Année 2018 - 2019

BENNOUR Alexandre

KLEIN Christopher

Rapport du projet d’Algorithmes et Complexité

Ce rapport rend compte du projet d’algorithmes et complexité de Master Informatique 1 de BENNOUR Alexandre et KLEIN Christopher. Ce projet consistait à l’implémentation de différents algorithmes choisissant un arbre couvrant d’un graphe. De plus, nous utilisons ces algorithmes pour une application ludique, les labyrinthes.

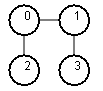
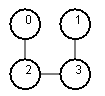
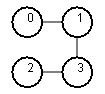
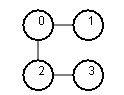
Ce rapport contient les réponses aux différentes questions posés par nos enseignants ainsi qu’une explication de la répartition du travail dans le groupe.

Vous pouvez retrouvez les sources java de notre projet sur le dépôt git suivant :

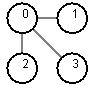
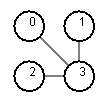
<https://github.com/christo57/Algo>

Question 1 :

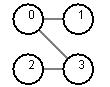
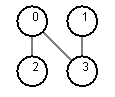
Notre graphe représente un carré avec une diagonale. Donc on peut déjà obtenir 4 arbres couvrants en retirant la diagonale et une arrête du carré à chaque fois.



On a ensuite deux arbres couvrants en prenant la diagonale et deux arrêtes adjacentes.

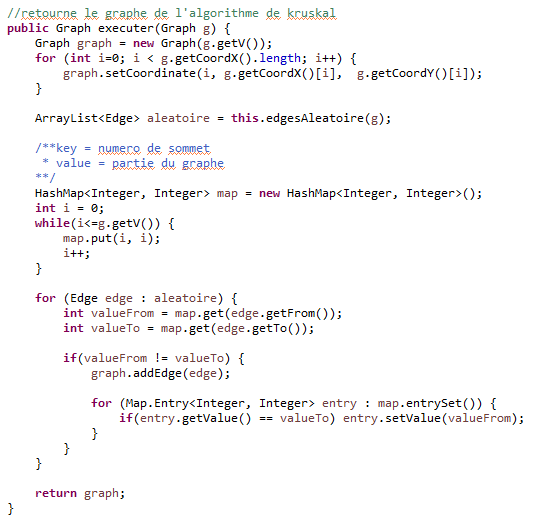


Et pour finir deux autres en prenant la diagonale et les arrêtes opposées.

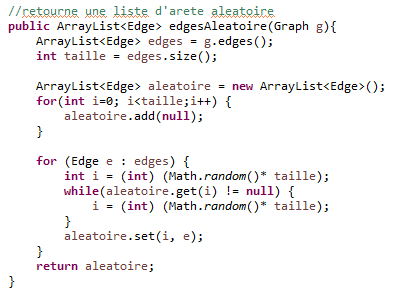


Ce qui nous fait un totale de 8 arbres couvrants pour le graphe G1

Question 2 :

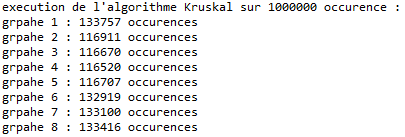
Voici l’implémentation de l’algorithme de Kruskal : 

La fonction edgesAleatoire(Graph) est une fonction que nous avons écrit pour obtenir une liste aléatoire d’arrêtes qui sont dans le Graph passé en paramètre :



Question 3 :

En testant l’algorithme un million de fois sur le graph G1 et en comptant l’apparition de chaque arbre couvrant obtenu, on obtient les résultats suivants :



Donc sur les 8 arbres couvrants obtenus on voit bien que les fréquences d’apparitions ne sont pas les même, ils n’ont pas tous la même probabilité d’apparaitre.

Q4)

JE SAIS PAS PROUVER

Si on prend le graphe G1 :

