Année : 2018 - 2019

KLEIN Chirstopher

BENNOUR Alexandre

Rapport : Projet d’Algorithmes

et Complexité

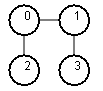
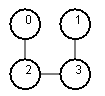
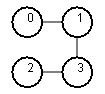
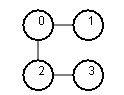
Dans ce rapport, nous allons répondre aux différentes questions qui nous ont été posé par nos enseignants et expliquer en particulier la répartition du travail dans le groupe

Vous pouvez retrouvez les sources java du projet sur notre dépôt git a l’url :

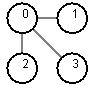
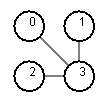
https://github.com/christo57/Algo

Question 1 :

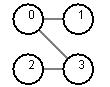
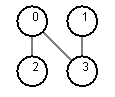
Notre graphe représente un carré avec une diagonale. Donc on peut déjà obtenir 4 arbres couvrants en retirant la diagonale et une arrête du carré à chaque fois.



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

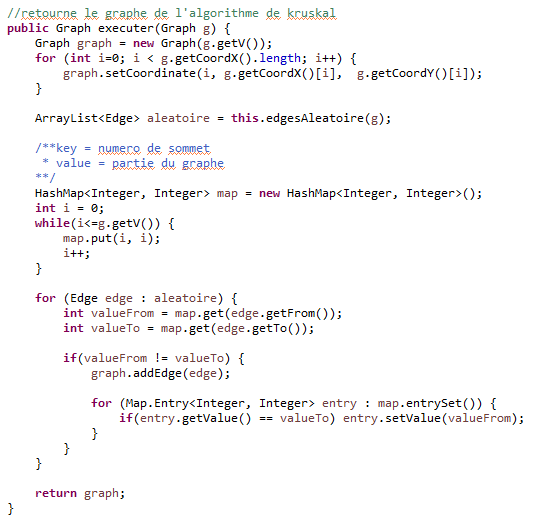


Et pour finir deux autres en prenant la diagonale et les arrêtes opposées.

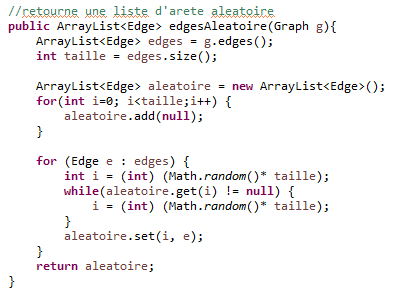


Ce qui nous fait un totale de 8 arbres couvrants pour le graphe G1

Question 2 :

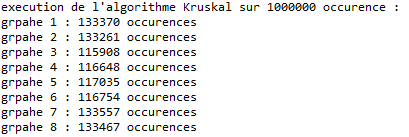
Voici l’implémentation de l’algorithme de Kruskal : 

La fonction edgesAleatoire(Graph g) 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 :

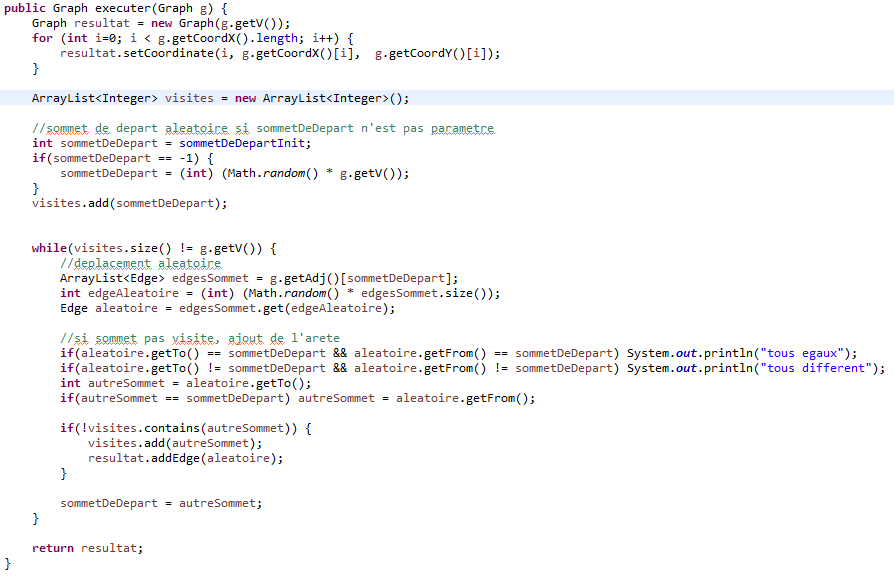


Il y a 5 arrêtes : (0-1),(0-3),(0-2),(1-3),(2-3).

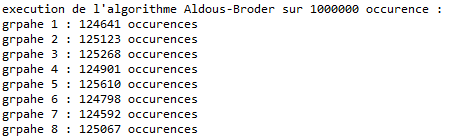
Donc 1/5 chance pour chaque arrête d’être prise en première.

