MA351 : Introduction à la théorie des graphes

Arbres

Yann Kieffer

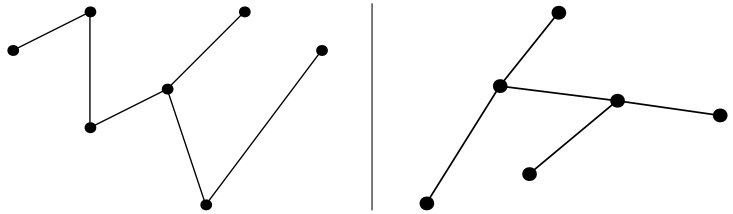
ESISAR

Arbres : définition, existence de feuilles

Définition d'arbre

Un *arbre* est un graphe non-orienté connexe sans cycle.

Notez que par définition, un arbre a toujours au moins un sommet.
On utilise souvent la notation $T = (V, E)$ pour un arbre (anglais : tree).
Exemples.



Un premier théorème

Théorème

Un arbre ayant au moins une arête a au moins deux feuilles.

Démonstration.

Soit $T = (V, E)$ un arbre ayant au moins une arête xy. Si on lance $(s, r) = \mathit{Cycle\_ou\_Feuille}(T, x, xy)$, alors on sait que s sera "Feuille", puisque T n'a pas de cycle ; et r sera une feuille de T.
Puisque r est une feuille de T, r a un unique voisin qu'on note v.
Alors en lançant $(s, u) = \mathit{Cycle\_ou\_Feuille}(T, r, rt)$, on aura à nouveau que u est une feuille. Il nous reste à vérifier que $u \neq r$.
Autrement dit, il faut vérifier que Cycle_ou_Feuille() ne renvoit jamais comme feuille le sommet qu'on lui a passé en paramètre. Il est facile de s'en convcaincre en déroulant l'algorithme sur un exemple ; il est un peu plus difficile d'en faire la preuve ; elle vous est laissée à titre d'exercice. □

Un théorème de décomposition

Théorème

Tout arbre $T = (V, E)$ est soit l'arbre trivial $T_0 = (\{*\}, \emptyset)$, soit obtenu à partir d'un autre arbre $T' = (V', E')$ en adjoignant à T' une nouvelle feuille :

$$V = V' \cup \{f\}; \quad E = E' \cup \{sf\}; \quad f \notin V'; \quad s \in V'$$

Démonstration.

Si T a exactement un sommet, alors on est dans le premier cas. Sinon, T a au moins 2 sommets. Comme c'est un graphe connexe, il a au moins une arête. D'après le théorème précédent, il a au moins deux feuilles ; notons f une de ses feuilles, et notons s l'unique voisin de f. On définit :

$$V' = V \setminus \{f\}; \quad E' = E \setminus \{sf\}; \quad T' = (V', E')$$

Alors :

- T' est connexe ;
- T' est sans cycle ;
- on a bien les relations attendues entre T et T'.

□

Applications de ce théorème

Ce théorème de décomposition va nous permettre de faire des preuves par récurrence sur les arbres. Par exemple :

Théorème

Si $T = (V, E)$ est un arbre, alors $|V| = |E| + 1$.

Démonstration.

On fait la preuve par récurrence sur $|V|$. Si $|V| = 1$, alors T est l'arbre trivial, et le résultat est bien vérifié. Supposons que la propriété annoncée par le théorème soit vérifiée pour tout arbre ayant n sommets ; et soit T un arbre ayant $n + 1$ sommets.
Alors, d'après le théorème précédent, il existe un arbre T' avec :

$$V = V' \cup \{f\}; \quad E = E' \cup \{sf\}; \quad f \notin V'; \quad s \in V'$$

Comme T' a n sommets, on sait que $|V'| = |E'| + 1$. Donc :

$$|V| = |V'| + 1 = |E'| + 2 = |E| + 1$$

□

Arbres couvrants de poids minimum

Un problème classique concernant les arbres est de trouver un sous-graphe d'un graphe non-orienté qui soit un arbre, et dont le poids total des arêtes est minimum.
Ce problème a par exemple une application dans le domaine des réseaux, où une stratégie de routage est de n'utiliser que des liens formant un arbre pour transmettre les paquets. Une des qualités de cette stratégie est qu'il n'y a alors qu'une unique route pour transférer un paquet d'un sommet de l'arbre à un autre sommet.
Pour bien aborder le problème de l'arbre couvrant de poids minimum, nous avons besoin de quelques définitions.