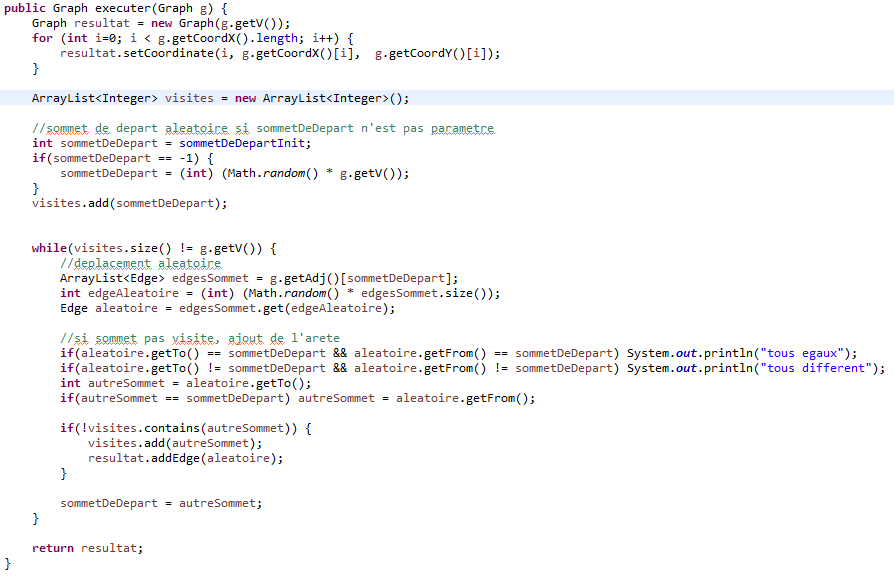


Il y a 5 arrêtes : (0-1),(0-3),(0-2),(1-3),(2-3).

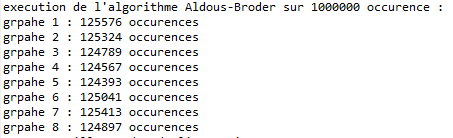
Donc 1/5 chance pour chaque arrête d’être prise en première.

Question 5 :

Voici l’implémentation de l’algorithme d’Aldous-Broder :



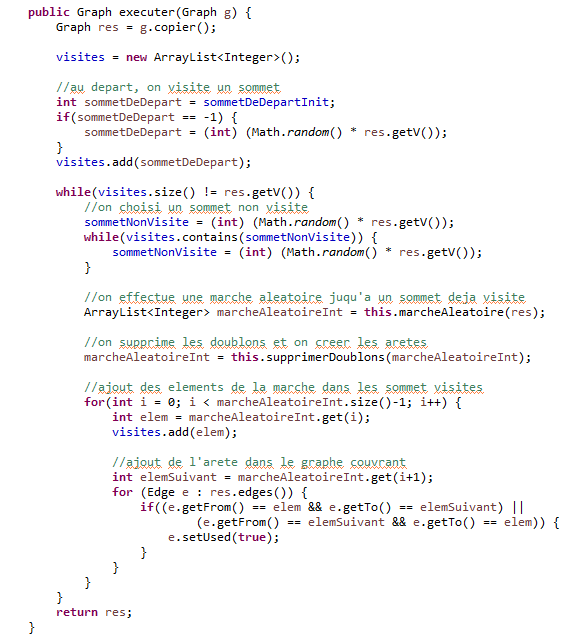
En testant l’algorithme un million de fois sur le graph G1 et en comptant l’apparition de chaque arbre couvrant obtenu, on obtient les résultats suivants :



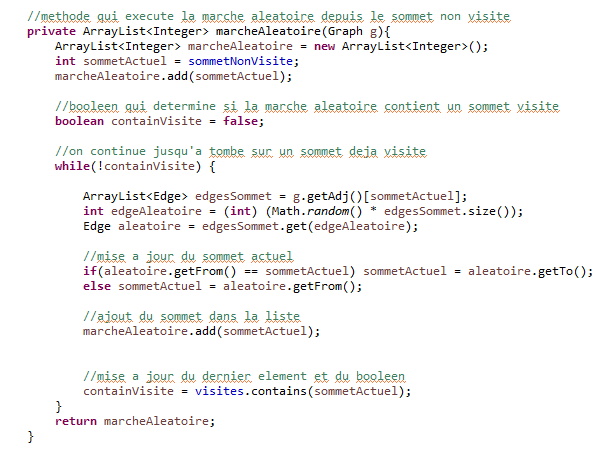
Donc sur les 8 arbres couvrants obtenus on voit bien que les fréquences d’apparitions sont sensiblement les mêmes, donc avec l’algorithme d’Aldous-Broder, on obtient des arbres couvrant avec une probabilité équivalente d’apparaître.

Question 6 :

Voici l’implémentation de l’algorithme d’Aldous-Broder :



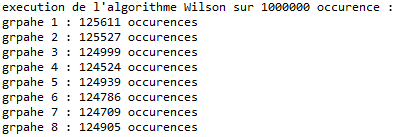
Cette méthode utilise plusieurs autres fonctions : la fonction marcheAleatoire(Graph g) qui permet d’exécuter une marche aléatoire depuis le sommet non visité jusqu’à un sommet déjà visité :



De plus, on utilise aussi une autre fonction, supprimerDoublons(ArrayList<Integer> liste) qui permet de supprimer tous les doublons d’une liste d’entier. Cette fonction utilise aussi une autre fonction, getDoublon(ArrayList<Integer> liste) qui permet de récupérer le premier doublons de la liste entrée en paramètres.



