Plus courts chemins

Algorithmique – L3

François Laroussinie

7 décembre 2010

Plan

Plan

Plus courts chemins

$G = (S, A, w)$: orienté et valué ($w : A \to \mathbb{R}$).

Pour $\rho \stackrel{\text{def}}{=} v_0 \to v_1 \to \ldots \to v_k$, on note $w(\rho)$ sa longueur ou sa distance :

$$w(\rho) \stackrel{\text{def}}{=} \sum_{i=1,\ldots,k} w(v_{i-1}, v_i)$$

Plus courts chemins

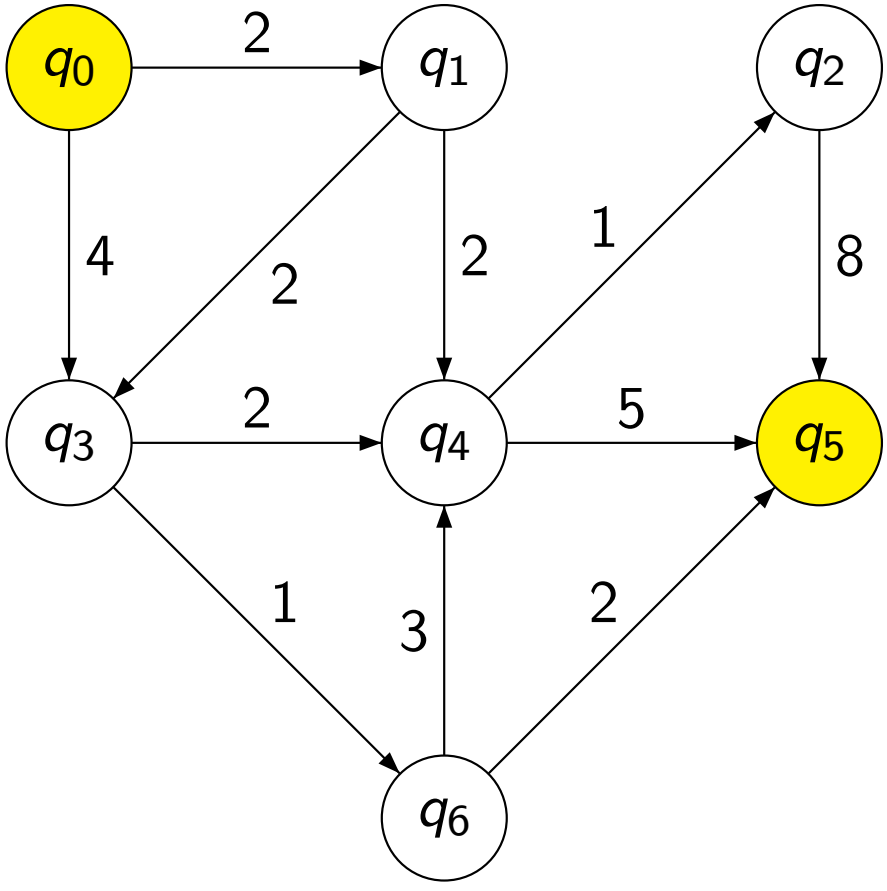
$G = (S, A, w)$: orienté et valué ($w : A \to \mathbb{R}$).

Pour $\rho \stackrel{\text{def}}{=} v_0 \to v_1 \to \ldots \to v_k$, on note $w(\rho)$ sa longueur ou sa distance :

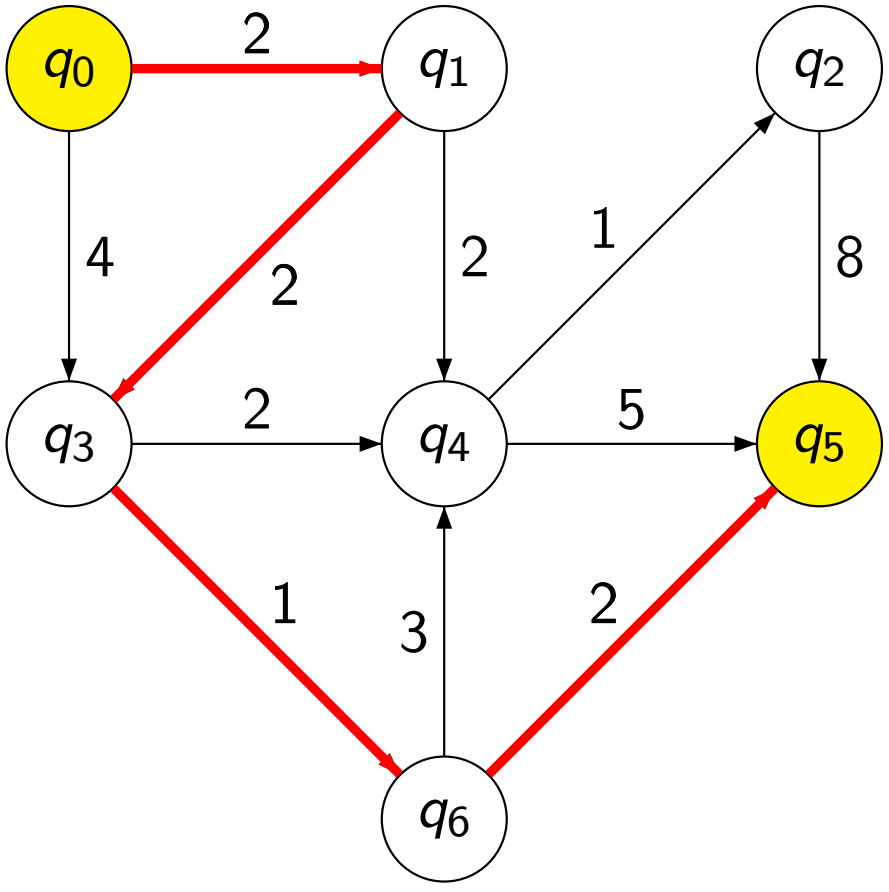
$$w(\rho) \stackrel{\text{def}}{=} \sum_{i=1,\ldots,k} w(v_{i-1}, v_i)$$

Définitions : Un chemin ρ de u à v est un plus court chemin (PCC) de u à v ssi, pour tout chemin π de u à v, on a $w(\pi) \geq w(\rho)$.

