: |

**Opgave 1** (25%)

I denne opgave betragter vi først orienterede *pyramide-grafer* med ikke-negative vægte på kanterne. En pyramide-graf består af en række lag af knuder. I det nederste lag (lag 1) er der $n$ knuder, og i hvert lag opefter aftager antal knuder med én. Den $i$'te knude i lag $j$ har kanter til den $i$'te knude i lag $j+1$ og den $i+1$'te knude i lag $i$ og lag $i-1$ (hvis disse knuder findes). Det antages, at grafen er givet ved incidenslister, hvor incidenslisterne er sorteret alfabetisk.

**Spørgsmål a:** Angiv et BFS-træ for ovenstående graf, når BFS-gennemløbet starter i knuden $A$. Angiv kanterne i BFS-træet og BFS-numrene for knuderne. □

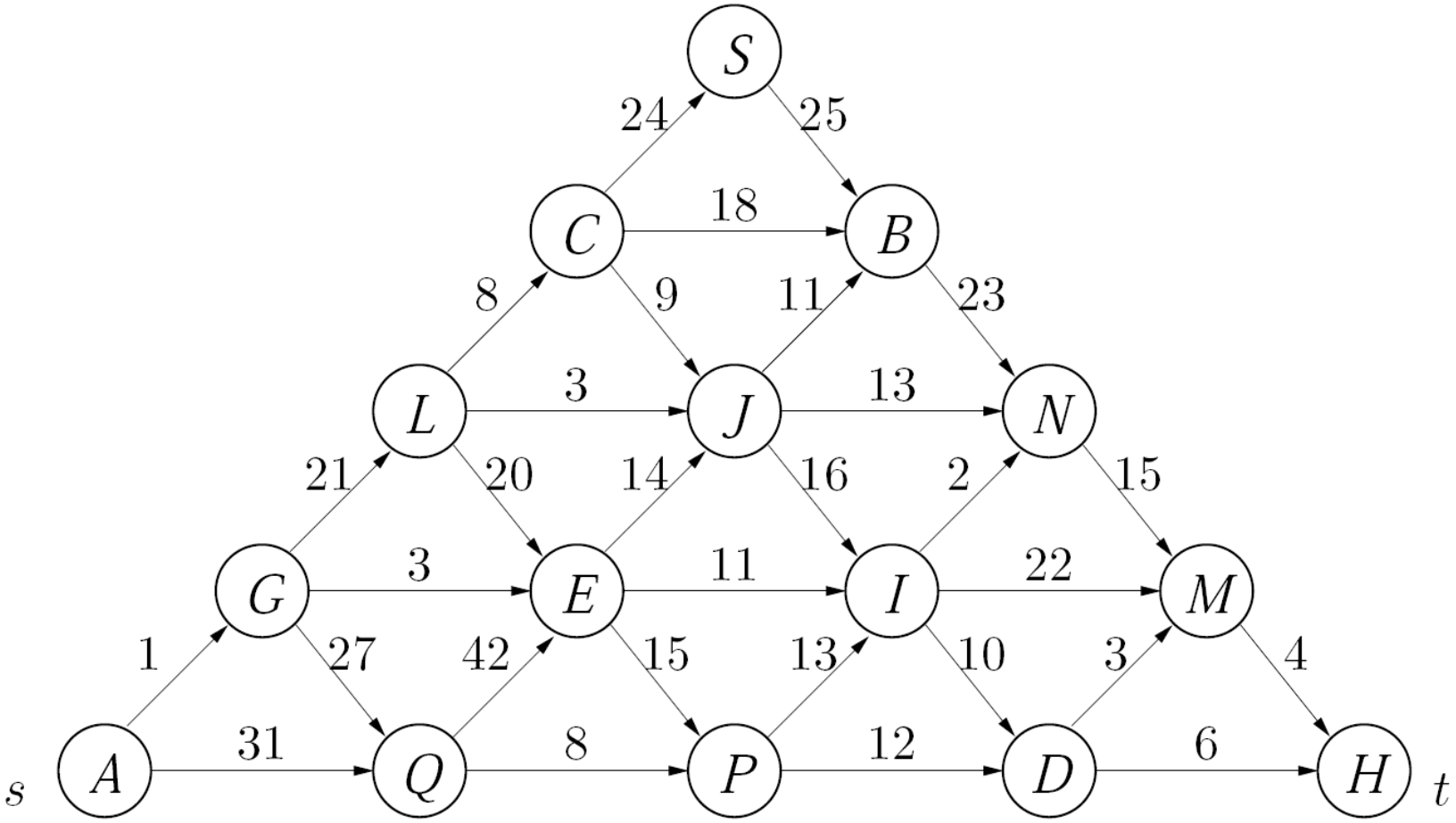
**Spørgsmål b:** Angiv et DFS-træ for ovenstående graf, når DFS-gennemløbet starter i knuden $A$, og en DFS-nummerering af knuderne. Angiv for hver knude "discovery time" og "finishing time". □

**Spørgsmål c:** Angiv et korteste veje træ for ovenstående graf, når kortest veje beregningen sker med hensyn til startknuden $A$. For hver knude $v$ angiv også afstanden fra startknuden $A$ til $v$. □

