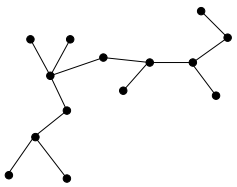
Université Clermont Auvergne
BUT Info 1A
R207 - Graphes
Année 2021-2022

Florent Foucaud
Malika More
Thibault Ralet
Carine Simon
Thierry Trévisan

Graphes
Arbres couvrants de poids minimum

---

1 Arbres



Construction Récursive

- Un graphe avec un seul sommet et zéro arête est un arbre.
- À partir d'un arbre avec au moins 1 sommet, en accrochant une feuille à n'importe quel sommet, on obtient un arbre avec un sommet et une arête de plus.

Remarques

- Un arbre à n sommets possède exactement $n-1$ arête.
- Un arbre avec au moins deux sommets a forcément au moins une feuille.

2 Arbre couvrant

Définition.

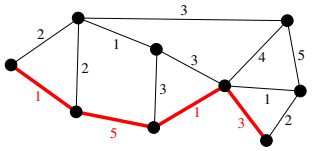
- graphe non orienté G
- arbre T $\begin{cases} \text{sommets : tous les sommets de } G \\ \text{arêtes : certaines arêtes de } G \end{cases}$

Remarques.

1. Un graphe peut avoir plusieurs arbres couvrants.
2. Un graphe non connexe n'a aucun arbre couvrant.
   (Autrement dit, un graphe qui a un arbre couvrant est forcément connexe.)
3. Un arbre n'a qu'un seul arbre couvrant, lui-même.
4. Un graphe connexe a forcément (au moins) un arbre couvrant.

3 Graphe valué

Définition (Graphe valué). Chaque arête a un *poids* (> 0).
Le poids d'une chaîne/d'un arbre est la somme des poids des arêtes qui la/le composent.



$1 + 5 + 1 + 3 = 10$

Question étant donné un graphe valué connexe, on veut construire un arbre couvrant de poids total le plus petit possible.

4 Algorithme de Prim

Méthode

- Initialiser T avec $\begin{cases} \text{sommets : un sommet de } G \text{ qu'on choisit} \\ \text{arêtes : aucune} \end{cases}$
- Répéter :
  - ⋆ Trouver toutes les arêtes de G qui relient un sommet de T et un sommet extérieur à T
  - ⋆ Parmi celles-ci, choisir une arête de poids le plus petit possible
  - ⋆ Ajouter à T cette arête et le sommet correspondant
- S'arrêter dès que tous les sommets de G sont dans T
- Retourner T

L'algorithme est bien défini car si la condition d'arrêt «*Tous les sommets de G sont dans T*» n'est pas réalisée, alors il existe au moins une arête qui relie un sommet v extérieur à T à un sommet de T (car G est connexe). Ce qui signifie qu'il sera possible d'ajouter une arête dans la boucle.

L'algorithme se termine car on ajoute à chaque passage un sommet et une arête, donc au bout d'un nombre fini d'itérations, la condition d'arrêt «*Tous les sommets de G sont dans T*» est réalisée.

L'algorithme renvoie un arbre couvrant Au départ, T est réduit à un sommet, c'est donc un arbre.

À chaque passage dans la boucle, on ajoute une arête et une feuille à l'arbre T, donc l'objet construit demeure encore un arbre.

Finalement, quand on sort de la boucle on retourne T qui est bien un arbre.

Comme l'arbre T obtenu à l'issue de l'algorithme a pour sommets tous les sommets de G et pour arêtes certaines arêtes de G, c'est bien un arbre couvrant de G.

Il reste à vérifier que l'arbre couvrant T est de poids minimal (hors programme). On peut le montrer par récurrence en montrant la propriété ci-dessous, et montrer qu'elle reste vraie à chaque passage dans la boucle : L'ensemble des arêtes choisies par l'algorithme de Prim est contenu dans un arbre couvrant de G de poids minimal. Ceci nous va nous assurer qu'à la fin de l'exécution de l'algorithme, l'ensemble des arêtes composant T est contenu dans un arbre couvrant de G de poids minimal. Comme T constitue lui-même un arbre couvrant de G, c'est celui qu'on cherche.