Arbres couvrants de poids minimum : concepts

Définition

Un *sous-graphe partiel* d'un graphe $G = (V, E)$ est un graphe obtenu à partir de G en gardant tous les sommets et en ne conservant que certaines arêtes :

$$G' = (V, F) \quad \text{avec } F \subseteq E$$

Définition

Un *arbre couvrant* d'un graphe G est un sous-graphe partiel de G' qui soit un arbre.

Le problème qui nous intéresse peut alors s'énoncer comme suit.

Arbre couvrant de poids minimum

Etant donnés un graphe $G = (V, E)$, et des poids sur les arêtes de ce graphe : $p : E \to \mathcal{R}$, trouver (le poids d')un arbre couvrant de G qui soit de poids minimum parmi tous les arbres couvrants de G.

Le *poids* d'un sous-graphe partiel est la somme des poids de ses arêtes.

Algorithme de Prim

Il y a deux algorithmes classiques qui permettent de résoudre le problème de l'arbre couvrant de poids minimum.
Nous présentons d'abord l'algorithme de Prim (Jarník, 1930) :

Prim ($G = (V, E)$, $s \in V$, $p : E \to \mathcal{R}$) :
- $S \leftarrow \{s\}$
- $F \leftarrow \emptyset$
- $Ce \leftarrow \emptyset$
- $P \leftarrow \{uv \in E / u \in S, v \notin S\}$
- **TantQue** $P \neq \emptyset$, **Faire** :
  - soit xy une arete de P de poids minimal
  - $S \leftarrow S \cup \{y\}$
  - $F \leftarrow F \cup \{xy\}$
  - $P \leftarrow P \setminus \{xy\}$
  - **PourTout** uv dans P avec $u \in S, v \in S$, **Faire** :
    - $C \leftarrow$ Chaine $((S, F), u, v)$
    - $Ce \leftarrow Ce \cup \{(uv, C)\}$
  - $P \leftarrow \{uv \in E / u \in S, v \notin S\}$
- **Retourner** (S, F, Ce)

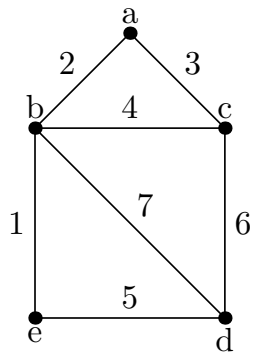
Recherche de chaîne dans un arbre

L'algorithme de Prim fait appel à une routine de recherche de chaîne dans un arbre. Voici cette routine :

```
Chaine( G = (V,E), x ∈ V, y ∈ V ):                    1
– (M,o) = Accessibles3(G,x)                           2
– Retourner Reconstruire_chemin(o,x,y)                3
```


Déroulons cet algorithme

Je vous propose de dérouler l'algorithme sur le graphe non-orienté avec poids ci-dessous. Faites une exécution avec a comme sommet de départ ; et une avec d comme sommet de départ.



A propos de l'algorithme de Prim

- A l'issue de l'algorithme, S est la composante connexe de s ; pouvez-vous justifier pourquoi ? On supposera dans la suite que G est connexe ;
- cet algorithme a un lien de parenté certain avec Accessibles() ;
- l'intention de l'algorithme est qu'à l'issue du calcul, (S, F) soit un arbre couvrant de poids minimum de G pour les poids p.

Il nous faut maintenant prouver la correction de cet algorithme, et donc que cette intention est bien atteinte.

Théorème

A l'issue de l'appel :
$(S, F, Ce) = Prim(G = (V, E), s \in V, p : E \to \mathcal{R})$,
où le graphe G est connexe, $S = V$, et (V, F) est un arbre couvrant de poids minimum de G pour les poids p.

Théorèmes préliminaires à la preuve de Prim()

Lemme

Soit $G = (V, E)$ un graphe, et x et y deux sommets de G tels que xy ne soit pas une arête de G. On note $G' = (V, E \cup \{xy\})$. Alors exactement une des propositions suivantes est vraie :

1. *G' a un cycle passant par $\{xy\}$, et G et G' ont le même nombre de composantes connexes ;*
2. *G' n'a pas de cycle passant par $\{xy\}$, et G' a une composante connexe de moins que G.*

Démonstration.

