Ouder Nathan
Mathieu Gabin
S2D

Rapport de la saé 2.2 : exploration algorithmique d’un problème

Partie 1 : représentation d'un graphe

Dans cette partie, nous avons créé les classes Arc, Arcs, et GrapheListe, ainsi que l'interface Graphe. Nous avons testé l'insertion de noeuds et d'arcs dans le graphe, ainsi que sa mise en forme dans la méthode toString().

Partie 2 : calcul du plus court chemin par point fixe

Nous avons créé la classe BellmanFord contenant une méthode de parcours de graphe par point fixe. La méthode a ensuite été testée dans la classe de tests TestBellman, où nous avons vérifié si la méthode fonctionnait dans des graphes classiques, si le graphe ne contenait pas de noeud, si le noeud de départ n'existait pas, si le noeud de départ n'avait pas de suivants, ou si il existait plusieurs chemins ayant le même coût.

Question 8 :

Voici l'algorithme que nous avons créé pour parcourir un graphe avec la méthode du point fixe :

```
fonction PointFixe(Graphe g InOut, Noeud depart) :

début :
    idDepart <- 0
        tant que idDepart < longueur(g.listeNoeuds())-1 et
g.listeNoeuds()[idDepart] != depart faire :
            idDe
    part <- idDepart + 1
    fpour
    j <- 0
    g.valeurs[idDepart].valeur <- 0
    pour j de 0 à longueur(hg.valeurs)-1 faire :
        g.valeurs[j].valeur <- +∞
    fpour

    pour i de 1 à longueur(g.listeNoeuds())-1 faire :
        pour chaque arc dans g.tousLesArcs() faire :
            si g.valeurs[arc.depart].valeur + arc.cout <
g.valeurs[arc.destination].valeur alors :
                g.valeurs[arc.destination].valeur <-
g.valeurs[arc.depart].valeur + arc.cout
                g.valeurs[arc.destination].parent <- arc.depart
            fin si
        fin pour
    fin pour

    pour chaque arc dans g.tousLesArcs() faire :
```

```
        si g.valeurs[arc.depart].valeur + arc.cout <
g.valeurs[arc.destination].valeur alors :
            retourner "Le graphe contient un cycle de poids négatif"
        fin si
    fin pour

    retourner g.valeurs

fin PointFixe()
```

Partie 3 : calcul du meilleur chemin par Dijkstra

Pour cette partie, nous avons essentiellement fait la même chose qu'avec la méthode de bellman-Ford et avons effectués les tests suivants : (mettre tests)

Partie 4 : validation et expérimentation

Question 16 :

Pour comparer les deux méthodes, nous avons fait parcourir 50 graphes de 250 noeuds disposés aléatoirement à chacun des algorithmes. Les résultats sont les suivants.

Pour la méthode de Bellman-Ford (sortie de la console) :

```
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1419
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1233
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1257
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1185
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1185
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1196
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1249
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1201
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1220
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1191
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1275
ms
```

Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1232 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1205 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1248 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1187 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1231 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1187 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1177 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1202 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1204 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1189 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1175 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1202 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1171 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1180 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1171 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1170 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1182 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1198 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1163 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1181 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1217 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1202 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1177 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1166 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1182 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1216 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1172 ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1245 ms

```
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1252
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1326
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1250
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1176
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1178
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1168
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1169
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1181
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1226
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1183
ms
Temps d'exécution de l'algorithme de BellmanFord sur un grand graphe : 1200
ms
l'algorithme de BellmanFord a pris en moyenne 1207.04 ms pour 50 tests
```

Pour l'algorithme de dijkstra (sortie de la console) :

```
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 23 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 10 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 11 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 10 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 12 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 12 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 11 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 12 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 11 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 11 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 10 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 10 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 7 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 6 ms
```

```
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 7 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 10 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 7 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 10 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 6 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 7 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 10 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 7 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 7 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 9 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 8 ms
Temps d'exécution de l'algorithme de Dijkstra sur un grand graphe : 7 ms
l'algorithme de dijkstra a pris en moyenne 9.04 ms pour 50 tests
```

Ainsi nous avons pu remarquer que l'algorithme de Dijkstra est bien plus performant que celui de Bellman-Ford pour un grand graphe généré aléatoirement (nb : les graphes devaient contenir 1000 noeuds mais l'algorithme de Bellman-Ford mettait plus de 10 secondes pour afficher les résultats d'un seul test tandis que celui de Dijkstra mettait environ deux dixièmes de secondes)

Question 17:

D'après les résultats, c'est l'algorithme de Dijkstra qui est le plus performant.
En effet, cet algorithme ne passe pas en revue tous les points même si ceux-ci ont déja été assignés à leur chemin minimal contrairement à celui de Bellman-Ford qui vérifie tous les points à chaque mise à jour du graphe.

Question 18 :

Selon nous, l'algorithme qui devrait être le plus efficace est celui de Dijkstra, pour les mêmes raisons que nous avons évoqué dans la question précédente.

Conclusion générale :

Cette saé nous a appris à comparer la vitesse d'exécution de deux algorithmes dans des graphes. Le déroulement de la saé s'est déroulée sans difficulté majeure, mis à part l'implémentation de la méthode de Bellman-Ford qui a posé quelques soucis.