M1 Info - Graphes et programmation dynamique

Cours 2 - Plus courts chemins a origine unique

BFS et Dijkstra

Semestre Automne 2022-2023 - Université Côte D'azur

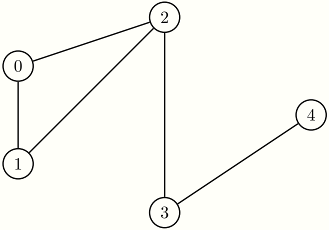
Christophe Crespelle

christophe.crespelle@univ-cotedazur.fr

UNIVERSITÉ CÔTE D'AZUR

DIGITAL SYSTEMS FOR HUMANS
GRADUATE SCHOOL AND RESEARCH

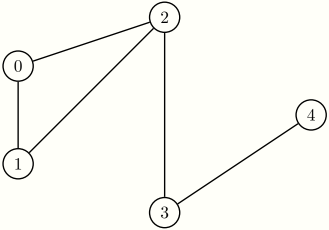
Representation des graphes en memoire



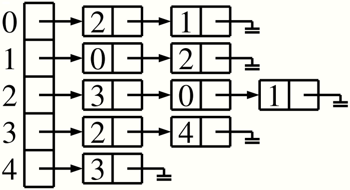
- Matrice d'adjacence

	0	**1**	**2**	**3**	**4**
0	0	1	1	0	0
1	1	0	1	0	0
2	1	1	0	1	0
3	0	0	1	0	1
4	0	0	0	1	0

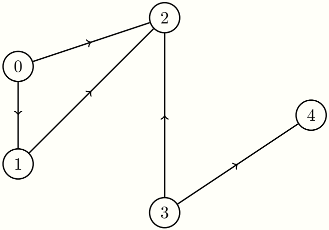
Representation des graphes en memoire



- Listes d'adjacence



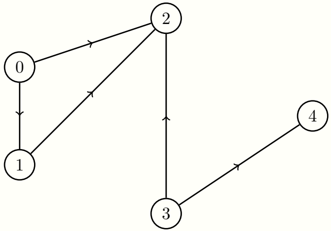
Representation des graphes orientes en memoire



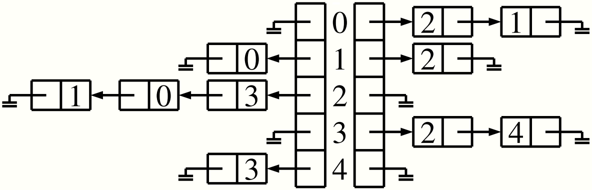
- Matrice d'adjacence

	0	**1**	**2**	**3**	**4**
0	0	1	1	0	0
1	0	0	1	0	0
2	0	0	0	0	0
3	0	0	1	0	1
4	0	0	0	0	0

Representation des graphes orientes en memoire



- Listes d'adjacence



Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?

Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$

Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$
    - Listes : $O(min\{d(u), d(v)\})$

Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$
    - Listes : $O(min\{d(u), d(v)\})$
  - Requete de voisinage : tous les voisins de u ?

Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$
    - Listes : $O(min\{d(u), d(v)\})$
  - Requete de voisinage : tous les voisins de u ?
    - Matrice : $O(n)$

Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$
    - Listes : $O(min\{d(u), d(v)\})$
  - Requete de voisinage : tous les voisins de u ?
    - Matrice : $O(n)$
    - Listes : $O(d(u))$

Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$
    - Listes : $O(min\{d(u), d(v)\})$
  - Requete de voisinage : tous les voisins de u ?
    - Matrice : $O(n)$
    - Listes : $O(d(u))$
- En espace

Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$
    - Listes : $O(min\{d(u), d(v)\})$
  - Requete de voisinage : tous les voisins de u ?
    - Matrice : $O(n)$
    - Listes : $O(d(u))$
- En espace
  - Matrice : $O(n^2)$