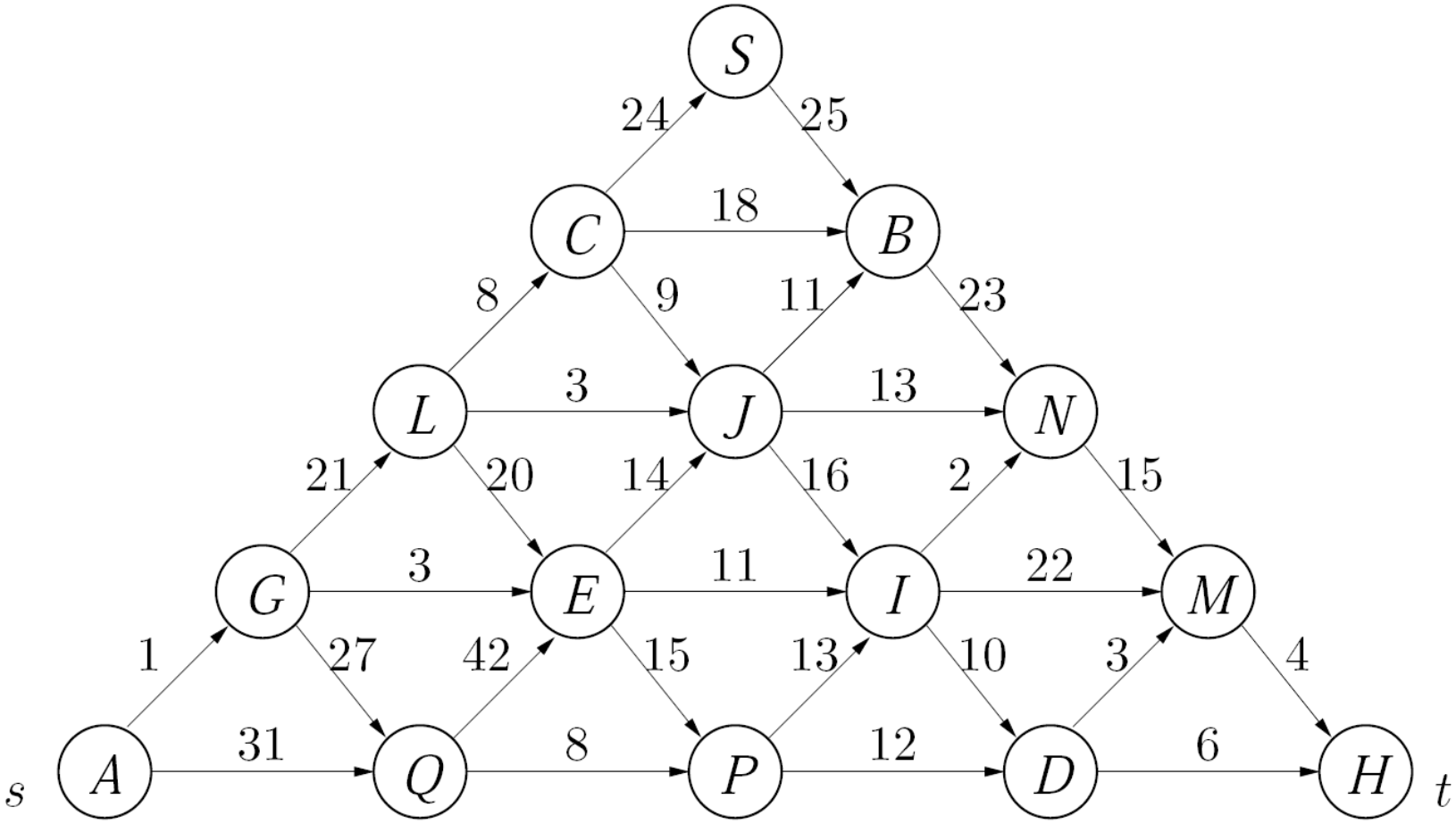
**Spørgsmål d:** For en pyramide-graf med $n$ knuder i det nederste lag, angiv en algoritme til at finde den korteste vej fra nederste venstre hjørne ($s$) til nederste højre hjørne ($t$) med udførselstid $O(n^2)$. □

**Spørgsmål e:** Angiv kanterne i et minimum udspændende træ for ovenstående uorienterede pyramide-graf med vægte på kanterne. □