Question 5 :

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



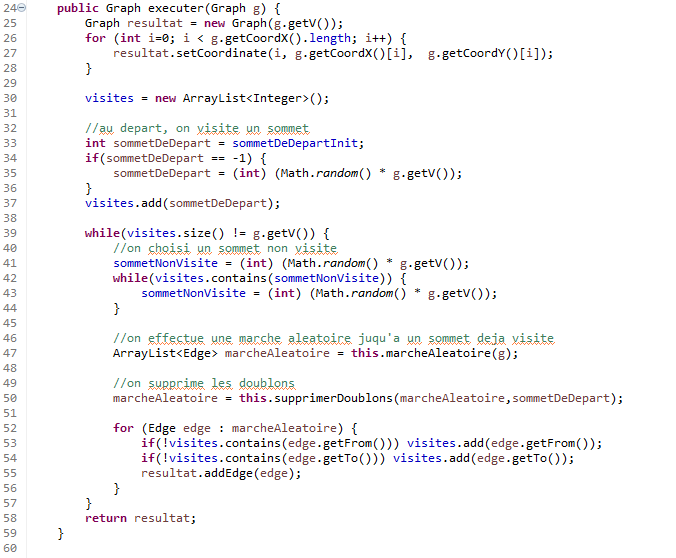
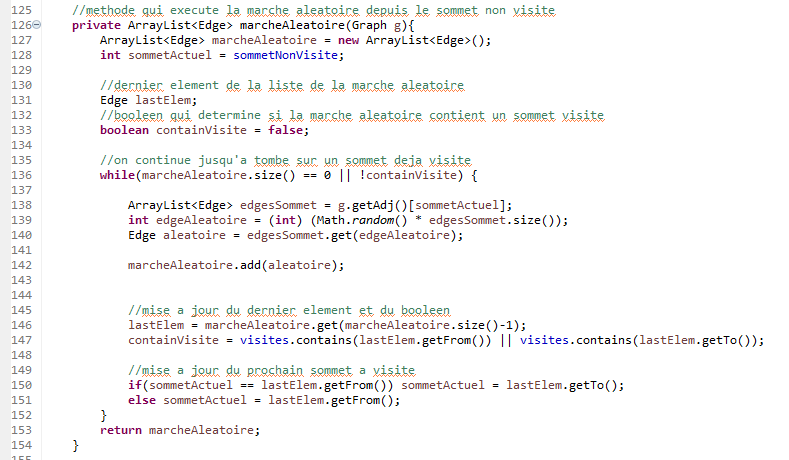
En testant l’algorithme un million de fois sur le graph G1 et en comptant l’apparition de chaque arbre couvrant obtenu, on obtient les résultats suivants :

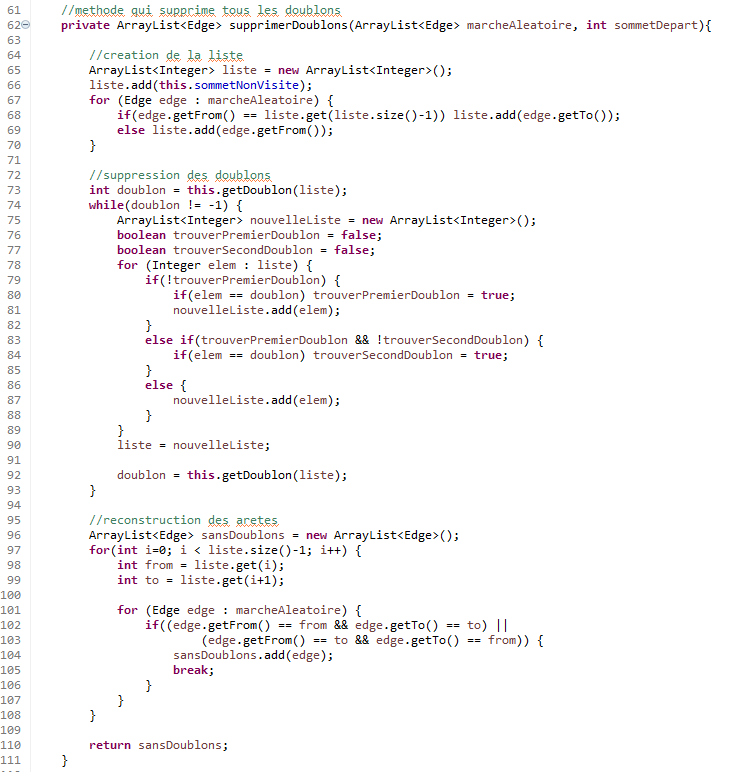


Donc sur les 8 arbres couvrants obtenus on voit bien que les fréquences d’apparitions sont sensiblement les mêmes, donc avec l’algorithme d’Aldous-Broder, on obtient des arbres couvrant avec une probabilité équivalente d’apparaître.

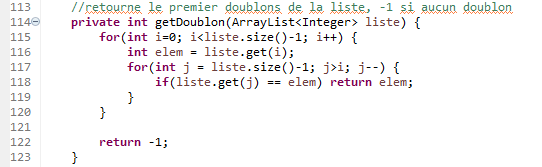
Question 6 :

Voici l’implémentation de l’algorithme de Wilson:

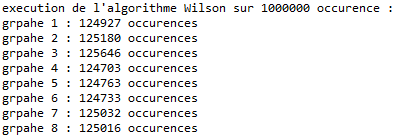
La fonction marcheAleatoire(Graph g) permet d’obtenir la marche aléatoire depuis le sommet non visité choisi jusqu’à un sommet déjà visité :

La fonction supprimerDoublons(ArrayList<Edge> marcheAleatoire, int sommetDepart) permet de supprimer tous les doublons de la marche aléatoire faite :

Cette méthode utilise aussi une autre méthode, getDoublon(ArrayList<Integer> liste) qui retourne le premier doublons de la liste :



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 :

