Matroïdes et Algorithmes Gloutons

Rencontre « Algorithmique et Programmation »

CIRM (Luminy), 5-9 mai 2003

Pierre BÉJIAN

pierre.bejian@univ.u-3mrs.fr
bejian@free.fr
http://bejian.free.fr

GLOUTON, -ONNE adj. et n. XIe siècle, *gloton* ; XVIIe siècle, comme nom d'animal. Issu du latin impérial *glut(t)onem*, « glouton », dérivé de *glut(t)us*, « gosier ».
1. Adj. Qui mange avec avidité et excès, voracement. *Un enfant glouton*. Par méton. *Un appétit glouton*. *Une faim gloutonne*. Subst. *Cet homme est un glouton*. Fig. Avide. *Une jeunesse gloutonne de plaisirs*. *Une curiosité gloutonne*. **2.** N. m. Mammifère carnivore de la famille des Mustélidés. *Les gloutons vivent dans les régions arctiques.*

Plan

1. Le principe glouton
2. Arbres couvrants de poids maximum
3. Matroïdes
4. Généralisation : gloutonoïdes

1. Le principe glouton

– pour un problème d'optimisation
– construction séquentielle
– à chaque étape on fait le meilleur choix local

Pas de retour en arrière : directement vers une solution.

Le résultat n'est pas toujours optimal.

Progression descendante = choix **puis** résolution d'un problème plus petit.

Premier exemple : location d'un camion

Un véhicule unique est proposé à la location.
On veut maximiser le nombre de clients satisfaits.

Soit E l'ensemble des demandes. Pour $e \in E$ on note $d(e)$ (resp. $f(e)$) la date de début (resp. de fin) de la demande e.

Les demandes e_1 et e_2 sont compatibles ssi

$$]d(e_1), f(e_1)[\cap]d(e_2), f(e_2)[= \varnothing$$

On cherche un sous-ensemble de E de demandes deux à deux compatibles, de cardinal maximum.

LocationCamion(E)

Trier les élements de E par date de fin croissante
$f(e_1) \leq f(e_2) \leq \cdots \leq f(e_n)$
$s_1 \leftarrow e_1$ // *on choisit la demande qui termine le plus tôt*
$k \leftarrow 1$ // *k désigne le nombre de clients actuellement satisfaits*
pour $i = 2$ à n **faire**
  si $d(e_i) \geq f(s_k)$ **alors**
    $s_{k+1} \leftarrow c_i$
    $k \leftarrow k + 1$
  fin si
fin pour
retourner $\{s_1, ..., s_k\}$

Théorème : L'algorithme donne bien un résultat optimal.

Complexité : $\mathscr{O}(n \log n)$

Exemple :

	e_1	e_2	e_3	e_4
d	0	0	2	3
f	1	3	6	6

Remarques :

– ne marche pas pour maximiser le temps total de location
– idem si on trie par durée décroissante

2. Arbres couvrants de poids maximum

Vocabulaire : graphe (non-orienté), chemin, cycle, composante connexe, ...

Graphe acyclique (forêt) : sans cycle élémentaire non trivial
Arbre $=$ forêt connexe
Si $G = (S, A)$ connexe, alors G est un arbre $\Leftrightarrow a = s - 1$

Arbre couvrant de $G =$ qui contient tous les sommets de G
Couvrant $\Leftrightarrow$ maximal pour l'inclusion (faux pour les forêts).

Pondération $w : A \to \mathbb{R}_+$. Pour $X \subset A$ le poids de X est

$$f(X) = \sum_{x \in X} w(x).$$

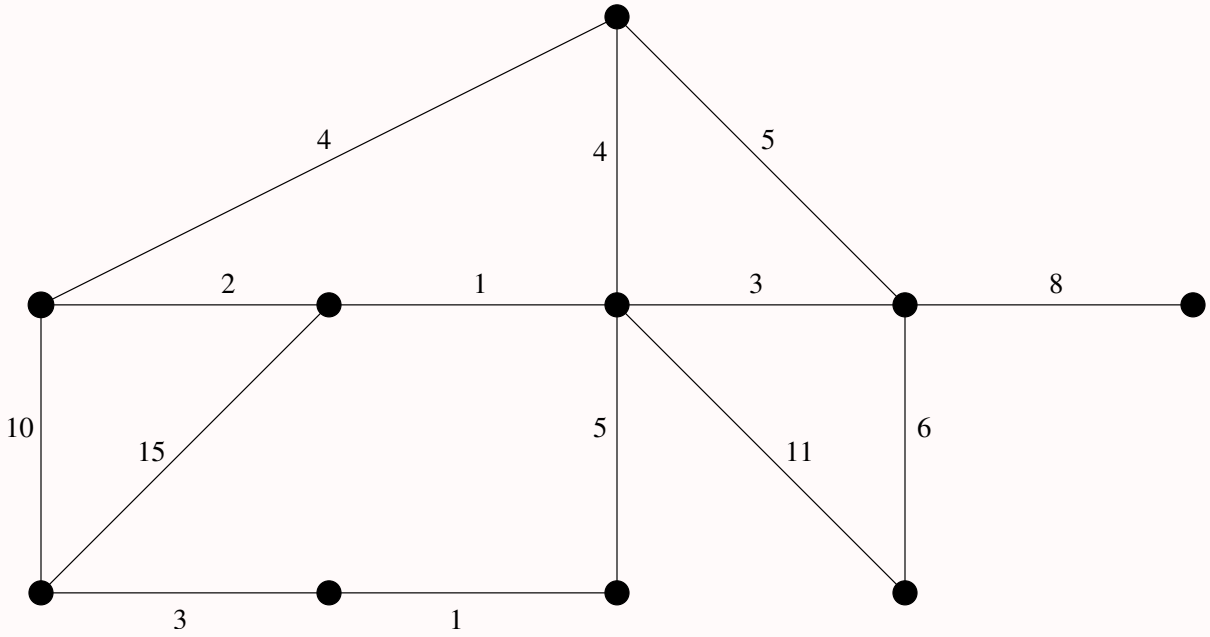
But : trouver un arbre couvrant de G de poids maximum.

