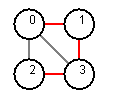
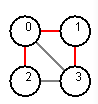
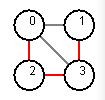
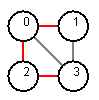
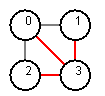
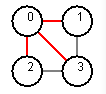
Q1)

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. Les arrêtes rouges correspondent à l’arbre couvrant trouvé.



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

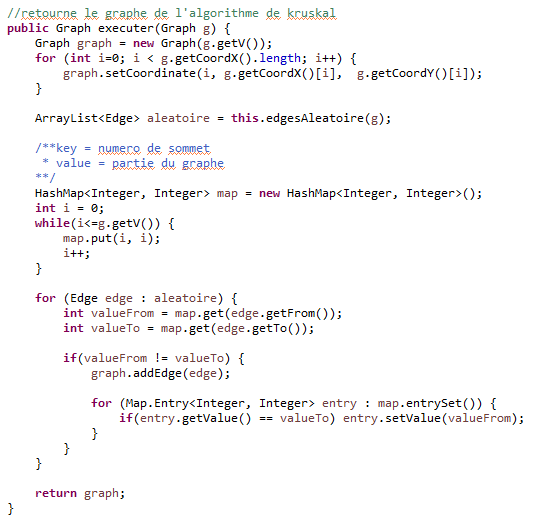


Et pour finir deux autres en prenant la diagonale et les arrêtes opposées.

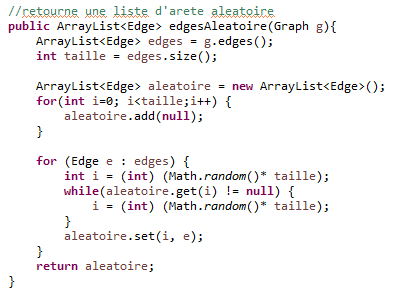


Ce qui nous fait un totale de 8 arbres couvrants pour le graphe G1

Q2)

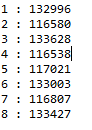
Voici l’implémentation de l’algorithme de Kruskal : 

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 :



Q3)

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 :

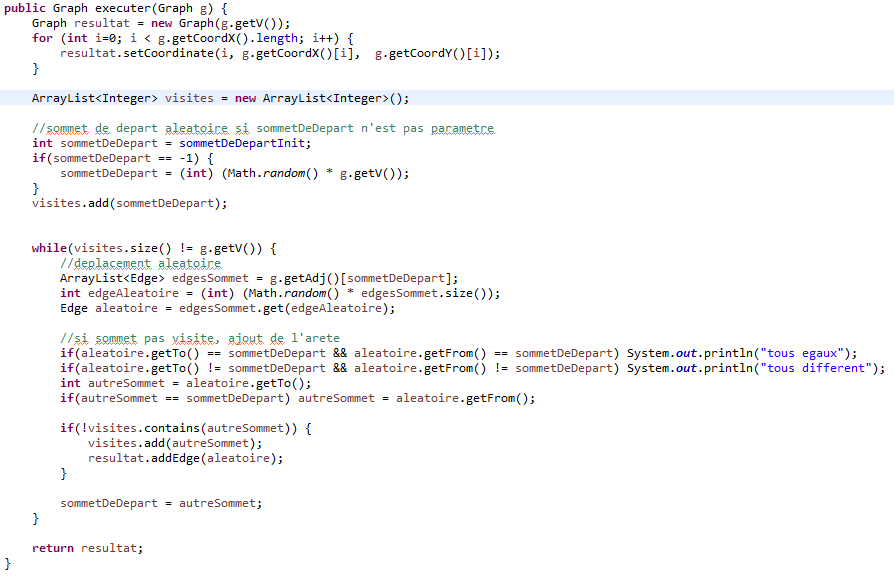


Il y a 5 arrêtes : (0-1),(0-3),(0-2),(1-3),(2-3).

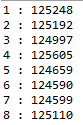
Donc 1/5 chance pour chaque arrête d’être prise en première.

