**Introduction**

Dans ce laboratoire, nous devons tester le package algs4. Nous nous concentrons sur trois méthodes spécifiques : eulerianCycle, bipartite et regular. Nous effectuerons nos tests en boîte noire selon la méthode de la catégorie partition. Nous commencerons par EC pour ensuite faire AC. Nous regarderons finalement la couverture des branches des différentes méthodes pour comparer celles-ci. Nous procéderons à obtenir une couverture de 100% avec des tests en boîte blanche.

**Théorie**

Pour bien pouvoir faire nos tests, nous avons discuté des concepts théoriques afin d’en arriver à une compréhension commune.

Graphe

Tout ensemble de sommets et d’arc. Un graphe sans sommet se doit d’avoir aucun arc.

Connexe

Il s’agit de la propriété qui indique s’il est possible d’atteindre tous les sommets à partir de n’importe quel sommet, c’est-à-dire que les arcs relient tous les sommets entre eux.

Cycle eulérien

Pour les cycles eulériens, il s’agit de la capacité à passer à travers tous les arcs et revenir à son point de départ. Contrairement au graphe, celui-ci doit avoir au moins un sommet et un arc afin de pouvoir un cycle. En effet, bien qu’un graphe n’ayant aucun arc soit eulérien, il n’a y pas de cycle. Il n’est pas requis d’être connexe. Toutefois, tous les sommets de degré non nul se doivent d’avoir leur degré pair et de faire partie d’une seule composante connexe.

Biparti

Pour un graphe biparti, nous devons pouvoir séparer les sommets en deux ensembles où aucun arc ne relie des sommets d’un même ensemble. Ainsi, il est évident qu’un graphe sans arc sera biparti puisqu’aucun arc n’est présent pour relier les sommets d’un même ensemble. Il n’est pas requis d’être connexe. De plus, il est admissible d’avoir des ensembles vides (aucun sommet). Toutefois, s’il y a un ensemble vide, aucun arc ne doit être présent.

Régulier

Un graphe régulier est un graphe où tous les sommets ont le même nombre de voisins. Une condition nécessaire et suffisante est que le produit du nombre de sommets et du nombre de voisins est pair ET que le nombre de sommets supérieur au nombre de voisin.

**Partitions**

Eulérien

Il y a deux paramètres en entrée : le nombre de sommets et le nombre d’arcs. Comme mentionné précédemment, il se doit d’avoir un sommet et un arc. Pour l’ensemble des sommets, elle sera divisée en trois partitions : v < 1, v = 1 et v > 1. Pour les arcs, cela sera les trois mêmes partitions que les sommets.

Biparti

Il y a trois paramètres en entrée : le nombre de sommets dans un ensemble et le nombre de sommets dans le second ensemble ainsi que le nombre d’arc. Les partitions pour les sommets sont : v < 0, v = 0 et v > 0. Pour les arcs, un cas additionnel apparaît : celui d’un graphe sans sommet les partitions sont : e < 0, e = 0 et e > 0.

Régulier

Il y deux paramètres en entrée : le nombre de sommets et le degré de ceux-ci. Encore une fois, les conditions limites se situeront à 0 pour le nombre de sommets. Les partitions seront : v < 0, v= 0 et v > 0. Il est à noter que les degrés diffèrent du nombre de voisins. Nous nous contenterons de respecter la contrainte qui stipule que le produit du nombre de sommets et d’arcs soit pair. Celui-ci signifie que la somme des degrés est paire, ce qui est nécessaire puisque que tout arc dans un graphe contribue à augmenter de deux le total de degrés du graphe. L’autre condition peut être respecté en réduisant le nombre de voisins au besoin. En effet, le lien entre le nombre de voisins et le degré est que le nombre de voisins est limité par le nombre de degré. De plus, s’il n’y de sommets, le degré doit être nul. Donc la partition résultante est : k < 0, k = 0, k : V\*k pair et k : V\*k impair. Il serait possible de séparer la partition k = 0 en fonction de V nul et positif, mais la partition k = 0 && V = 0 serait absorbé par V \* k pair.

**Tests en boîte noire**

Eulérien

Les valeurs négatives retournent des erreurs. Nous nous attendons à recevoir une erreur de type IllegalArgumentException puisqu’il s’agit de valeur impossible pour nos arguments. Pour la vérification du cycle eulérien, nous avons adapté un code trouvé sur internet [1]. Il consiste principalement à trouver un sommet avec un degré non nul et à parcourir tous les voisins qui n’ont pas été parcouru de façon récursive.

Biparti

Les valeurs négatives retournent des erreurs. Pour la vérification qu’il s’agit d’un graphe biparti, nous avons encore utilisé un code provenant d’internet [2] tout en l’adaptant à nos besoins. Il s’agit de générer une matrice d’adjacence pour ensuite constituer nos deux ensembles.

Régulier

Les valeurs négatives retournent des erreurs, ainsi qu’un produit du nombre de sommets et d’arcs impair. Pour le graphe régulier, nous avons créé une fonction pour qui utilise la méthode adj de la classe graph. Nous calculons le nombre de voisins et s’assurons que tous les sommets ont le même nombre. Nous n’utilisons pas les degrés puisque nous admettons la possibilité d’avoir des boucles et des liens multiples entre deux sommets. Nous soulignons le fait que la condition V\*k pair survient seulement ici puisque dans les autres méthodes, le paramètre est le nombre d’arcs. Ce dernier implique nécessaire un résultat pair puisque tout arc augmente le degré global du graphe de deux.

Une erreur survient dans nos tests. En effet, le cas où il n’y aucun sommet, mais un degré positif non nul ne retourne pas une erreur, mais bel et bien un graphe. Nous considérons qu’il s’agit d’une erreur d’implémentation.

EC

Nous avons considéré les cas qui retournent les erreurs. Pour les valeurs qui retournent qui des valeurs admissibles, nous avons cherché à minimiser le nombre de cas possible. Voici le nombre de cas de tests pour les trois méthodes testées : eulerian = 1 + 1 + 2 = 4, bipartite = 1 + 1 + 1 + 1 + 2 = 6 et regular = 1 + 1 + 2 \* 2 = 6.

AC

Encore une fois, nous avons fait un test distinct pour chaque valeur retournant des erreurs. Il n’est pas pertinent de tester toutes les possibilités avec ces valeurs négatives puisque les autres paramètres sont sans incidence. Voici le nombre de cas de tests pour les trois méthodes testées : eulerian = 1 + 1 + 2 \* 2 = 6, bipartite = 1 + 1 + 1 + 2 \*2 \*2 = 11 et regular = 1 + 1 + 2 \* 2 = 6. On remarque que pour le régulier, les cas EC et AC possèdent le même nombre de tests. Ceci s’explique par le fait que pour deux des quatre partitions de k retournent des erreurs. Ainsi, pour traiter chaque cas valide de la partition de V, il y a seulement la partition V\*k pair qui est admissible.

**Couverture des branches**

Nous n’obtenons pas une couverture de 100% puisque certaines contraintes auxquelles nous ne nous attendions pas surviennent. Par exemple, dans la fonction bipartite, il y a la condition E > V1 \* V2. Cela implique que le graphe retourné doit être simple, soit aucune possibilité d’avoir plusieurs arcs entre deux sommets. Nous ignorerons pourquoi cette contrainte est présente.

[1] https://www.geeksforgeeks.org/eulerian-path-and-circuit/

[2] <https://www.geeksforgeeks.org/bipartite-graph/>

**CFG EulerianCycle**

A screenshot of a cell phone

Description automatically generated

**CFG Bipartite**

**CFG Regular**