Algorithme de Kruskal

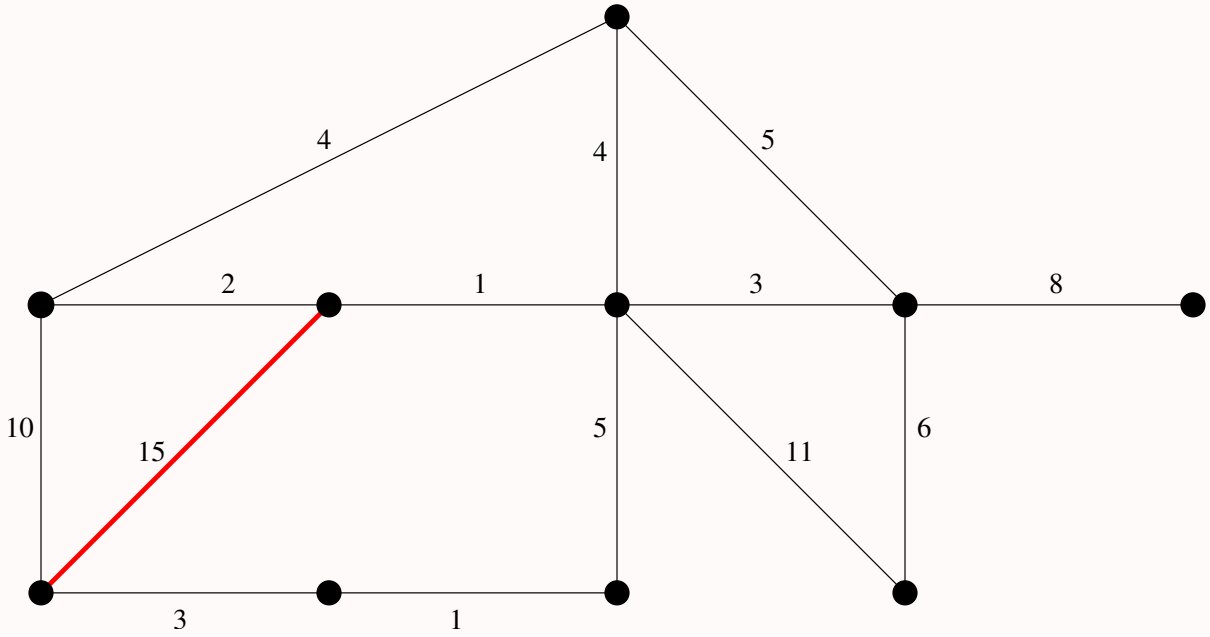
KRUSKAL(G,w)
  Trier les élements de A par ordre de poids décroissant
    $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_a)$
  $T \leftarrow \varnothing$
  pour $i = 1$ à a **faire**
    si $T \cup \{e_i\}$ est acyclique **alors**
      $T \leftarrow T \cup \{e_i\}$
    fin si
  fin pour
  retourner T

Remarque : à chaque étape, T est une forêt.

