Algorithme de Kruskal

Antoine Dequay

21 septembre 2022

Notes

— Prof : Nicolas Markey.

— Leçon : 925, 927.

— Références :

  — Aho, Hopcroft, Ullman, *The Design and Analysis of Computer Algorithms*,

  — Chrétienne pour α.

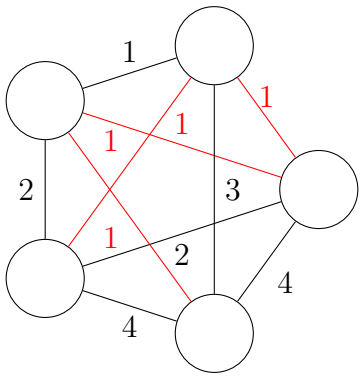
Soit $G = (S, A)$ un graphe non-orienté connexe pondéré par une fonction $\omega : A \longrightarrow \mathbb{R}^*$.

On cherche $B \subset A$ un sous-ensemble d'arrêtes de A qui forme un arbre couvrant de A, tel que $\omega(B) := \sum_{b \in B} \omega(b)$ soit minimal.

C'est à dire qu'on cherche $B \subset A$ tel que (S, B) soit connexe et tel que $\omega(B)$ soit minimum. La propriété d'arbre vient alors du fait que s'il existe un cycle dans B, l'une des arrête est inutile pour assurer la connexité de (S, B), et est de poids strictement positif. La propriété sur la minimisation de ω est alors mise en défaut.

Prenons l'exemple suivant (ici le graphe est complet mais ce n'est pas le cas en général) :

Exemple 1



L'idée la plus élémentaire pour trouver un tel arbre est d'essayer de prendre successivement les arrêtes de plus petit poids pour le construire. On ne les rajoute alors qu'à la condition que le nouvel ensemble créé soit toujours un arbre.

Cela donne l'algorithme suivant, qui utilise la structure d'Union-Find et le paradigme de l'algorithmie gloutonne :

Algorithme 2

```
Entrée : G=(S,A), ω.
Sortie : E un arbre couvrant minimal.
Kruskal(S, A, ω):
    E = ∅
    for s dans S:
        create(s)
    Trier A de façon croissante par rapport à ω
    for (u,v) dans A pris dans l ordre croissant pour ω:
        if Find(u) ≠ Find(v):
            E = E ∪ {(u,v)}
            Union(u,v)
    return E
```

Théorème 3 L'algorithme de KRUSKAL est correct et se termine.

Preuve.

— **Terminaison** :

A et S étant de ensembles finis, c'est immédiat.

— **Correction** :

Soit $G = (A, S)$. On note K l'ensemble renvoyé par l'algorithme de KRUSKAL. Par ce qui précède et la structure de l'algorithme, on sait que c'est un arbre couvrant de G. Montrons qu'il est minimal. Soit E un arbre couvrant minimal de G.

— Si E et K ont les mêmes arrêtes, alors on a le résultat.

— Sinon, soit $q \in K \backslash E$ **minimale pour** ω.

Alors, $E \cup \{q\}$ possède un unique cycle (car E est un arbre couvrant), noté $q, e_1, \ldots, e_k$.

Si pour tout $j \in [\![1, k]\!]$, $e_j \in K$, alors K posséderait un cycle, ce qui est absurde (car K est un arbre). Soit donc $e_{j_0} \in E \backslash K$, on considère $E \backslash \{e_{j_0}\} \cup \{q\}$.

— Si $\omega(e_{j_0}) > \omega(q)$, on a diminué $\omega(E)$ de $\omega(e_{j_0}) - \omega(q)$, ce qui est absurde par minimalité de E.

— Si $\omega(e_{j_0}) < \omega(q)$, e_{j_0} formerait dans $K \cup \{e_{j_0}\}$ un cycle avec des arêtes de K de poids $\leq \omega(e_{j_0})$, par structure de l'algorithme. Or, par choix de q, ces arêtes étant aussi de E, ce cycle serait également dans E. C'est donc absurde.

Ainsi, $\omega(e_{j_0}) = \omega(q)$.

On peut donc progressivement substituer toutes les arêtes de K dans E, ce qui implique bien $\omega(K) = \omega(E)$ (car $|K| = |E| = |S| + 1$, donc le processus est fini).

□

Remarque On vient en fait de prouver que l'invariant de boucle suivant : Si on note A_k les arrêtes déjà examinées à la k-ème itération de la boucle et E_k l'ensemble correspondant à E construit au bout de la k-ème itération de la boucle, alors (S, E_k) est un arbre couvrant minimal de (S, A_k).

Théorème 4 la complexité temporelle de l'algorithme de KRUSKAL est un $O(|S|+|A|log(|A|)+|A|\alpha(|S|))$ dans le cas de la meilleure implémentation de la structure d'Union-Find, où α est la "réciproque" de la fonction $n \longmapsto A(n, n)$, où A est la fonction d'ACKERMANN ($\alpha \leq 5$ en pratique).

Remarque Un autre algorithme de calcul d'un arbre couvrant minimal est celui de PRIM, qui a une complexité temporelle en $O(|S|log(|S|) + |A|log(|A|))$.