Projet

Algorithmique et Complexité

Arbres couvrants

**M1 Informatique**

KIMM Raphaël

ROTON Théo **2020-2021**

Table des matières

[Questions 3](#_Toc60255450)

[Question 1 3](#_Toc60255451)

[Question 2 4](#_Toc60255452)

[Question 3 5](#_Toc60255453)

[Question 4 6](#_Toc60255454)

[Question 5 6](#_Toc60255455)

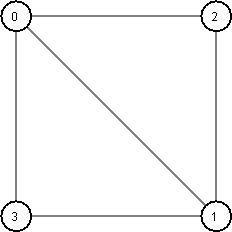
[Question 6 8](#_Toc60255456)

[Sources 10](#_Toc60255457)

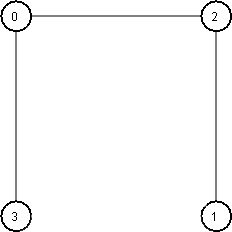
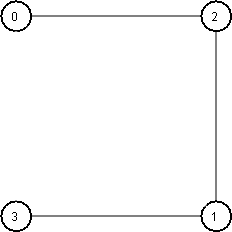
# Questions

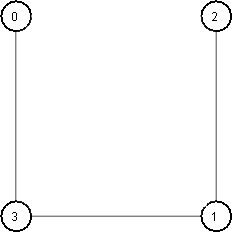
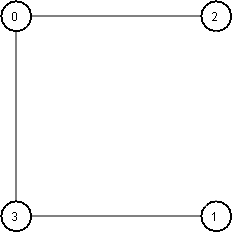
## Question 1

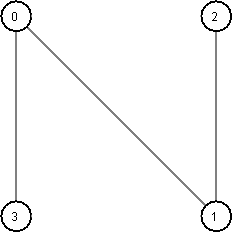
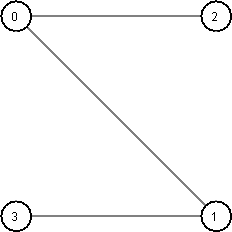
Nous avons le graphe G1 suivant :

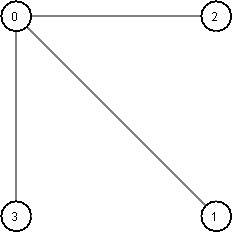
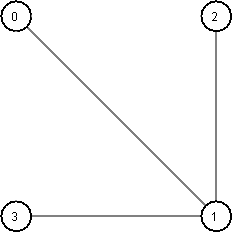


On peut trouver, pour ce graphe, 8 arbres couvrants différents. Voici les différents arbres que l’on peut avoir pour le graphe G1 :

**1.  2.

3. 4. 

5. 6. 

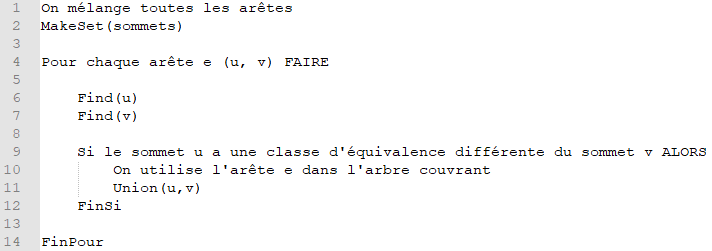
7.  8. 

Nous avons mis un numéro à chaque arbre couvrant (de 1 à 8). Ces numéros nous seront utiles dans les prochaines questions pour faire référence à ces arbres.

## Question 2

Dans cette question nous allons implémenter l’algorithme de Kruskal avec la structure « Union-Find » où on utilise un tableau d’entier pour faire la référence vers le nœud père. Nous allons utiliser 3 fonctions venant de la structure « Union-Find » :

* MakeSet(sommets) : qui initialise le tableau d’entier des références pour chaque sommet. Chaque sommet aura comme classe d’équivalence lui-même.
* Find(sommet) : qui détermine à quelle classe d’équivalence appartient le sommet.
* Union(sommet1, sommet2) : qui unit 2 classes d’équivalence en une seule.

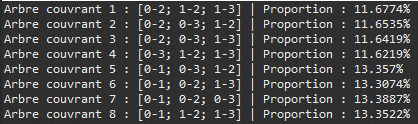
**Pseudo-algorithme :**

L’algorithme de Kruskal a été implémenté dans le fichier « Kruskal.java ».

## Question 3

Dans cette question, nous allons exécuter l’algorithme de Kruskal un million de fois sur le graphe G1 afin de voir la fréquence d’apparition de chaque arbre couvrant. Etant donné que nous avons 8 possibilités d’arbre couvrant pour le graphe G1, la fréquence d’apparition de chaque arbre devrait être d’environ 12,5%.

Après l’exécution de l’algorithme de Kruskal un million de fois, nous obtenons le résultat suivant :



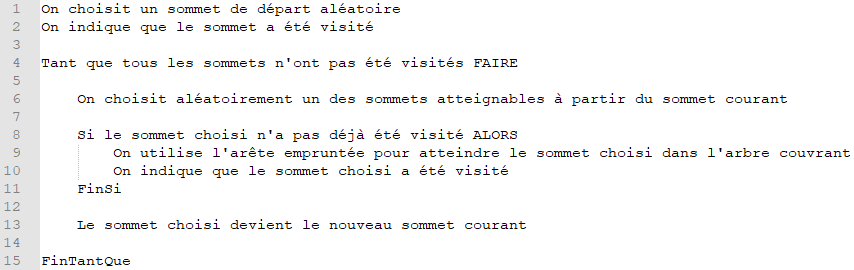
Le numéro de l’arbre couvrant fait référence aux arbres couvrants trouvés dans la question 1. Pour chaque arbre, on a la liste des arêtes le formant entre crochets. Chaque arbre est représenté avec sa proportion d’apparition en utilisant l’algorithme de Kruskal. Le graphique suivant représente la fréquence d’apparition de chaque arbre couvrant :

On peut voir sur ce graphique que les arbres couvrants n’ont pas tous la même probabilité d’apparaître. Si on regarde en détail le graphique, on remarque que les 4 premiers arbres couvrants ont approximativement la même probabilité d’apparaître (≈ 11,65%). On peut aussi remarquer le même phénomène pour les 4 derniers arbres couvrants (≈ 13,35%). En utilisant l’algorithme de Kruskal, on voit donc que tous les arbres couvrants n’ont pas la même probabilité d’être trouvé.

