Notes de cours IFT 2125
L'arbre couvrant minimum.
version 2.3
le 25 mai 2023

# 1 Arbre couvrant minimum

Voici un théorème qui permet de prouver que les algorithmes de Kruskal et de Prim (pour trouver un arbre couvrant minimum d'un graphe connexe pondéré) font ce qu'ils doivent faire.

Commençons par quelques définitions et rappels de notations.

**Définition 1.1** *Soit $X$ un ensemble. Une* partition *de $X$ est une famille $(X_i)_{i\in I}$ de parties (sous-ensembles) de $X$ vérifiant*

- $\cup_{i\in I} X_i = X$
- $X_i \cap X_j = \emptyset$ *quand* $i \neq j \in I$

*On appelle les ensembles $X_i$ les* classes *de la partition.*

L'ensemble d'indices $I$ peut être vu, pour nos fins, comme $\{0, 1, 2, \ldots, k\}$ pour $k \leq |X|$ quand $X$ est fini. On écrit $X \dot\cup Y$ pour indiquer la réunion de $X$ et $Y$ dijsoints.

On aimerait supposer connues les définitions de *graphe*, *chemin*, *cycle*, *arbre* (attention! un arbre n'as pas de racine, ni des enfants gauche et droit) etc., mais on va quand même faire quelques rappels. Notons que pour un ensemble $X$, $\binom{X}{2} = \{\{x, y\} : x \neq y \in X\}$. On suppose que tous nos graphes sont finis.

**Définition 1.2** *Soit $G = (V, E)$ un graphe (rappellons que $E \subseteq \binom{V}{2}$, il n'y a pas de boucles ni d'arêtes multiples).*

- *le graphe $G$ est* connexe *si pour toute partition de $V$ en deux parties non-vides $X_1$ et $X_2$ il existent $x_i \in X_1$ et $x_2 \in X_2$ tels que $x_1x_2 \in E$;*
- *un sous-graphe de $G$ est un graphe $H = (U, F)$ tel que $U \subseteq V$ et $F \subseteq E$ (ne pas oublier qu'avec la définition d'un graphe, ceci veut dire que $F \subseteq E \cap \binom{U}{2}$);*
- *un sous-graphe de $G$* induit par *$F \subseteq E$ est le graphe $G\langle F\rangle = (V_F, F)$ avec $V_F = \cup_{e\in F} e = \cup\{u, v\} \in F\{u, v\}$;*
- *un sous-graphe de $G$* induit par *$U \subseteq V$ est le graphe $G\langle F\rangle = (U, E \cap \binom{U}{2})$;*
- *un sous-graphe $H$ connexe de $G$ est* maximal *si pour tout $x \in V \setminus V(H)$, le graphe $H + x = (V(H) \cup \{x\}, E(H) \cup \{ux : u \in V(H)\} \cap E$ n'est plus connexe;*
- *une* composante connexe *$H$ de $G$ est un sous-graphe induit connexe maximal de $G$;*
- *un* arbre couvrant *de $G$ est un sous-graphe $T = (V, F)$ de $G$ qui est un arbre.*

**Lemme 1.1** *Tout graphe connexe $G = (V, E)$ contient un arbre couvrant.*

*Démonstration.* On en construit un récursivement. Soit $T_0 = (V_0, E_0)$ l'arbre avec $V_0 = \{v_0\}$ pour un sommet arbitraire $v_0 \in V$ et $E_0 = \emptyset$. Ce sous-graphe $T_0$ est bien un arbre. Pour $i = 1, \ldots, n-1$, on définit $T_i = (V_i, E_i)$ à partir de $T_{i-1}$. Soit $(V_{i-1}, V \setminus V_{i-1})$ une partition de $V$. Puisque $G$ est connexe, il existe $u_i \in V_{i-1}$ et $v_i \in V \setminus V_{i-1}$ tels que $u_iv_i \in E$. Soit $V_i = V_{i-1} \cup \{v_i\}$ et $E_i = E_{i-1} \cup \{u_iv_i\}$. On voit que $T_i$ est connexe car $T_{i-1}$ l'est et donc il y a un chemin de $x$ vers $y$ dans $T_{i-1}$ pour tout $x, y =\in V_{i-1}$ et les chemins de $x$ vers $u_i$ peuvent être rallongés pour donner des chemins de $x$ vers $v_i$. Si $T_{i-1}$ est sans cycle, $T_i$ l'est

aussi car $v_i \notin V_{i-1}$. Donc $T_i$ est un arbre pout tout $i = 0, \ldots, n-1$. On note que $|V_i| = |V_{i-1}| + 1 = i + 1$ pour $i > 0$, donc $T_{n-1}$ contient tous les $n$ sommets. On conclue que $T_{n-1}$ est un arbre couvrant de $G$. □