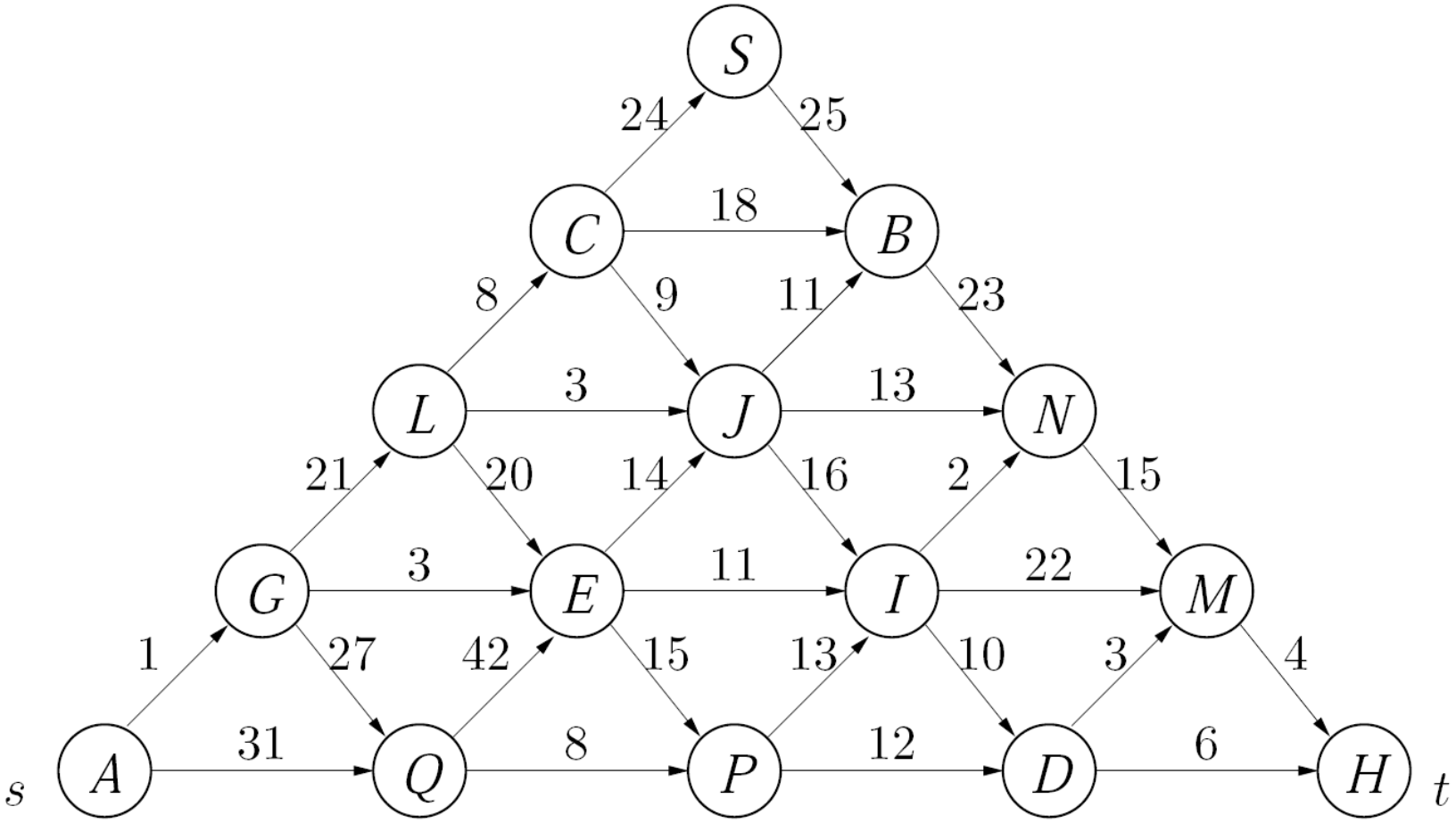
# 1(a) BFS


S
C
B
L
J
N
G
E
I
M
A
Q
P
D
H
s
t
24
25
18
8
9
11
23
3
13
21
20
14
16
2
15
3
11
22
1
27
42
15
13
10
3
4
31
8
12
6

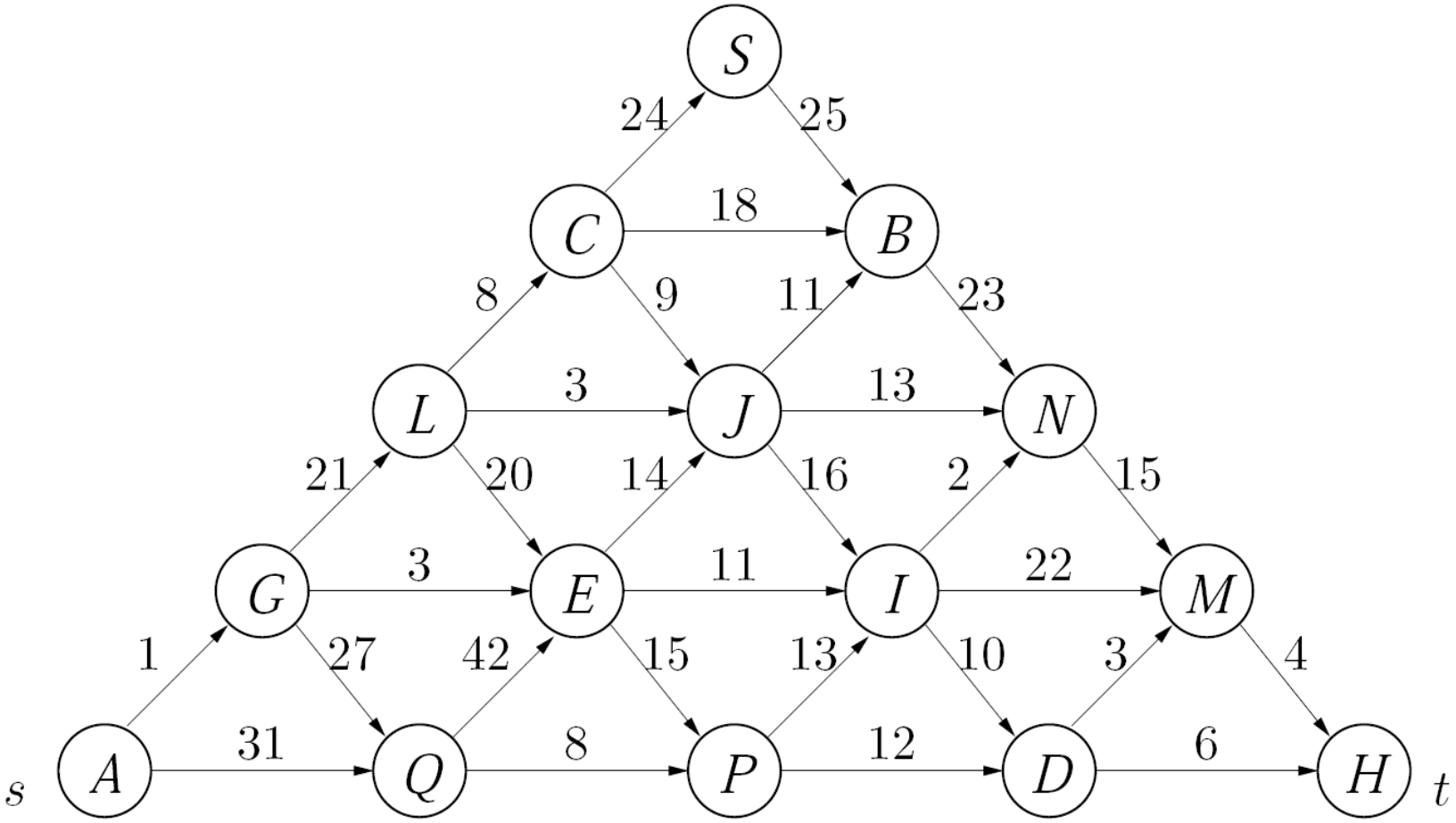
# 1(b) DFS


S
24
25
C
18
B
8
9