Si x et y sont dans la même composante connexe de G, alors on tombe dans le cas (1) : comme il y a une chaîne de x à y dans G, l'ajout de l'arête xy ferme un cycle, et laisse inchangées les composantes connexes. Si x et y sont dans des composantes connexes différentes, l'ajout de l'arête xy fusionne les composantes connexes de x et de y. L'arête xy ne participe à aucun cycle, puisqu'il n'y a pas de chaîne de x à y dans G. Les deux cas sont clairement exclusifs l'un de l'autre, en raison de leur définition ! □

Théorèmes préliminaires à la preuve de Prim()

Théorème

Soit $G = (V, E)$ un graphe non-orienté. Deux des propositions suivantes suffisent à entraîner la troisième :

1. *G est connexe ;*
2. *G est sans cycle ;*
3. $|V| = |E| + 1$.

Démonstration

Il s'agit réellement de trois théorèmes en un. Le premier, $(1) + (2) \Rightarrow (3)$ est le dernier théorème que nous avons démontré.
Montrons $(2) + (3) \Rightarrow (1)$.
Rangeons les arêtes de E dans un ordre quelconque : $E = \{e_1, \ldots, e_m\}$.
Définissons la suite de graphes : $G_i = (V, E_i)$, où $E_i = \{e_1, \ldots, e_i\}$. Nous prenons $E_0 = \emptyset$.
Alors $G_m = G$; et G_0 est un graphe sans arête : il a donc $|V| = m + 1$ composantes connexes.

Triple-théorème : suite et fin.

On passe de G_i à G_{i+1} par l'ajout d'une arête. Le lemme précédent indique que par cet ajout, soit on crée un cycle, soit on réduit le nombre de composantes connexes. Comme le graphe final $G_m = G$ est sans cycle, on en déduit qu'à chaque étape, on réduit de 1 le nombre de composantes connexes. Or on part de $m + 1$ composantes ; et il y a m ajout d'arêtes ; donc le nombre de composantes dans G_m est 1 : c'est-à-dire que G est connexe.

Montrons maintenant $(1) + (3) \Rightarrow (2)$.
On raisonne sur la même suite de graphes qu'au point précédent. Cette fois-ci, on utilise l'hypothèse de connexité. Or il y a m transitions possibles pour passer de G_0 à $G_m = G$; et on sait que G a une composante, alors que G_0 en a $m + 1$. Or une transition réduit au mieux de 1 le nombre de composantes connexes. Il faut donc perdre une composante connexe à chaque transition ; ceci signifie qu'on est à chaque transition dans le second cas du lemme précédent. Donc aucun ajout d'arête, en allant de G_0 à G_m ne crée de cycle. Donc G est sans cycle.

Trois lemmes sur les arbres - 1

Lemme

Si $T = (V, E)$ et $T' = (V, E')$ sont deux arbres sur le même ensemble de sommets, et si $e \in E' \setminus E$, alors il existe $f \in E \setminus E'$ tel que $H = (V, E \cup \{e\} \setminus \{f\})$ est un arbre.

Démonstration.

Soit $G = (V, E \cup \{e\})$. Si on note $e = xy$, comme il y a une unique chaîne de x à y dans T, il y a un unique cycle C dans G passant par e. Comme T' est sans cycle, il y a au moins une arête de $C \setminus \{xy\}$ qui n'est pas dans T'. Soit f une telle arête. Alors $H = (V, E \cup \{e\} \setminus \{f\})$ est un graphe sans cycle, puisqu'on a brisé le seul cycle de G ; et il vérifie la relation $|V(H)| = |E(H)| + 1$. Donc d'après le lemme précédent, c'est un graphe connexe ; et donc un arbre. □