Exemple de PCC



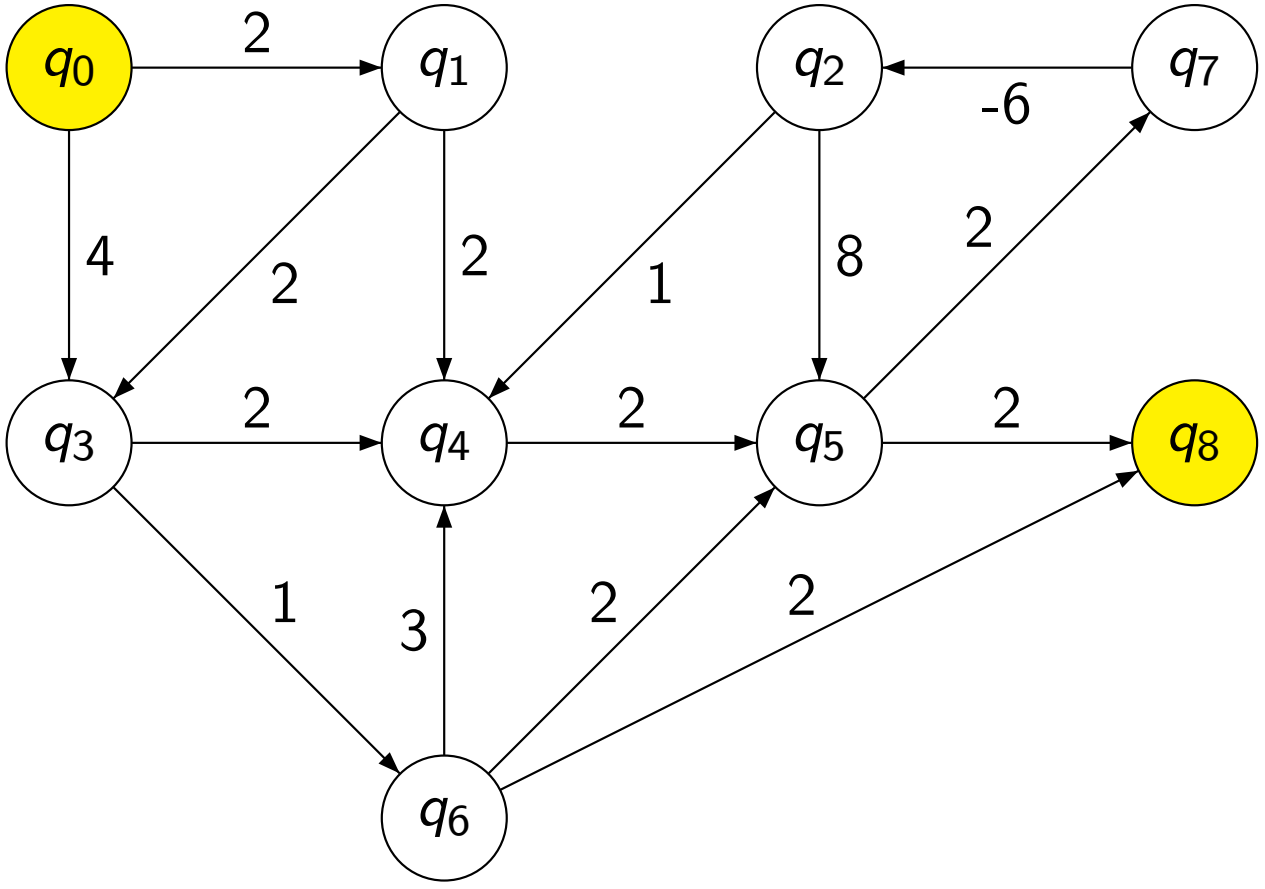
Exemple de PCC



distance $= 7$

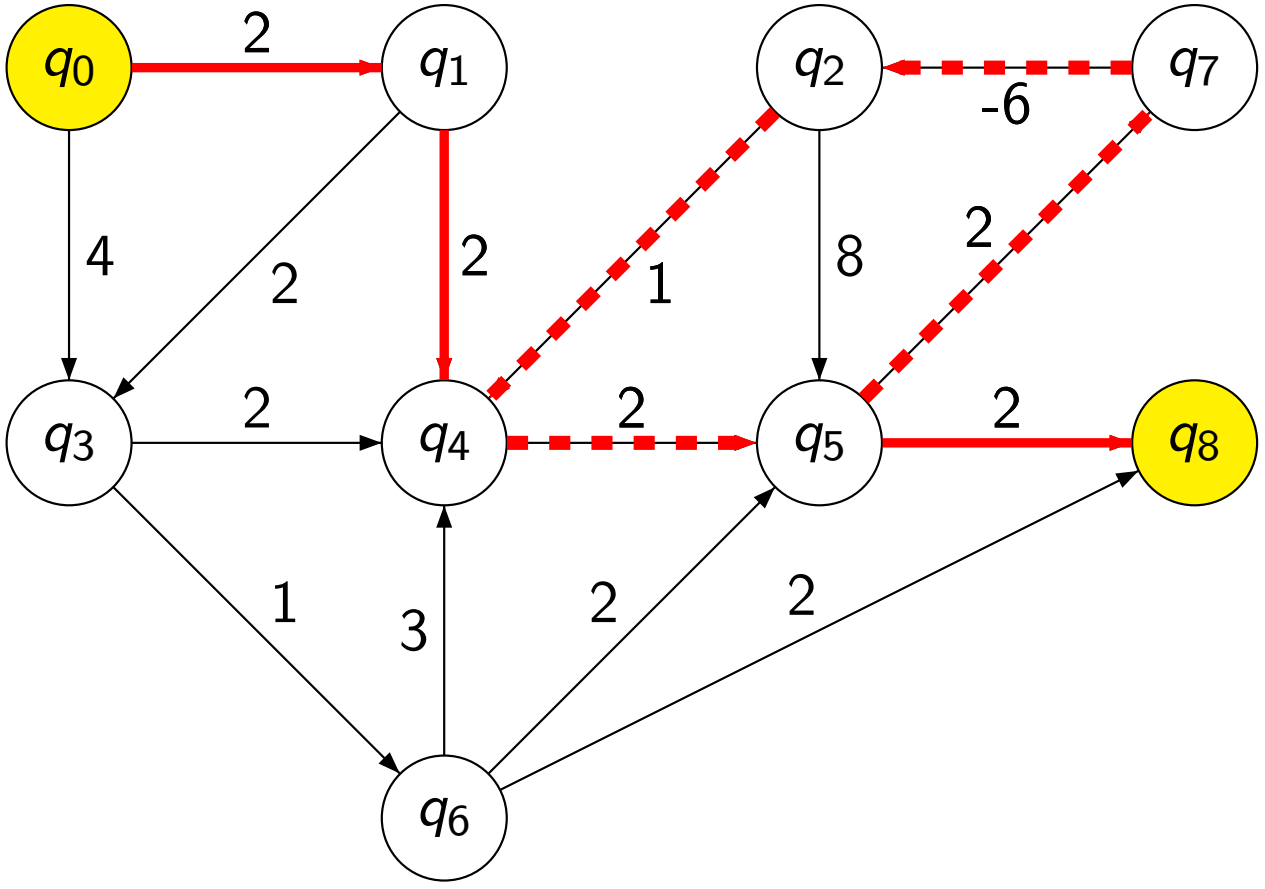
Exemple de PCC

Un PCC de q_0 à q_8 ?



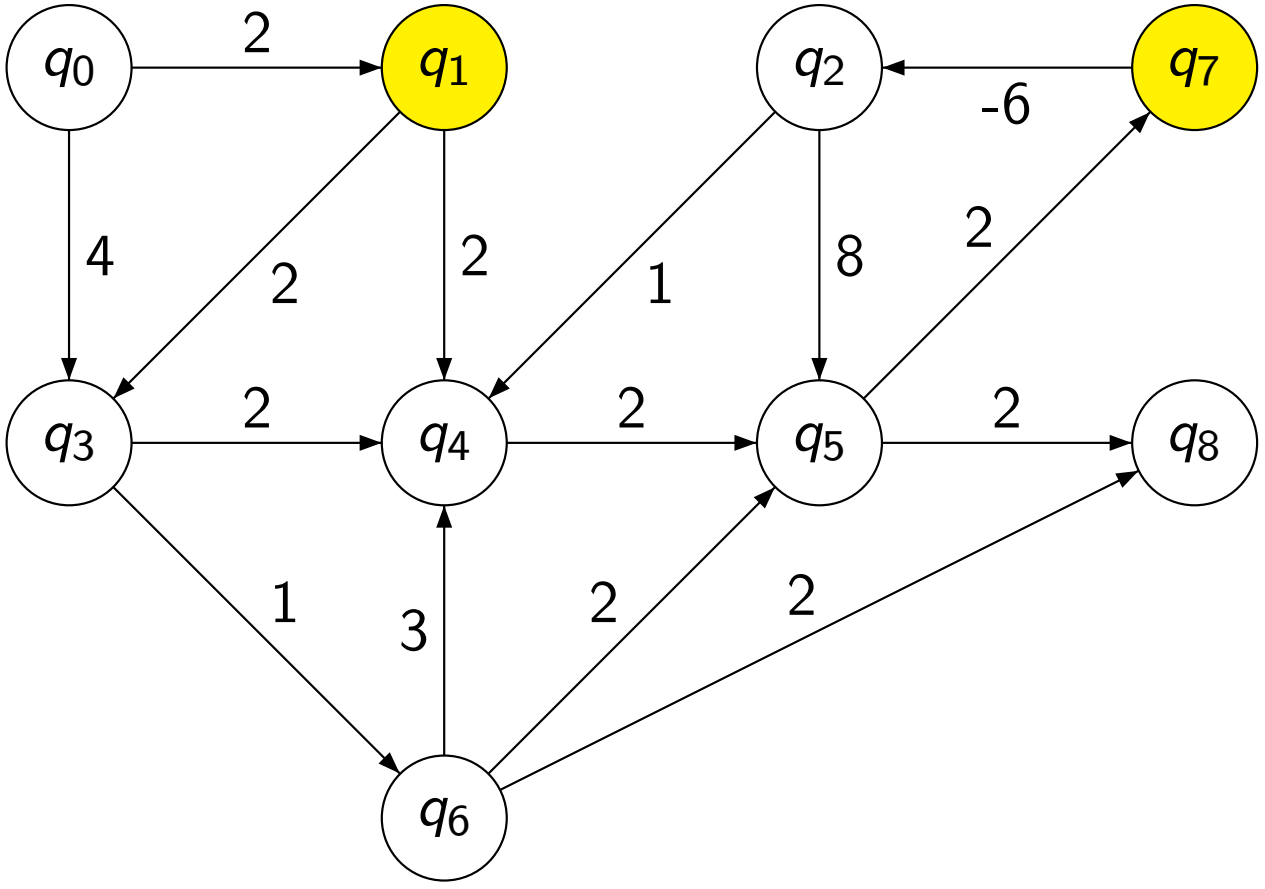
Exemple de PCC

Un PCC de q_0 à q_8 ? $-\infty$! il n'en existe pas ! !



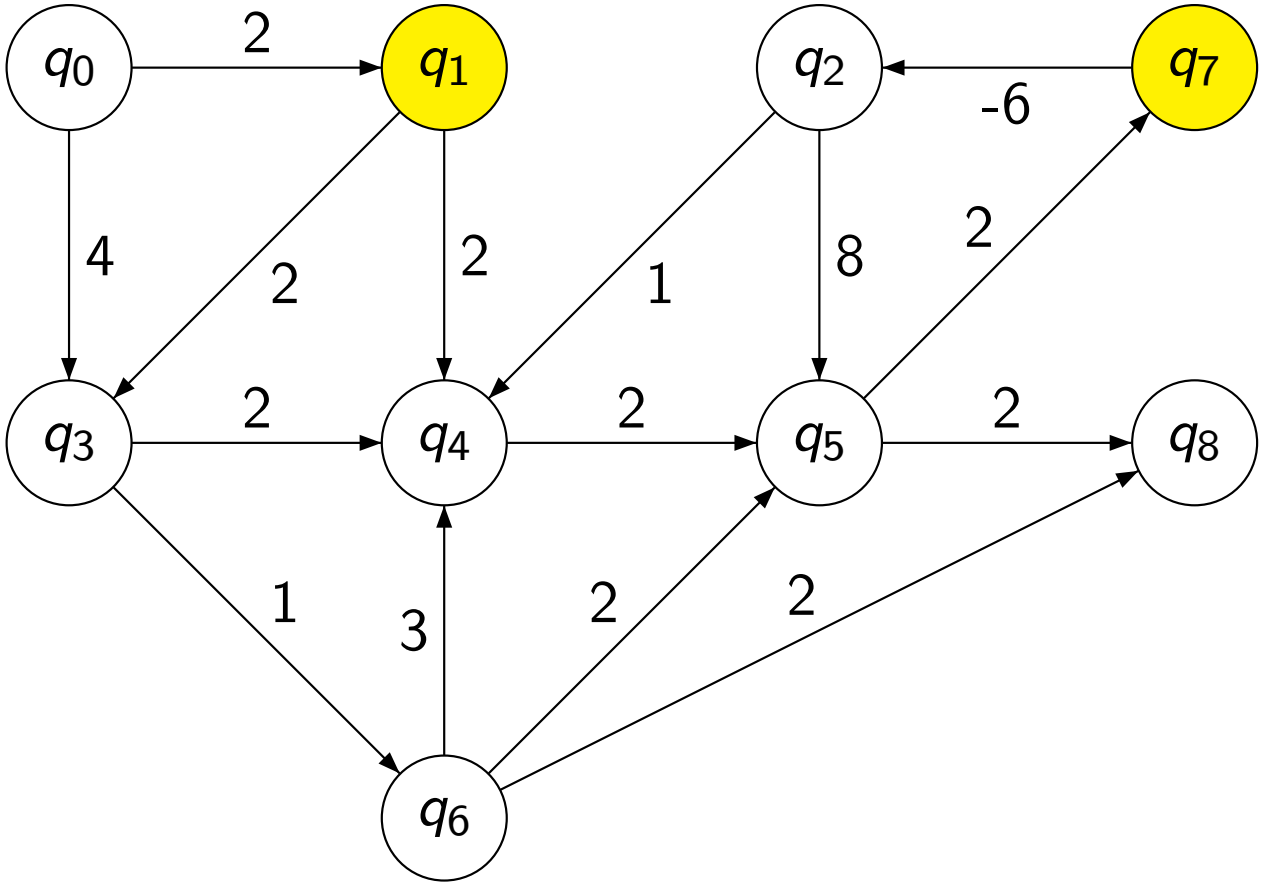
Exemple de PCC

Un PCC de q_7 à q_1 ?



Exemple de PCC

Un PCC de q_7 à q_1 ? ∞ ! il n'en existe pas ! !



Existence de PCC

Propriété :Existence d'un PCC

Il existe un PCC entre u et v ssi

(a) v est atteignable depuis u (*i.e.* $\exists u \to^* v$), et

(b) il n'existe pas de cycle strictement négatif $c : z \to^* z$ et un chemin $u \to^* z \to^* v$.

Dans la suite jusqu'à l'algorithme de Floyd-Warshall, on suppose qu'ils

Existence de PCC

Propriété :Existence d'un PCC

Il existe un PCC entre u et v ssi

(a) v est atteignable depuis u (*i.e.* $\exists u \to^* v$), et

(b) il n'existe pas de cycle strictement négatif $c : z \to^* z$ et un chemin $u \to^* z \to^* v$.

Dans la suite jusqu'à l'algorithme de Floyd-Warshall, on suppose qu'il n'existe pas de cycle strictement négatif dans le graphe G.