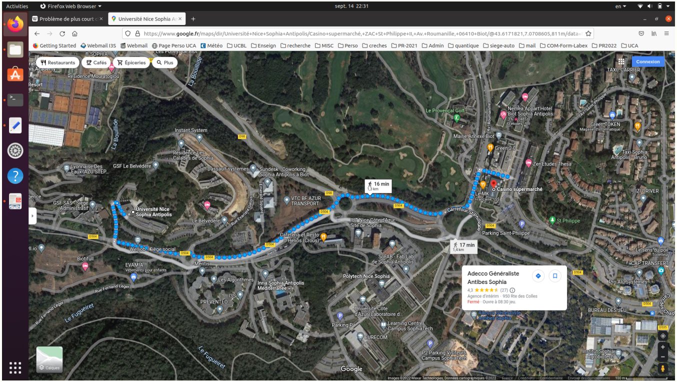
Comparaison des performances des deux representations

Un graphe G a n sommets et m aretes

- En temps
  - Requete d'adjacence : u et v sont-ils adjacents ?
    - Matrice : $O(1)$
    - Listes : $O(min\{d(u), d(v)\})$
  - Requete de voisinage : tous les voisins de u ?
    - Matrice : $O(n)$
    - Listes : $O(d(u))$
- En espace
  - Matrice : $O(n^2)$
  - Listes : $O(n + m)$

Plus court chemin dans les graphes

- **Entrée :** un graphe $G = (V, E)$, deux sommets $s, t \in V$



- **Sortie :** un plus court chemin de s a t dans G

Graphes non orientes et non ponderes

Plus courts chemins a origine unique

Notion de distance dans les graphes

- **Cadre :** graphes non-orientes et non-ponderes

Plus courts chemins a origine unique

Notion de distance dans les graphes

- **Cadre :** graphes non-orientes et non-ponderes
- **Longueur d'un chemin C entre u et v :** nombre d'aretes sur le chemin C

Plus courts chemins a origine unique

Notion de distance dans les graphes

- **Cadre :** graphes non-orientes et non-ponderes
- **Longueur d'un chemin C entre u et v :** nombre d'aretes sur le chemin C
- **Distance $dist(u, v)$ entre u et v :** longueur du plus court chemin entre u et v

Plus courts chemins a origine unique

Notion de distance dans les graphes

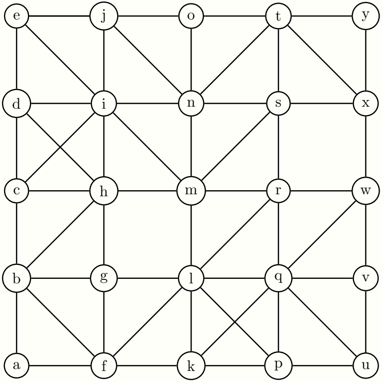
- **Cadre :** graphes non-orientes et non-ponderes
- **Longueur d'un chemin C entre u et v :** nombre d'aretes sur le chemin C
- **Distance $dist(u, v)$ entre u et v :** longueur du plus court chemin entre u et v

Probleme des plus courts chemins a origine unique

- **Entrée :** un graphe $G = (V, E)$ non oriente et non pondere, un sommet $s \in V$

- **Sortie :** la distance $dist(s, u)$ pour tout les sommets $u \in V$

Exemple



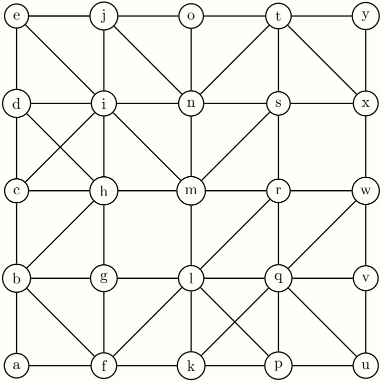
- Representation (en memoire) de la sortie ?

Algorithme : parcours en largeur (BFS)

Algorithme 1 : Plus courts chemins a origine unique **BFS(G,s)**

```
Un tableau dist de taille n ; F ← ∅ une file;
pour u de 0 a n − 1 faire
    dist[u] ← +∞ ; coul[u] ← blanc;
fin
dist[s] ← 0 ; enfiler(s, F) ; coul[s] = gris;
tant que F ≠ ∅ faire
    u ← premier(F);
    pour v ∈ N(u) faire
        si coul[v] = blanc alors
            dist[v] ← dist[u] + 1 ; enfiler(v, F) ; coul[v] = gris;
        fin
    fin
    defiler(F);
fin
```

Exemple d'execution de BFS(G,s)



Preuve de correction de l'algorithme

Theoreme : A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ entre u et s.

