Algorithmique I
Année 2007-2008

A. Benoit, B. Depardon
JF. Pineau, C. Rezvoy

TD n°8 - Recherche de plus courts chemins

1 L'algorithme de Bellman-Ford

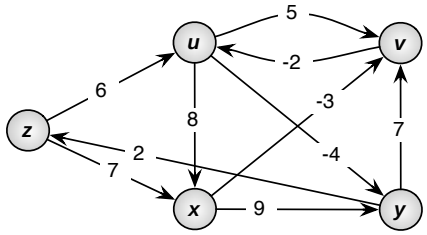
L'algorithme de Bellman-Ford résout le problème des plus courts chemins avec origine unique dans le cas le plus général où les poids des arcs peuvent avoir des valeurs négatives. Étant donné un graphe orienté pondéré $G = (V, E)$, de fonction de poids w, et une origine s, l'algorithme retourne une valeur booléenne indiquant s'il existe un circuit de poids négatif accessible depuis s. S'il n'en existe pas, l'algorithme donne les plus courts chemins ainsi que leurs poids.

Les notations sont les suivantes : $\pi[v]$ contient le prédécesseur de v sur le chemin (NIL s'il n'y en a pas), $\delta(u, v)$ est le poids du plus court chemin de u vers v (∞ s'il n'existe pas), $d[v]$ est une variable qui est une borne supérieure du poids du plus court chemin de s vers v (cf. question 4). On définit le poids d'un chemin $p = \langle v_0, v_1, \ldots, v_k \rangle$, est $w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$.

Algorithme 1 Bellman-Ford(G, w, s)

```
1: pour tout sommet v ∈ V faire // Initialisation
2:    d[v] ← ∞, π[v] ← NIL
3: d[s] ← 0
4: pour i de 1 à |V| − 1 faire
5:    pour tout arc (u, v) ∈ E faire // relâchement de l'arc (u, v)
6:       si d[v] > d[u] + w(u, v) alors
7:          d[v] ← d[u] + w(u, v), π[v] ← u
8: pour tout arc (u, v) ∈ E faire // détection des circuits négatifs
9:    si d[v] > d[u] + w(u, v) alors
10:      retourner Faux
11: retourner Vrai
```

Question 1.1 On considère le graphe ci-dessous. Faire tourner l'algorithme en prenant comme source le sommet z. Même question en changeant le poids de l'arc (y, v) à 4.



Question 1.2 Quelle est la complexité de cet algorithme ?

Question 1.3 Montrer qu'après le relâchement de l'arc (u, v), $d[v] \leq d[u] + w(u, v)$.

Question 1.4 Propriété du majorant : Montrer que $d[v] \geq \delta(s, v)$ pour tout sommet v à tout instant de l'algorithme. Montrer que lorsque $d[v]$ a atteint sa borne inférieure $\delta(s, v)$, il n'est plus jamais modifié.

Question 1.5 Propriété de convergence : Considérons un plus court chemin $(s, \ldots, u, v)$ pour un sommet v donné. Montrer que si $d[u] = \delta(s, u)$ à un moment précédent l'appel du relâchement de (u, v) dans l'algorithme, alors on a toujours $d[v] = \delta(s, v)$ après l'appel.

Question 1.6 Montrer que, si G ne contient aucun circuit de poids négatif accessible depuis s, alors, à la fin de l'exécution de l'algorithme, $d[v] = \delta(s, v)$ pour tout sommet v accessible depuis s.

Question 1.7 Montrer que pour chaque sommet v, il existe un chemin de s vers v si et seulement si BELLMAN-FORD se termine avec $d[v] < \infty$.

Nous pouvons à présent démontrer la validité de l'algorithme.

Question 1.8 Montrer que si G ne contient aucun circuit de poids négatif accessible depuis s, alors l'algorithme retourne VRAI, on a $d[v] = \delta(s, v)$ pour tous les sommets $v \in V$, et le sous-graphe G_π des sommets v dont $\pi(v) \neq$ NIL et des arcs $(\pi(v), v)$ est une arborescence des plus courts chemins de racine s. Montrer que si G contient un circuit de poids négatif accessible à partir de s, l'algorithme renvoie FAUX (on pourra raisonner par l'absurde).

