OUDIN Clement
DOAN Lancelot
S2C

Rapport SAE Graphe

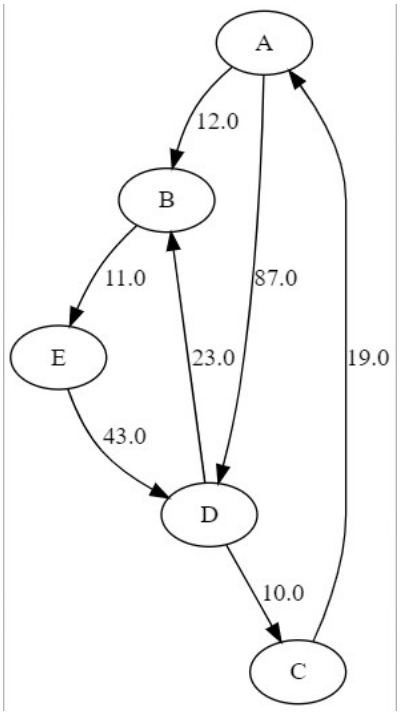
Représentation d'un graphe :

Au cours de cette partie, nous avons mettre en place le début du projet. Pour ce faire, nous avons dû créer les classes Noeud, Arc, GrapheListe, ainsi que l'interface Graphe, qui nous permettront de nous représenter un graphe. L'interface Graphe sera implémentée par la classe GrapheListe qui permettra de manipuler plus facilement les graphes.

Ensuite, nous avons dû créer une classe Main pour pouvoir vérifier si nos classes fonctionnaient bien, ainsi qu'une méthode toString, qui nous permettrait de renvoyer une chaine de caractére pouvant être lu via GraphViz.

Le test réalisé pour pouvoir vérifier la véracité de nos classes a eu pour but de vérifier pour chacuns des noeuds si le noeud qui le précéde est le bon et si le nombre de noeuds contenus dans le graphe était correcte.

A la fin de cette partie, nous devions écrire un constructeur permettant de générer un graphe à partir d'un fichier texte.



Point fixe :

Au cours de cette partie, nous devions tout d'abord réaliser l'algorithme de Bellman Ford.

Question :

[Question 13]

```
Fonction pointFixe(Graphe g InOut, Noeud depart)
   Debut
      Pour chaque sommet v de G faire
         L(v) <- infini
         parent(v) <- null
      Fin Pour
      L(depart) <- 0
      continuer <- vrai
      Tant que continuer faire
         continuer <- false
         Pour chaque arc(x, y, poids) de G faire
            si (L(y) > L(x) + poids alors
               L(y) <- L(x) + poids
               parent(y) <- x
               continuer <- vrai
            fsi
         Fin Pour
      FTQ
   Fin
```

Lexique :
g : Graphe, Un graphe sur lequel on va se baser
depart : Noeud, le noeud par lequel on va commencer
v : entier, indice d'iteration
continuer : booleen, condition d'arret de la boucle
poids : entier, la distance entre deux noeuds

Ensuite, nous avons du manipuler la classe Valeur, qui aura été fournie au préalable, pour pouvoir écrire le code de la méthode de résolution de graphe de point fixe.
Un main aura été réalisé, pour pouvoir vérifier si les valeurs des arcs sont les bonnes.

Le test unitaire réalisé portera sur la vérification des noeuds, de la valeur des arcs, et de la vérification de prédécesseurs.

Pour conclure cette partie, il était demandé d'écrire le code de la méthode calculerChemin(String destination), qui correspondra au chemin de noeuds pour arriver à destination.

Dijkstra :

Cette partie consistera à de la traduction de l'algorithme de Dijkstra à partir de l'algorithme donné dans le polycopié.

Le test réalisé sera le même que pour la méthode de point fixe.

A la fin de cette partie, le but aura été de réaliser une classe MainDijkstra, dont le but est de pouvoir lire un graphe à partir d'un fichier texte.
Il sera aussi présenté un calcul des chemins les plus courts ainsi que l'affichage des chemins pour des noeuds donnés.

Validation et expérimentation :

[Question 21]

On pourra remarquer que la différence entre ces deux algorithmes est que, lors de l'utilisation du point fixe, les valeurs de chaques noeuds ne sont pas les valeurs définitives au cours des itérations. Elles peuvent changer à tout moment.

