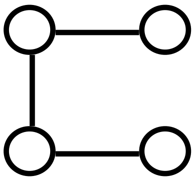
MAHR Benjamin
GINESTRA Alex

Compte Rendu :
Algorithme et Complexité

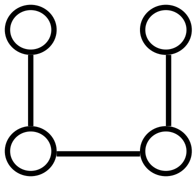
Partie 1:

Q1) Voici les arbres couvrants du graphe G1: (les sommets étant numérotés par ligne)

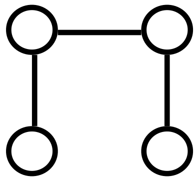
g1:



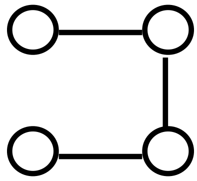
g2:



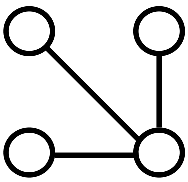
g3:



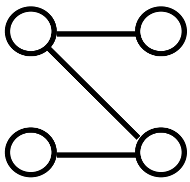
g4:



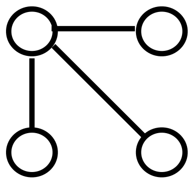
g5:



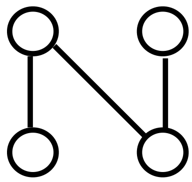
g6:



g7:



g8:

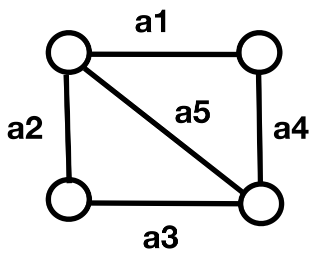


Q2) Cf. méthode algorithmeKruskal( ) de la classe Kruskal.

Q3) Cf. méthode testQ3( ) de la classe Kruskal. (avec g[n-1], n étant le numéro du graphe de la question 1)
Résultats:

Nombre total de graphes classes: 1000000
graphe num 1 : 116628
graphe num 2 : 117160
graphe num 3 : 116713
graphe num 4 : 116735
graphe num 5 : 133305
graphe num 6 : 132938
graphe num 7 : 133189
graphe num 8 : 133332

Q4) Dans l'algorithme de Kruskal, les arêtes sont au départ mélangés de façon aléatoire. Donc il y a autant de probabilité (chance) pour chaque arête d'être choisie en premier (tête de liste). Prenons l'exemple de départ chaque arête a une probabilité 1/5 d'apparaitre en premier dans la liste.
On numérote chaque arête dans notre exemple.



Dans l'algorithme de Kruskal peu importe le résultat du mélange aléatoire, les deux premières arêtes de la liste triés seront ajoutés dans le graphe final.
En effet, nous n'ajoutons pas une arête uniquement si elle produit un cycle. Or un cycle se produit avec au minimum avec 3 arêtes. Donc dans tout les cas, nous ajoutons donc les 2 premières arêtes de la liste triée.

Donc considérons les 2 premières arêtes et regardons les arbres pouvant être formés à partir de celles-ci.
Dans cette exemple, 10 couples d'arêtes distinctes peuvent être ajoutés au départ. On notera la probabilité d'apparition de chaque arbre dans chaque cas (couple d'arête).

Les numéros des arbres correspondent aux différents arbres que nous avons défini à la première page de ce rapport.

Couple de départ (arête n°)	Arbres possibles à partir du Couple	Probabilité d'apparition de chaque arbre pour ce couple
a1 - a2	Arbres : 1,3,7	1/3
a1 - a3	Arbres : 1,4,6	1/3
a1 - a4	Arbres : 3,4	1/2
a1 - a5	Arbres : 6,7	1/2
a2 - a3	Arbres : 1,2	1/2
a2 - a4	Arbres : 2,3,8	1/3
a2 - a5	Arbres : 7,8	1/2
a3 - a4	Arbres : 2,4,5	1/3
a3 - a5	Arbres : 5,6	1/2
a4 - a5	Arbres : 5,8	1/2

On calcule la probabilité totale de chaque arbre d'apparaitre.
On note p1 la probabilité de l'arbre numéro 1 d'apparaitre, et ainsi de suite pour les autres arbres.
On multiplie par 1/10 chacune des probabilité du tableau pour chaque arbre arbre de chaque couple car on a 1/10 que le couple de départ correspondant soit ajouté en premier.

p1 = (1/3 + 1/3 + 1/2)*1/10 = **7/60**
p2 = (1/3 + 1/3 + 1/2)*1/10 = **7/60**
p3 = (1/3 + 1/2 + 1/3)*1/10 = **7/60**
p4 = (1/3 + 1/2 + 1/3)*1/10 = **7/60**

p5 = (1/3 + 1/2 + 1/2)*1/10 = **8/60**
p6 = (1/3 + 1/2 + 1/2)*1/10 = **8/60**
p7 = (1/2 + 1/3 + 1/2)*1/10 = **8/60**
p8 = (1/3 + 1/2 + 1/2)*1/10 = **8/60**

On constate donc que tous les arbres couvrants n'ont pas la même probabilité d'apparaitre.

Q5) Cf. méthode testQ5( ) de la classe AldousBroder. (avec g[n-1], n étant le numéro du graphe de la question 1)
Résultats:

Nombre total de graphes classes: 1000000
graphe num 1 : 124115
graphe num 2 : 125247
graphe num 3 : 125304
graphe num 4 : 125187
graphe num 5 : 124763
graphe num 6 : 125723
graphe num 7 : 124619
graphe num 8 : 125042

Q6) Cf. méthode testQ6( ) de la classe Wilson. (avec g[n-1], n étant le numéro du graphe de la question 1)
Résultats:

Nombre total de graphes classes: 1000000
graphe num 1 : 125020
graphe num 2 : 125439
graphe num 3 : 124942
graphe num 4 : 125335
graphe num 5 : 124796
graphe num 6 : 124327
graphe num 7 : 125124
graphe num 8 : 125017

Q7) Cf. methode testQ7( ) de la classe Labyrinthe. Les fichiers .tex résultants sont nommés « Q7Kruskal.tex » et « Q7Wilson.tex »

Q8)
Résultats pour 1000 Labyrinthe de 20x20:

Kruskal: pas vers la sortie = 49 cul de sac = 119
Wilson: pas vers la sortie = 51 cul de sac = 114

Répartition du travail:

Nous avons réalisé l'ensemble des questions ainsi que la réalisation des algorithmes en duo.