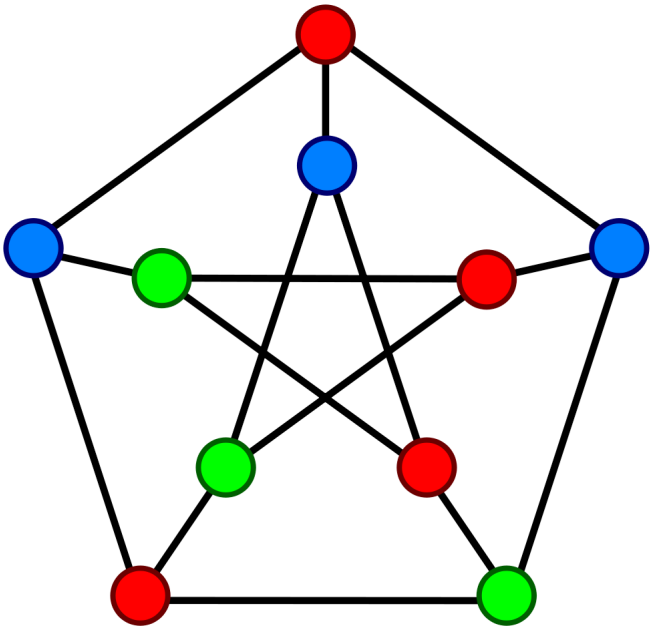
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Cours GM3
2022/2023

Optimisation discrète



Cours d'optimisation discrète d'Arnaud KNIPPEL,
rédigé par Paul MEHAUD

Table des matières

Chapitre 1

Théorie des graphes

Chapitre 2

Cheminement dans un graphe

I. Présentation : plus courts chemins

Soit G un graphe orienté valué sur les arcs.
$G = (X,U)$, on note alors $n = |X|$ et $m = |U|$.
On a également $l : U \to \mathbb{R}$, une fonction longueur (poids, distance, coût...).

Notation fonctionnelle : $l(u)$ ou $l(ij)$ avec $u = ij$, $i \to j$.

Notation fonctionnelle : l_u ou l_{ij}.
Soient s,$t \in X$ distincts.

Définition 1.1. On appelle chemin de s à t une suite finie d'arcs $\mu : i_0i_1, \ldots, i_{p-1}i_p$, avec $i_0 = s$ et $i_p = t$.

Définition 1.2. La longueur d'un chemin μ est

$$l(\mu) = \sum_{u \in \mu} l(u)$$

Le problème du plus court chemin consiste donc à trouver un chemin de s à t de longueur minimale.

Définition 1.3. Un chemin est simple si chacun de ses arcs est utilisé au plus une fois.

Définition 1.4. Un chemin est élémentaire si chacun de ses sommets est utilisé au plus une fois.

Remarque. Un chemin élémentaire est également simple. Mais la réciproque est fausse.

Définition 1.5. Un circuit est un chemin $\mu : i_0i_1, \ldots, i_{p-1}i_p$, avec $i_0 = i_p$.

Définition 1.6. Un circuit est simple si chacun de ses arcs est utilisé au plus une fois.

Définition 1.7. Un circuit est élémentaire si chacun de ses sommets est utilisé au plus une fois.

Définition 1.8. Un chemin eulérien est un chemin qui passe exactement une fois par chaque arc.

Définition 1.9. Un chemin hamiltonien est un chemin qui passe exactement une fois par chaque sommet.

Définition 1.10. Un circuit eulérien est un circuit qui passe exactement une fois par chaque arc.

Définition 1.11. Un circuit hamiltonien est un circuit qui passe exactement une fois par chaque sommet.

Conditions d'existence d'un plus court chemin :

- Il existe un chemin de s à t ;
- Il n'y a pas de circuit absorbant.

Définition 1.12. Un circuit absorbant est un circuit de longueur strictement négative.

Définition 1.13. Un circuit absorbant est un circuit de longueur strictement négative.