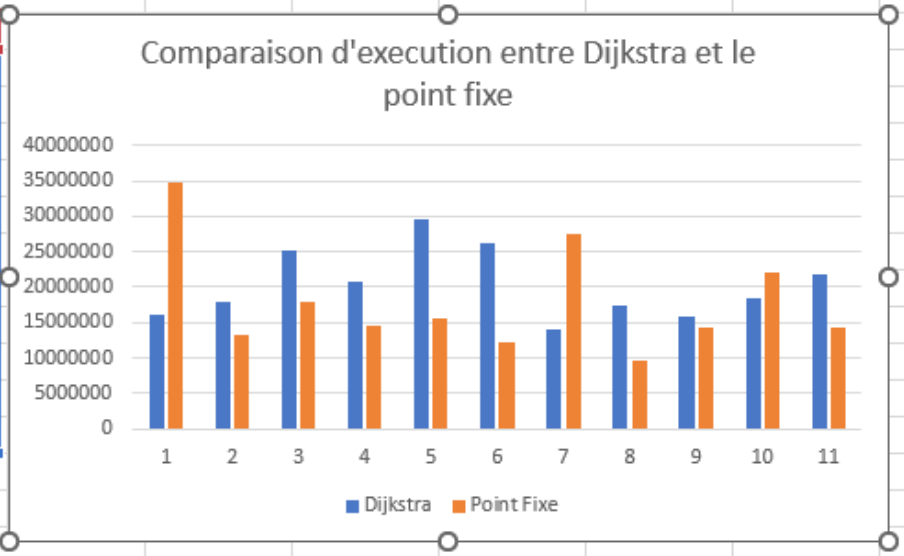
Lors de l'utilisation de l'algorithme de Dijkstra, les valeurs qui sont retenus sont les valeurs définitives de chaque noeuds.
De ce fait, dans certains cas, l'algorithme de Dijkstra est le plus rapide.

[Question 22]

On pourra donc retirer comme conclusion que l'algorithme de Dijkstra est plus rapide de l'algorithme de point fixe étant donné qu'il nécessite moins d'itération.

[Question 23]

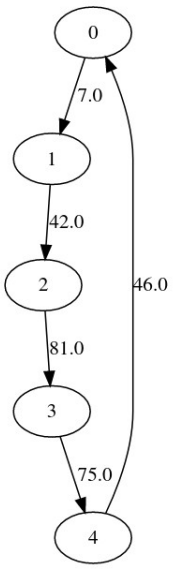
Dijkstra	Point Fixe
15995100	34793500
17982200	13320000
25211200	17781800
20845300	14546700
29452000	15458700
26252800	12112300
13994300	27481000
17404300	9567300
15726200	14377300
18432400	22050200
21727700	14238100

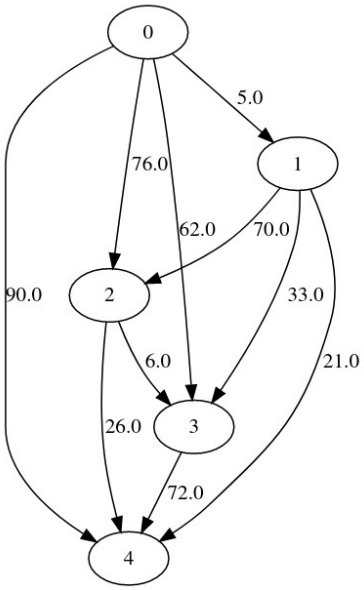


Grace à la méthode System.nanoTime, nous avons pu comparer la rapidité des deux méthodes. Ainsi, la méthode de point fixe semble être la plus efficace de manière générale au cours des divers itérations réalisées.
Ainsi, la méthode de point fixe semble être la plus efficace parce qu'elle met moins de temps à s'exécuter.

Ensuite, il a fallu générer des graphes de façon aléatoire.
Pour cela, nous avons du créer une classe GenererGraphe qui implémentera l'interface Graphe.