Preuve de correction de l'algorithme

<u>Theoreme :</u> A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ entre u et s.

<u>Propriete 1 :</u> La distance $dist[v]$ est initialement $+\infty$ et si v est accessible depuis s alors $dist[v]$ devient finie au cours de l'algorithme.

Preuve de correction de l'algorithme

Theoreme : A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ entre u et s.

Propriete 1 : La distance $dist[v]$ est initialement $+\infty$ et si v est accessible depuis s alors $dist[v]$ devient finie au cours de l'algorithme.

Propriete 2 : Lorsque $dist[v] \leftarrow dist[u] + 1$ a la ligne 10, v est accessible depuis s par un chemin de longueur $dist[v]$.

Preuve de correction de l'algorithme

<u>Theoreme :</u> A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ entre u et s.

<u>Propriete 1 :</u> La distance $dist[v]$ est initialement $+\infty$ et si v est accessible depuis s alors $dist[v]$ devient finie au cours de l'algorithme.

<u>Propriete 2 :</u> Lorsque $dist[v] \leftarrow dist[u] + 1$ a la ligne 10, v est accessible depuis s par un chemin de longueur $dist[v]$.

<u>Corollaire :</u> A la fin de l'algorithme, pour tout u, on a $dist[u]$ est finie ssi u est accessible depuis s, et on a de plus $dist(s, u) \leq dist[u]$.

Preuve de correction de l'algorithme

Propriete 3 : Au cours de l'algorithme, la valeur $dist[v]$ du sommet v qui est enfiler dans F (ligne 10) est croissante. De plus, a chaque instant de l'algorithme la difference entre les valeur $dist[.]$ associee au premier *prem* et du dernier *dern* sommet de F est au plus 1 : $|dist[prem] - dist[dern]| \leq 1$.

Preuve de correction de l'algorithme

<u>Theoreme :</u> A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ entre u et s.

Démonstration.

Considerons la premiere fois au cours de l'algorithme que la distance affectee a un sommet v (ligne 10) n'est pas correcte et soit u le sommet affecte a la ligne 7 et qui a decouvert v.

□

Preuve de correction de l'algorithme

Theoreme : A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ entre u et s.

Démonstration.

Considerons la premiere fois au cours de l'algorithme que la distance affectee a un sommet v (ligne 10) n'est pas correcte et soit u le sommet affecte a la ligne 7 et qui a decouvert v.

On a a la fois $dist[v] > dist(s, v)$, d'apres le corollaire precedent, et $dist[u] = dist(s, u)$ par definition de v.

□

Preuve de correction de l'algorithme

<u>Theoreme :</u> A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s,u)$ entre u et s.

Démonstration.

Considerons la premiere fois au cours de l'algorithme que la distance affectee a un sommet v (ligne 10) n'est pas correcte et soit u le sommet affecte a la ligne 7 et qui a decouvert v.

On a a la fois $dist[v] > dist(s,v)$, d'apres le corollaire precedent, et $dist[u] = dist(s,u)$ par definition de v.

Considerons un plus court chemin de s a v, et notons u' le precedent de v sur ce chemin. On a $dist(s,u') = dist(s,v) - 1 < dist[v] - 1 = dist[u] = dist(s,u)$.

□

Preuve de correction de l'algorithme

Theoreme : A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ entre u et s.

Démonstration.
Considerons la premiere fois au cours de l'algorithme que la distance affectee a un sommet v (ligne 10) n'est pas correcte et soit u le sommet affecte a la ligne 7 et qui a decouvert v.

On a a la fois $dist[v] > dist(s, v)$, d'apres le corollaire precedent, et $dist[u] = dist(s, u)$ par definition de v.

Considerons un plus court chemin de s a v, et notons u' le precedent de v sur ce chemin. On a
$dist(s, u') = dist(s, v) - 1 < dist[v] - 1 = dist[u] = dist(s, u)$.