On définit $\delta(s,u)$ la distance d'un PCC de s à u :

$$\delta(s,u) \stackrel{\text{def}}{=} \begin{cases} \min\{w(\rho) \mid s \to_\rho u\} & \text{si } \exists\, s \to^* u \\ \infty & \text{sinon} \end{cases}$$

Propriété fondamentale des PCC

Propriété

Si $\rho : v_0 \to v_1 \to \ldots \to v_k$ est un PCC entre v_0 et v_k, alors tout sous-chemin $v_i \to \ldots \to v_j$ (avec $0 \leq i < j \leq k$) de ρ est un PCC de v_i à v_j.

Pourquoi ?

Les problèmes de PCC

- les PCC à origine unique : On cherche tous les PCC depuis un sommet de départ s ;
- les PCC à destination unique : On cherche tous les PCC menant à un sommet d'arrivée s ; et
- les PCC pour toutes les paires de sommets de G.

Les problèmes de PCC

- les PCC à origine unique : On cherche tous les PCC depuis un sommet de départ s ;
- les PCC à destination unique : On cherche tous les PCC menant à un sommet d'arrivée s ; et
- les PCC pour toutes les paires de sommets de G.

Algo. de Dijkstra = pour les PCC à origine unique.

Algo de Floyd = pour les PCC entre tous les sommets.

Plan

Algorithme de Dijkstra

Le cadre : les graphes orientés $G = (S, A, w)$ et valués avec $w : A \to \mathbb{R}_+$

(évidemment sans cycle négatif !)

Le problème :
Etant donné un sommet s, trouver la distance d'un PCC entre s et tout autre sommet $u \in S \ldots$

Algorithme de Dijkstra

Le cadre : les graphes orientés $G = (S, A, w)$ et valués avec $w : A \to \mathbb{R}_+$

(évidemment sans cycle négatif !)

Le problème :
Etant donné un sommet s, trouver la distance d'un PCC entre s et tout autre sommet $u \in S$... Et construire ces PCC !

Algorithme de Dijkstra

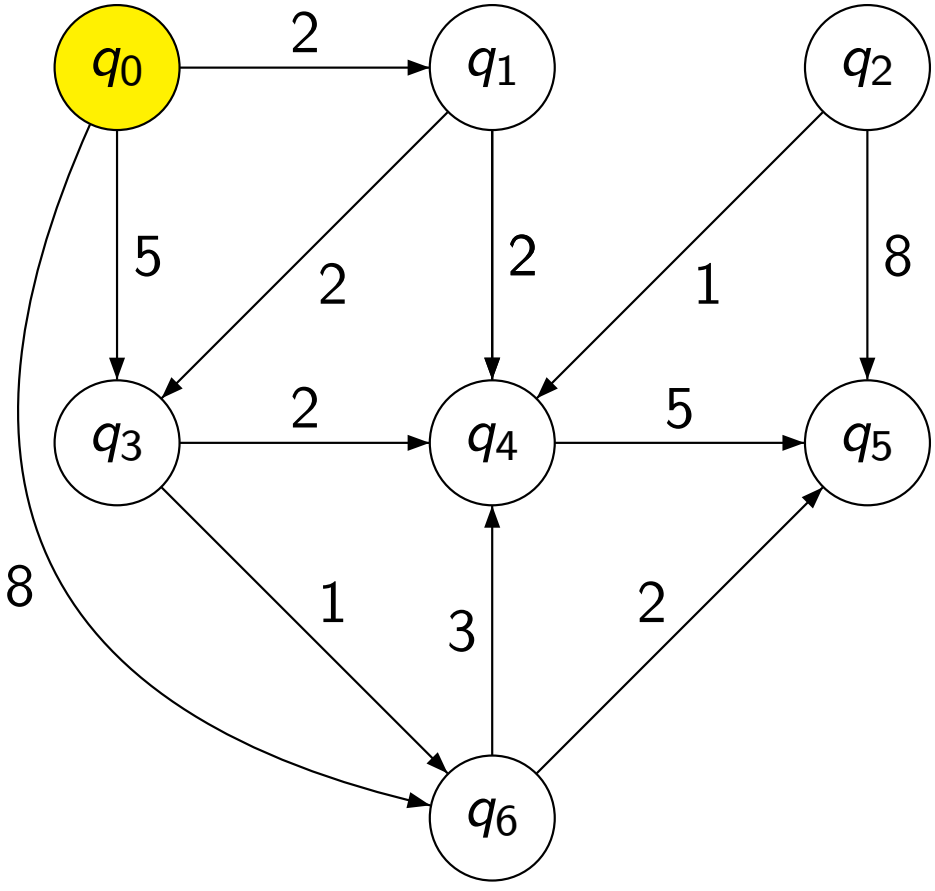
Le cadre : les graphes orientés $G = (S, A, w)$ et valués avec $w : A \to \mathbb{R}_+$

(évidemment sans cycle négatif !)

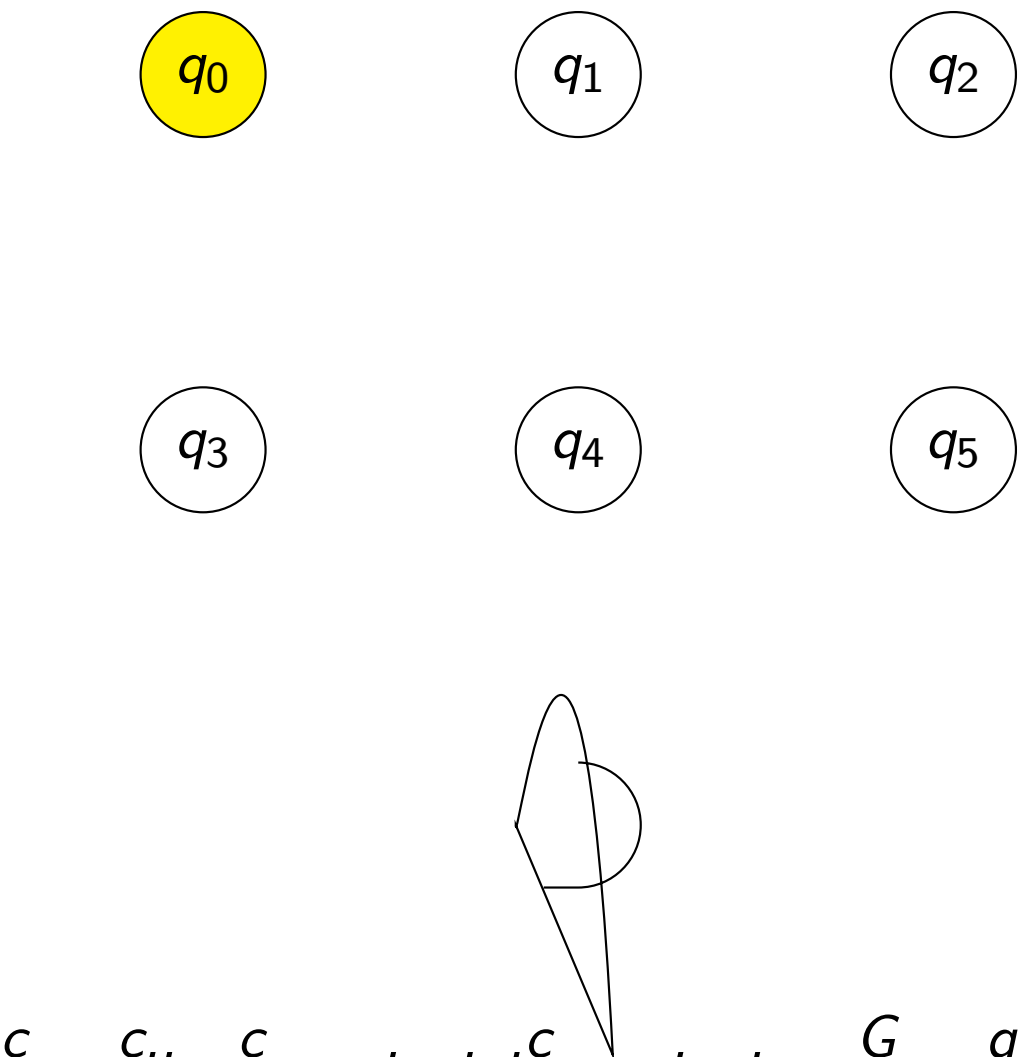
Le problème :
Etant donné un sommet s, trouver la distance d'un PCC entre s et tout autre sommet $u \in S$... Et construire ces PCC !

On va construire une arborescence des PCC $T = (S, A')$:
T est un arbre de racine s et contenant tous les sommets x accessibles depuis s et tels que : tout chemin de s à x dans T est un PCC de G.

Arborescence des PCC



Arborescence des PCC



Arborescence des PCC

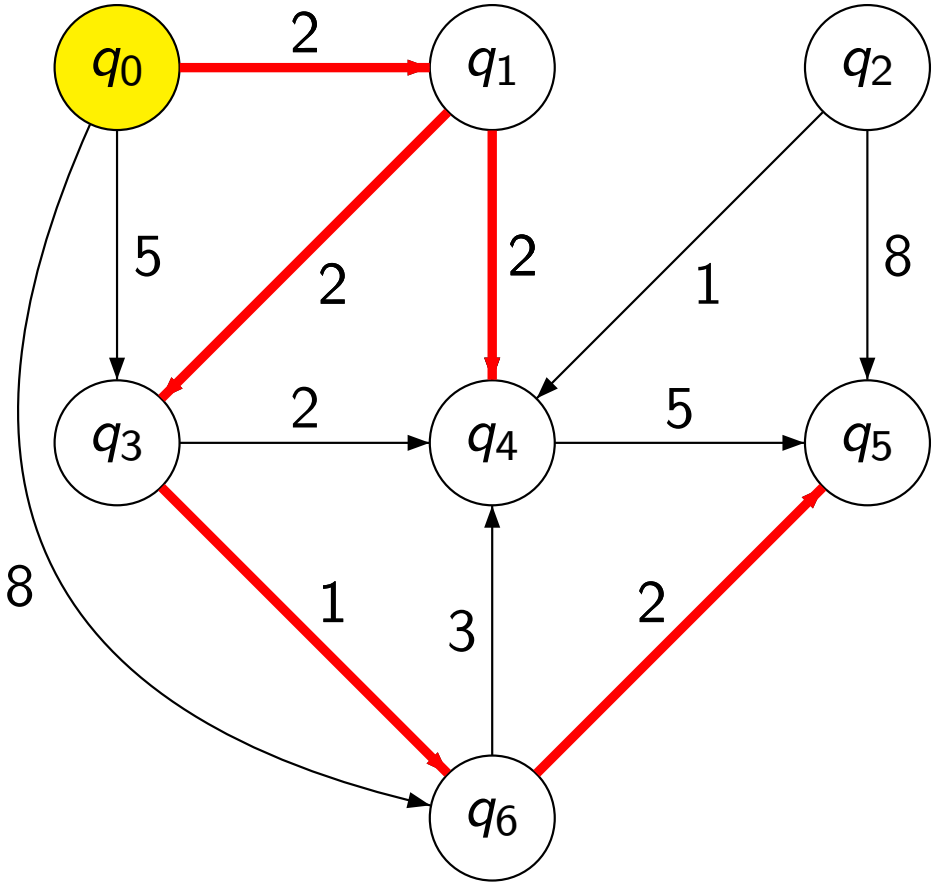


Table des prédécesseurs

$\Pi[q_0] = \text{undef}$

$\Pi[q_1] = q_0$

$\Pi[q_2] = \text{undef}$

$\Pi[q_3] = q_1$

$\Pi[q_4] = q_1$

$\Pi[q_5] = q_6$

$\Pi[q_6] = q_3$

Algorithme de Dijkstra

(Structure très proche de celle de l'algorithme de Prim)

On utilise une file de priorité F pour stocker les sommets :

la clé d[x] de x est sa distance minimale à s en ne passant que par des sommets dont on a déjà trouvé un PCC depuis s et qui ont été déjà extraits de F.

