**Représentation d’un graphe**

Ecriture de la classe Nœud et des méthodes de Noeud

Ecriture de la classe Arc

Ecriture de l’interface Graphe

Ecriture de la classe GrapheListe

Ecriture de test unitaire

**Calcul du plus court chemin par point fixe**

Ecriture de l’algorithme du point fixe

Ecriture de la classe Main

Ecriture de test unitaire

**Calcul du meilleur chemin par Dijkstra**

Ecriture de la classe Dijkstra

Ecriture de test unitaire

Ecriture de MainDijkstra

**Validation et expérimentation**

**Question 22)**

|  |  |
| --- | --- |
| BellmanFord: Graphe1.txt | Dijkstra: Graphe1.txt |
| 1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:1.7976931348623157E308 p:null  4 -> V:1.7976931348623157E308 p:null  5 -> V:1.7976931348623157E308 p:null  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:1.7976931348623157E308 p:null  4 -> V:15.0 p:1  5 -> V:1.7976931348623157E308 p:null  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:1.7976931348623157E308 p:null  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:1.7976931348623157E308 p:null  7 -> V:32.0 p:3  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:1.7976931348623157E308 p:null  7 -> V:32.0 p:3  8 -> V:21.0 p:3  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:1.7976931348623157E308 p:null  7 -> V:32.0 p:3  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:24.0 p:3  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:1.7976931348623157E308 p:null  7 -> V:32.0 p:3  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:24.0 p:3  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:1.7976931348623157E308 p:null  7 -> V:20.0 p:4  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:24.0 p:3  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:20.0 p:4  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:20.0 p:4  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:24.0 p:9  7 -> V:20.0 p:4  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:24.0 p:9  7 -> V:20.0 p:4  8 -> V:18.0 p:9  9 -> V:16.0 p:3 | 1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:1.7976931348623157E308 p:null  4 -> V:1.7976931348623157E308 p:null  5 -> V:1.7976931348623157E308 p:null  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:1.7976931348623157E308 p:null  4 -> V:15.0 p:1  5 -> V:1.7976931348623157E308 p:null  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:1.7976931348623157E308 p:null  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:1.7976931348623157E308 p:null  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:1.7976931348623157E308 p:null  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:1.7976931348623157E308 p:null  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:22.0 p:5  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:22.0 p:5  8 -> V:1.7976931348623157E308 p:null  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:22.0 p:5  8 -> V:21.0 p:3  9 -> V:1.7976931348623157E308 p:null  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:22.0 p:5  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:25.0 p:5  7 -> V:20.0 p:4  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:24.0 p:9  7 -> V:20.0 p:4  8 -> V:21.0 p:3  9 -> V:16.0 p:3  1 -> V:0.0 p:null  10 -> V:21.0 p:5  2 -> V:4.0 p:1  3 -> V:13.0 p:2  4 -> V:15.0 p:1  5 -> V:12.0 p:2  6 -> V:24.0 p:9  7 -> V:20.0 p:4  8 -> V:18.0 p:9  9 -> V:16.0 p:3 |

L’algorithme de BellmanFord parcours l’entièreté des Nœuds à chaque fois, tout en modifiant le chemin minimal menant à un Nœud si le chemin minimal déjà affecté est plus grand. Alors que l’algorithme de Dijkstra lui affecte une valeur minimale aux parents du Nœud sur lequel il est actuellement, puis le parent avec le chemin minimal devient le Nœud actuel et réitère les mêmes opérations sans jamais repasser par les Nœuds pour les lesquels il a déjà effectué cette opération.

Nœud = Sommet.

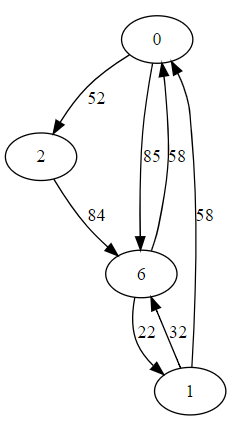
**Question 23 et 24)**

D’après l’observation de la question précédente, on peut en conclure que l’algorithme de BellmanFord est plus performant que celui de Dijkstra pour les graphes avec peu de chemins différents possibles, car il effectuera moins d’itérations, mais par conséquents pour les graphes avec beaucoup de chemins différents possibles, le second algorithme sera plus performant.

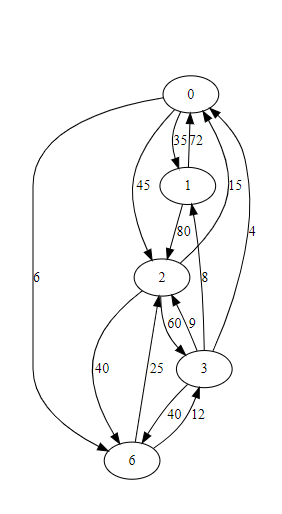
Par conséquent l’efficacité des algorithmes dépendent du nombre d’arc moyen par Sommet.

**Question 26)**

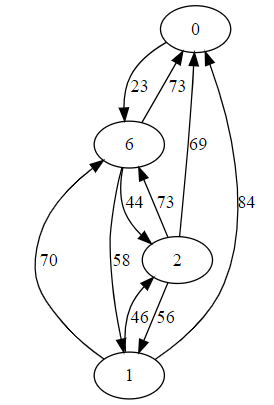
g.genererGraphe("2","6",3);



g.genererGraphe("2","6",4);



g.genererGraphe("2","6",3);



**Question 27 et 28)**

Une image contenant texte, capture d’écran, menu

Description générée automatiquement

Une image contenant texte, capture d’écran, menu

Description générée automatiquement

(Lorsque le ratio est positif cela signifie que l’algorithme de Dijkstra est plus performant)

Comme on peut le voir dans l’extrait des résultats obtenus, l’algorithme le plus efficace est très souvent celui de Dijkstra, avec une moyenne de 23,805µs de différences par nombre d’arc moyen par Nœud, les rares fois où l’algorithme de BellmanFord est plus performant sont observé lorsque le nombre d’arc moyen par nœud est faible, est que par conséquent peu de chemins sont possibles.

**Question 29)**

On en conclu donc que pour les graphes où de nombreux chemins possibles existent l’algorithme de Dijkstra est le plus performant, cependant lorsque le nombre de chemins est très faible l’algorithme de BellmanFord est plus performant, par conséquent il vaut mieux utiliser celui de Dijkstra.

**Conclusion générale**

Cette SAE nous a appris l’utilisation des Map et TreeMap mais également à ne pas sous-estimer les tâches à faire. Nous avons rencontré plusieurs problèmes, notamment sur les algorithmes comme des boucles infinies ou des résultats faux.

Le bilan que nous pouvons de ce travail est qu’il ne faut pas se décourager dû au fait de ne pas réussir les questions car avec un peut de recul, on peut trouver le problème et continuer d’avancer. De plus, cette SAE nous a appris à ne pas sous-estimer les questions car c’est ce que nous avons un petit peut fais pour les algorithmes et au final ce sont les questions qui nous ont posé le plus de problèmes.