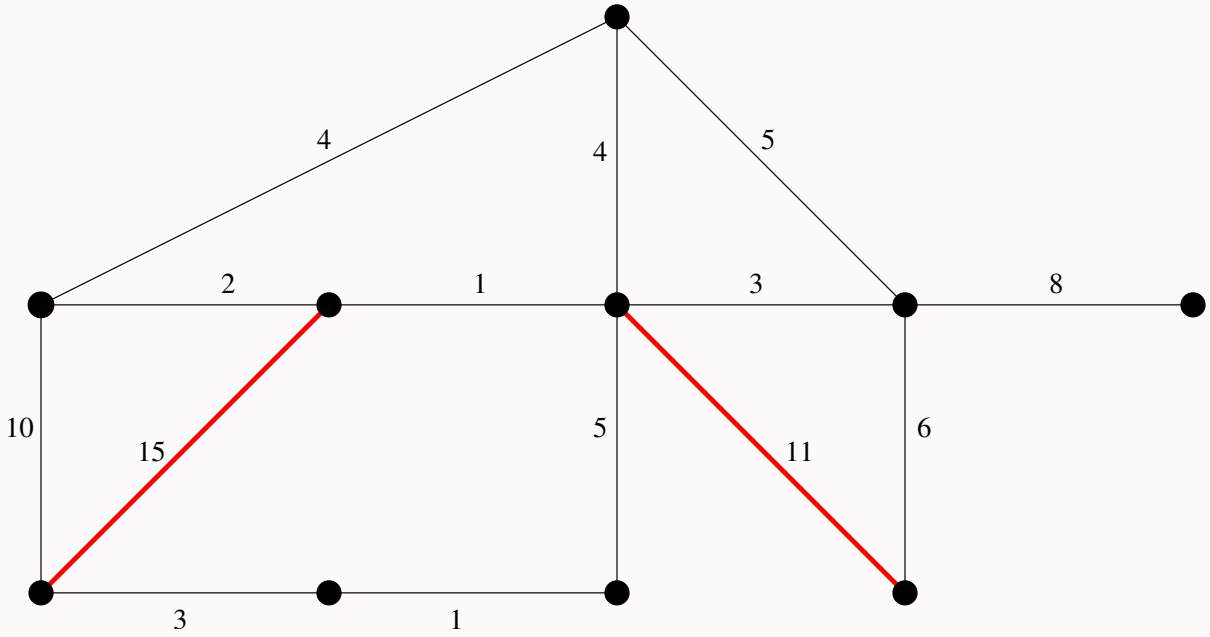
Algorithme de Kruskal



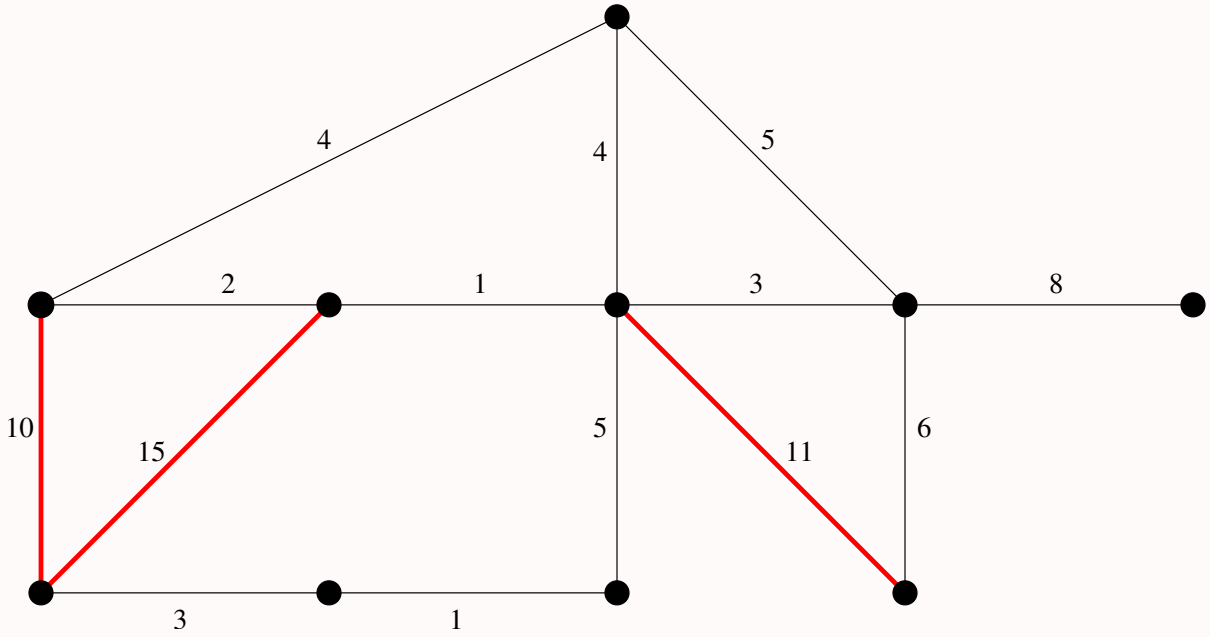
Algorithme de Kruskal



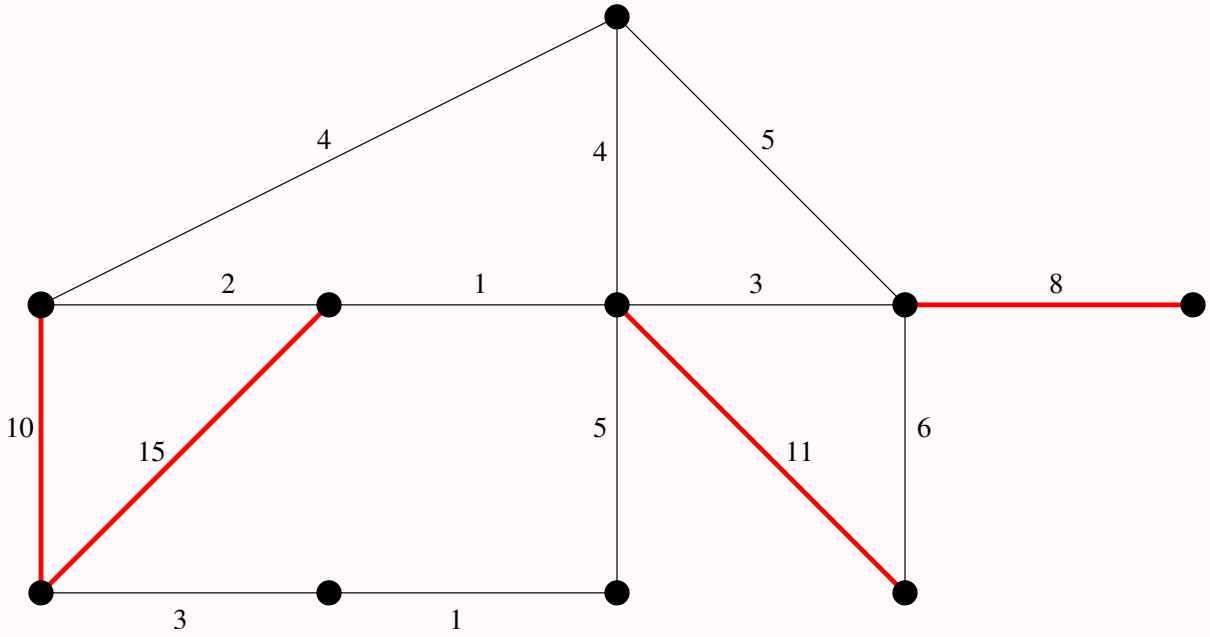
Algorithme de Kruskal



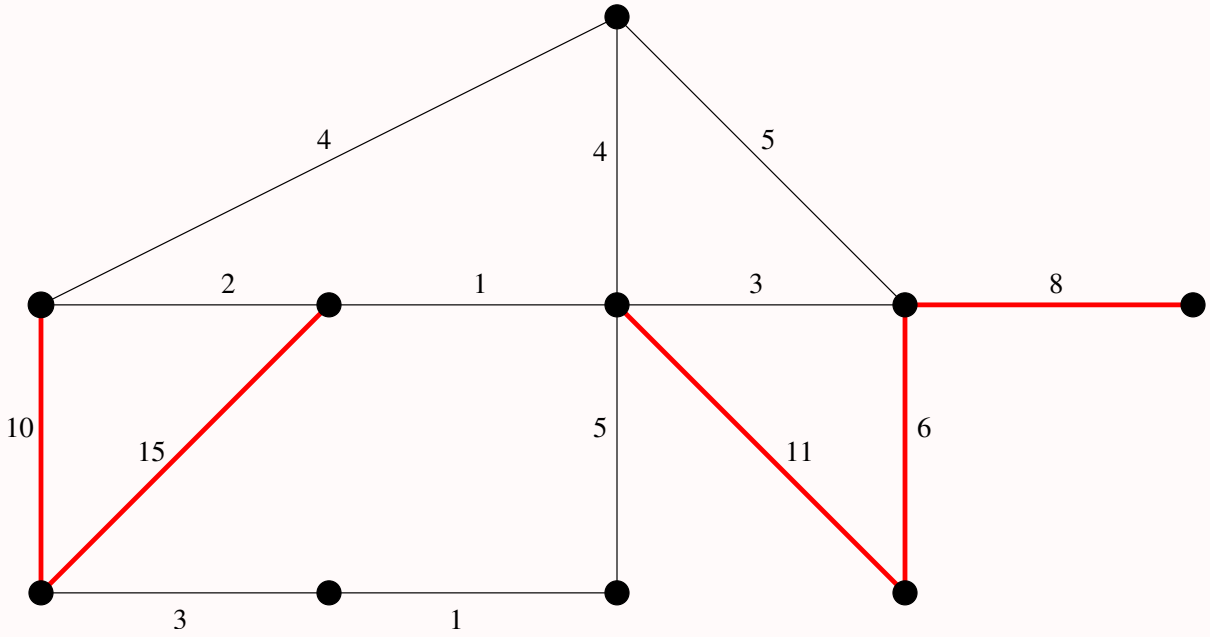
Algorithme de Kruskal