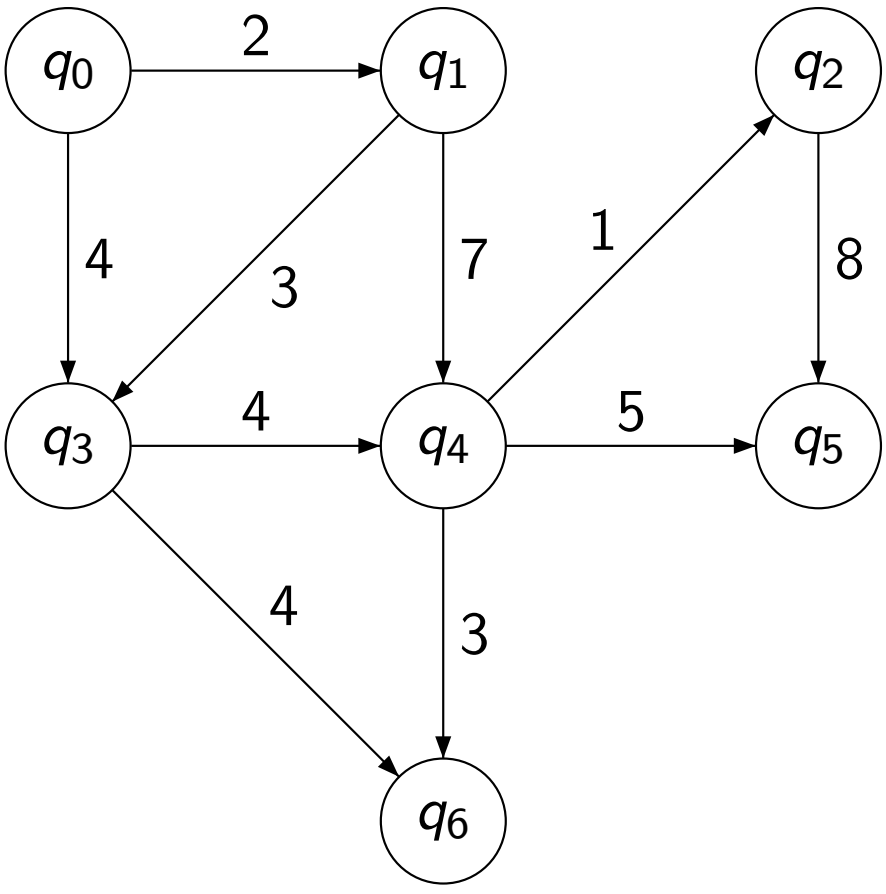
$\Rightarrow$ d[x] est une sur-approximation de $\delta(s, x)$: d[x] $\geq \delta(s, x)$

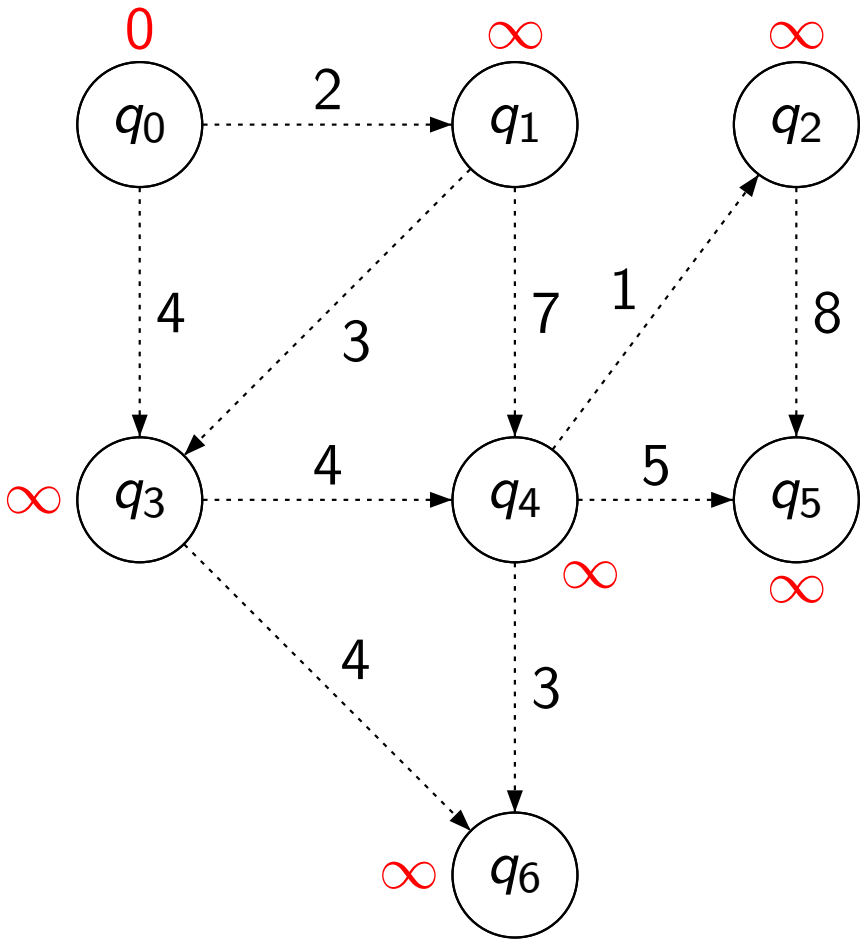
Algorithme de Dijkstra

Procédure `PCC-Dijkstra`(G, s)
pour chaque $u \in S$ **faire**
    $\Pi[u] := \text{nil}$
    $\text{d}[u] := \begin{cases} 0 & \text{si } u = s \\ \infty & \text{sinon} \end{cases}$
$F := \text{File}(S, d, \text{IndiceDansF})$
tant que $F \neq \emptyset$ **faire**
    $u :=$ Extraire-Min(F)
    $\text{IndiceDansF}[u] := -1$
    pour chaque $(u, v) \in A$ **faire**
        si $d[v] > d[u] + w(u, v)$ **alors**
            $\text{d}[v] := \text{d}[u] + w(u, v)$
            $\Pi[v] := u$
            MaJ-F-Dijkstra$(F, d, v, \text{IndiceDansF})$
return d, Π

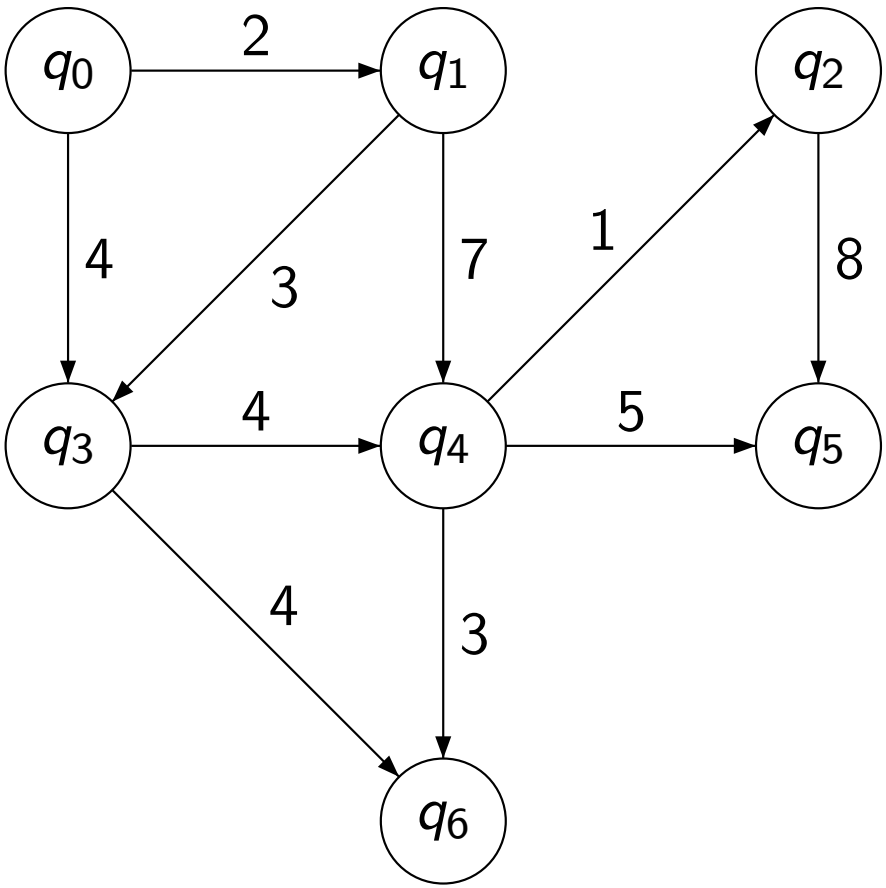
Procédure MaJ-F-Dijkstra($F, d, v,$ IndiceDansF)
begin
    //$F[i]$ désigne le i-ième sommet de F.
    $i :=$ IndiceDansF$[v]$
    tant que $(i/2 \geq 1) \wedge (d[F[i/2]] > d[F[i]])$ **faire**
        $F[i] \leftrightarrow F[i/2]$
        IndiceDansF$[F[i]] := i$
        IndiceDansF$[F[i/2]] := i/2$
        $i := i/2$;
end