Or d'apres la propriete 3, les sommets entrent dans F, et donc en sortent, a distance croissante. Ainsi, comme
$dist(s, u') < dist(s, u)$, u' aurait donc du etre traite, et decouvrir v, avant u : contradiction. $\square$

Analyse de complexite de l'algorithme

- toutes les instructions s'executent en temps $O(1)$
- boucle ligne 2 : $O(n)$
- Boucle ligne 6 : au plus n fois
  - car u est premier de F (ligne 7) au plus une fois
- boucle ligne 8 : $d^\circ(u)$ fois
- au total, la boucle ligne 6 prend un temps $\leq O(n) + O(\sum_{u \in V} d^\circ(u)) = O(n+m)$
- complexite totale de l'algo : $O(n+m)$

Graphes orientes (toujours non ponderes)

- **Cadre :** graphes orientes et non-ponderes
- **Longueur d'un chemin oriente C de u a v :** nombre d'arcs sur le chemin C
- **Distance $dist(u, v)$ entre u et v :** longueur du plus court chemin oriente de u a v
  - Note : $dist(u, v) \neq dist(v, u)$

Probleme des plus courts chemins orientes a origine unique

- **Entrée :** un graphe oriente $G = (V, A)$ non pondere, un sommet $s \in V$

- **Sortie :** la distance $dist(s, u)$ pour tout les sommets $u \in V$

Graphes orientes (toujours non ponderes)

Meme solution mais $N(u)$ devient $N^+(u)$

Algorithme 2 : BFS(G,s) dans un graphe oriente

Un tableau *dist* de taille n ; $F \leftarrow \emptyset$ une file;
pour u *de* 0 *a* $n-1$ **faire**
    $dist[u] \leftarrow +\infty$; $coul[u] \leftarrow blanc$;
fin
$dist[s] \leftarrow 0$; $enfiler(s, F)$; $coul[s] = gris$;
tant que $F \neq \emptyset$ **faire**
    $u \leftarrow premier(F)$;
    pour $v \in N^+(u)$ **faire**
        si $coul[v] = blanc$ **alors**
            $dist[v] \leftarrow dist[u] + 1$; $enfiler(v, F)$; $coul[v] = gris$;
        fin
    fin
    defiler(F);
fin

Graphes ponderes positivement

Plus courts chemins a origine unique

Notion de distance dans les graphes ponderes positivement

- **Cadre :** graphes orientes et ponderes positivement
  - une application $w : A \to \mathbb{R}_+$

Plus courts chemins a origine unique

Notion de distance dans les graphes ponderes positivement

- **Cadre :** graphes orientes et ponderes positivement
  - une application $w : A \to \mathbb{R}_+$
  - graphes non orientes obtenus comme cas particulier : graphes orientes symetriques avec fonction de poids symetrique $\forall (u,v) \in A, w(u,v) = w(v,u)$.

Plus courts chemins a origine unique

Notion de distance dans les graphes ponderes positivement

- **Cadre :** graphes orientes et ponderes positivement
  - une application $w : A \to \mathbb{R}_+$
  - graphes non orientes obtenus comme cas particulier : graphes orientes symetriques avec fonction de poids symetrique $\forall (u, v) \in A, w(u, v) = w(v, u)$.
- **Longueur d'un chemin oriente pondere C de u a v :** somme des poids des arcs du chemin oriente C

Plus courts chemins a origine unique

Notion de distance dans les graphes ponderes positivement

- **Cadre :** graphes orientes et ponderes positivement
  - une application $w : A \to \mathbb{R}_+$
  - graphes non orientes obtenus comme cas particulier : graphes orientes symetriques avec fonction de poids symetrique $\forall (u, v) \in A, w(u, v) = w(v, u)$.
- **Longueur d'un chemin oriente pondere C de u a v :** somme des poids des arcs du chemin oriente C
- **Distance $dist(u, v)$ de u a v :** longueur du plus court chemin oriente pondere de u a v