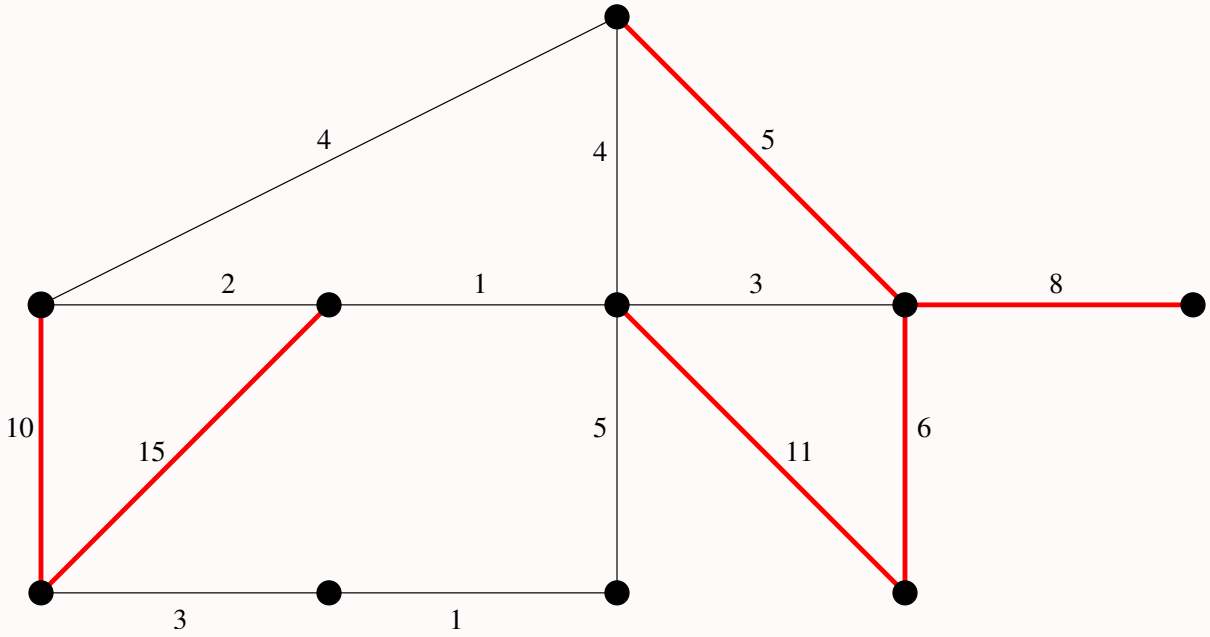
Algorithme de Kruskal



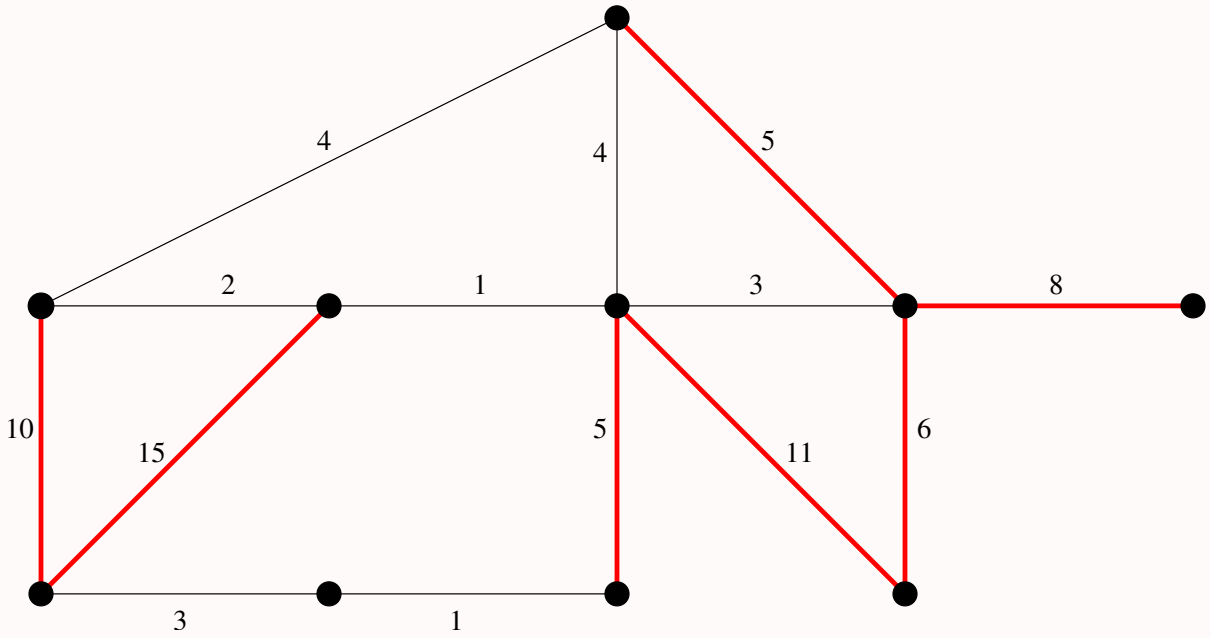
Algorithme de Kruskal



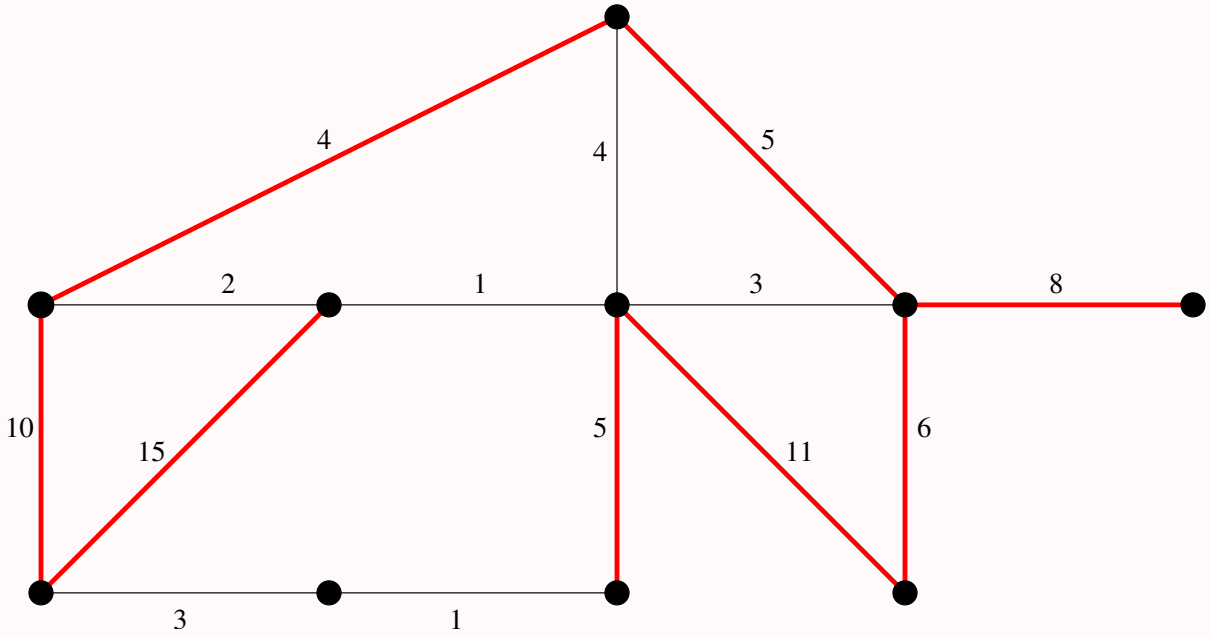
Algorithme de Kruskal



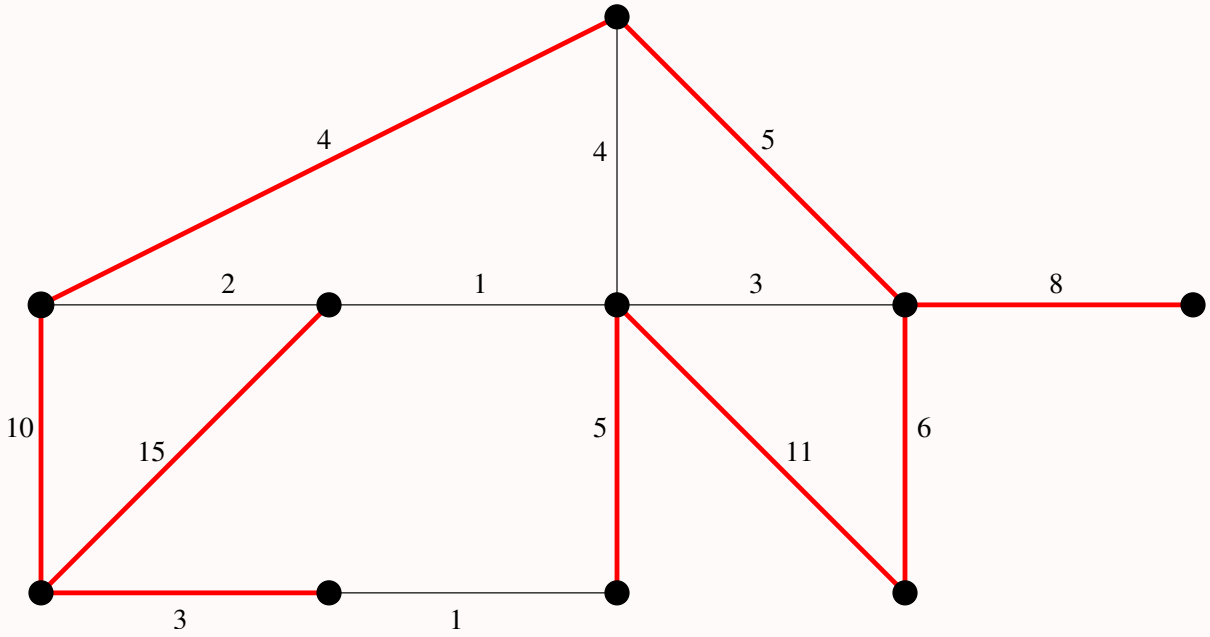
Algorithme de Kruskal



Algorithme de Kruskal



Algorithme de Kruskal



Optimalité : voir section 3 (matroïdes)

Remarques :

– arbre couvrant de poids minimum
– fonctionne si G n'est pas connexe
– poids négatifs (et G connexe)

Complexité : $\mathscr{O}(a \log s)$ (voir biblio)