Proposition 1.14. *Propriété d'optimalité de Belman : Un sous chemin d'un plus court chemin est un plus court chemin.*

Démonstration. Soit μ un plus court chemin de s à t.

$$\mu : s = i_0 \to i_1 \to \cdots \to i_r = t$$

Soit μ' un sous chemin de μ, de $s' = i_k$ à $t' = i_l$.

$$\mu : s = i_0 \to i_1 \to \cdots \to \underbrace{s' = i_k \to \cdots \to t' = i_0}_{\mu'} \to \cdots \to i_r = t$$

Soit μ'' un plus court chemin de s' à t'.

$$\mu : s = i_0 \to i_1 \to \cdots \to s' = i_k \to \cdots \to t' = i_k \to \cdots \to i_r = t$$

Il en existe forcément un car, si on ne peut pas minorer la longueur d'un chemin de s' à t', alors on ne le peut pas non plus de s à t.

Nous allons montrer que $l(\mu') \leqslant l(\mu'')$.

Soit μ''' le chemin construit à partir de μ en rentrant μ' et en ajoutant μ''.

$$l(\mu''') = \sum_{j=0}^{k-1} l(i_j i_{j+1}) + l(\mu'') + \sum_{j=l}^{n-1} l(i_j i_{j+1})$$

Or μ'' est un plus court chemin de s' à t' : $l(\mu'') \leqslant l(\mu')$ et μ est le plus court chemin de s à t : $l(\mu) \leqslant l(\mu''')$. On en déduit : $l(\mu') \leqslant l(\mu'')$. □

II. Potentiel

Définition 2.1. $\pi : X \to \mathbb{R}$ est potentiel si : $\forall ij \in U$, $\pi(j) - \pi(i) \leqslant l(ij)$.

Selon le contexte, on note $\pi(i)$ ou π_i.

Proposition 2.2. *Soit π un potentiel et μ un chemin de s à i :*

$$l(\mu) \geqslant \pi(i) - \pi(s)$$

Démonstration. Notons $\mu_i : i_0 \to i_1 \to \cdots \to i_k = i$.

$$l(\mu) = \sum_{n=1}^{k} l(i_{n-1} i_n) \geqslant \sum_{n=1}^{k} (\pi(i_n) - \pi(i_{n+1})) = \pi(i_k) - \pi(i_0) = \pi(i) - \pi(s)$$

On parle de relation min-max. □

Corollaire 2.3. *Si π^* est un potentiel et μ^* un chemin de s à i tels que $l(\mu^*) = \pi^*(i) - \pi^*(s)$ alors μ^* est un plus court chemin de s à i, et de plus $\pi^*(i) - \pi^*(s) = \max\{\pi(i) - \pi(s) \,|\, \pi \text{ potentiel}\}$.*

III. Algorithme de FORD

Cet algo fournit s'il existe un plus court chemin de s à i, pour tout $i \in X$.

Initialisation

Deux vecteurs : π et pred de taille $n = |X|$.

$\pi_s \leftarrow 0$;

$$\pi_i \leftarrow \begin{cases} l_{si} & \text{si } i \in \Gamma^{-1}(s) \\ +\infty & \text{sinon} \end{cases}$$

$$\text{pred}_i \leftarrow \begin{cases} s & \text{si } i \in \Gamma^{-}(s) \\ \emptyset & \text{sinon} \end{cases}$$

Fonctionnement

TANT QUE $k < n$ (π n'est pas un potentiel)
  $k \leftarrow k + 1$
  POUR TOUT $ij \in U$ /*dans un ordre arbitraire*/
    SI $\pi_j - \pi_i > l_{ij}$
      ALORS $\pi_j \leftarrow \pi_i + l_{ij}$
        $\text{pred}_j \leftarrow i$
    FIN SI
  FIN POUR
FIN TANT QUE

Théorème 3.1. *A la fin de l'algorithme de Ford, si $k < n$, on obtient un plus court chemin de s à i, $\forall i$.*

Démonstration. Cf. corollaire. □

Question : Si $k = n$ à la fin de l'algorithme, que peut-on en déduire ?

IV. Algorithme de DIJKSTRA

Spécificité par rapport à FORD : A chaque itération, on a une partition de X en deux parties : $X = S \cup \overline{S}$ avec $S \cap \overline{S} = \emptyset$ ($\overline{S} = X \backslash S$). S est l'ensemble des sommets i tels que $\pi_i = \pi_i^*$. Pour $i \in \overline{S}$, $\pi_i = \min_{k \in S \cap \Gamma^{-}(i)} (\pi_k + l_{k_i})$.