<u>Exemples de graphes générés aléatoirement :</u>



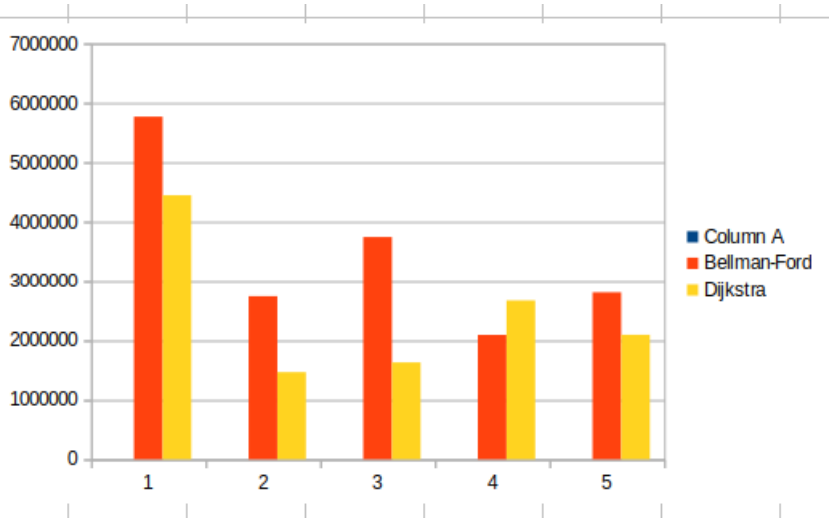


[Question 26]

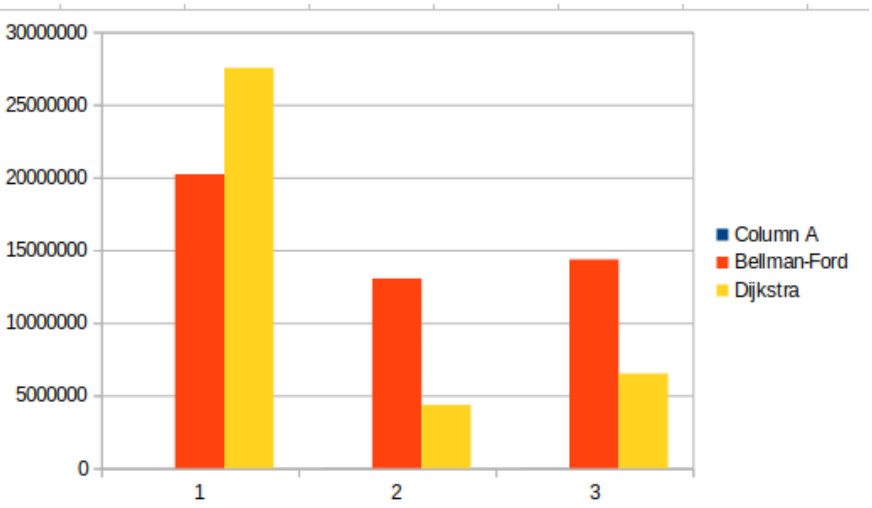
Pour comparer les deux algorithmes, nous avons réalisé plusieurs itérations sur un graphe ayant peu de noeuds, et une itération sur un graphe avec plusieurs noeuds.

Ainsi, nous avons pu constater que pour un graphe avec peu de noeuds, le parcours de graphe à l'aide du point fixe est plus long que le parcours de graphe à l'aide de Dijkstra.

	Bellman-Ford	Dijkstra
0	5766505	4444977
1	2745923	1468058
2	3741693	1632368
3	2094563	2675449
4	2812184	2096462
	17160868	12317314
	0,33602641778	0,360872265



	Bellman-Ford	Dijkstra
0	20216096	27532551
1	13037991	4348724
2	14354043	6498629
	47608130	38379904
	0,42463537215	0,717368939



On pourra remarquer que la méthode de Dijkstra est aussi la plus rapide lorsqu'il s'agit d'un graphe contenant plusieurs noeuds.

[Question 27]

Ratio de performance de Bellman-Ford avec peu de noeud : 0,33
Ratio de performance deDijkstra avec peu de noeud : 0,36

Ration de performance de Bellman-Ford avec beaucoup de noeud : 0,42
Ration de performance de Dijkstra avec beaucoup de noeud : 0,71

Ce rapport n'est pas constant, il dépend donc du nombre de noeuds.
En effet, on a pu voir que le ratio de performance n'était pas le même dans ces deux situations.

[Question 28]

On peut donc tirer comme conclusion de cette étude que l'algorithme de Dijkstra sera le plus efficace dans la majorité des situations.
Cependant, l'algorithme de point fixe peut s'avérer plus efficace, mais cela dépendra du nombre de noeuds, mais aussi éventuellement du nombre d'arcs.

Conclusion :

Au cours de cette SAE, nous avons donc appris que selon l'algorithme utilisé, le temps d'exécution sera plus ou moins long. Le temps d'exécution va aussi varier selon la taille du graphe, ce qui influence grandement celui-ci.

Il n'y a eu qu'une grande difficultée durant cette SAE, celle-ci étant la réalisation de l'algorithme de point fixe, notamment pour la compréhension de celui-ci.
La traduction d'algorithme n'était pas un réel problème, cette partie étant juste chronophage.