Plus courts chemins a origine unique

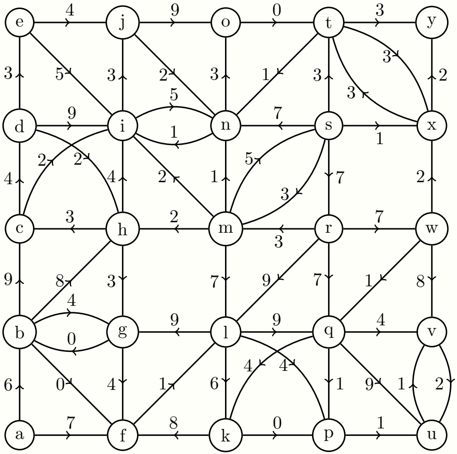
Notion de distance dans les graphes ponderes positivement

- **Cadre :** graphes orientes et ponderes positivement
  - une application $w : A \to \mathbb{R}_+$
  - graphes non orientes obtenus comme cas particulier : graphes orientes symetriques avec fonction de poids symetrique $\forall(u, v) \in A, w(u, v) = w(v, u)$.
- **Longueur d'un chemin oriente pondere C de u a v :** somme des poids des arcs du chemin oriente C
- **Distance $dist(u, v)$ de u a v :** longueur du plus court chemin oriente pondere de u a v

Probleme des plus courts chemins a origine unique

- **Entrée :** un graphe $G = (V, E)$ oriente et pondere positivement, un sommet $s \in V$
- **Sortie :** la distance $dist(s, u)$ pour tout les sommets $u \in V$

Exemple

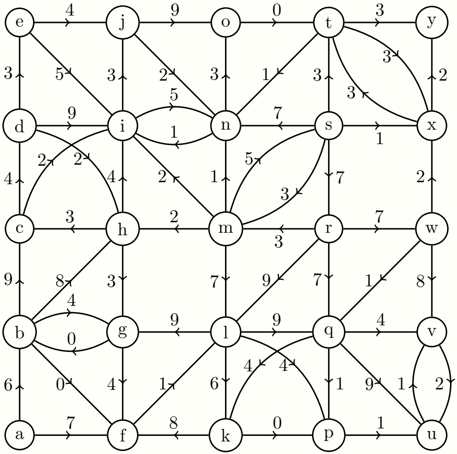

e
j
o
t
y
d
i
n
s
x
c
h
m
r
w
b
g
l
q
v
a
f
k
p
u
4
9
0
3
3
5
3
2
3
1
3
3
3
2
9
5
1
7
1
2
2
4
2
1
5
3
7
2
3
2
3
7
9
8
3
7
9
7
1
8
4
0
9
9
4
6
0
4
1
6
4
4
1
9
1
2
7
8
0
1

Algorithme : Dijkstra

Algorithme 3 : Dijkstra(G,s)

Un tableau *dist* de taille n ; Un tableau *coul* de taille n;
pour u *de* 0 *a* $n-1$ **faire**
    $dist[u] \leftarrow +\infty$; $coul[u] \leftarrow blanc$;
fin
$dist[s] \leftarrow 0$; $Q \leftarrow \{s\}$; $coul[s] = gris$;
tant que $Q \neq \emptyset$ **faire**
    $u \leftarrow min_{dist}(Q)$; $Q \leftarrow Q \setminus \{u\}$; $coul[u] \leftarrow$ noir;
    pour $v \in N^+(u)$ **faire**
        si $dist[u] + w(u,v) < dist[v]$ **alors**
            $dist[v] \leftarrow dist[u] + w(u,v)$;
            si $coul[v]$=blanc **alors**
                $Q \leftarrow Q \cup \{v\}$; $coul[v] \leftarrow$ gris;
        fin
    fin
fin

Exemple d'execution de Dijkstra(G,s)



Preuve de correction de l'algorithme

Theoreme : A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ de s a u.

Preuve de correction de l'algorithme

Theoreme : A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ de s a u.

Lemme : A la ligne 7 de l'algorithme, $dist[u]$ est exactement la distance de s a u dans G.

Preuve de correction de l'algorithme

<u>Theoreme :</u> A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ de s a u.

<u>Lemme :</u> A la ligne 7 de l'algorithme, $dist[u]$ est exactement la distance de s a u dans G.

Démonstration.

Par l'absurde, supposons que cela ne soit pas vrai pour le sommet u de la ligne 7.