Algo de DIJKSTRA :

Initialisation

$S = \{s\}$

$\overline{S} = X \backslash \{s\}$

$$\pi_i \leftarrow \begin{cases} 0 & \text{si } i = s \\ l_{si} & \text{si } i \in \Gamma^{-1}(s) \\ +\infty & \text{sinon} \end{cases}$$

$$\text{pred}_i \leftarrow \begin{cases} s & \text{si } i \in \Gamma^{-}(s) \\ \emptyset & \text{sinon} \end{cases}$$

Fonctionnement

TANT QUE $\overline{S} \neq \emptyset$
  $j \leftarrow \text{argmin}_{i \in \overline{S}} \pi_i$;
  $\overline{S} \leftarrow \overline{S} \backslash \{j\}$; $S \leftarrow S \cup \{j\}$;
  SI $\overline{S} \neq \emptyset$
    ALORS POUR Tout $i \in \Gamma^{+}(j) \cap \overline{S}$

SI $\pi_i - \pi_j > l_{ji}$;
ALORS $\pi_i \leftarrow \pi_j + l_{ji}$;
$\text{pred}_i \leftarrow j$;
FIN SI
FIN POUR
FIN SI
FIN TANT QUE

Complexité : $O(n+m)$

V. Cas des graphes sans circuits : Algo de BELLMAN

Définition 5.1. On dit qu'un graphe $G = (X,U)$ orienté possède un ordre topologique si on peut numéroter les sommets de 1 à n de façon à ce que $(ij \in U) \implies (i < j)$.

Proposition 5.2. *G est sans circuit ssi il possède un ordre topologique.*

Démonstration. On montre le sens direct (réciproque immédiate).

Soit G un graphe orienté sans circuit.

Pour deux sommets r et s distincts quelconques, on ne peut avoir à la fois un chemin de r vers s et un chemin de s vers r (sinon ils forment un circuit).

S'il y a un chemin de r vers s : on place r avant s.

S'il y a un chemin de s vers r : on place s avant r.

S'il n'y a pas de chemin de s vers r, ni de r vers s : on fait comme on veut !

On construit un ordre topologique en prenant un sommet i_1 qui n'a pas de prédécesseur (il y en a forcément un car sinon on construit un circuit en remontant les arcs). Puis, par récurrence, en retirant les sommets au fur et à mesure. □

Remarque. La mise en œuvre algorithmique : parcours de graphe $O(m+n)$.

Algo de BELLMAN :

- Trouver un ordre topologique
- Initialiser π et pred comme dans les autres algorithmes

Fonctionnement

POUR Tout sommet i de 1 à n dans l'ordre topologique
POUR Tout $j \in \Gamma^+(i)$
SI $\pi_j - \pi_i > l_{ij}$
ALORS $\pi_j \leftarrow \pi_i + l_{ij}$;
$\text{pred}_j \leftarrow i$;
FIN SI
FIN POUR
FIN POUR

Complexité : $O(n+m)$ car chaque arc est examiné exactement une fois.

VI. Application aux problèmes d'ordonnancement

Problème central de l'ordonnancement (scheduling).

Etant donné une réalisation décomposée en tâches (travaux élémentaires) déterminer l'ordre et le calendrier d'exécution des différentes tâches, compte tenu des contraintes auxquelles est soumise l'exécution des tâches et afin d'éviter les pertes de temps.

Exemple 6.1. Construction (maison...), projet informatique, mise en place d'une chaîne de fabrication...

On suppose qu'on peut décomposer le projet en tâches élémentaires qui doivent être exécutées sans interruption (tâches non préemptives).

Données :

- n tâches : $a_1, \ldots, a_n$;
- la durée des tâches : d_i, $i \in \{1, \ldots, n\}$;
- contraintes de précédance : contraintes « potentielles » de type : a_k précède a_l.

Résultat : un ordonnancement des tâches : une date t_i de début pour chaque tâche a_i.

La méthode potentiels-tâches : Bernard ROX, 1960.

Définition 6.2. *Le graphe potentiels-tâches :*

- Chaque sommet correspond à une tâche (et symbolise le début de la tâche) ;
- Chaque arc $a_i a_j$ correspond à une contrainte de précédance, valuée de la durée de la tâche.

Proposition 6.3. *Ce graphe est sans circuit.*

Conséquence : Il existe au moins un sommet sans prédécesseur et un sommet sans successeur.

On crée deux sommets fictifs : α symbolise le début du projet et ω la fin.

On crée un arc de α vers tout sommet sans prédécesseur avec une valeur 0.