Algorithme de Prim

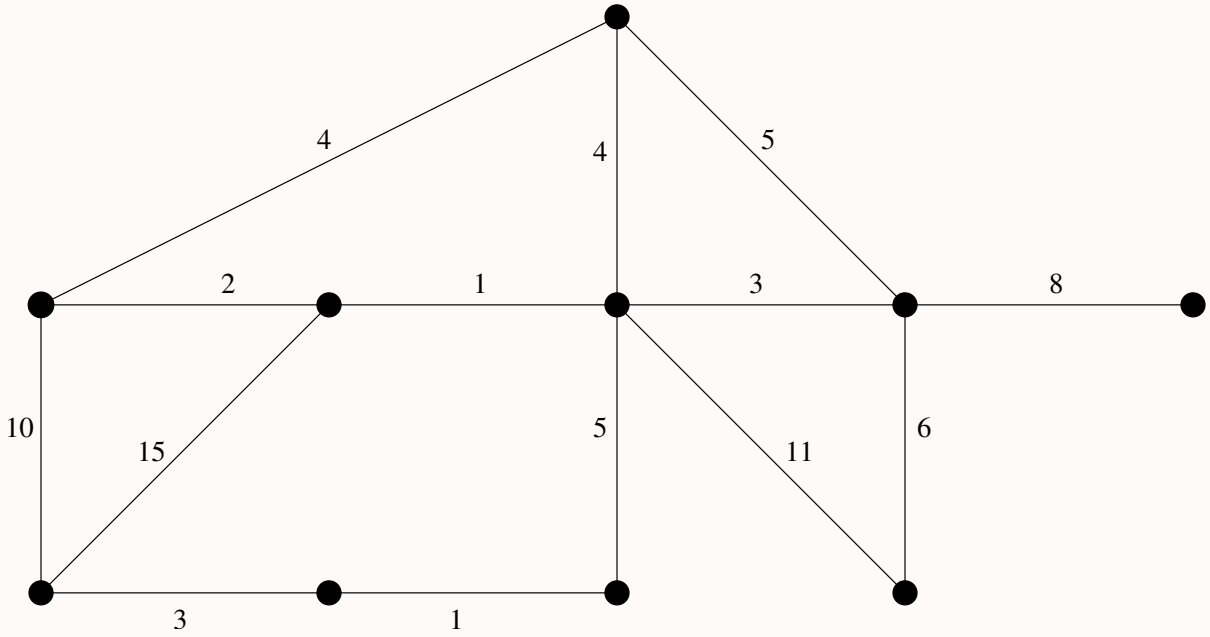
G est supposé connexe

PRIM(G,w,v_0)
  $T \leftarrow \varnothing$
  tant que T n'est pas couvrant **faire**
    Choisir une arête e de poids maximum telle que $T \cup \{e\}$ soit un arbre contenant v_0
    $T \leftarrow T \cup \{e\}$
  fin tant que
  retourner T

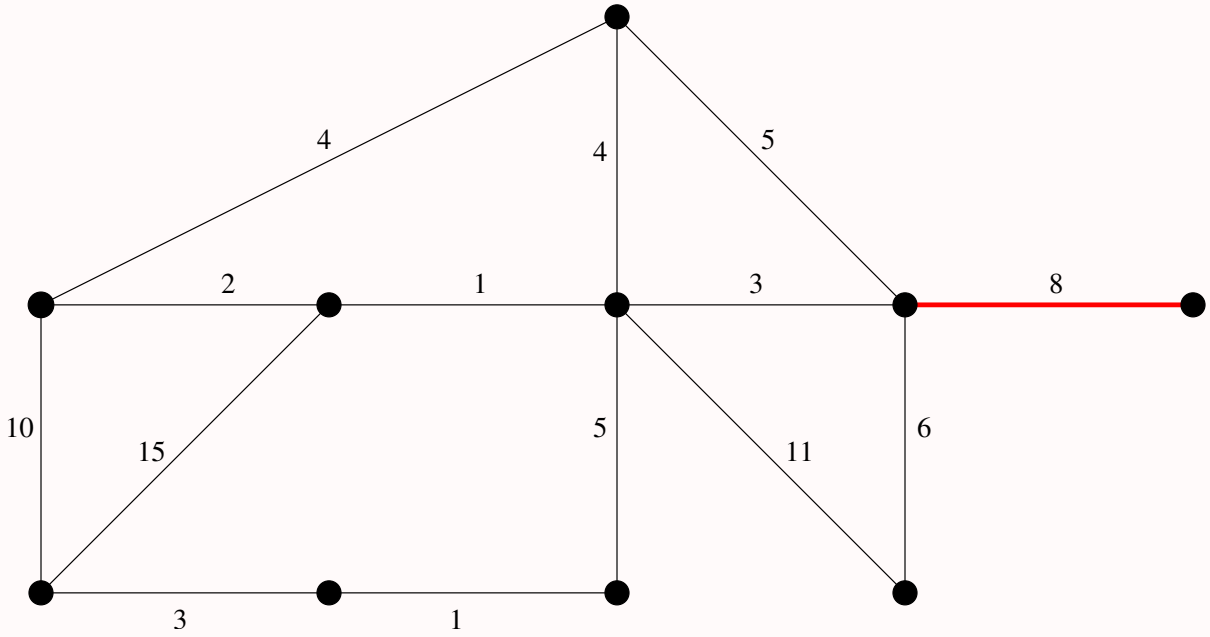
Remarque : à chaque étape T est un arbre contenant v_0