$\square$

Preuve de correction de l'algorithme

<u>**Theoreme :**</u> A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ de s a u.

<u>Lemme :</u> A la ligne 7 de l'algorithme, $dist[u]$ est exactement la distance de s a u dans G.

Démonstration.
Par l'absurde, supposons que cela ne soit pas vrai pour le sommet u de la ligne 7.
Considerons un plus court chemin de s a u et le premier sommet non noir v sur ce chemin.

$\square$

Preuve de correction de l'algorithme

Theoreme : A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ de s a u.

Lemme : A la ligne 7 de l'algorithme, $dist[u]$ est exactement la distance de s a u dans G.

Démonstration.

Par l'absurde, supposons que cela ne soit pas vrai pour le sommet u de la ligne 7.

Considerons un plus court chemin de s a u et le premier sommet non noir v sur ce chemin.

Alors v est gris et $dist[v] < dist[u]$

$\square$

Preuve de correction de l'algorithme

<u>**Theoreme :**</u> A la fin de l'algorithme, la distance $dist[u]$ affectee a chaque sommet u est exactement la distance $dist(s, u)$ de s a u.

<u>Lemme :</u> A la ligne 7 de l'algorithme, $dist[u]$ est exactement la distance de s a u dans G.

Démonstration.

Par l'absurde, supposons que cela ne soit pas vrai pour le sommet u de la ligne 7.
Considerons un plus court chemin de s a u et le premier sommet non noir v sur ce chemin.
Alors v est gris et $dist[v] < dist[u]$
$\Longrightarrow$ contradiction : l'algo aurait du choisir v plutot que u.

□

Analyse de complexite de l'algorithme

- Toutes les instructions sont en temps $O(1)$, sauf
  - $u \leftarrow min_{dist}(Q)$, a la ligne 7
  - $dist[v] \leftarrow dist[u] + w(u, v)$, a la ligne 10
  - qui dependent de la structure de donnees pour extraire le min

Analyse de complexite de l'algorithme

- Toutes les instructions sont en temps $O(1)$, sauf
  - $u \leftarrow min_{dist}(Q)$, a la ligne 7
  - $dist[v] \leftarrow dist[u] + w(u, v)$, a la ligne 10
  - qui dependent de la structure de donnees pour extraire le min
- au total : $O(n + m)$ + temps utilisation structure de donnees

Analyse de complexite de l'algorithme

- Toutes les instructions sont en temps $O(1)$, sauf
  - $u \leftarrow min_{dist}(Q)$, a la ligne 7
  - $dist[v] \leftarrow dist[u] + w(u, v)$, a la ligne 10
  - qui dependent de la structure de donnees pour extraire le min
- au total : $O(n + m)$ + temps utilisation structure de donnees

Differentes implementations de la file de priorite

- Tableau des poids : $O(n^2)$
  - Total algo : $O(n^2)$

Analyse de complexite de l'algorithme

- Toutes les instructions sont en temps $O(1)$, sauf
  - $u \leftarrow min_{dist}(Q)$, a la ligne 7
  - $dist[v] \leftarrow dist[u] + w(u, v)$, a la ligne 10
  - qui dependent de la structure de donnees pour extraire le min
- au total : $O(n + m)$ + temps utilisation structure de donnees

Differentes implementations de la file de priorite

- Tableau des poids : $O(n^2)$
  - Total algo : $O(n^2)$

- Tas binaire : $O((n + m) \log n)$
  - Total algo : $O((n + m) \log n)$

Analyse de complexite de l'algorithme

- Toutes les instructions sont en temps $O(1)$, sauf
  - $u \leftarrow min_{dist}(Q)$, a la ligne 7
  - $dist[v] \leftarrow dist[u] + w(u, v)$, a la ligne 10
  - qui dependent de la structure de donnees pour extraire le min
- au total : $O(n + m)$ + temps utilisation structure de donnees

Differentes implementations de la file de priorite

- Tableau des poids : $O(n^2)$
  - Total algo : $O(n^2)$
- Tas binaire : $O((n + m) \log n)$
  - Total algo : $O((n + m) \log n)$
- Tas fibonacci : $O(m + n \log n)$
  - Total algo : $O(m + n \log n)$