Observons que la construction de la preuve décrit bien les algorithmes de fouille en abstrait. Il suffit de préciser un peut plus *quelle* arête $u_i v_i$ on prend à chaque étape. Cela permet d'obtenir soit une fouille en profondeur, soit une fouille en largeur (mais pas seulement).

**Définition 1.3** *Soit $G = (V, E)$ un graphe pondéré par $c : E \longrightarrow \mathbb{R}^{\geq 0}$. Un* arbre couvrant minmum *de $G$ est un arbre couvrant $T = (V, F)$ tel que pour tout arbre couvrant $T' = (V, F')$, $\sum_{e \in F} c(e) \leq \sum_{e \in F'} c(e)$.*

**Corollaire 1.1** *Soit $G = (V, E)$ un graphe connexe pondéré par $c : E \longrightarrow \mathbb{R}^{\geq 0}$. Alors $G$ contient un arbre couvrant minimum.*

*Démonstration.* Puisque le graphe est fini, le nombre d'arbres couvrants possibles est fini et on peut choisir un arbre de poids (coût) minimum parmi eux. Une méthode longue et inefficace donnant la réponse est de choisir $n-1$ arêtes parmi toutes les arêtes du graphe avec $n$ sommets et $m \geq n-1$ arêtes, vérifier si elles induisent un arbre, et si oui, calculer son coût. Faire ceci pour tout ensemble de $n-1$ arêtes et retenir l'arbre de coût minimum courant. □

**Théorème 1.1** *Soit $G = (V, E)$ un graphe connexe pondéré par $c : E \longrightarrow \mathbb{R}^{\geq 0}$. Soit $T = (V, F)$ un arbre couvrant minimum de $G$, $F' \subset F$ et soit $C_0 \dot\cup C_1 \dot\cup \ldots \dot\cup C_{k-1}$ une partition de l'ensemble des composantes connexes de $G_{F'} = (V, F')$. Soit $uv \in E \setminus F'$ une arête de coût minimum parmi celles qui relient les classes $C_i$ différentes. Alors il existe un ACM $T^* = (V, F^*)$ tel que $F' \cup \{uv\} \subseteq F^*$.*

*Démonstration.* Soit $G, T, V, E, F, F', C_0, \ldots C_{k-1}, uv$ comme dans l'énnoncé. On va montrer comment obtenir l'arbre $T^*$ à partir de $T$ et $uv$.

1. Puisque $T$ est connexe, il existe un chemin $P$ dans $T$ de $u$ à $v$. Si $uv$ est sur $P$ (en fait, est $P$), on a fini: il suffit de prendre $T^* = T$. Supposons alors que $uv \notin F$.

2. Soit $f \in E \setminus F'$ une arête reliant deux classes différentes de composantes connexes de $G_{F'}$. Alors par le choix de $uv$ comme une arête de coût minimum, on a
$$c(uv) \leq c(f).$$

3. Soit $f \in E(P)$ (donc dans $T$) une arête reliant deux classes différentes de composantes connexes de $G_{F'}$ . Alors - par la minimalité de $T$ - on a
$$c(f) \leq c(uv).$$
Pour le voir, on se rappelle que l'ajout d'une arête à un arbre crée un cycle dans le graphe ainsi obtenu et donc le graphe $T' = (V, F")$ avec $F" = (F \cup \{uv\}) \setminus \{f\}$ est un arbre couvrant de $G$ et
$$c(f) + \sum_{f \neq e \in F} c(e) = c(T) \leq c(T') \leq c(uv) + \sum_{f \neq e \in F} c(e)$$
ce qui donne ce qu'on voulait.