2 L'algorithme de Johnson

On s'intéresse maintenant au problème consistant à calculer les plus courts chemins pour tout couple de sommets du graphe. L'algorithme de Johnson a une bonne complexité pour les graphes creux (peu d'arêtes en comparaison du nombre de sommets). Il utilise les algorithmes de Dijkstra et de Bellman-Ford, et renvoie soit la matrice des poids des plus courts chemins, soit l'assertion que le graphe possède un circuit de poids négatif.

La technique utilisée consiste à se ramener au cas avec uniquement des poids positifs, puis de faire tourner l'algorithme de Dijkstra en partant de chaque sommet. Si les poids ne sont pas tous positifs, on les redéfinit pour pouvoir se ramener au premier cas.

Soit w la fonction des poids avant redéfinition. La nouvelle fonction de poids $\hat{w}$ doit respecter deux propriétés :

1. Pour chaque couple de sommets $u, v \in V$, un plus court chemin de u à v en utilisant la fonction de poids w est également un plus court chemin pour la fonction $\hat{w}$.
2. Pour chaque arête (u, v), le nouveau poids $\hat{w}(u, v)$ n'est pas négatif.

Dans la suite, on note comme dans l'exercice précédent δ pour la fonction des poids des plus courts chemins (avec w), et $\hat{\delta}$ représente la fonction des poids des plus courts chemins en utilisant la fonction de poids $\hat{w}$.

Voici l'algorithme de Dijkstra. Avec $\forall u \in V, adjacent[u]$ la liste des sommets adjacents à u.

Algorithme 2 Dijkstra(G, w, s)

1: **pour tout** sommet $v \in V$ **faire** // Initialisation
2: $d[v] \leftarrow \infty$, $\pi[v] \leftarrow$ NIL
3: $d[s] \leftarrow 0$
4: $E \leftarrow \emptyset$
5: $F \leftarrow V[G]$ // ensemble des sommets de G
6: **tantque** $F \neq \emptyset$ **faire**
7: $u \leftarrow v | d[v] = \min\{d[x] | x \in F\}$ // on choisit le sommet avec la plus petite valeur de d
8: $F \leftarrow F - \{u\}$
9: $E \leftarrow E \cup \{u\}$
10: **pour tout** sommet $v \in adjacent[u]$ **faire** // relâchement de l'arc (u, v)
11: **si** $d[v] > d[u] + w(u, v)$ **alors**
12: $d[v] \leftarrow d[u] + w(u, v)$, $\pi[v] \leftarrow u$

Question 2.1 Propriété 1
Soit $h : V \rightarrow \mathbb{R}$ une fonction quelconque faisant correspondre un réel à chaque sommet. Pour chaque arête $(u, v) \in E$, on définit $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$.

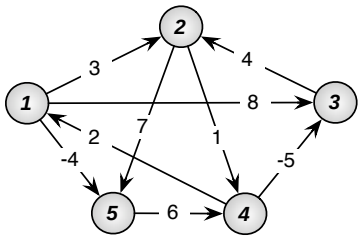
Soit $p = \langle v_0, v_1, ..., v_k \rangle$ un chemin allant de v_0 à v_k. Montrer que $w(p) = \delta(v_0, v_k)$ si et seulement si $\hat{w}(p) = \hat{\delta}(v_0, v_k)$. De plus, montrer que G possède un circuit de poids négatif en utilisant w si et seulement si G possède un circuit de poids négatif en utilisant $\hat{w}$.

Question 2.2 Propriété 2
On désire maintenant définir $\hat{w}$ vérifiant également la deuxième propriété (fonction non négative). On crée un nouveau graphe $G' = (V', E')$ avec $V' = V \cup \{s\}$ $(s \notin V)$ et $E' = E \cup \{(s, v) : v \in V\}$. La fonction de poids w est étendue de manière à ce que $w(s, v) = 0$ pour chaque $v \in V$.

De plus, si G n'a pas de circuit de poids négatif, on définit $h(v) = \delta(s, v)$ pour chaque $v \in V'$.

2.2.1 A partir des indications précédentes, définir une nouvelle fonction de poids et montrer qu'elle vérifie les deux propriétés.

2.2.2 Générer G' à partir du graphe G ci-dessous, et calculer les nouveaux poids.



Question 2.3 Algorithme de Johnson
Écrire l'algorithme de Johnson en s'inspirant des questions précédentes. Faire tourner l'algorithme sur le graphe ci-dessus. Justifier la correction de l'algorithme et calculer sa complexité. On rappelle que l'algorithme de Dijkstra implémenté par un tas de Fibonacci a une complexité en $O(VlogV + E)$.