Algorithmique — L3 — TD 9
Plus courts chemins : la méthode Dijkstra contre la méthode Bellman-Ford

1 Dijkstra

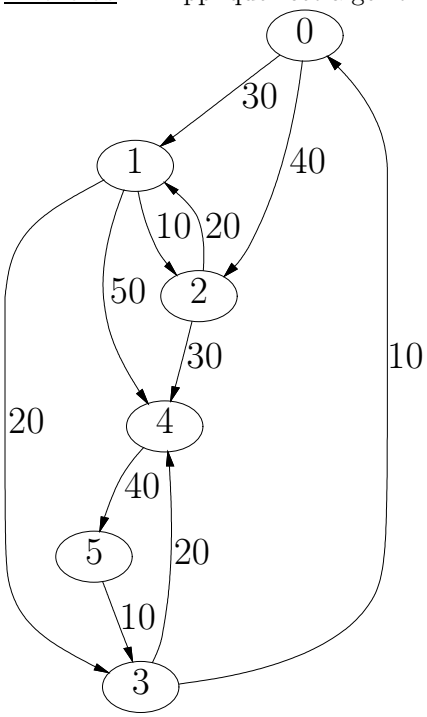
On rappelle l'algorithme de Dijkstra :

```
Dijkstra(  G: graphe, w: fonction de ponderation, s:sommet)
   soit n le nombre de sommets de G
   soit d[ ] un tableau de n entiers initialise a +infini
   soit Pi[ ] un tableau de n pointeurs sur sommet initialise a null
   soit S une file de priorite

d[s]=0
Initialiser F avec tous les sommets et d comme clef

Tant que S non vide
    u = Extraire-Min(F)
    pour tout voisin v de u
          si d[u] + w(u,v) < d[v]
               d[v] = d[u] + w(u,v)
               Pi[v] = u
               MaJ (F, v, d)
```

Exercice 1 : Appliquer cet algorithme au graphe suivant (pour les plus courts chemins depuis 0) :



Exercice 2 : Rappeler sa complexité en temps. Ne pas oublier les fonctions de manipulation de la file de priorité.

Exercice 3 : Pourquoi l'algorithme de Dijkstra n'est-il pas valable lorsque les arêtes peuvent avoir des poids négatifs ? Trouver un exemple.

2 Bellman-Ford

Et voici le concurrent : l'algorithme de Bellman-Ford.

```
Bellman-Ford(  G: graphe, w: fonction de ponderation, s:sommet)
   soit n le nombre de sommets de G
   soit d[] un tableau de n entiers initialise a +infini
   soit Pi[] un tableau de n sommets initialise a nil

d[s]=0
pour i de 1 a n faire
  pour chaque arc (u,v) de G faire
     si d[u] + w(u,v) < d[v]
               d[v] = d[u] + w(u,v)
               Pi[v] = u

pour chaque arc (u,v) de G faire
    si d[u] + w(u,v) < d[v] retourner Faux
retourner vrai
```

Exercice 4 : Appliquer l'algorithme au graphe de la page précédente.

Exercice 5 : Quelle est la complexité de cet algorithme (au mieux, au pire, en moyenne) ?

Exercice 6 : A quoi sert la dernière boucle ?
Donnez un exemple de graphe où la valeur FAUX est retournée.

Exercice 7 : Prouvez l'invariant

S'il existe un plus court chemin de s à u de k arcs,
alors après le tour k dans la boucle principale d[u] vaut $\delta(s, u)$

Servez-vous en pour prouver la validité de Bellman-Ford.

3 Taux de change optimal

On souhaite convertir de l'argent d'une devise dans une autre. Le problème est que toutes les conversions ne sont pas possibles : pour deux monnaies A et B, on peut parfois convertir de l'argent de A en B, parfois non. On considère donc un *graphe de change* $G = (V, E)$ entre monnaies donnant les conversions possibles. Ce graphe est orienté (parfois on peut convertir A en B mais pas B en A).

La *fonction de change* est une fonction c telle que
– une somme S en monnaie A vaut $S \cdot c(A, B)$ en monnaie B (les taxes éventuelles sont incluses).

– $c(A, B)$ est défini si et seulement si (A, B) est un arc du graphe de change.
– $c(A, B) > 0$

Le *graphe de change étendu* est le graphe $G' = (V, E, c)$ pondéré par la fonction de change. Une *séquence de change* est la conversion d'une monnaie A_1 en monnaie A_k en passant par les monnaies intermédiaires $A_2...A_{k-1}$ (en supposant bien sûr toutes ces conversions possibles). Il lui correspond un chemin dans le graphe de change.

Exercice 8 : Quel est le taux de change de A_1 en A_k dans une séquence de change $A_1, A_2, ...A_k$?

Exercice 9 : Dans quelle condition (exprimée sur G') quelqu'un peut-il devenir *infiniment riche* en changeant de l'argent ?

Étant données deux séquences de change différentes de la monnaie A en la monnaie B, la meilleure des deux est celle qui a le taux le plus élevé.

Exercice 10 : Supposons que l'on connaisse une séquence de change S_1 de la monnaie A en la monnaie B, d'une part, et une séquence S_2 de la monnaie A en la monnaie C d'autre part. Supposons que l'arc (C, B) existe, dans le graphe de change. Écrivez une condition de *relaxation* en comparant les taux des séquences S_1 d'une part, S_2 puis (C, B) d'autre part, et gardant la meilleure.

Exercice 11 : Écrivez une version modifiée de l'algorithme de Bellman-Ford, utilisant cette condition de relaxation modifiée, donnant les meilleurs taux de change d'une monnaie A vers toutes les autres.
Pourquoi est-ce correct ?
indication : log(ab)=log(a)+log(b)

Exercice 12 : Même question avec Dijkstra : est-ce que ça marche ?

4 Plus longs chemins

Exercice 13 : Quel(s) algorithme faut-il modifier pour avoir un arbre de plus **longs** chemins : Dijkstra ? Bellman-Ford ?