4. Donc: pour toute arête $f$ qui est dans $P$ et qui relie deux classes différentes de composantes connexes de $G_{F'}$ on a
$$c(uv) = c(f).$$

5. On peut alors prendre $T^* = (V, F^*)$ où
$$F^* = (F \setminus \{f\}) \cup \{uv\}$$
pour n'importe quelle arête $f$ décrite en 4. Ceci donnera
$$\sum_{e \in F} = c(T) = c(T^*) = \sum_{e \in F^*} c(e)$$

6. Pour terminer il suffit de noter que si $uv \neq P$ alors une telle arête $f$ existe.

□

Le théorème nous permet de commencer par $F' = \emptyset$ (sûrement une partie de $F$ pour tout arbre couvrant de $G$ et d'y ajouter une arrête à la fois tout en étant assuré que l'ensemble d'arrêtes obtenu est, à chaque étape une partie de l'ensemble $F^*$ d'arrêtes d'un arbre couvrant minimum. N'oublions pas que $F^*$ peut changer d'une étape à l'autre. Prouvons qu'un bon algorithme existe pour trouver un arbre couvrant minimum d'un graphe $G = (V, E)$ pondéré par $c : E \longrightarrow \mathbb{R}^{\geq 0}$. Pour simplifier l"ecriture, soit $F_i \subseteq E$ et soit $CC_i$ l'ensemble de composantes connexes du graphe $G_{F_i} = (V, F_i)$. Soit $|V| = n$ et $|E| = m \geq n - 1$.

**Algorithme 1.1**

$F_0 = \emptyset$
`pour` $i = 1..n-1$ `faire`
 `choisir une partition de` $CC_{i-1}$
 `choisir une arête` $e$ `de poids minimum entre 2 classes distinctes de` $CC_i$
 $F_i = F_{i-1} \cup \{e\}$

**Théorème 1.2** *L'algorithme 1.1 s'arrête et*

1. $|CC_i| = |CC_{i-1}| - 1$*;*
2. $G_{F_i} = (V, F_i)$ *est une forrêt;*
3. $G_{F_{n-1}} = (V, F_{n-1})$ *est un arbre couvrant minimum de* $G$*.*

*Démonstration.* Bien évidemment, on fait une récurrence sur $i$ pour les premières deux parties; la troisième en découle directement.
Pour $i = 0$, on a $F_i = \emptyset$ et le graphe $G_{F_i}$ n'a aucune arête. Les composantes connexes sont donc des sommets isolés et il y en a $n = n - i$. Un sommet seul est un arbre, le graphe est donc une forrêt. Si les deux énoncés sont vérifiés pour $i - 1 \geq 0$, ils le sont pour $i$ car en reliant deux classes de la partition, l'algorithme relie deux composantes connexes. Puisque les deux extremités de l'arête $e$ choisie sont dans des composantes différentes $C$ et $C'$, l'ajout de $e$ ne peut pas créer un cycle. La composante crée par l'ajout de $e$ est donc un arbre. Cet ajout enlève deux composantes $(C, C')$ et en ajoute une ($C$ reliée à $C'$). Donc $|CC_i| = |CC_{i-1}| - 2 + 1 = |CC_{i-1} - 1$. On conclue que $|CC_{n-1}| = 1$ et la composante connexe est un arbre. Sa minimalité découle de théorème 1.1 car à chaque itération de l'algorithme, ce théorème nous garantit que l'ensemble $F_i$ fait partie de l'ensemble d'arêtes d'un arbre couvrant minimum de $G$ et ceci est vrai en particulier pour $F_{n-1}$. □

**Corollaire 1.2** *Les algorithmes de Kruskal et de Prim trouvent un arbre couvrant minimum du graphe pondéré donné en entrée.*

*Démonstration.* Soit $C_0^i \dot\cup C_1^i \dot\cup \ldots \dot\cup C_{k_i-1}^i$ la partition choisie par l'algorithme à la $i-$ème itération. Pour Kruskal, la partition choisie dans l'algorithme consiste d'une composante connexe par classe, i.e., $|CC_i| = k$. Pour Prim, $k = 2$ avec $C_0^i = G\langle F_{i-1}\rangle$ et $C_1^i = \{\{v\} : v \in V \setminus V(G\langle F_{i-1}\rangle)\}$. Cette partition garantit que $G\langle F_i\rangle$ est un arbre pour tout $i = 0, \ldots, n-1$. □

On peut donner une preuve plus courte si on se permet de laisser l'agorithme implicite.