11
23
L
3
J
13
N
21
20
14
16
2
15
G
3
E
11
I
22
M
1
27
42
15
13
10
3
4
s
A
31
Q
8
P
12
D
6
H
t

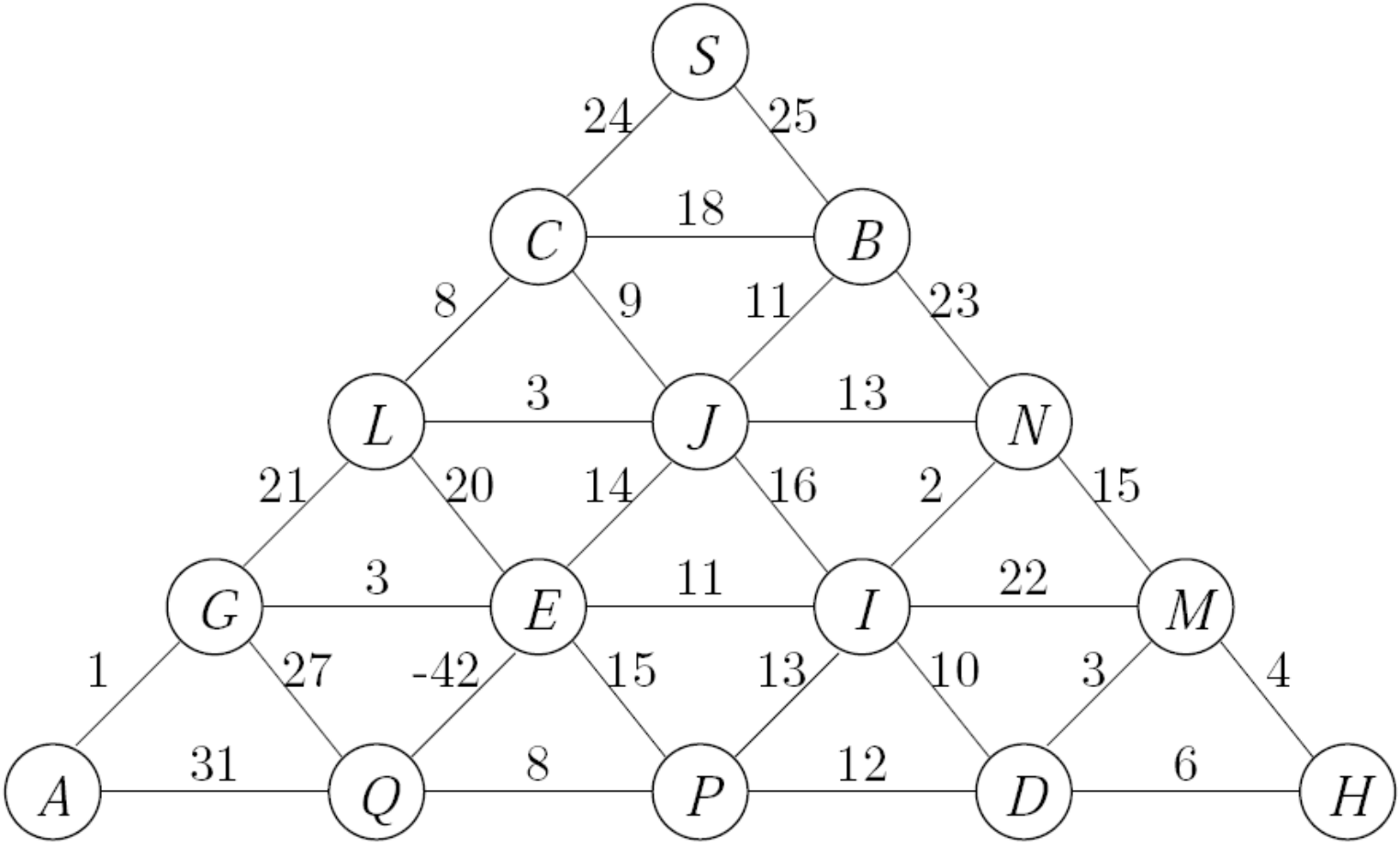
# 1(c) Korteste veje



# 1(d) Korteste veje

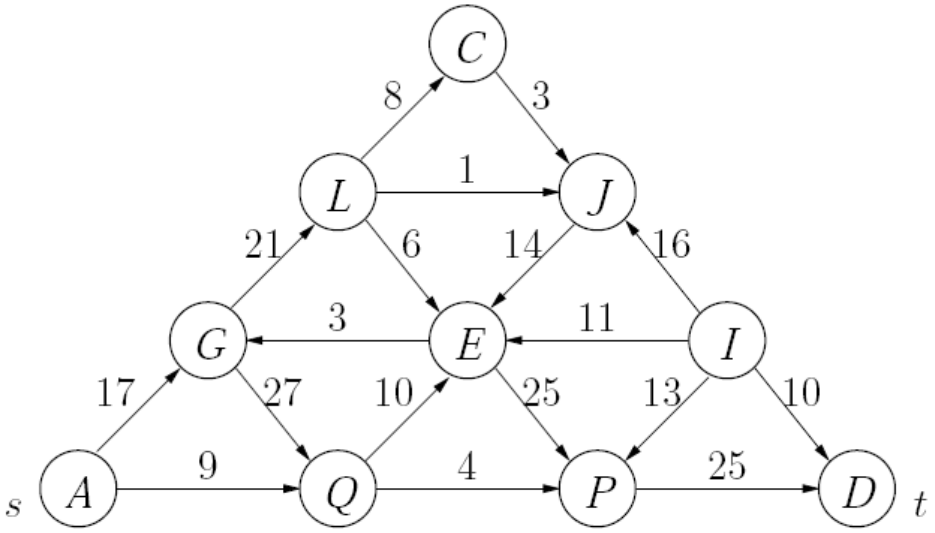


# 1(e) MST


S