Illustration pour les 3 lemmes

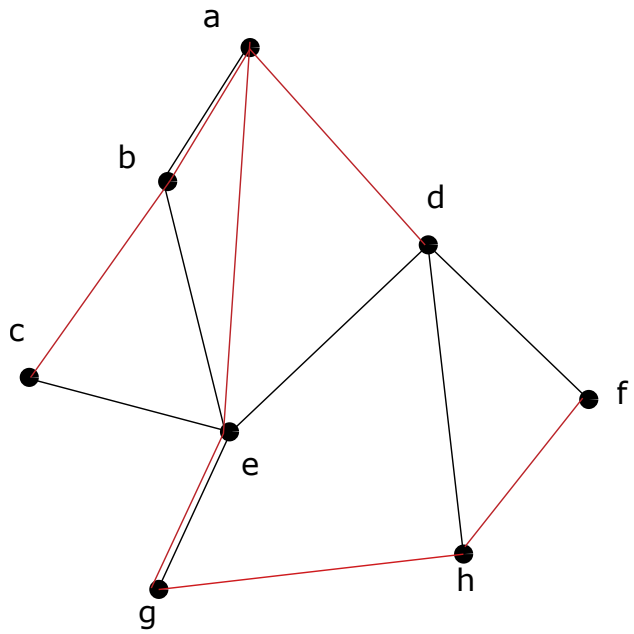
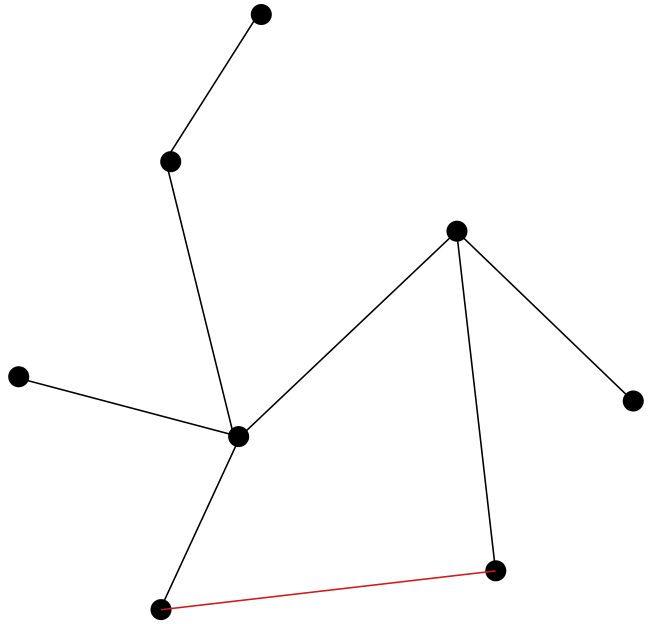


Illustration pour le lemme 1



Trois lemmes sur les arbres - 2

Lemme

Si $T = (V, E)$ et $T' = (V, E')$ sont deux arbres sur le même ensemble de sommets, et si $e \in E' \setminus E$, alors il existe $f \in E \setminus E'$ tel que $H = (V, E' \setminus \{e\} \cup \{f\})$ est un arbre.

Démonstration.

Soit $G = (V, E' \setminus \{e\})$. Ce graphe a exactement deux composantes connexes. Si on note $e = xy$, alors x est dans une des composantes, appelons-la C_1, et y est dans l'autre composante de G, qu'on appelle C_2.
Soit C une chaîne de x à y dans T. Alors il y a au moins une arête de C dont une extrémité est dans C_1, et l'autre dans C_2, puisque x est dans l'un, et y dans l'autre.
Soit f une telle arête. Cette arête ne peut pas être une arête de G.
Alors l'ajout de f à G produit le graphe H qui est connexe, et qui vérifie la bonne relation numérique entre nombre de sommets et d'arêtes ; c'est donc un arbre. □