**Théorème 1.3** *Les algorithmes de Kruskal et de Prim trouvent un arbre couvrant mnimum dans un graphe pondéré.*

*Démonstration.* Soit $G = (V, E)$ un graphe avec $n$ sommets pondéré par $c : E \longrightarrow \mathbb{R}^{\geq 0}$. Soit $A$ l'algorithme de Kruskal qui trouve un arbre couvrant dans $G$. On prouve que l'abre trouvé est minimum. La preuve est la même pour les deux algortihmes, avec un petit ajout pour Prim.

Soit $\hat{T}$ un arbre (couvrant) trouvé par $A$. Soit $e_1, e_2, \ldots, e_{n-1}$ les arêtes de $\hat{T}$ dans l'ordre dans lequel elles sont prises par $A$ et, pour $k \in \{1, \ldots, k\}$, soit $E_k = \{e_1, \ldots e_{k-1}\}$. Soit $V_k = \cup_{i=1}^{k-1} e_i = \cup_{e \in E_k} e$ (rappel : une arête est une partie à deux éléments de $V$, donc $V_k$ est l'ensemble de sommets incidentes aux arêtes $e_1, e_2, \ldots, e_{k-1}$).

Pour un arbre couvrant minimum $T$, soit $f(T)$ le plus petit indice tel que $e_{f(T)} \notin E(T)$ (donc $e_1, \ldots, e_{f(T)-1} \in E(T) \cap E(\hat{T})$). Parmi tous les arbres couvrants minimaux, prenons $T^*$ tel que $f(T^*)$ est maximum et mettons $f(T^*) = k$. Soit $e_k = \{u, v\}$. Dans $T^* + e_k$ il y a un cycle unique $C$ (propriété d'un arbre) qui doit contenir des arêtes qui ne sont pas dans $\hat{T}$ (sinon, $C$ serait un cycle dans le graphe acyclique $\hat{T}$). Soit $e$ une telle arête. On a que $c(e) \leq c(e_k)$ car sinon, $c(T^* + e_k - e) < c(T^*)$ et puisque $T^* + e_k - e$ est un arbre couvrant de $G$, ceci contredirait la minimalité de $T^*$.

On voudrait prouver que $c(e) \geq c(e_k)$ car cela nous permettra de dire que $T' = T^* + e_k - e$ est un arbre couvrant tel que $c(T') = c(T^*)$ et $f(T') > k$, contredisant le choix de $T^*$. Si $A$ est l'algorithme de Kruskal, ceci est clair, car sinon $e$ aurait été prise par $A$ plutôt que $e_k$.

Si $A$ est l'algorithme de Prim, il faut un peu plus - il faut chosir $e$ avec soin afin de pouvoir dire que $e$ aurait été prise plutôt que $e_k$. Soit le cycle $C = u_0 u_1 \ldots u_l$ avec $u_0 = u$ et $u_1 = v$. Par le choix de $e_k$ par Prim, exactement un de $u_0, u_1$ est dans $V_k$, l'ensemble de sommets de l'arbre construit avant l'ajout de $e_k$. Supposons, sans perte de généralité, que ce soit $u_1$. On observe que les arêtes de $C$ sont chacune soit dans $E_k$, soit dans $E(T^*) \setminus E(\hat{T})$, par le choix de $T^*$, et on se souvient qu'elles ne sont pas toutes dans $E_k$. Soit $s$ le plus petit index tel que $\{u_s, u_{s-1}\} \in E(T^*) \setminus E(\hat{T})$ (i.e. tel que $u_s \notin V_k$) et mettons $e = \{u_s, u_{s-1}\}$. On a maintenant que $c(e) \geq c(e_k)$ car sinon Prim l'aurais prise avant de $e_k$ et que $c(e) \leq c(e_k)$ car sinon $c(T^* + e_k - e) < c(T^*)$, comme on a vu au début. On peut alors conclure, comme on le voulait, que $T' = T^* + e_k - e$ est un arbre couvrant tel que $c(T') = c(T^*)$ et $f(T') > k$, contredisant le choix de $T^*$. □