24
25
C
18
B
8
9
11
23
L
3
J
13
N
21
20
14
16
2
15
G
3
E
11
I
22
M
1
27
-42
15
13
10
3
4
A
31
Q
8
P
12
D
6
H

**Opgave 2** (15%)

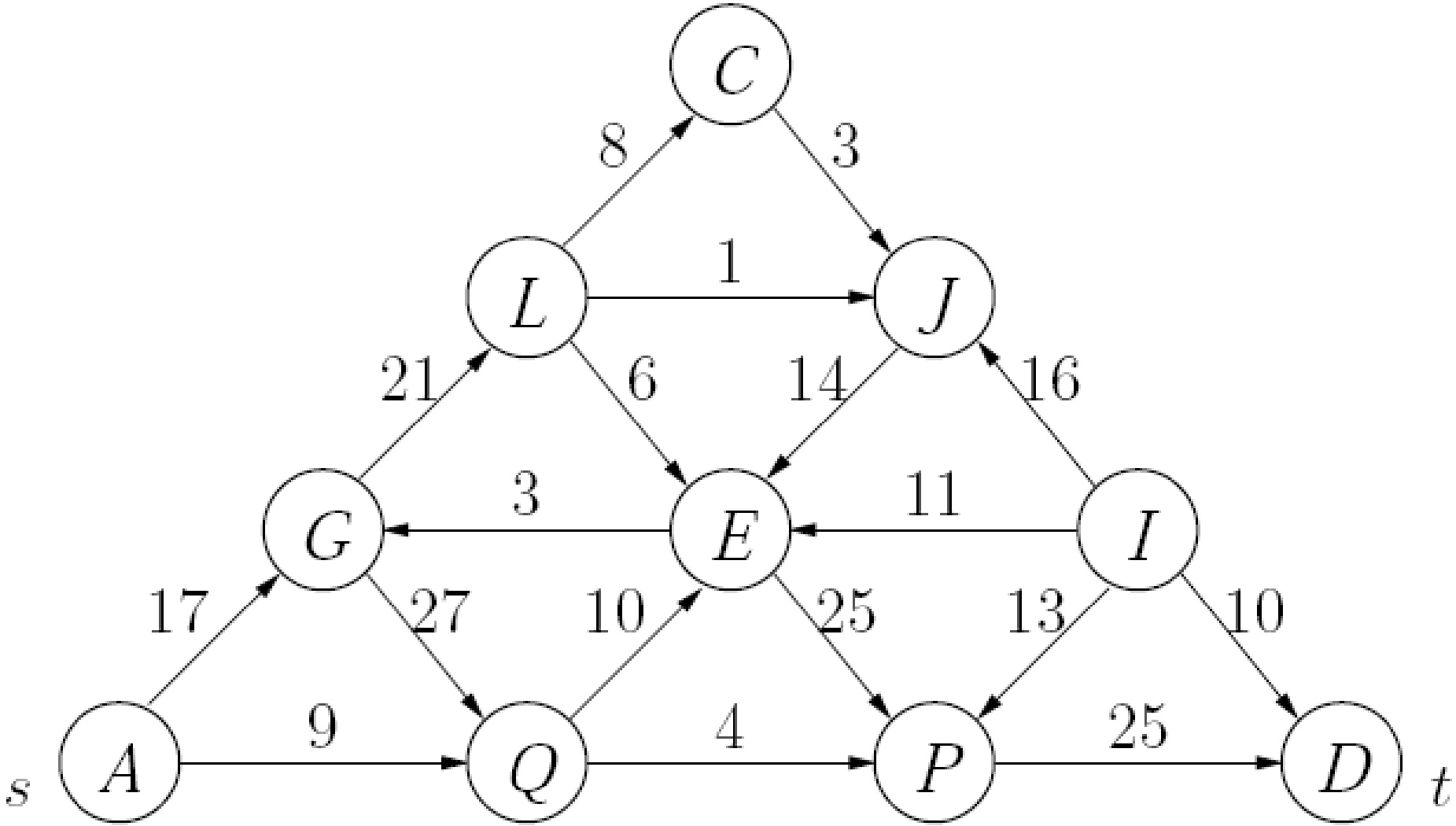
Betragt nedenstående netværk med de angivne kapaciteter på kanterne.