Illustration pour le lemme 2

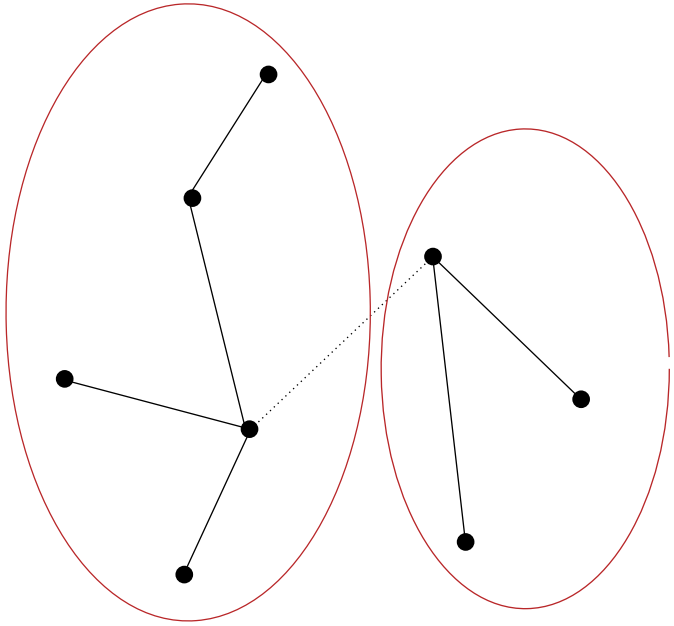
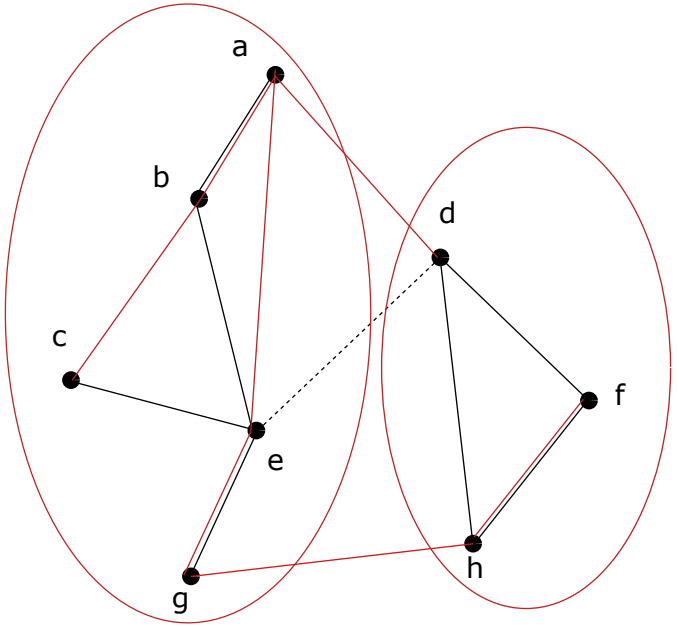


Illustration pour le lemme 3



Trois lemmes sur les arbres - 3

Lemme

Si $T = (V, E)$ et $T' = (V, E')$ sont deux arbres, et si $e \in E' \setminus E$, alors il existe $f \in E \setminus E'$ tel que $H_1 = (V, E' \setminus \{e\} \cup \{f\})$ est un arbre, et $H_2 = (V, E \cup \{e\} \setminus \{f\})$ est un arbre.

Démonstration.

Notons $e = xy$, et $G_1 = (V, E' \setminus \{e\})$. Alors G_1 a deux composantes connexes : celle de x, qu'on note C_1 ; et celle de y, qu'on note C_2.
Soit C une chaîne de x à y dans T. Alors il y a au moins une arête de C dont une extrémité est dans C_1, et l'autre dans C_2, puisque x est dans l'un, et y dans l'autre.
Soit f une telle arête. Cette arête ne peut pas être une arête de G_1.
Alors l'ajout de f à G_1 produit le graphe H_1 qui est connexe, et qui vérifie la bonne relation numérique entre nombre de sommets et d'arêtes ; c'est donc un arbre.
D'autre part, le retrait de cette arête au graphe $G_2 = (V, E \cup \{e\})$ lui enlève son seul cycle. Donc H_2 est également un arbre.

□

Certificat d'optimalité

Un *certificat d'optimalité* est une donnée fournie en complément d'une solution supposée optimale, et permettant de prouver son optimalité.

Pour le problème de l'arbre couvrant de poids minimum, nous proposons les données suivantes comme candidates à être un certificat d'optimalité, ce que nous prouvons à la diapo suivante.

Certificat d'optimalité pour 'arbre couvrant de poids minimum'

si $T = (V, F)$ est un arbre couvrant supposé optimal de $G = (V, E)$, un certificat d'optimalité $\mathcal{C}$ pour T et p est la donnée d'une fonction $f : (E \setminus F) \to V^*$ qui à chaque arête e non prise dans T associe une chaîne C de T joignant les extrémités de cette arête e, et telle que chaque arête de C est de poids inférieur ou égal à celui de e.

Preuve d'optimalité (1)

Nous allons maintenant montrer que lorsqu'on dispose d'une solution supposée optimale $\widetilde{T}$ et d'un certificat d'optimalité $\mathcal{C}$ pour $\widetilde{T}$, alors $\widetilde{T}$ est bien un arbre couvrant de poids minimum.

Théorème

Soit $\widetilde{T} = (V, \widetilde{F})$ un arbre couvrant de G, $\mathcal{C}$ un certificat d'optimalité pour $\widetilde{T}$, G et p, et $T = (V, F)$ un arbre couvrant quelconque. Alors $p(\widetilde{T}) \leq p(T)$.

