IFT 2015 Devoir 3

Liwaa Zebian (20213839)
Tarek Radwan (20231177)

25 juillet 2024

Auto-évaluation

Objectifs

L'objectif de ce projet était de mettre en œuvre un programme en Java pour trouver un Arbre de Recouvrement Minimal (ARM) dans un graphe. Nous avons utilisé l'algorithme de Kruskal pour garantir l'efficacité en termes de temps et de mémoire.

Analyse de la complexité algorithmique

Notre solution utilise l'algorithme de Kruskal pour trouver l'ARM du graphe. L'algorithme de Kruskal peut être divisé en plusieurs étapes avec les complexités suivantes :

1. **Tri des arêtes** : Nous devons trier toutes les arêtes en fonction de leur coût. La complexité de cette étape est $O(m \log m)$, où m est le nombre d'arêtes.
2. **Opérations de l'ensemble disjoint** :

- **Création des ensembles** : Initialisation des ensembles pour chaque nœud, ce qui prend $O(n)$ temps.
- **Recherche et union** : Pour chaque arête, nous effectuons des opérations de recherche et d'union sur les ensembles disjoints. La complexité de chaque opération de recherche ou d'union est pratiquement constante.

Étant donné que nous effectuons ces opérations pour chaque arête, la complexité totale pour cette partie est $O(m)$.

Complexité Totale

La complexité totale de l'algorithme de Kruskal, en combinant les étapes de tri et les opérations sur les ensembles disjoints, est :

$$O(m \log m) + O(m)$$

Puisque la seconde partie est linéaire, la complexité totale est principalement dominée par le terme de tri :

$$O(m \log m)$$

Justification

- **Tri des arêtes** : Nous devons trier les arêtes pour les traiter dans l'ordre croissant de coût. Cette étape utilise un algorithme de tri, comme le tri rapide (quicksort) ou le tri fusion (mergesort), qui a une complexité de $O(m \log m)$.
- **Opérations de l'ensemble disjoint** : Les opérations de création d'ensembles, de recherche et d'union sont essentielles pour vérifier si une arête peut être ajoutée à l'ARM sans créer de cycle. Ces opérations sont réalisées en $O(m)$, car chaque opération est pratiquement constante.

Conclusion

En conclusion, notre algorithme pour trouver l'ARM en utilisant l'algorithme de Kruskal est efficace avec une complexité temporelle de $O(m \log m)$. Cette complexité est raisonnable pour les graphes denses et permet de trouver une solution optimale de manière efficace.