TP3 - IFT2015

Mathias La Rochelle & Tom Stanic

Juillet 2024

1 Analyse de la complexité algorithmique

Nous n'avons pas utilisé la compression de chemin dans cette analyse car cela ferait intervenir la constante d'Ackermann, ce qui dépasse le cadre de ce cours et complexifie l'analyse de manière inutile.

1.1 Principaux facteurs du temps d'exécution

1. **Tri des arêtes par poids croissant** : Cette étape peut être réalisée en $O\Big(|E| \log\big(|E|\big)\Big)$ grâce à des algorithmes de tri efficaces comme le tri par fusion ou le tri rapide.

2. **Initialisation des structures de données pour l'union-find** : Il suffit de faire $|V|$ opérations pour assigner chaque sommet à un noeud distinct. En effet, au début de l'union-find, aucun sommet n'est relié à un autre, ils sont tous indépendants. Cela prend donc $O\big(|V|\big)$.

3. **Opération `find`** : Sans la compression de chemin, le pire cas de `find` pourrait être $O\Big(\log\big(|V|\big)\Big)$ en utilisant seulement l'union par rang, car cela garantit que la profondeur de n'importe quel arbre est au plus $\log\big(|V|\big)$.

4. **Détection des cycles** : Une arête crée un cycle si et seulement si ses deux sommets sont dans la même composante. Cette vérification prend $O\Big(\log\big(|V|\big)\Big)$. En effet, on effectue `find` pour les deux sommets et on regarde s'ils ont le même représentant dans l'union-find.

5. **Opération `union`** : Sans la compression de chemin, mais avec l'union par rang, cette opération se réalise en $O\Big(\log\big(|V|\big)\Big)$. En effet, on effectue deux opérations `find` pour trouver les représentants des ensembles, puis on compare et attache l'arbre de plus petite profondeur sous celui de plus grande profondeur.

1.2 Complexité temporelle en notation grand O

- $O\Big(|E| \log\big(|E|\big)\Big)$ pour le tri initial des arêtes.
- Entre $|V| - 1$ et $|E|$ tours de boucle, chaque tour prenant $O\Big(\log\big(|V|\big)\Big)$ pour les opérations `find` et `union`.
- Donc, $O\Big(|E| \log\big(|V|\big)\Big)$ pour toute la boucle.

En combinant ces étapes, on obtient :

$$O\Big(|E| \log\big(|E|\big)\Big) + O\Big(|E| \log\big(|V|\big)\Big) = O\Big(|E| \log\big(|V|\big)\Big)$$

étant donné que $|E| \in O\big(|V|^2\big)$.

Ainsi, la complexité totale de l'algorithme de Kruskal est :

$$O\Big(|E| \log\big(|V|\big)\Big)$$