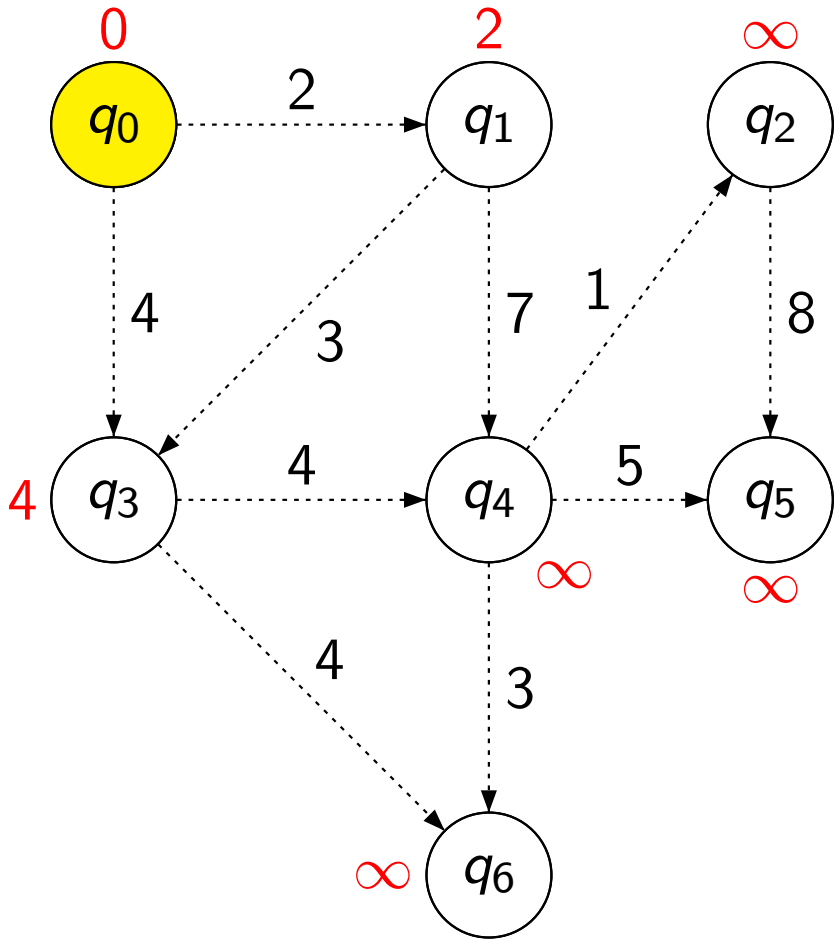
Exemple



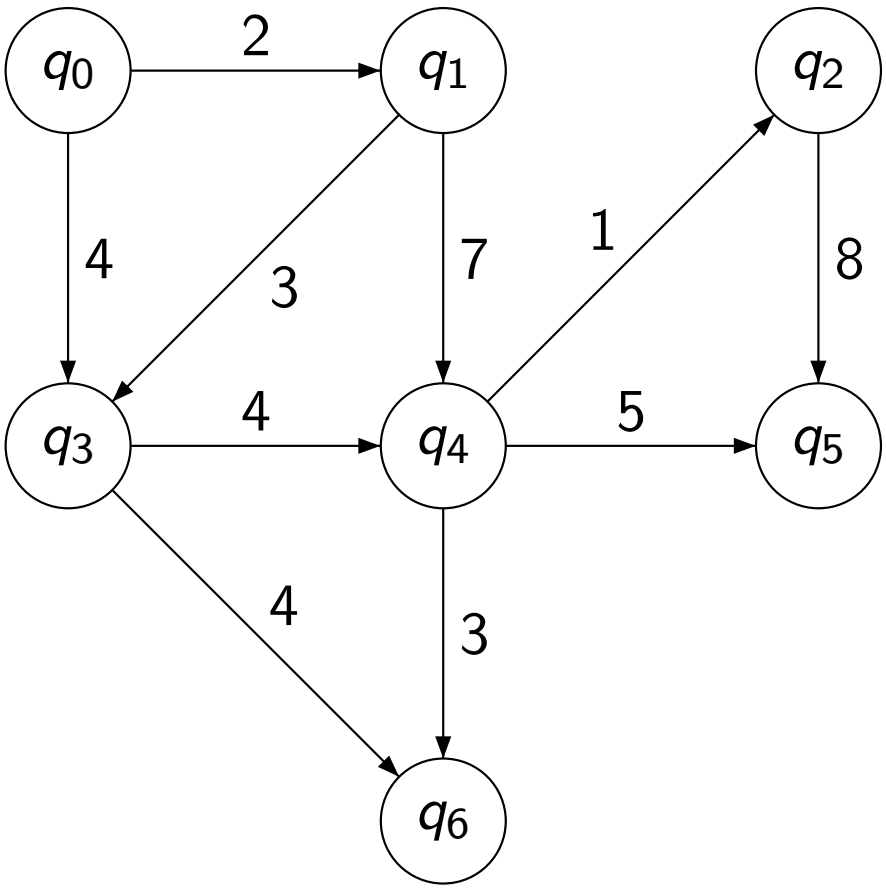


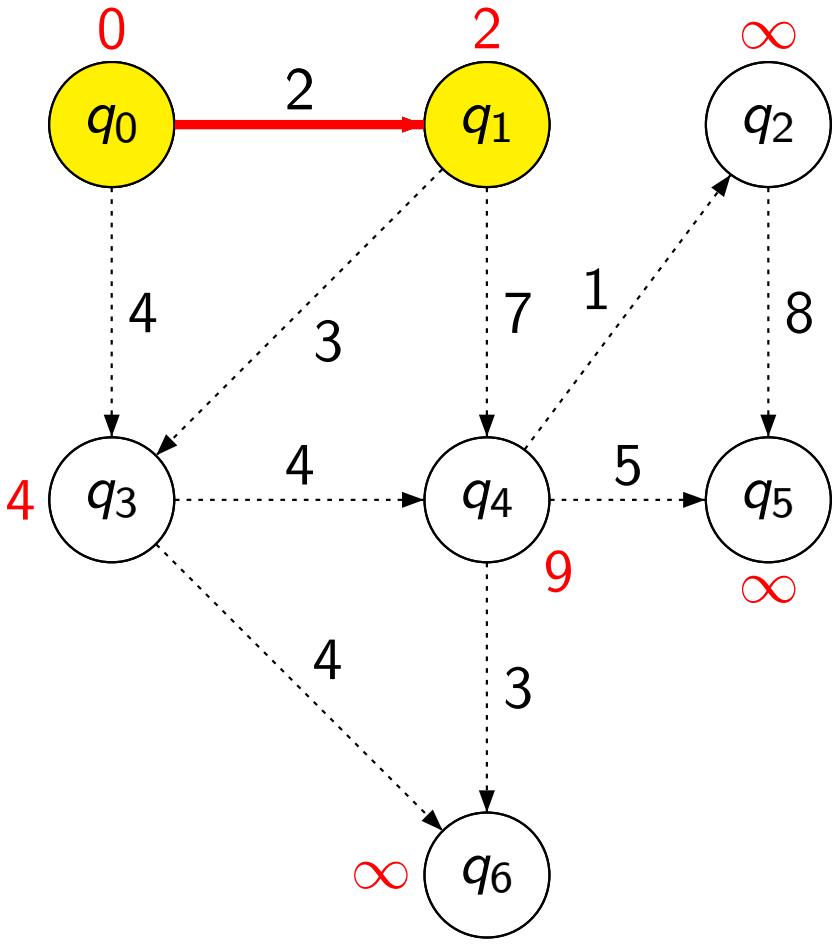
Example


q_0
q_1
q_2
q_3
q_4
q_5
q_6
2
4
3
7
1
8
4
5
4
3

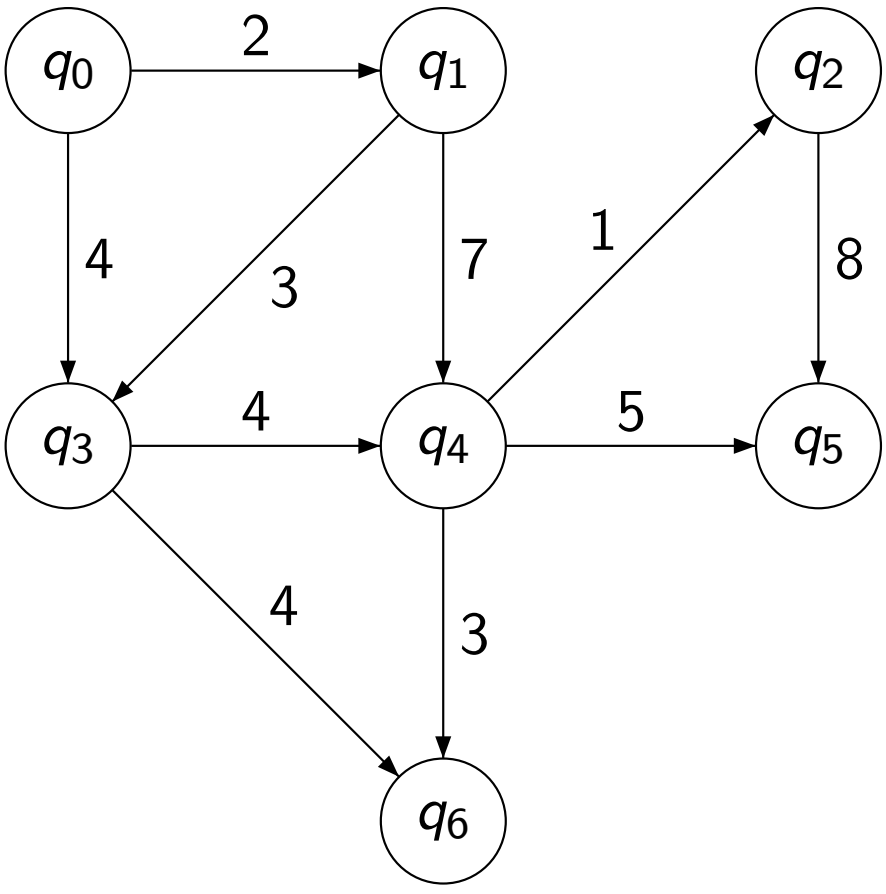

0
2
∞
q_0
q_1
q_2
4
q_3
q_4
q_5
∞
∞
∞
q_6
2
4
3
7
1
8
4
5
4
3

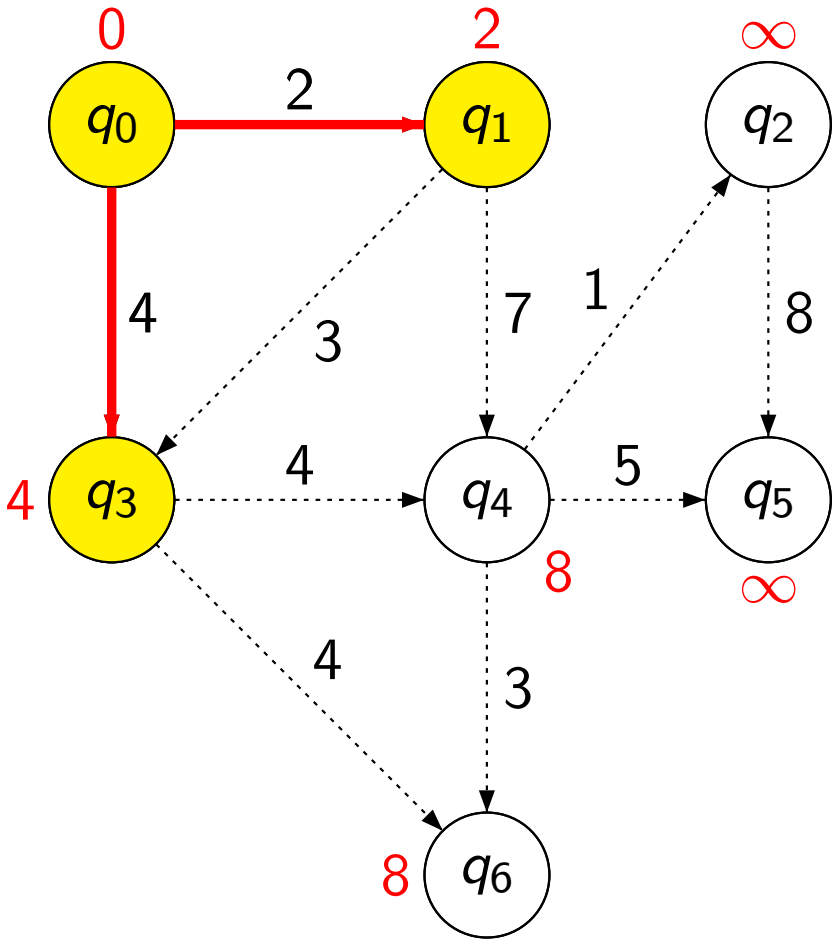
Example


q_0
q_1
q_2
q_3
q_4
q_5
q_6
2
4
3
7
1
8
4
5
4
3

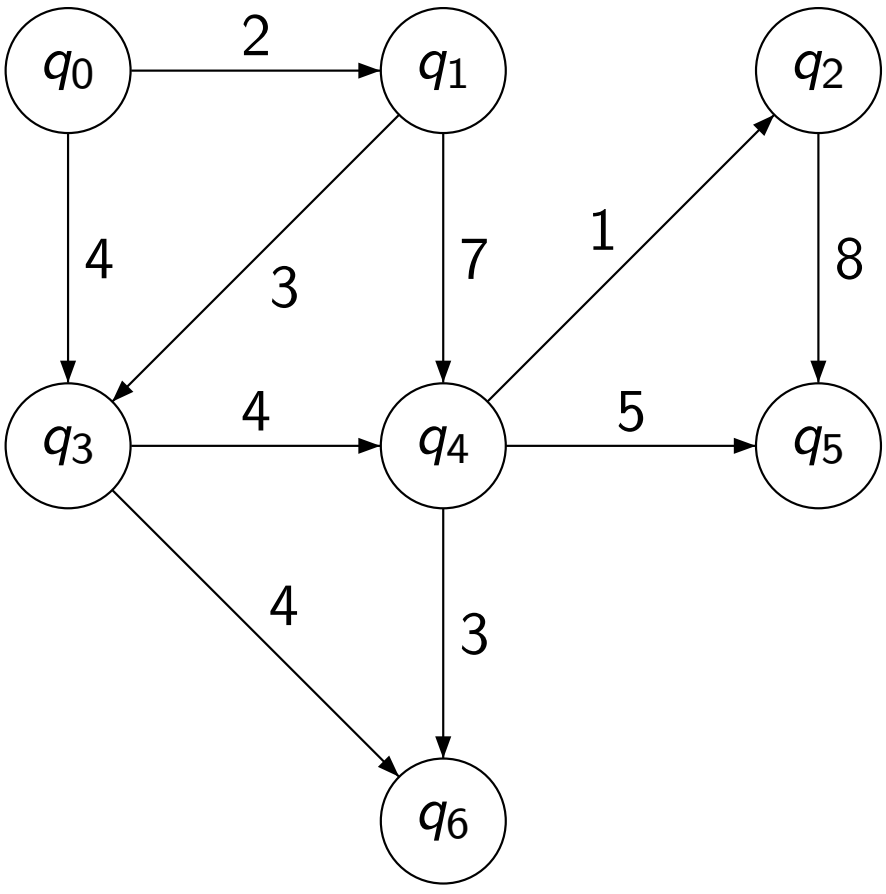

0
2
∞
q_0
q_1
q_2
4
3
7
1
8
4
q_3
4
q_4
5
q_5
9
∞
4
3
∞
q_6

