INF2010 - Structures de données et algorithmes

Travail Pratique #5
Graphes, Bellman
Automne 2018

Contexte du laboratoire

Dans le cadre de ce Laboratoire, vous allez implémenter un graphe, et utiliser l'algorithme de Bellman pour trouver le chemin le plus court entre un sommet source et les autres nœuds d'un graphe. Un graphe est un ensemble de nœuds (nodes) et d'arcs (edges). Les sources qui vous sont remises incluent cinq classes: **Main.java, Node.java, Edge.java, Graph.java**, **Bellman.java** et un fichier **« graphe.txt».** Main permet de construire un graph (**Graph.java**) et tester l'algorithme Bellman (**Bellman.java**). Attention, il ne faut pas modifier que les méthodes demandées, vous avez le droit d'ajouter des attributs et des méthodes, mais pas changer les signatures de classes ou de méthodes.

Exercice 1 : Création de graphe (1 points)

Dans la classe « **Graph** » complétez la méthode « **getOutEdges** » qui permet de retourner tous les arcs sortants d'un nœud « **source** ». D'une manière similaire complétez la méthode « **getInEdges** » qui permet de trouver tous les arcs entrants à un nœud « **dest** ». Complétez aussi la méthode « **readFromFile**» de la classe« **Graph** » qui permet de créer le graphe à partir d'un fichier « **graphe.txt**».

Exercice 2 : Implémentation de l'algorithme de Bellman (3 points)

L'algorithme de Bellman trouve le plus court chemin entre un nœud de départ **S** et tous les autres nœuds. Pour implémenter l'algorithme, il faut suivre plusieurs itérations, dont chacune consiste à calculer la distance entre le nœud de départ et ses successeurs, ensuite ces derniers et leurs successeurs, etc. L'algorithme ci-dessous décrit les étapes de l'algorithme de Bellman :

Algorithme de Bellman

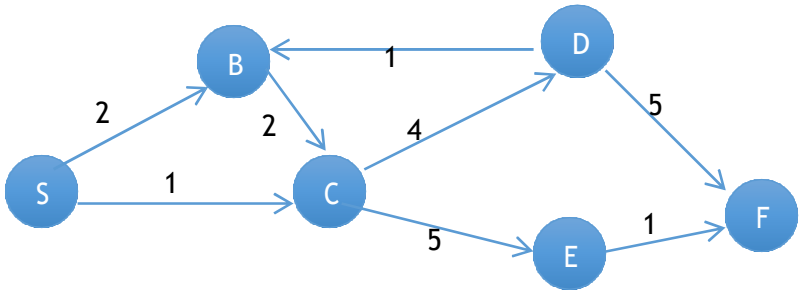
1. $\pi^0(s) := 0$; $R^0(s) := \infty$; $k := 1$;
2. $\pi^0(x) := \infty$ et $R^0(x) := \infty$ pour tout $x \neq s$;
3. $\pi^k(x) := min\{\pi^{k-1}(x), \min_{y \in N^-(x)}\{\pi^{k-1}(y) + d_{yx}\}\}$;
4. Si $\pi^{k-1}(x) > \min_{y \in N^-(x)}\{\pi^{k-1}(y) + d_{yx}\}$ poser $R^k(x) := y$;
5. Si $\pi^k(x) = \pi^{k-1}(x)$ pour tout x alors **STOP**;
6. Si $k \leq n-1$ poser $k := k+1$ et aller à (3);
7. Si $k=n$ **STOP :** il existe un circuit de longueur négative

Où

- **s** est le sommet source.
  π^k est le tableau de calcul du plus court chemin à l'itération **k**;
- $\pi^k(x)$ est le coût (distance totale) du plus court chemin entre le sommet **s** et **x** à l'itération k;
- R^k est un tableau contenant le prédécesseur de chaque nœud dans le plus court chemin à l'itération **k;**
- $N^-(x)$ **est** l'ensemble des nœuds prédécesseurs du nœud **x.**
- d_{yx} est la distance entre les deux sommets **x** et **y**.

Exemple 1: Soit le graphe sans circuit négatif suivant:

Graphe 1



- Chaque ligne du tableau ci-dessous représente un tableau π^k

Itération (k)	S	B	C	D	E	F
0	0	∞	∞	∞	∞	∞
1	0	2	1	∞	∞	∞
2	0	2	1	5	6	∞
3	0	2	1	5	6	7
4	0	2	1	5	6	7

STOP $\pi^k = \pi^{k-1}$

Le coût du plus court chemin de **S** à **F** est **7**

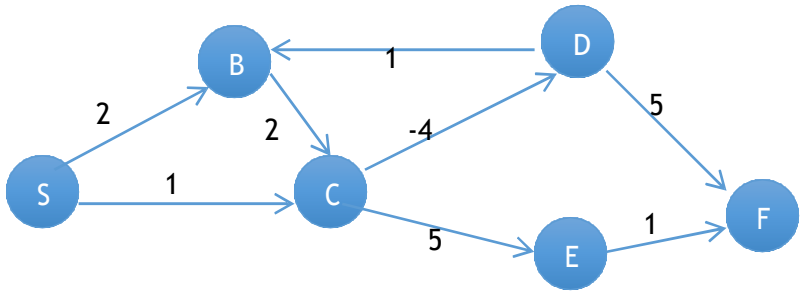
- Chaque ligne du tableau ci-dessous représente un tableau R^k