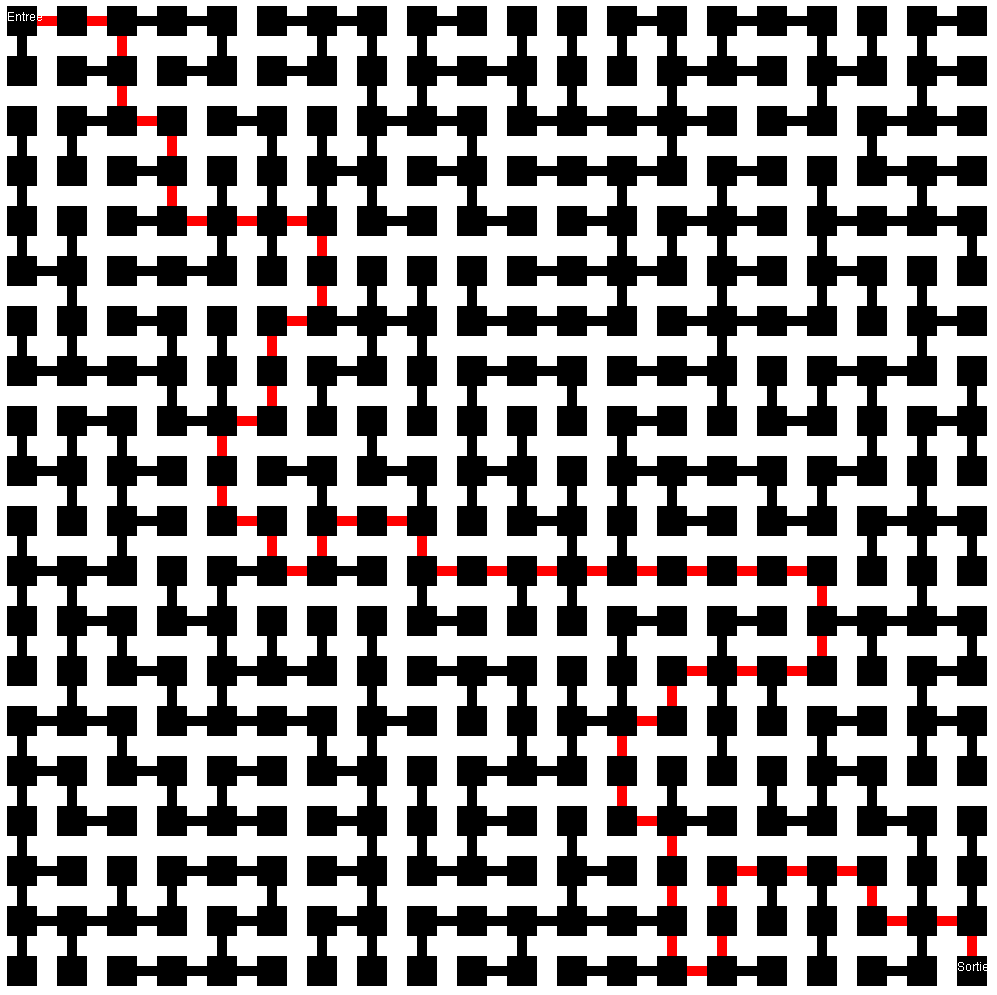
En testant l’algorithme un million de fois sur le graph G1 et en comptant l’apparition de chaque arbre couvrant obtenu, on obtient les résultats suivants :



Donc sur les 8 arbres couvrants obtenus on voit bien que les fréquences d’apparitions sont sensiblement les mêmes, donc avec l’algorithme de Wilson, on obtient des arbres couvrant avec une probabilité équivalente d’apparaître.

Question 7 :

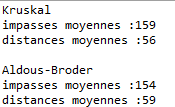
Voici un labyrinthe généré à partir d’un arbre couvrant d’un graphe de 20x20 avec l’algorithme de Kruskal :



Le chemin rouge correspond au chemin entre l’entrée et la sortie.

Question 8 :

En testant avec 1000 labyrinthes générés par l’algorithme de Kruskal et 1000 autres générés par l’algorithme d’Aldous-Broder, on obtient les moyennes suivantes :



Kruskal donne une distance inferieur à Aldous-Broder mais fournit plus d’impasses.