Afin de démontrer le résultat, nous allons construire une suite d'arbres telle que :

- $T_0 = T$;
- $T_k = \widetilde{T}$;
- $\forall i \in \{1, \ldots, k\}, \quad p(T_{i-1}) \geq p(T_i)$.

Si $T = \widetilde{T}$, alors $k = 0$ et on a fini !
Sinon, supposons qu'on ait constuit $T_0, \ldots, T_p$, avec $T_p \neq \widetilde{T}$.
Puisque $T_p \neq \widetilde{T}$, il y a une arête e qui est dans T_p, mais pas dans $\widetilde{T}$. On note $e = xy$.

Preuve d'optimalité (2)

On note U le graphe $(V(T_p), E(T_p) \setminus \{e\})$. On considère alors la chaîne $f(e)$, où f est la fonction apparaissant dans le certificat d'optimalité $\mathcal{C}$.
C'est une chaîne joignant x à y dans $\widetilde{T}$.
Or x et y sont dans des composantes connexes disjointes de U. Donc il y a (au moins) une arête de $f(e)$ qui joint les deux composantes connexes de U. Soit e' une telle arête.
On définit alors $T_{p+1} = (V, E(U) \cup \{e'\})$.
Par le choix de e', il est clair que T_{p+1} est connexe. Comme la relation numérique entre arêtes et sommets est la bonne, c'est un arbre. D'après la propriété du certificat, $p(e) \geq p(e')$, donc $p(T_p) \geq p(T_{p+1})$.
On a fini notre construction. On en déduit que :

$$p(\widetilde{T}) = p(T_k) \leq p(T_{k-1}) \leq \cdots \leq p(T_0) = p(T)$$

Début de la preuve de correction de Prim()

Exécutabilité

En parcourant ligne à ligne l'algorithme, on constate qu'il n'y a pas de problème d'exécutabilité.

Arrêt

A chaque fois qu'on choisit une arête xy, y n'est pas encore dans S ; puis y est placé dans S. Comme ce sont les seules modifications de S, et que $S \subseteq V$, on en déduit que le nombre de passages dans la boucle est borné par le nombre d'éléments dans V.

Reste de la preuve

Correction des valeurs retournées : c'est un arbre

La preuve du fait que $S = V$ ne sera pas détaillée ici. C'est un bon exercice que de la rédiger.
Nous allons maintenant montrer, d'une part que $T = (V, F)$ est un arbre ; et d'autre part, que c'est un arbre de poids minimum parmi tous les arbres couvrants pour G et p.
Tout d'abord, observons qu'avant chaque entrée dans la boucle, le graphe (S, F) est un graphe connexe. En effet, sa valeur initiale est le graphe $(\{s\}, \emptyset)$ qui est connexe ; et le passage d'une valeur à la suivante consiste à ajouter un sommet, et une arête joignant ce nouveau sommet à un sommet déjà présent. D'autre part, la même observation montre que ce graphe a un sommet de plus que d'arête : c'est donc un arbre, d'après le triple-théorème. Or T est la valeur finale de (S, F) : c'est donc un arbre.

L'arbre est de poids minimum

Soit $T = (V, F)$ l'arbre retourné par Prim, et $\mathcal{C}$ le candidat-certificat d'optimalité retourné par Prim.
Nous allons vérifier que $\mathcal{C}$ est bien un certificat d'optimalité.
Il est facile de vérifier que $\mathcal{C}$ a la bonne forme : c'est une fonction qui à certaines arêtes de G associe des chaînes de T. Cette fonction est définie sur les arêtes de P dont les deux extrémités sont dans S, après ajout à S d'un sommet. A chaque étape de Prim, P est l'ensemble des arêtes candidates ; en fin de boucle, une arête $\{uv\}$ non choisie, et qui ne sera plus jamais candidate, se voit affecter une chaîne dans `Ce`. Comme u et v sont dans S, et le graphe (S, F) est un arbre, il y a bien une chaîne dans (S, F) joignant les deux extrémités de $\{uv\}$.
A la fin de l'exécution de Prim, `Ce` contient donc bien une fonction des arêtes de G non sélectionnées dans T vers des chaînes de T.