Itération (k)	S	B	C	D	E	F
0	-	-	-	-	-	-
1	-	S	S	-	-	-
2	-	S	S	C	C	-
3	-	S	S	C	C	E
4	-	S	S	C	C	E

Pour trouver le plus court chemin de **S** à **F** on parcourt le graphe du sommet **F** jusqu'au sommet **S** en utilisant la ligne R^4 du tableau ci-dessus. On trouve **F**$\longrightarrow$ R^4**(F)=E** $\longrightarrow$ R^4**(E)=C** $\longrightarrow$ R^4 **(C)=S,** ensuite on inverse le chemin pour obtenir **S** $\longrightarrow$ **C** $\longrightarrow$ **E** $\longrightarrow$ **F.**

Exemple 2: Soit le graphe avec circuit négatif (B→ C→ D → B) suivant:

Graphe 2



Chaque ligne du tableau ci-dessous représente un tableau π^k

Itération (k)	S	B	C	D	E	F
0	0	∞	∞	∞	∞	∞
1	0	2	1	∞	∞	∞
2	0	2	1	-3	6	∞
3	0	-2	1	-3	6	2
4	0	-2	0	-3	6	2
5	0	-2	0	-4	5	2
6	0	-3	0	-4	5	1

STOP k = n

Chaque ligne du tableau ci-dessous représente un tableau R^k

Itération (k)	S	B	C	D	E	F
0	-	-	-	-	-	-
1	-	S	S	-	-	-
2	-	S	S	C	C	-
3	-	D	S	C	C	D
4	-	D	B	C	C	D
5	-	D	B	C	C	D
6	-	D	B	C	C	D

Pour afficher le circuit négatif, on parcourt tous les sommets **x** tel que $\pi^n(x) < 0$ et on construit le chemin à partir de **x**. Si le chemin trouvé est un circuit alors **stop**. Sinon choisir un autre sommet **x** tel que $\pi^n(x) < 0$.

Par exemple si on choisit le sommet **B** on trouve le circuit **B**⟶ R^4 **(B)=D** ⟶ R^4 **(D)=C** ⟶ R^4**(C)=B**

Complétez la méthode « **displayTables** » qui affiche la table « **piTable** » et « **rTable** » en respectant le format ci-dessous :

- ✓ « **piTable** » du **graphe2** :

```
<< PITable >>:
              S      B      C      D      E      F
0      :      0.0    inf    inf    inf    inf    inf
1      :      0.0    2.0    1.0    inf    inf    inf
2      :      0.0    2.0    1.0    -3.0   6.0    inf
3      :      0.0    -2.0   1.0    -3.0   6.0    2.0
4      :      0.0    -2.0   0.0    -3.0   6.0    2.0
5      :      0.0    -2.0   0.0    -4.0   5.0    2.0
6      :      0.0    -3.0   0.0    -4.0   5.0    1.0
```

- ✓ « **rTable** » du **graphe2** :

```
<< RTable >>:
k      :      S      B      C      D      E      F
0      :      -      -      -      -      -      -
1      :      -      S      S      -      -      -
2      :      -      S      S      C      C      -
3      :      -      D      S      C      C      D
4      :      -      D      B      C      C      D
5      :      -      D      B      C      C      D
6      :      -      D      B      C      C      D
```

Exercice 3 : Afficher le chemin le plus court (1 points)

Maintenant, une fois que vous avez complété les tables, il faut afficher le plus court chemin de **S** vers tous les sommets. Pour cela, utiliser une pile « path» pour retrouver les plus courts chemins. Complétez la méthode « **diplayShortestPaths** ».
Format d'affichage demandé est :

- ✓ Si le graphe contient un circuit négatif (i.g **graphe 2**):

```
==>  le graphe contient un circuit de coût négatif

[B - B] : B -> C -> D -> B
```

- ✓ Si le graphe ne contient pas de circuit négatif (i.g **graphe1**):

```
=>  Les chemins sont :

[S - B] 2.0 : S -> B
[S - C] 1.0 : S -> C
[S - D] 5.0 : S -> C -> D
[S - E] 6.0 : S -> C -> E
[S - F] 7.0 : S -> C -> E -> F
```

Notez que votre code doit implémenter l'algorithme de Bellman en générale, pas juste pour l'exemple de TP (d'autres graphes seront testés durant la correction).

Instructions pour la remise:

Le travail doit être fait par équipe de 2 personnes idéalement et doit être remis via Moodle :

Groupe (01) : 06 Décembre avant 23h55

Groupe (02) : 29 Novembre avant 23h55

Groupe (03) : 05 Décembre avant 23h55

Groupe (04) : 28 Novembre avant 23h55

Groupe (05) : 09 Décembre avant 23h55

Veuillez envoyer vos fichiers .java seulement dans une archive de type *.**zip** qui portera le nom **inf2010_lab5_MatriculeX_MatriculeY** (de sorte que **MatriculeX < MatriculeY**).

Les travaux en retard seront pénalisés de 20 % par jour de retard. Aucun travail ne sera accepté après 4 jours de retard.