On crée un arc de tout sommet sans successeur vers ω avec comme valeur la durée de la tâche.

Ordonnancement au plus tôt : On veut minimiser la date de fin de projet = horizon du projet (make_span). Cet horizon est $\geqslant$ à la longueur de tout chemin de α à ω.

$$H \geqslant \max_{\mu \text{ chemin de } \alpha \text{ à } \omega} (l(\mu))$$

Pour trouver cet ordonnancement, on cherche un chemin critique : un chemin de α à ω de plus longue durée.

Idée : On peut prendre l'opposé des durées et appliquer l'algorithme de Belman.

Remarque. On peut garder les valeurs positives (les durées) et changer le sens des inégalités de potentiel.

Chapitre 3

Flots dans les graphes

I. Flot

Soit s et $t \in X$ deux sommets distincts, soit un graphe $G = (X,U)$ orienté valué par une fonction

$$\begin{array}{rccc} C: & U & \to & \mathbb{R}^+ \\ & ij & \mapsto & C(ij) = C_{ij} \end{array}$$

C_{ij} est la capacité de l'arc ij.

Définition 1.1. Un vecteur $f \in \mathbb{R}_+^{|U|}$ est un st-flot tel que

$$\forall i \in X \setminus \{s,t\}, \sum_{ij \in \delta^+(i)} f_{ij} = \sum_{ji \in \delta^-(i)} f_{ji}$$

(loi de Kirchhoff aux nœuds).

Remarque. Si on relâche la condition de positivité : $f \in \mathbb{R}^{|U|}$, espace vectoriel.

Définition 1.2. Soit f un st-flot. On appelle valeur du flot f la quantité :

$$v(f) = \sum_{sj \in \delta^+(s)} f_{sj} - \sum_{js \in \delta^-(s)} f_{js}$$

Remarque. Dans beaucoup d'applications, s n'a pas de prédécesseur, t n'a pas de successeur. On parle alors de « réseau de transport ».

Proposition 1.3.

$$v(f) = \sum_{sj \in \delta^+(s)} f_{sj} - \sum_{js \in \delta^-(s)} f_{js} = \sum_{jt \in \delta^-(t)} f_{jt} - \sum_{tj \in \delta^+(t)} f_{tj}$$

Démonstration. Soit

$$\alpha = \sum_{i \in X} \left(\sum_{ij \in \delta^+(i)} f_{ij} - \sum_{ji \in \delta^-(i)} f_{ji} \right) = v(f) + \sum_{tj \in \delta^+(t)} f_{tj} - \sum_{jt \in \delta^-(t)} f_{jt}$$

grâce à la définition du flot. Or

$$\alpha = \sum_{ij \in U} f_{ij} - \sum_{ji \in U} f_{ji} = 0$$

car chaque arc est entrant pour un sommet et sortant pour un sommet. □

Définition 1.4. On dit qu'un st-flot est compatible (admissible, réalisable, faisable, respecte les capacités) si :

$$\forall ij \in U,\ f_{ij} \leqslant C_{ij}$$

Exemple.

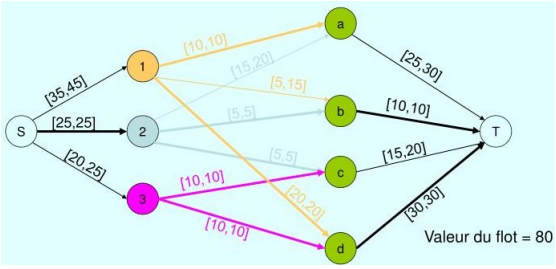


FIGURE 3.1. Flot avec : $[f_{ij}; C_{ij}]$

Définition 1.5. On dit que l'arc $ij \in U$ est saturé si $f_{ij} = C_{ij}$.

II. Coupe d'un graphe

Définition 2.1. On appelle *st*-coupe du graphe G tout ensemble d'arcs déterminés par une bipartition des sommets $\left(S,\overline{S}\right)$ avec $s \in S$ et $t \in \overline{S}$: c'est l'ensemble des arcs ayant leur origine dans S et leur destination dans $\overline{S}$.

Notation. $\delta^+(S)$ est la coupe déterminée par $\left(S,\overline{S}\right)$ $(\overline{S} = X \backslash S)$.

Remarque. Par abus de notation, on écrira parfois « la coupe $\left(S,\overline{S}\right)$ ».