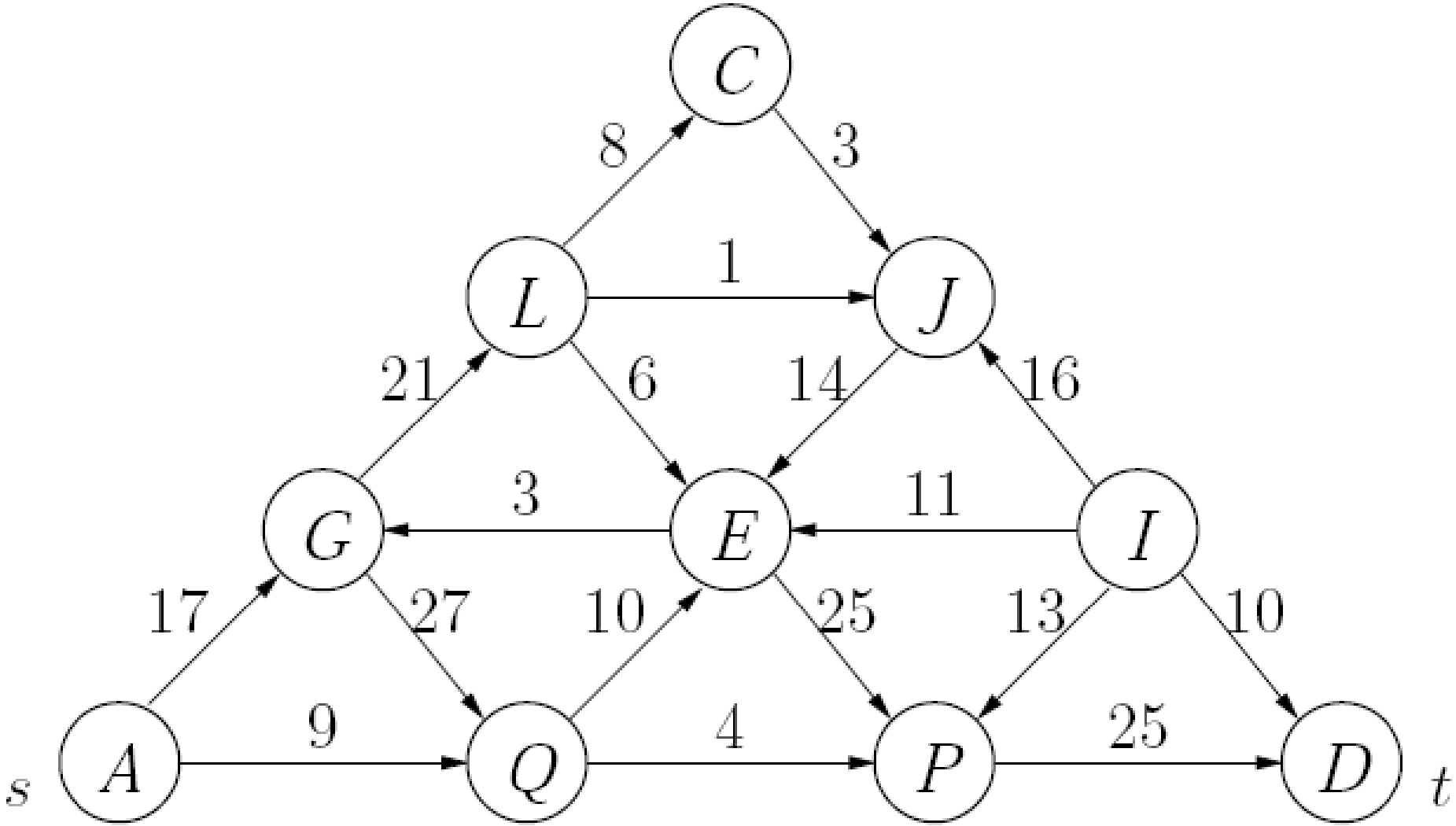

**Spørgsmål a:** Angiv en maksimal strømning fra $s$ til $t$ i netværket (angiv for hver kant strømningen langs kanten), angiv værdien af en maksimal strømning, og angiv et snit (dvs. opdeling af knuderne i to disjunkte mængder) hvor kapaciteten af snittet er lig værdien af en maksimal strømning. □