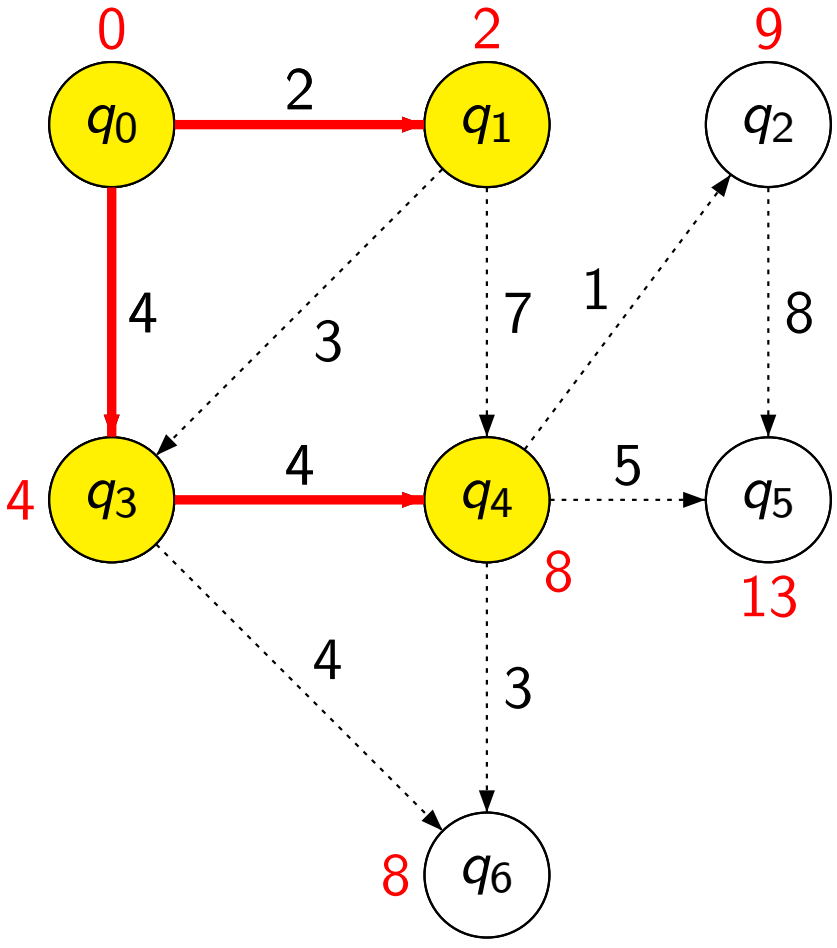
Example


q_0
q_1
q_2
q_3
q_4
q_5
q_6
2
4
3
7
1
8
4
5
4
3

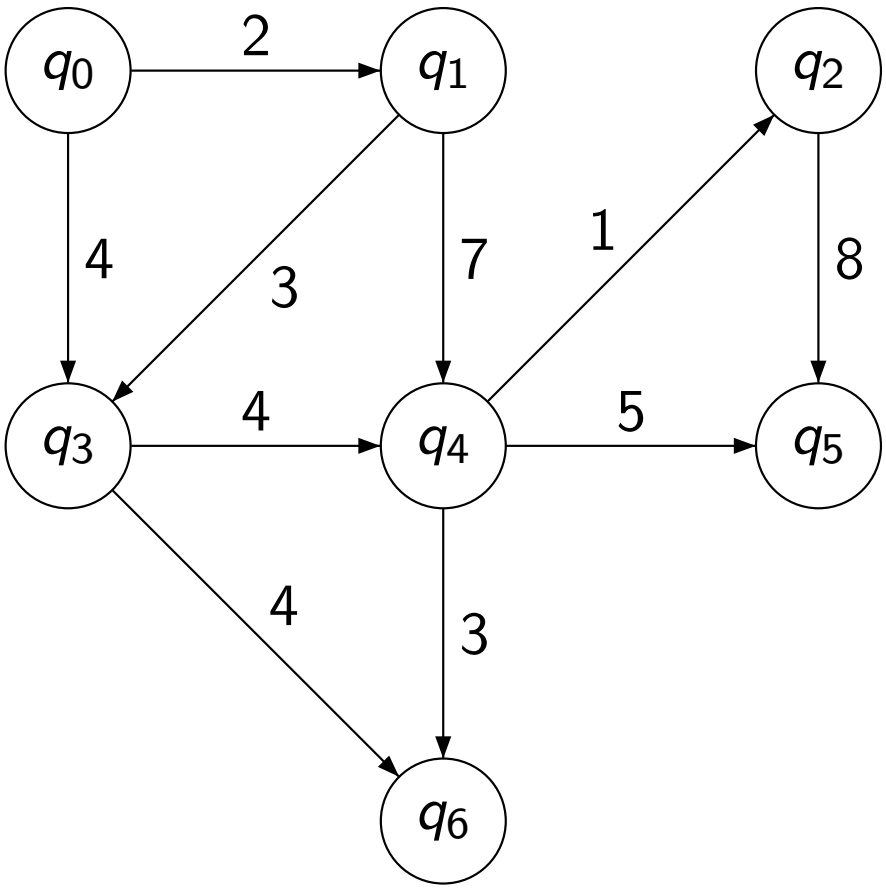

0
2
∞
q_0
q_1
q_2
4
q_3
q_4
q_5
8
∞
8
q_6
2
4
3
7
1
8
4
5
4
3

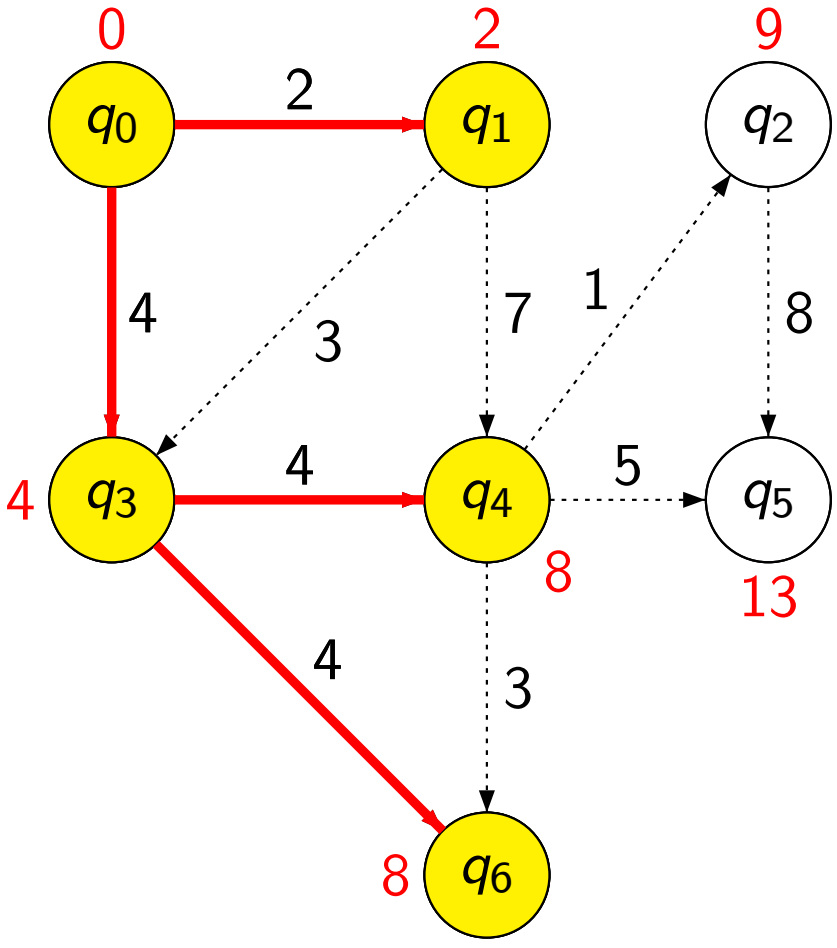
Example


q_0
q_1
q_2
q_3
q_4
q_5
q_6
2
4
3
7
1
8
4
5
4
3

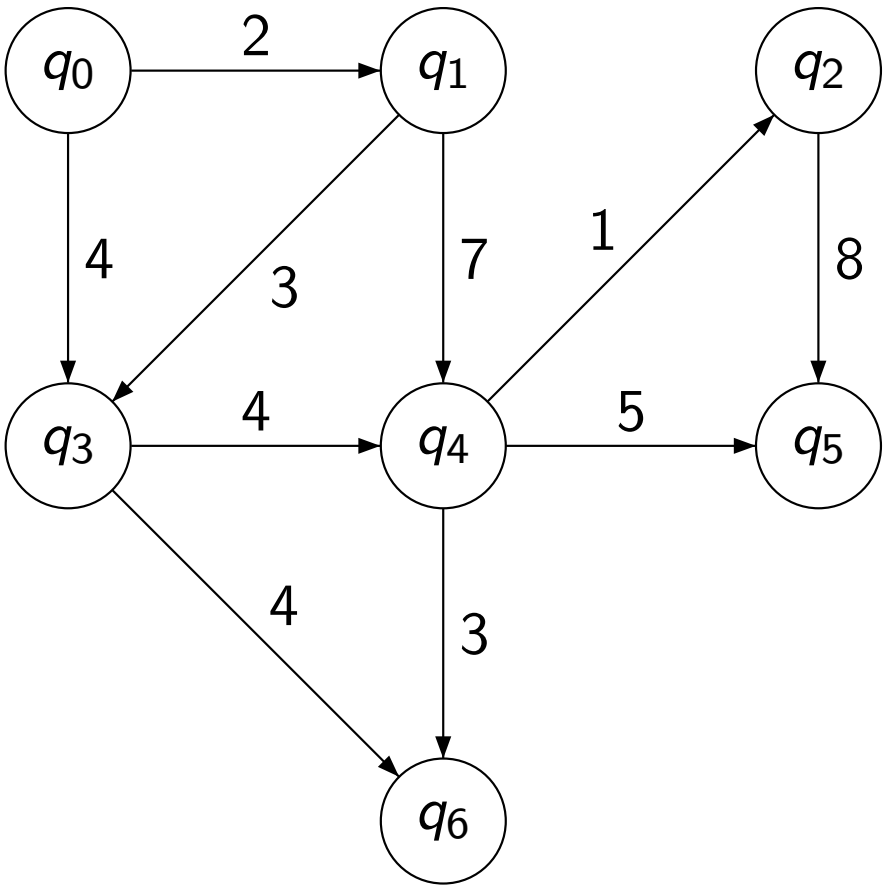

0
2
9
q_0
q_1
q_2
4
q_3
4
q_4
5
q_5
8
13
q_6
2
4
3
7
1
8
4
5
4
3

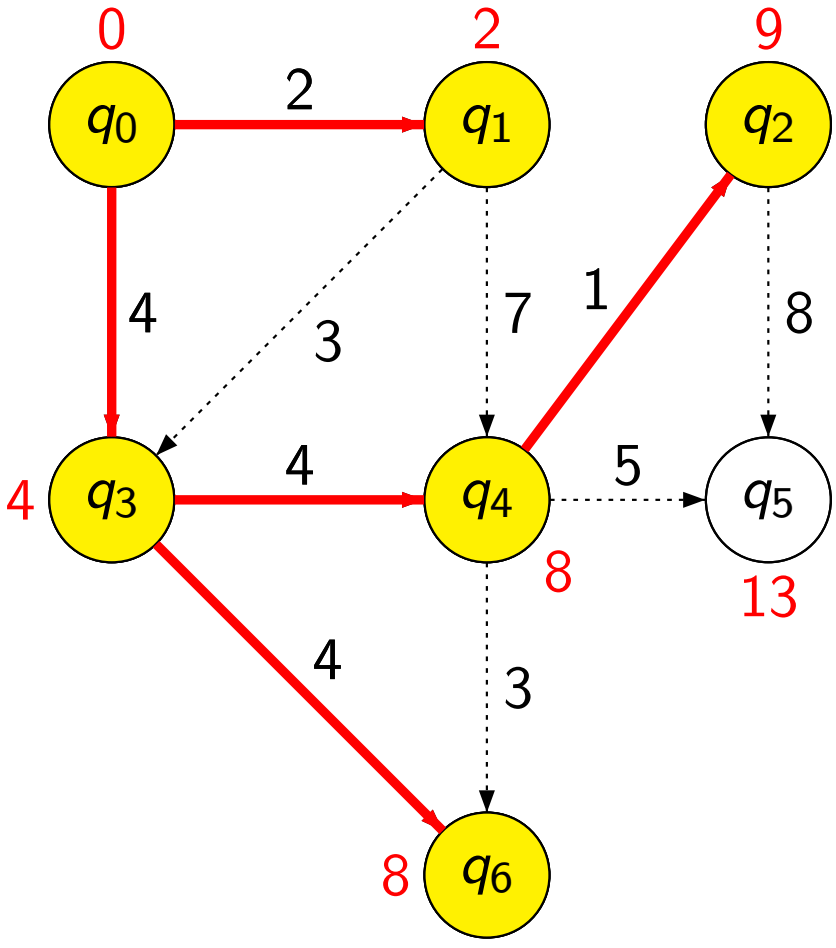
Exemple


q_0
q_1
q_2
q_3
q_4
q_5
q_6
2
4
3
7
1
8
4
5
4