Notation. $\delta^+(s)$ est l'ensemble des arcs sortant d'un sommet s.
$\delta^-(s)$ est l'ensemble des arcs entrant d'un sommet s.

Définition 2.2. La capacité d'une coupe $\delta^+(S)$ est

$$C\left(\delta^+(S)\right) = \sum_{ij \in \delta^+(S)} C_{ij}$$

Proposition 2.3. *Pour tout st-flot compatible f et toute st-coupe $\delta^+(S)$, on a :*

$$v(f) \leqslant c\left(\delta^+(S)\right)$$

Autrement dit, la valeur d'un st-flot compatible est inférieure ou égale à la capacité d'une st-coupe.

Remarque. C'est une relation *min-max*.

Démonstration.

$$v(f) = \left(\sum_{sj \in \delta^+(s)} f_{sj} - \sum_{js \in \delta^-(s)} f_{js}\right) + \sum_{i \in S \backslash \{s\}} \underbrace{\left(\sum_{ij \in \delta^+(i)} f_{ij} - \sum_{ji \in \delta^-(i)} f_{ji}\right)}_{0}$$

On a rajouté la somme (nulle) des flux nets sortant des autres sommets de S. Dans cette somme, les flux des arcs entre deux sommets de S sont comptés deux fois : une fois avec le signe « + » et une fois avec le signe « − ». Il reste les flux entrant et les flux sortant de S. Donc

$$v(f) = \sum_{ij \in \delta^+(S)} f_{ij} - \sum_{ji \in \delta^-(S)} f_{ji}$$

Comme $f_{ij} \geqslant 0$, $\forall ij \in U$, on a

$$v(f) \underbrace{\leqslant}_{(1)} \sum_{ij \in \delta^+(S)} f_{ij} \underbrace{\leqslant}_{(2)} \sum_{ij \in \delta^+(S)} C_{ij} = C\left(\delta^+(S)\right)$$

(1) : égalité ssi $f_{ij} = 0$, $\forall ji \in \delta^-(S)$. (2) : égalité ssi $f_{ij} = C_{ij}$, $\forall ij \in \delta^+(S)$. □

Corollaire 2.4. *Soit un st-flot compatible f et une st-coupe $\delta^+(S)$. On a*

$$v(f) = C\left(\delta^+(S)\right) \Longleftrightarrow \begin{cases} \forall ij \in \delta^-(S) & f_{ij} = 0 \\ \forall ij \in \delta^+(S) & f_{ij} = C_{ij} \end{cases}$$

Questions : 1) Peut-on toujours trouver de tels f et S ? 2) Combien y-a-t'il de *st*-coupe dans un graphe ?

Réponses : 2) Le nombre de bipartitions est $2^n - 2$.

III. Chaîne augmentante

Définition 3.1. Une *st*-chaîne augmentante (chaîne augmentante de s à t) est une chaîne μ entre s et t telle que :

Si ij est un arc de μ, orienté de s vers t alors $f_{ij} < C_{ij}$.

Si ij est un arc de μ, orienté de t vers s alors $f_{ij} > 0$.

Pour une *st*-chaîne augmentante, on peut augmenter la valeur du flot de

$$\alpha = \inf\left(\min_{ij \in \mu \text{ de } s \text{ vers } t} C_{ij} - f_{ij},\ \min_{ij \in \mu \text{ de } t \text{ vers } s} f_{ij}\right)$$

Pour un sommet i de la chaîne μ on a alors 4 cas possibles :

$$\begin{array}{cc} \overset{+\alpha}{\rightarrow}\ i\ \overset{+\alpha}{\rightarrow} & \overset{+\alpha}{\rightarrow}\ i\ \overset{-\alpha}{\leftarrow} \\ \underset{-\alpha}{\leftarrow}\ i\ \underset{-\alpha}{\leftarrow} & \underset{-\alpha}{\leftarrow}\ i\ \underset{+\alpha}{\rightarrow} \end{array}$$

Théorème 3.2. *Un st-flot compatible est de valeur maximale ssi il n'admet pas de st-chaîne augmentante.*

Démonstration. Si f admet une chaîne augmentante de s à t, alors on peut augmenter $v(f)$, donc f n'est pas de valeur maximale.

Réciproquement, soit f un *st*-flot compatible n'admettant pas de *st*-chaîne augmentante.