Q5)

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 :

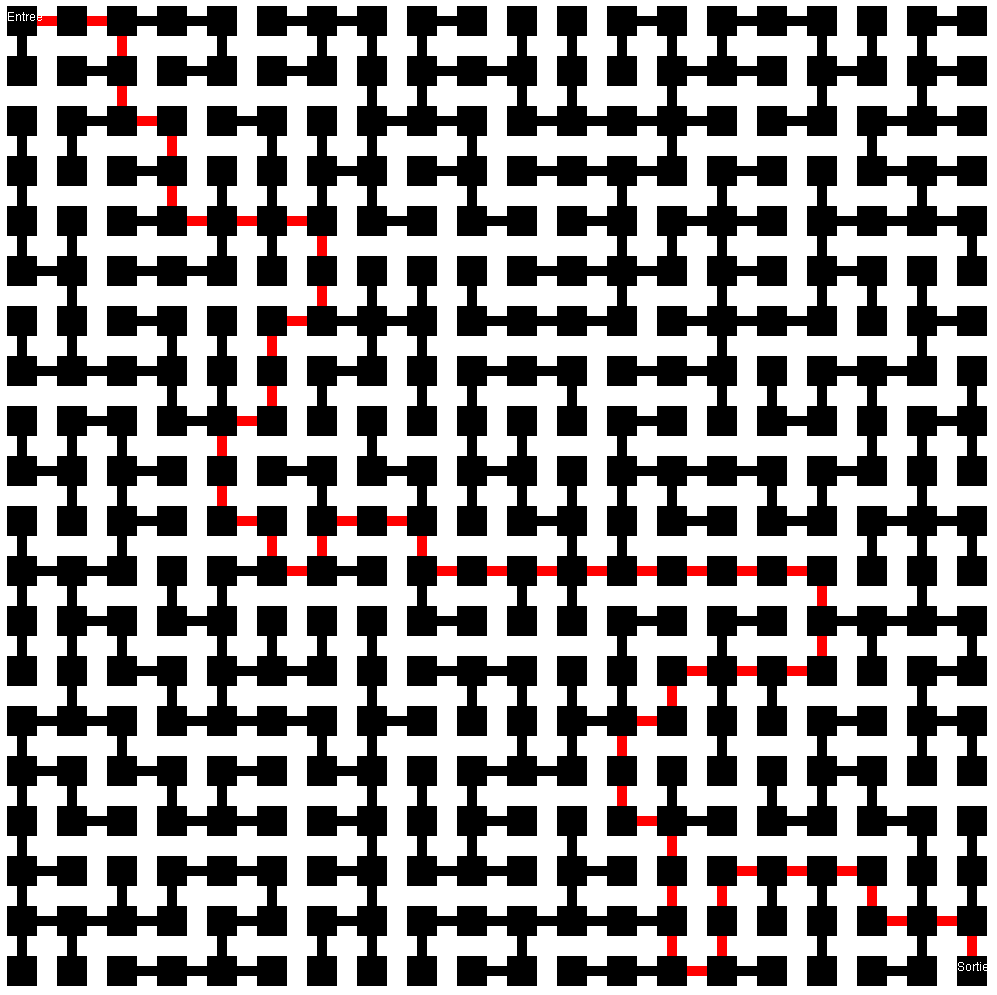


Donc sur les 8 arbres couvrants obtenus on voit bien que les fréquences d’apparitions sont sensiblement les mêmes, donc avec l’algorithme d’Aldous-Broder, on obtient des arbres couvrant avec une probabilité équivalente d’apparaître.

Q6)

Q7)

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

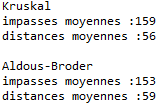
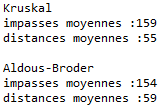
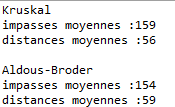


Le chemin rouge correspond au chemin entre l’entrée et la sortie.

L’algorithme qui permet de compter le nombre d’impasses et la distance vers la sortie consiste à dans un premier temps supprimer les premières impasses et les compter. Ces impasses correspondent aux nœuds (sommets du graph) qui n’ont qu’une seule arrête sans prendre en compte l’entrée et la sortie. Ensuite tant qu’il reste ce type de nœuds on continu à les supprimer, cette fois ci sans le compter car ils font partie des premières impasses dénombrées. Une fois toutes les impasses supprimées, on obtient le chemin qui mène de l’entrée à la sortie et il suffit de compter le nombre d’arrêtes pour avoir la distance.

Q8)

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.