**Spørgsmål b:** Betragt Edmonds-Karp algoritmen anvendt på ovenstående graf til beregning af en maksimal strømning. Angiv de forbedrende stier der anvendes under udførslen af Edmonds-Karp algoritmen. For hver forbedrende sti angiv knuderne på stien og strømningen, man forbedrer med, langs stien. □

# 2(a) Max-Flow



# 2(b) Edmonds-Karp



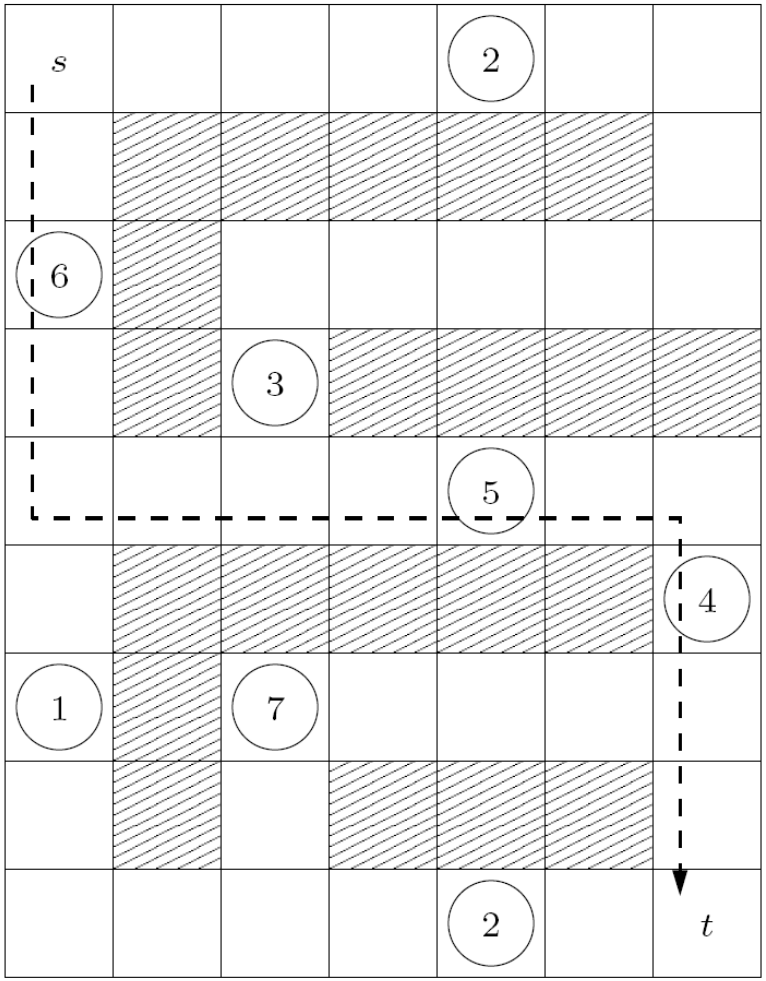
**Opgave 3** (20%)

I denne opgave betragter vi labyrinter som er givet ved et gitter af $m \times n$ celler, hvor cellerne kan være blokerede (skraverede) eller fri. Fra en given celle kan man gå til cellen oven over, neden under, venstre for, og højre for cellen, hvis disse er fri. Nedenstående eksempel (til ventre) er en $9 \times 7$ labyrint. Vi ønsker at komme fra øverste venstre hjørne ($s$) til nederste højre hjørne ($t$). I nogle af cellerne er der angivet et positivt tal. Dette tal angiver det antal *straf-point* man får ved at gå igennem cellen. F.eks. giver den stiplede sti $15 = 6+5+4$ straf-point. Målet er at finde en sti fra $s$ til $t$ med mindst mulig straf.

Labyrinten er givet ved et to-dimensionalt array $A$, hvor $A[i,j] = -1$ hvis cellen er blokeret, $A[i,j] = 0$ hvis cellen ikke giver straf-point, og ellers angiver $A[i,j] > 0$ det antal straf-point man får ved at gå igennem cellen (se figuren til højre).