Soit

$$S = \{x \in X \mid \text{il existe une chaîne augmentante de } s \text{ à } x\} \cup \{s\}$$

Comme il n'y a pas de *st*-chaîne augmentante : $t \notin S$. Soit $\overline{S} = X \backslash S$ et $\delta^+(S)$ la coupe définie par $\left(S,\overline{S}\right)$. Pour un arc xy avec $x \in S$ et $y \in \overline{S}$, l'arc est forcément saturé ($f_{xy} = C_{xy}$) car sinon on aurait $y \in S$ par construction.

Pour un arc $x'y'$ avec $x' \in \overline{S}$ et $y' \in S$, on a forcément $f_{x'y'} = 0$ car sinon on aurait $x' \in S$.

D'après le corollaire, on a $v(f) = C(\delta^+(S))$ et donc le flot f est de valeur maximale. □

Théorème 3.3. *de Ford-Fulkerson : La valeur maximale d'un st-flot compatible est égale à la capacité maximale d'une st-coupe.*

IV. Algorithme de Ford-Fulkerson

Pour le calcul d'un st-flot de valeur maximale.

Principe : On détecte une chaîne augmentante par un algorithme de marquage (parcours de graphe) et on augmente le flot sur cette chaîne de la valeur α. Lorsqu'on ne trouve plus de chaîne augmentante, c'est terminé.

On donne l'algorithme haut niveau. Il en existe bien sûr de nombreuses variantes et améliorations.

Initialisation
Un flot de s à t (par exemple le flot nul).

Fonctionnement
Booléen Arrêt ← FAUX
REPETER
  Marquage() ;
  SI t n'est pas marqué : Arrêt ← VRAI
    SINON Modifier le flot f sur la chaîne augmentante trouvée, de la valeur α.
  FINSI
JUSQU'A (Arrêt = VRAI).

L'algorithme de marquage peut s'écrire :
Algo Marquage()
Marquer + le sommet s
TANT QUE on peut marquer des sommets
  POUR TOUT arc ij
    SI i est marqué et $f_{ij} < C_{ij}$
      marquer j par $+i$
    FINSI
    SI j est marqué et $f_{ij} > 0$
      marquer i par $-j$
    FINSI
  FIN POUR
FIN TANT QUE

V. Arbre couvrant de poids minimal

Soit un graphe non orienté $G = (X,U)$. A chaque arête u de U, on associe un poids $c(u)$. On suppose que G est connexe. Le problème est de trouver un graphe partiel G qui soit un arbre et dont le poids total des arêtes est minimal.

Proposition 5.1. *Soit $G = (X,U)$ un graphe valué et soit $F = \{(X_1,U_1), (X_2,U_2), \ldots, (X_k,T_k)\}$ une forêt de G (i.e. un graphe partiel sans cycle). Soit xy une plus petite arête (au sens des poids) ayant exactement une extrémité dans X_1. Alors, parmi tous les arbres contenants les arêtes de F, et qui sont de plus petits poids, il y en a un qui contient xy.*

Démonstration. Cf. TD. □

Algorithme de Kruskal (1956) :

1. On part d'une forêt de n arbres réduits chacun à un sommet isolé (n est le nombre de sommets du graphe G) ;

2. A chaque itération, on ajoute à cette forêt la plus petite arête ne créant pas de cycle avec celles déjà choisies (autrement dit on obtient une nouvelle forêt en ajoutant cette arête) ;
3. On s'arrête quand on a choisi $(n-1)$ arêtes ou quand on ne trouve plus d'arête (le graphe n'est pas connexe).

Autre présentation de l'algo de Kruskal :

1. Trier les arêtes par poids croissants ;
2. Dans l'ordre de la liste triée des arêtes, si ajouter une arête ne crée pas de cycle, alors on l'adjoint $\Longleftrightarrow$ Si les extrémités de l'arête ne sont pas dans le même arbre, alors on ajoute l'arête.

Complexité : $O(m \log m)$

Algorithme de Kruskal (1957) :

1. On part de l'arbre initial T réduit à un sommet quelconque x ;
2. A chaque itération, on agrandi l'arbre T en le connectant au « plus proche » (au sens des poids) sommet non encore connecté ;
3. On s'arrête lorsqu'on a sélectionné $(n-1)$ arêtes ou qu'on ne trouve plus d'arêtes.