Algorithme de Prim

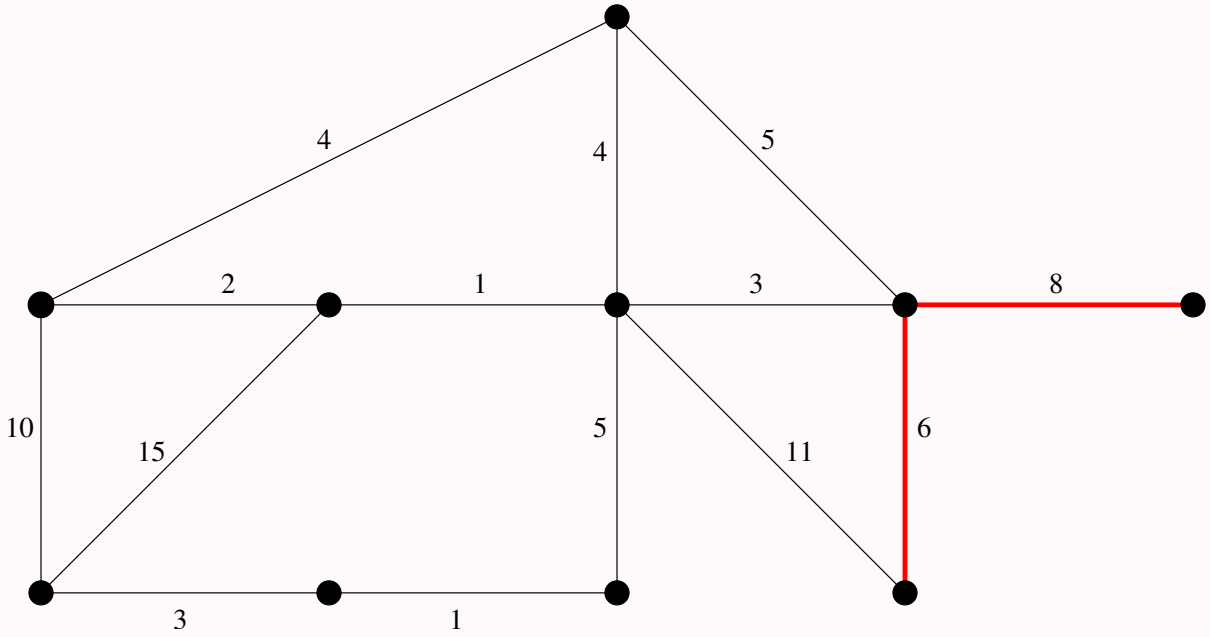
(v_0 est le sommet situé le plus à droite)