## Question 4

## Question 5

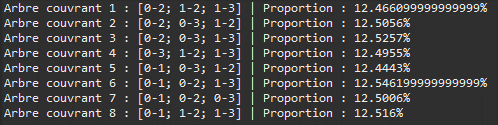
Dans cette question nous allons implémenter l’algorithme d’Aldous-Broder. Cet algorithme repose sur la notion de marche aléatoire : à chaque étape, on choisit un sommet atteignable aléatoirement (chaque sommet atteignable ayant la même probabilité d’être choisis).

**Pseudo-algorithme :**

L’algorithme d’Aldous-Broder a été implémenté dans le fichier « AldousBroder.java ».

Nous allons maintenant exécuter l’algorithme d’Aldous-Broder un million de fois sur le graphe G1 afin de voir la fréquence d’apparition de chaque arbre couvrant. Etant donné que nous avons 8 possibilités d’arbre couvrant pour le graphe G1, la fréquence d’apparition de chaque arbre devrait être d’environ 12,5%.

Après l’exécution de l’algorithme d’Aldous-Broder un million de fois, nous obtenons le résultat suivant :

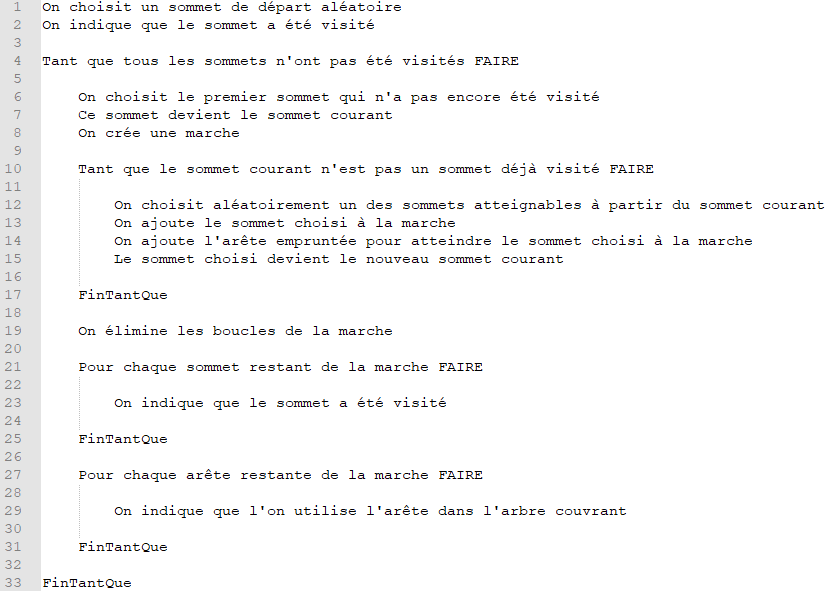


Le numéro de l’arbre couvrant fait référence aux arbres couvrants trouvés dans la question 1. Pour chaque arbre, on a la liste des arêtes le formant entre crochets. Chaque arbre est représenté avec sa proportion d’apparition en utilisant l’algorithme d’Aldous-Broder. Le graphique suivant représente la fréquence d’apparition de chaque arbre couvrant :

On peut voir sur ce graphique que les arbres couvrants ont approximativement la même probabilité (≈ 12,5%) d’apparaître. En utilisant l’algorithme d’Aldous-Broder, on voit donc que tous les arbres couvrants ont (approximativement) la même probabilité d’être trouvé.

## Question 6

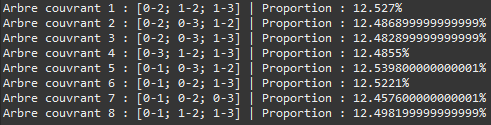
Dans cette question nous allons implémenter l’algorithme de Wilson. Cet algorithme repose aussi sur la notion de marche aléatoire, mais ici, l’algorithme va faire en sorte d’éliminer les boucles de la marche parcourue.

**Pseudo-algorithme :**

L’algorithme de Wilson a été implémenté dans le fichier « Wilson.java ».

Nous allons maintenant exécuter l’algorithme de Wilson un million de fois sur le graphe G1 afin de voir la fréquence d’apparition de chaque arbre couvrant. Etant donné que nous avons 8 possibilités d’arbre couvrant pour le graphe G1, la fréquence d’apparition de chaque arbre devrait être d’environ 12,5%.

Après l’exécution de l’algorithme de Wilson un million de fois, nous obtenons le résultat suivant :



Le numéro de l’arbre couvrant fait référence aux arbres couvrants trouvés dans la question 1. Pour chaque arbre, on a la liste des arêtes le formant entre crochets. Chaque arbre est représenté avec sa proportion d’apparition en utilisant l’algorithme de Wilson. Le graphique suivant représente la fréquence d’apparition de chaque arbre couvrant :

On peut voir sur ce graphique que les arbres couvrants ont approximativement la même probabilité (≈ 12,5%) d’apparaître. En utilisant l’algorithme de Wilson, on voit donc que tous les arbres couvrants ont (approximativement) la même probabilité d’être trouvé.

# Sources