L'arbre est de poids minimum : suite et fin

Il reste à vérifier que pour chaque arête a d'une telle chaîne $f(e)$, on a : $p(a) \leq p(e)$.
On sait que toutes les arêtes de la chaîne ont été choisies par Prim avant qu'on affecte une valeur dans `Ce` pour e. Notons C le cycle formé par l'arête e et la chaîne $f(e)$.
A chaque fois que Prim choisit une arête de C, il y a une autre arête de C qui est candidate. De plus, Prim ne peut pas prendre toutes les arêtes du cycle—puisqu'il construit un sous-graphe partiel sans cycle de G.
On en déduit que Prim ne peut pas prendre l'arête de poids dépassant celui de toutes les autres arêtes de C : au moment de prendre une telle arête, il y a une autre arête de poids inférieure qui est également candidate, et qui serait donc prise en priorité.
Donc pour toute arête a de $f(e)$, $p(a) \leq p(e)$: f est donc bien un certificat d'optimalité.
Donc d'après le théorème d'optimalité, T est un arbre de poids minimum dans G muni des poids p.

L'algorithme de Kruskal

Il existe un autre algorithme de calcul d'arbre couvrant de poids minimum. Il s'agit de l'algorithme de Kruskal (1956).

```
Kruskal(G = (V,E), p : E → R):                                   1
– Trier E par ordre de poids croissant :                         2
      p(e_1) ≤ p(e_2) ≤ ··· ≤ p(e_m)                             3
– F ← ∅                                                          4
– Ce ← ∅                                                         5
– Pour i de 1 a m, Faire :                                       6
  Si le graphe (V, F ∪ {e_i}) a un cycle,                        7
    alors :                                                      8
    * {x,y} ← e_i                                                9
    * Ce ← Ce ∪ { (e_i, Chaine((V,F),x,y)) }                     10
    sinon :                                                      11
    * F ← F ∪ {e_i}                                              12
– Retourner (F, Ce)                                              13
```


A propos de Kruskal

L'algorithme de Kruskal suit une logique très différente de celle de Prim. Il fonctionne en deux temps : tri des arêtes par ordre de poids croissant ; puis sélection une à une des arêtes qui ne forment pas de cycle avec les arêtes déjà retenues. Il n'utilise pas de sommet de départ, contrairement à Prim.
Notez également qu'il fabrique un certificat d'optimalité du même type que celui fabriqué par Prim.
Nous allons maintenant prouver cet algorithme.

Théorème

A l'issue de l'appel $(T, \mathcal{C}) = Kruskal(G, p)$, T est un arbre de poids minimum de G muni des poids p.

Preuve de Kruskal (1)

Exécutabilité

la seule instruction qui pose question est l'appel de Chaine à la ligne 10. Est-on sûr qu'il y a bien une chaîne de x à y dans (V, F) à ce moment-là ? Ceci est attesté par la branche du `Si` dans laquelle nous nous trouvons : nous savons que $(V, F \cup \{xy\})$ a un cycle ; donc (V, F) a bien une chaîne joignant x à y.

Arrêt

le nombre d'opérations de Kruskal peut être borné indépendamment des données d'entrées. Cet algorithme termine toujours.

Preuve de Kruskal (2)

Correction

Il nous reste à montrer que l'arbre T retourné par Kruskal est bien un arbre de poids minimum. Pour cela, nous allons utiliser le théorème d'optimalité. Il nous suffit donc de prouver que $\mathcal{C}$ est un certificat d'optimalité pour T.
Notons $T = (V, F)$. Il est facile de voir que pour une arête a de $E \setminus F$, $f(a)$ est bien une chaîne de T joignant les extrémités de a. Il reste à montrer que les arêtes de cette chaîne ont toutes un poids inférieur ou égal à a. Cela découle du fait que la chaîne est enregistrée dans f au moment où l'arête a est examinée ; or les arêtes de G sont examinées par ordre de poids croissant. Comme toutes les arêtes de la chaîne ont été examinées avant a, elles ont toutes un poids inférieur ou égal à celui de a. Donc $\mathcal{C}$ est bien un certificat d'optimalité pour T ; et donc T est bien un arbre couvrant de poids minimum de G muni des poids p.

Fin du cours

C'est la fin de ce cours d'introduction à la théorie des graphes. J'espère qu'il vous aura permis d'affiner votre sens de l'abstraction, votre aisance avec les algorithmes et les preuves mathématiques ; et aussi qu'il vous aura aidé à forger un sentiment de familiarité avec les graphes, orientés et non-orientés, comme outils de modélisation d'une situation.

Je vous souhaite une excellente poursuite d'études à l'Esisar !