**Spørgsmål a:** Beskriv en algoritme der finder en sti fra $s = A[1,1]$ til $t = A[m,n]$ med mindst mulige straf-point. Angiv algoritmens udførselstid. □

**Spørgsmål b:** Beskriv en algoritme der givet en labyrint (uden straf-point) finder det største samtidige antal stier man kan gå fra $s$ til $t$, således at alle par af stier kun har cellerne $s$ og $t$ til fælles. □




Labyrint

| | 1 | | | | | | $n$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| | 0 | −1 | −1 | −1 | −1 | −1 | 0 |
| | 6 | −1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | −1 | 3 | −1 | −1 | −1 | −1 |
| | 0 | 0 | 0 | 0 | 5 | 0 | 0 |
| | 0 | −1 | −1 | −1 | −1 | −1 | 4 |
| | 1 | −1 | 7 | 0 | 0 | 0 | 0 |
| | 0 | −1 | 0 | −1 | −1 | −1 | 0 |
| $m$ | 0 | 0 | 0 | 0 | 2 | 0 | 0 |

Array-repræsentation

**Opgave 4** (20%)

Lad $T$ være en streng af længde $n$, og lad $S_1, S_2, \ldots, S_k$ være $k$ strenge med samlet længde $N = \sum_{i=1}^{k} |S_i|$.

Vi ønsker at afgøre om $T$ kan skrives som en konkatenation af strenge fra $S_1, S_2, \ldots, S_k$, hvor hvert $S_i$ kan forekomme et vilkårligt antal gange (0, 1, eller flere gange).

F.eks. kan strengen

$$T = \underline{\texttt{A B B}}\,\underline{\texttt{B B}}\,\underline{\texttt{D}}\,\underline{\texttt{A B A}}\,\underline{\texttt{B B}}$$

skrives som en konkatenation af en delmængde af strengene

$$S_1 = \texttt{A B B} \qquad S_2 = \texttt{A C A A} \qquad S_3 = \texttt{B B} \qquad S_4 = \texttt{A B A} \qquad S_5 = \texttt{D}$$

For $0 \leq j \leq n$ lader vi $C(j)$ angive om $T[1..j]$ kan skrives som en konkatenation af strenge fra $S_1, \ldots, S_k$.

$$C(j) = \begin{cases} \textbf{sand} & \text{hvis } j = 0 \\ \textbf{sand} & \text{hvis } j > 0 \text{ og der findes } i \text{ hvor} \\ & \qquad |S_i| \leq j \ \wedge \ C(j - |S_i|) \ \wedge \ T[j - |S_i| + 1..j] = S_i \\ \textbf{falsk} & \text{ellers} \end{cases}$$

**Spørgsmål a:** Udfyld nedenstående tabel for $C(i)$ for $0 \leq i \leq 14$, når

$$T = \texttt{A B B B B B B A B A B A A A}$$

$$S_1 = \texttt{B B} \qquad S_2 = \texttt{B B B B B} \qquad S_3 = \texttt{A B} \qquad S_4 = \texttt{A B A} \qquad S_5 = \texttt{A A A}$$

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $C(i)$ | | | | | | | | | | | | | | | |

□

**Spørgsmål b:** Angiv en algoritme baseret på dynamisk programmering, der givet en streng $T$ og en mængde af strenge $S_1, S_2, \ldots, S_k$ afgør om $T$ kan skrives som en konkatenation af strenge fra $S_1, S_2, \ldots, S_k$, hvor hvert $S_i$ kan forekomme et vilkårligt antal gange. Angiv algoritmens udførselstid. □

**Spørgsmål c:** Udvid algoritmen fra spørgsmål b) til også at rapportere sekvensen af strenge fra $S_1, S_2, \ldots, S_k$ hvis konkatenation er lig $T$ – hvis sådan en løsning findes. Angiv algoritmens udførselstid. □

**Opgave 5** (20%)

Givet en streng $S$ angiver $\bar{S}$ strengen læst bagfra. F.eks. hvis $S = \texttt{A B C}$ så er $\bar{S} = \texttt{C B A}$.

I denne opgave ønsker vi givet en streng $T$ af længde $n$ og over et alfabet af størrelse $O(1)$ at finde en længst mulig delstreng $S$ således at både $S$ og $\bar{S}$ forekommer i $T$, og hvor der findes en forekomst af $S$ og $\bar{S}$ i $T$ som *ikke overlapper*.

F.eks. for $T = \texttt{D}\underline{\texttt{A B}}\texttt{C}\underline{\texttt{B A}}\texttt{C}$ forekommer $S = \texttt{A B}$ på position to i $T$ og $\bar{S} = \texttt{B A}$ på position 5 i $T$. Dette er en længst mulig streng hvor der findes ikke-overlappende forekomster af $S$ og $\bar{S}$. (Bemærk at $S = \texttt{A B C}$ og $\bar{S} = \texttt{C B A}$ begge forekommer i strengen $T$, men at der ikke er ikke-overlappende forekomster af $S$ og $\bar{S}$).

**Spørgsmål a:** Angiv for strengen

$$T = \texttt{A B C B A C B C A B B}$$

en længst mulig delstreng $S$, hvor der findes ikke-overlappende forekomster af $S$ og $\bar{S}$ i $T$. Angiv $S$ og positionen af en ikke-overlappende forekomst af $S$ og $\bar{S}$. □

I det følgende kan det antages at et suffikstræ for en streng af længde $n$ over et alfabet med $O(1)$ tegn kan konstrueres i $O(n)$ tid.

**Spørgsmål b:** Beskriv en algoritme der givet en streng $T$, finder en længst mulig delstreng $S$, hvor der findes ikke-overlappende forekomster af $S$ og $\bar{S}$ i $T$. Angiv algoritmens udførselstid. □