Initialement, l'ensemble des arêtes choisies par l'agorithme de Prim est vide, donc la propriété est trivialement vérifiée.

Supposons que c'est vrai aux étapes $1, \ldots, i$ mais par l'absurde, supposons que ce n'est pas le cas à l'étape $i + 1$. Donc l'arête $e = (u, v)$ choisie à l'étape $i + 1$ ne fait pas partie d'un arbre couvrant de G de poids minimal. Soit T_i l'arbre obtenu à l'étape i, avec u dans T_i. Comme la propriété est vraie pour les étapes $1, \ldots, i$, toutes les arêtes de T_i font bien partie d'un arbre

couvrant de poids minimal de G, appelé T_0. Considérons un chemin de u à v dans T_0 (il ne passe pas par e). Soit $u \dots x$ la sous-chaîne maximale qui appartient à T_i, aved $e' = xy$ la première arête de la chaîne qui est dans T_0 mais n'est pas dans T_i. On construit l'arbre T_1 à partir de T_0 en enlevant e' et ajoutant e. T_1 est bien un arbre couvrant. On prétend que T_1 est aussi de poids minimal. En effet, l'algorithme de Prim a choisi l'arête e à l'étape $i+1$. Il aurait ou choisir l'arête e'. C'est donc que le poids de e n'est pas plus grand que celui de e'. Donc, le poids de T_1 n'est pas plus grand que celui de T_0, et donc T_1 est de poids minimal. C'est une contradiction car on a supposé que e n'est dans aucun arbre couvrant de poids minimal, or e est bien dans T_1, et donc cette hypothèse était fausse. Donc la propriété voulue est aussi vraie à l'étape $i+1$.

5 Algorithme de Kruskal

Méthode

- Initialiser T avec $\begin{cases} \text{sommets : tous les sommets de } G \\ \text{arêtes : aucune} \end{cases}$
- Traiter les arêtes de G l'une après l'autre par poids croissant :
  - Si une arête permet de connecter deux composantes connexes de T,
    - alors l'ajouter à T
    - sinon ne rien faire
  - Passer à l'arête suivante
- S'arrêter dès que T est connexe
- Retourner T

L'algorithme s'arrête toujours, et renvoie bien un arbre couvrant

Le graphe obtenu T est bien sans cycle car on n'ajoute une arête que pour connecter deux composantes connexes, ce qui ne peut pas créer de cycle, et initialement T n'a pas d'arêtes, donc pas de cycles.

T finit par être connexe car G est connexe, donc en parcourant (au pire) toutes les arêtes, on arrive à connecter tous les sommets.

Donc T est bien un arbre, et de plus l'algorithme s'arrête toujours.

Il reste à vérifier que l'arbre couvrant T est de poids minimal (hors programme). Pour cela on peut procéder par l'absurde : on suppose qu'il existe un arbre couvrant T' de poids minimal, plus petit que T, mais qui a le plus d'arête en commun avec T. On considère la première arête e de T (dans l'ordre où les arêtes ont été ajoutées à T) qui n'est pas dans T'. On ajoute e à T' et on obtient un cycle dans T'. L'une des arêtes e' ($e \neq e'$) du cycle n'est pas dans T (puisque T n'a pas de cycle). On construit T'' à partir de T' en ajoutant e et en enlevant e'. T'' est bien couvrant, et ne peut pas avoir un poids plus petit que T' (qui est minimal) donc le poids de e est au moins celui de e'. Mais comme Kruskal a choisi e et non pas e', leurs poids sont les mêmes et donc T'' est aussi minimal. Mais il a plus d'arêtes en commun avec T que T', ce qui contredit le choix de T' et donc T' n'existe pas.