3

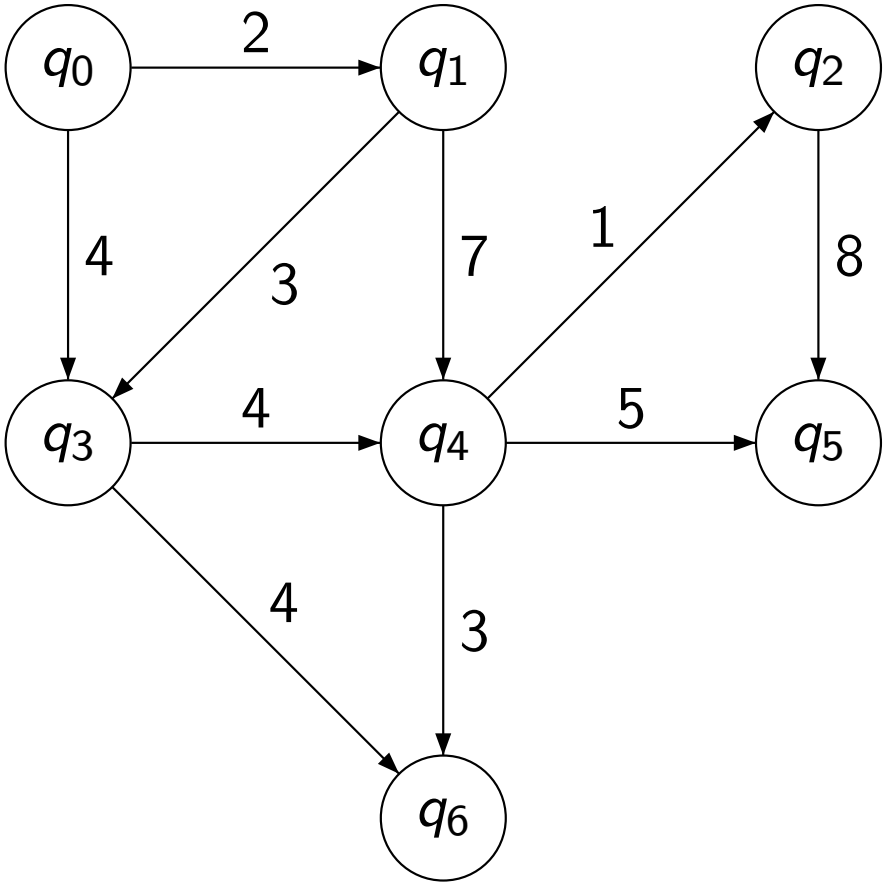

0
2
9
q_0
q_1
q_2
4
3
7
1
8
4
q_3
4
q_4
5
q_5
8
13
4
3
8
q_6

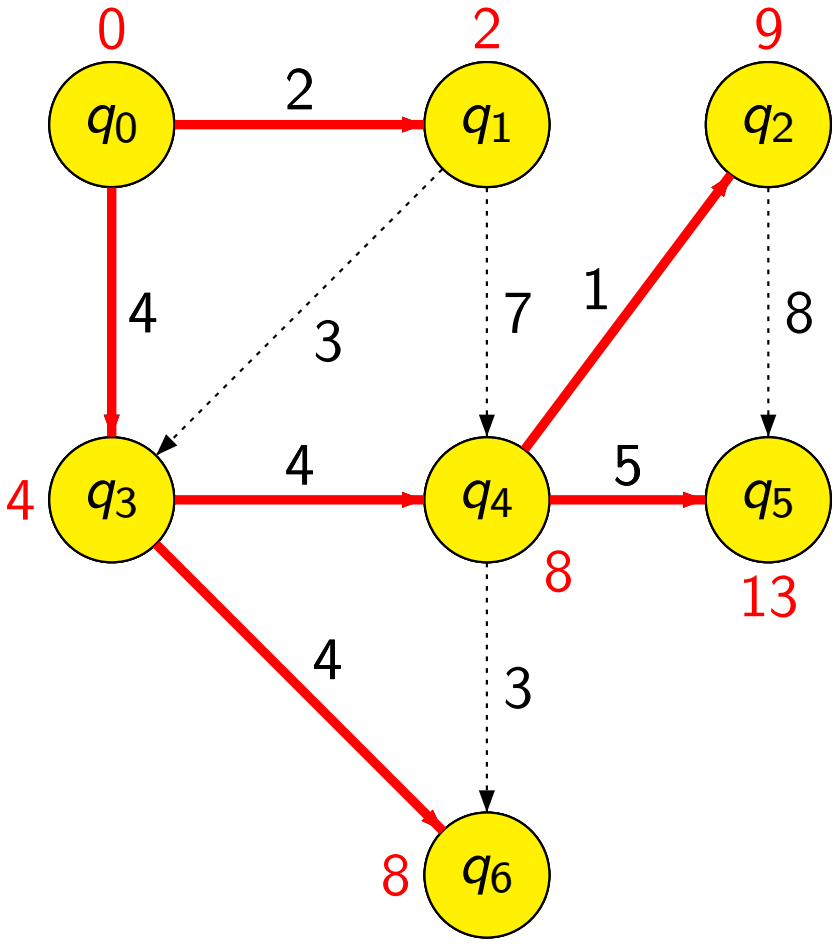
Exemple


q_0
q_1
q_2
q_3
q_4
q_5
q_6
2
4
3
7
1
8
4
5
4
3


0
2
9
q_0
q_1
q_2
4
3
7
1
8
4
q_3
4
q_4
5
q_5
8
13
4
3
8
q_6

Exemple


q_0
q_1
q_2
q_3
q_4
q_5
q_6
2
4
3
7
1
8
4
5
4
3


0
2
9
q_0
q_1
q_2
4
3
7
1
8
4
q_3
4
q_4
5
q_5
8
13
4
3
8
q_6

Propriété de l'algorithme de Dijkstra

Soit $d^i(v)$ la valeur de $d(v)$ au début de la i-ème itération de l'algo.

