

UNIVERSITÉ DE SHERBROOKE

IFT-436

ALGORITHME ET STRUCTURES DE DONNÉES

---

## Devoir 5

---

*Par :*

François BÉLANGER 94 245 437

Jérémie COULOMBE 13 061 991

Geneviève DOSTIE 12 078 306

*Présenté à :*

Richard ST-DENIS

20 juillet 2015

## **Introduction**

Dans le cadre du dernier travail, une étude de performance d'exécution de différents algorithmes pour un même type de problème était à faire. Chaque équipe devait choisir un problème et au moins trois algorithmes pour résoudre le problème choisi. L'équipe a choisi le problème du calcul de l'arbre sous-tendant de coût minimal. Les trois algorithmes sont : Borůvka, Kruskal et Prim. Les trois algorithmes ont été codés en Python avec le logiciel PyCharm.

Dans ce rapport, il sera présenté les outils de travail pour partager le code et faciliter l'avancement du travail, les algorithmes choisis, la conception du générateur d'échantillon pour faire les tests et les résultats de nos tests avec les hypothèses les algorithmes.

## **Outil de travail**

Pour ce travail, le langage choisi est le Python, version 2.7 . C'était un langage que certains connaissaient plus que d'autres, mais simple et rapide à comprendre. De plus, il semblait plus simple de trouver les algorithmes choisis codés afin de se concentrer plus sur les tests et les résultats.

Pour programmer en Python, le programme de programmation choisi est PyCharm. PyCharm donne accès à un outil Git pratique et facile à utiliser contrairement à d'autres programmes comme Visual Studio pour C++. GitHub fut utilisé avant tout pour le partage et faciliter l'accès au code, la distribution de tâches et permettre de garder un historique des changements au cas où il faudrait revenir en arrière.

## **Nos algorithmes**

Le choix des algorithmes fut en fonction de ce qui a été vu en classe et des algorithmes proposés dans l'énoncé du travail. Les algorithmes choisis pour ce travail sont Borůvka, Kruskal et Prim. Ce sont trois algorithmes de stratégie gloutonne, mais avec des complexités différentes que nous présenterons dans nos hypothèses.

Le seul algorithme dont le code a été trouvé sur Internet, c'est celui de Kruskal. Quant à Borůvka et Prim, aucun code acceptable a été trouvé, donc c'était préférable de les écrire pour ne pas perdre trop de temps sur cette tâche. Dans ce

cas-ci, un code raisonnable a été déterminé par le fait qu'il ne demande pas de modification majeur et qu'il est construit de manière optimal.

### Classe DisjointSet

La classe DisjointSet a été créé pour les algorithmes Borůvka et Kruskal, qui est tiré du code de l'algorithme de Kruskal. Celui-ci permet de faire des ensembles de sommets, trouver un sommet et relier 2 sommets ensembles. Cette partie de code était avant tout utilisée pour l'algorithme de Kruskal. Celle-ci a été mise dans une classe pour avoir la possibilité, lorsque possible, de l'utiliser dans d'autre algorithme.

### **Générateur d'échantillon**

Le générateur de graphe a été construit selon la documentation trouvé sur Internet : [https://en.wikipedia.org/wiki/Mersenne\\_Twister](https://en.wikipedia.org/wiki/Mersenne_Twister). Dans le code du générateur de graphe, la fonction « *random* » de Python est utilisée. Contrairement à la fonction de C++, celui-ci est efficace.

### **Expérience de travail**

Il sera présenté dans cette section les hypothèses puis les résultats de des tests. Il sera expliqué la méthodologie pour obtenir les résultats. Il sera considéré  $n$  le nombre de sommets et  $m$  le nombre d'arêtes.

### Hypothèses

D'abord, la complexité des algorithmes devraient être :  $O(m \log n)$  pour Borůvka,  $O(m \log m)$  pour Kruskal et  $O(m \log n)$  pour Prim. Il est à croire qu'étant de même complexité, Borůvka et Prim auront un temps de calcul similaire. Pour ce qui est de Kruskal, il devrait être le plus performant des trois algorithmes par sa différence  $\log m$ , car il y aura au plus  $\frac{n(n-1)}{2}$  arêtes.

### Méthodologie

Pour faire le temps de calcul pour un graphe de  $n$  sommets, une moyenne du temps est fait sur une boucle qui exécute 100 fois sur une même taille d'échantillon. Pour ce qui est de l'échantillonnage, il est fait par pas de  $10^i$ , pour  $i = 1, 2, \dots, k$ . Le  $k$  est déterminé selon la rapidité d'exécution de l'ordinateur, choisit dans le laboratoire informatique.

Pour recueillir les données sur le temps d'exécution versus le nombre de sommet, un « package » a été importé.

### Résultats

### **Conclusion**

En conclusion,