Propriété

- si v_j dénote le sommet extrait de F à l'itération j, et si $1 \leq j < k \leq |S|$, on a :

$$d^j[v_j] \leq d^k[v_k]$$

- Une fois extrait de F, le coefficient $d[-]$ d'un sommet n'est plus jamais modifié.

Propriété de l'algorithme de Dijkstra

Soit $d^i(v)$ la valeur de $d(v)$ au début de la i-ème itération de l'algo.

Propriété

- si v_j dénote le sommet extrait de F à l'itération j, et si $1 \leq j < k \leq |S|$, on a :

$$d^j[v_j] \leq d^k[v_k]$$

- Une fois extrait de F, le coefficient $d[-]$ d'un sommet n'est plus jamais modifié.

- On montre que l'on a $d^i[v_i] \leq d^{i+1}[v_{i+1}]$ par induction sur le numéro d'itération.
- Après l'extraction de v_i, toute modification à l'itération $j > i$ est conditionnée par $d^j[v_j] + w(v_j, v_i) < d^i[v_i]$...

Propriété de l'algorithme de Dijkstra

Propriété A tout moment de l'algorithme, on a $d[u] \geq \delta(s, u)$

c'est vrai lors de l'initialisation et c'est clairement maintenu à chaque itération...

Propriete de l'algorithme de Dijkstra

Propriété A tout moment de l'algorithme, on a $d[u] \geq \delta(s, u)$

c'est vrai lors de l'initialisation et c'est clairement maintenu à chaque itération...

Théorème :correction de l'algorithme de Dijkstra

$G = (S, A, w)$ orienté et valué tel que $w : A \to \mathbb{R}_+$.
L'algorithme de Dijkstra

1. termine,
2. à la fin, on a $d[u] = \delta(s, u)$ pour tout sommet $u \in S$, et
3. $\forall u \in S \backslash \{s\}$, si $d[u] < \infty$, alors il existe un PCC de s à u dont le dernier arc est $(\Pi[u], u)$.

L'algorithme PCC-Dijkstra prend un temps en $O\Big((|S| + |A|) \cdot \log(|S|)\Big)\ldots$

c'est à dire en $O(|A| \cdot \log(|S|))$ lorsqu'on suppose $|S| \leq |A|$.

Plan

Le problème

Objectif : calculer les PCC entre tous les sommets...

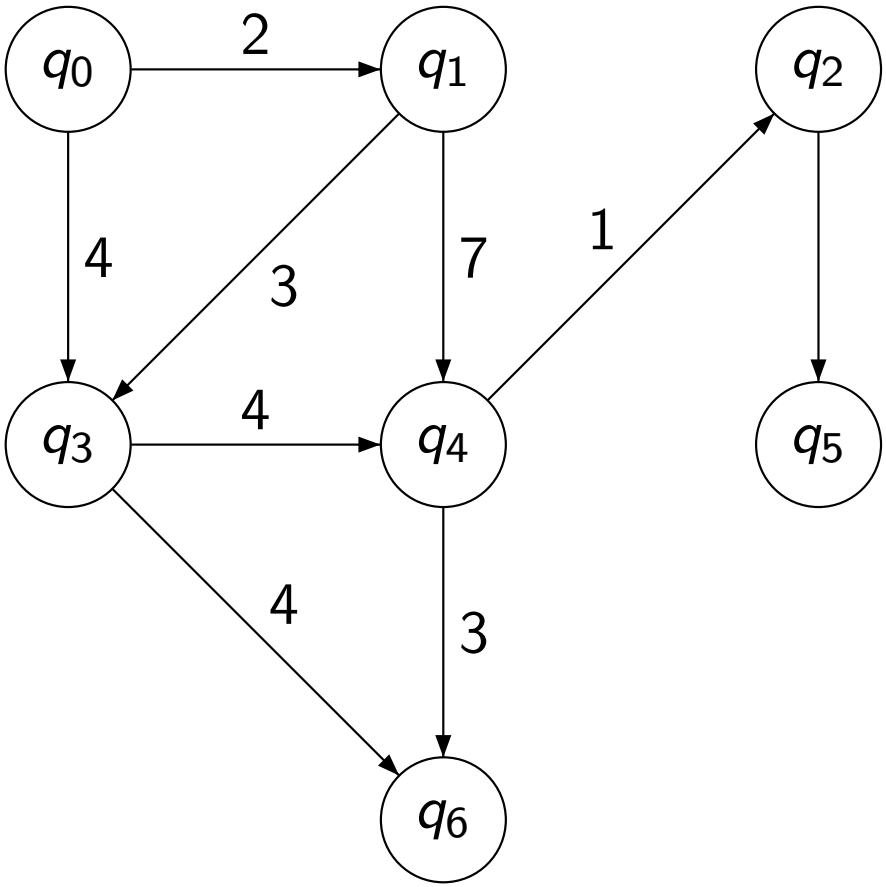
On suppose que $G = (S, A, w)$ est donné sous forme matricielle : (M_G, w)

- $S = \{x_1, \dots x_n\}$
- $w : A \to \mathbb{R}$
- $M_G = (\alpha_{ij})_{1 \leq i,j \leq n}$: α_{ij} décrit l'arc entre x_i et x_j.

$$\alpha_{ij} \overset{\text{def}}{=} \begin{cases} 0 & \text{si } i = j \\ w(x_i, x_j) & \text{si } i \neq j \text{ et } (x_i, x_j) \in A \\ \infty & \text{si } i \neq j \text{ et } (x_i, x_j) \notin A \end{cases}$$

Dans la suite, on notera δ_{ij} la distance $\delta(x_i, x_j)$ d'un PCC entre x_i et x_j.

Exemple



L'arborescence des PCC?

Pour représenter les PCC entre tous les sommets, on utilise une fonction $\ll$ prédécesseur $\gg$ Π.

Π est une matrice de taille $n \times n$ à valeurs dans S.
Ses coefficients sont les π

Programmation dynamique

On utilise une famille de matrices $n \times n$ $D^{(k)}$ $(k = 0, \ldots n)$ contenant les calculs intermédiaires.

Idée : $D^{(k)}[i, j]$ est la longueur d'un PCC entre x_i et x_j ne passant que par des états dans $\{x_1, \ldots, x_k\}$.

Programmation dynamique

On utilise une famille de matrices $n \times n$ $D^{(k)}$ $(k = 0, \ldots n)$ contenant les calculs intermédiaires.

Idée : $D^{(k)}[i,j]$ est la longueur d'un PCC entre x_i et x_j ne passant que par des états dans $\{x_1, \ldots, x_k\}$.

$$D^{(k)}[i,j] \stackrel{\text{def}}{=} \min\left(D^{(k-1)}[i,j] \; ; \; D^{(k-1)}[i,k] + D^{(k-1)}[k,j]\right)$$

Algorithme de Floyd-Warshall

Procédure `PCC-Floyd`(G)

pour $i = 1 \dots n$ **faire**

  pour $j = 1 \dots n$ **faire**

    $d_{ij}^{(0)} := \alpha_{ij}$

    si $\alpha_{ij} \neq \infty$ **alors** $\pi_{ij}^{(0)} := i$

pour $k = 1 \dots n$ **faire**

  pour $i = 1 \dots n$ **faire**

    pour $j = 1 \dots n$ **faire**

      si $d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ **alors**

        $d_{ij}^{(k)} := d_{ij}^{(k-1)}$

        $\pi_{ij}^{(k)} := \pi_{ij}^{(k-1)}$

      sinon

        $d_{ij}^{(k)} := d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

        $\pi_{ij}^{(k)} := \pi_{kj}^{(k-1)}$

Propriétés de l'algorithme

Soit $\rho : v_0 \to v_1 \to \ldots \to v_l$, l'intérieur de ρ est $\{v_1, \ldots, v_{k-1}\}$.

Propriété

Si G ne contient pas de cycle strictement négatif, alors pour tout $k = 0, \ldots, n$, on a :

- $d_{ij}^{(k)}$ est la distance d'un PCC entre x_i et x_j et d'intérieur inclu dans $\{x_1, \ldots, x_k\}$;
- $\pi_{ij}^{(k)}$ est le prédécesseur de x_j le long de ce PCC de x_i à x_j.

Par induction sur k.

Correction

Théorème : correction de l'algorithme de Floyd-Warshall Si G ne contient pas de cycle strictement négatif, alors $D^{(n)}$ contient les coefficients δ_{ij} des PCC.

Correction

Théorème : correction de l'algorithme de Floyd-Warshall Si G ne contient pas de cycle strictement négatif, alors $D^{(n)}$ contient les coefficients δ_{ij} des PCC.

Propriété G contient un cycle strictement négatif ssi il existe un coefficient $d_{ii}^{(n)}$ strictement négatif.

Algorithme simple

Procédure `PCC-Floyd`(G)

`//On initialise` D `avec` M`:`

pour $i = 1 \dots n$ **faire**
- **pour** $j = 1 \dots n$ **faire**
  - $d_{ij} := \alpha_{ij}$
  - **si** $\alpha_{ij} \neq \infty$ **alors** $\pi_{ij} := i$

pour $k = 1 \dots n$ **faire**
- **pour** $i = 1 \dots n$ **faire**
  - **pour** $j = 1 \dots n$ **faire**
    - **si** $d_{ij} > d_{ik} + d_{kj}$ **alors**
      - $d_{ij} := d_{ik} + d_{kj}$
      - $\pi_{ij} := \pi_{kj}$

return D, Π

Complexité de l'algorithme de Floyd-Warshall

$O(n^3)$!

Construction des PCC

Procédure `Construire-PCC`(Π, i, j)
`//`Π `: une matrice de prédécesseurs.`
`//`$1 \leq i, j \leq n$ `: deux indices de sommets`
begin
  si $i \neq j$ **alors**
    return *Construire-PCC*$(\Pi, i, \Pi[i, j]) \oplus (\Pi[i, j], j)$
end