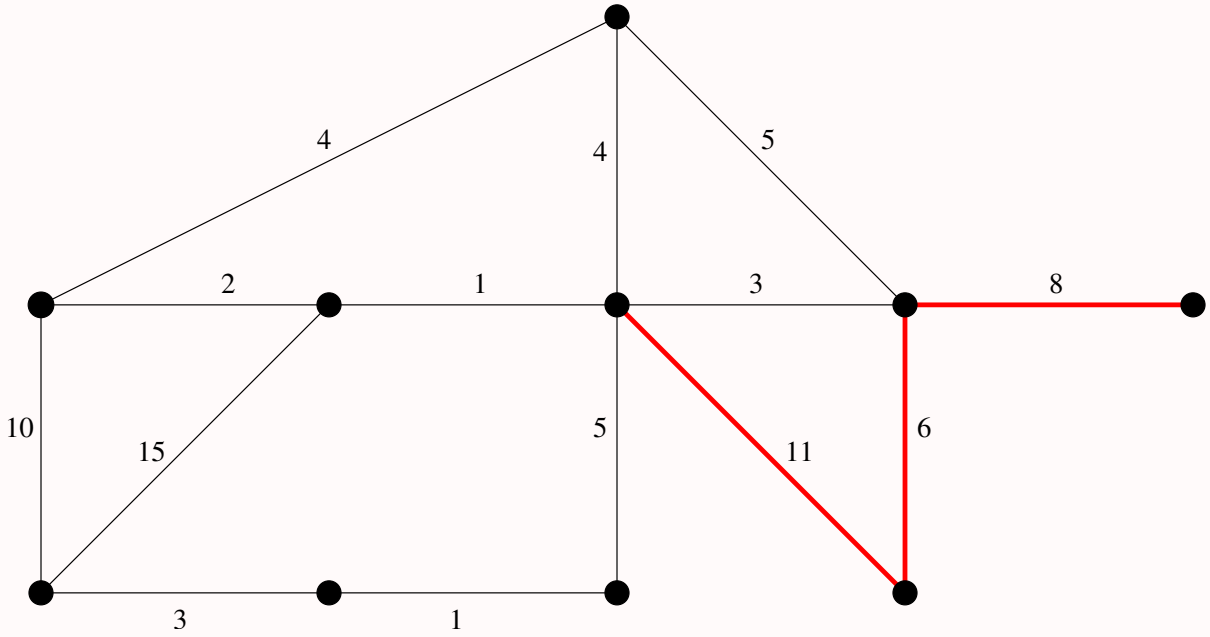
Algorithme de Prim



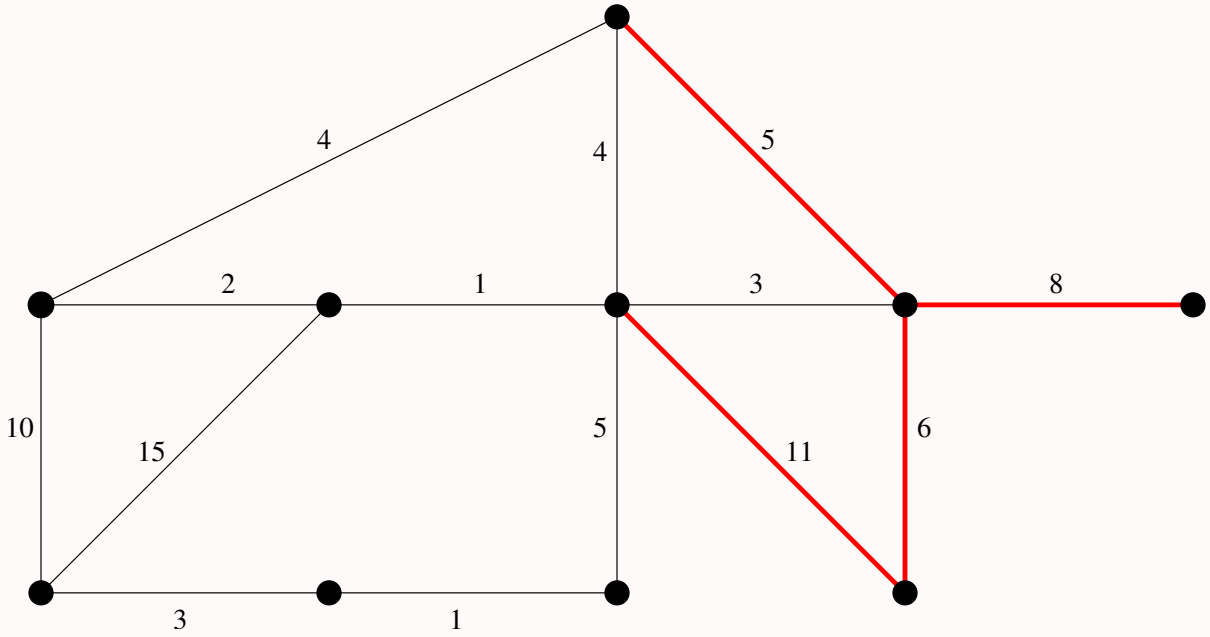
Algorithme de Prim



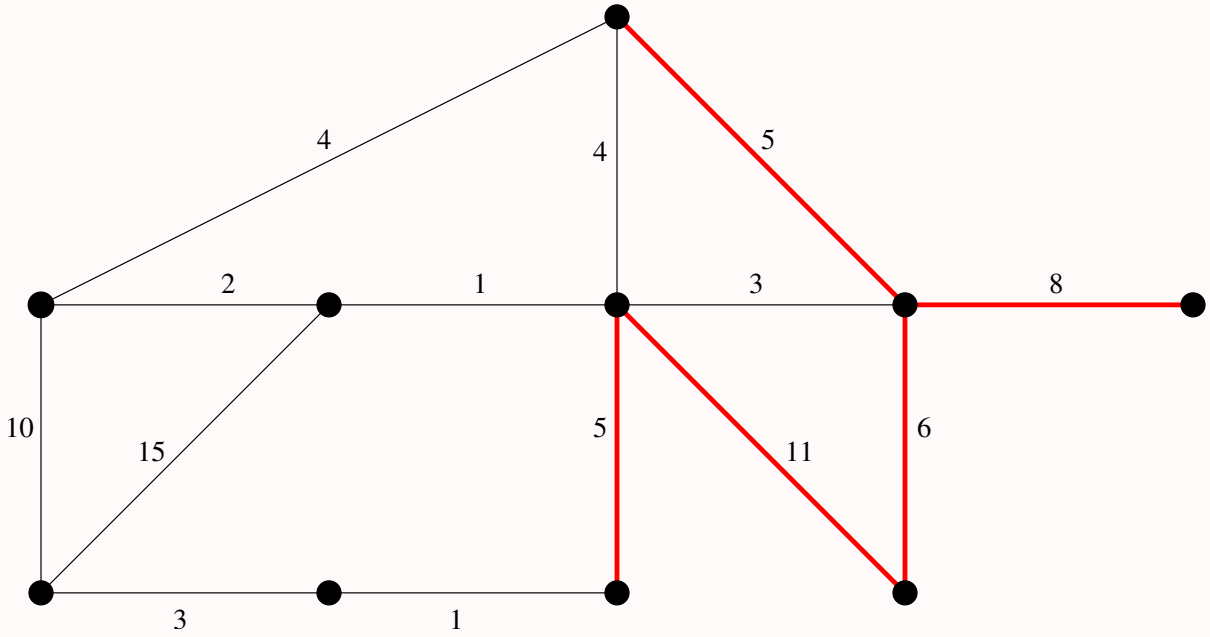
Algorithme de Prim



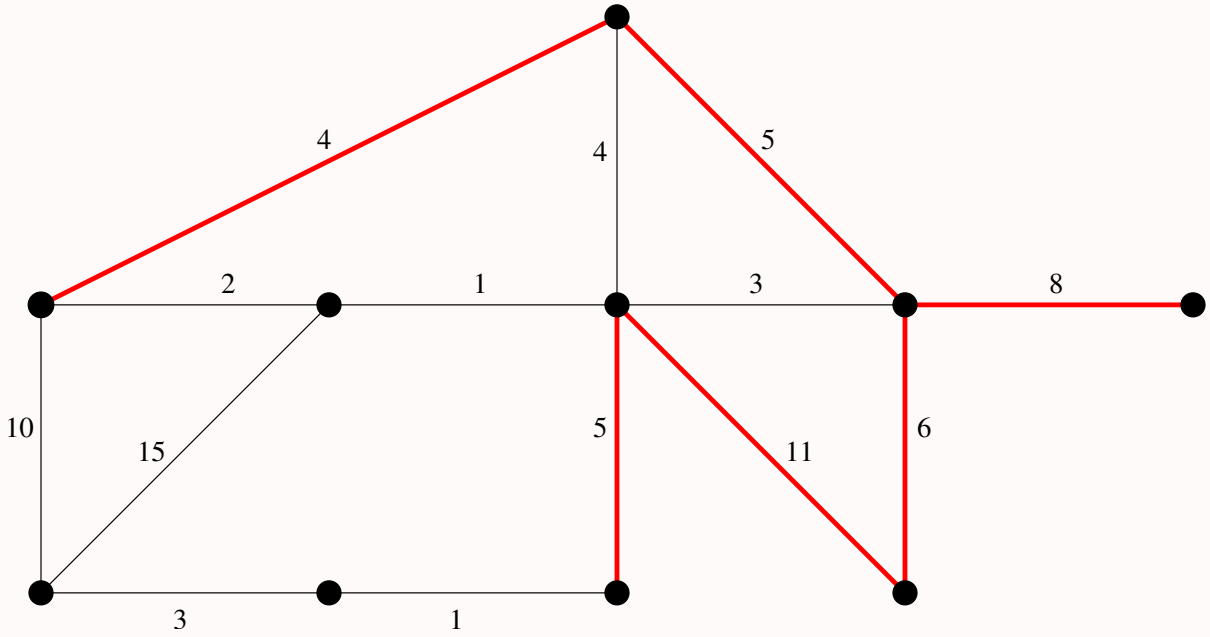
Algorithme de Prim



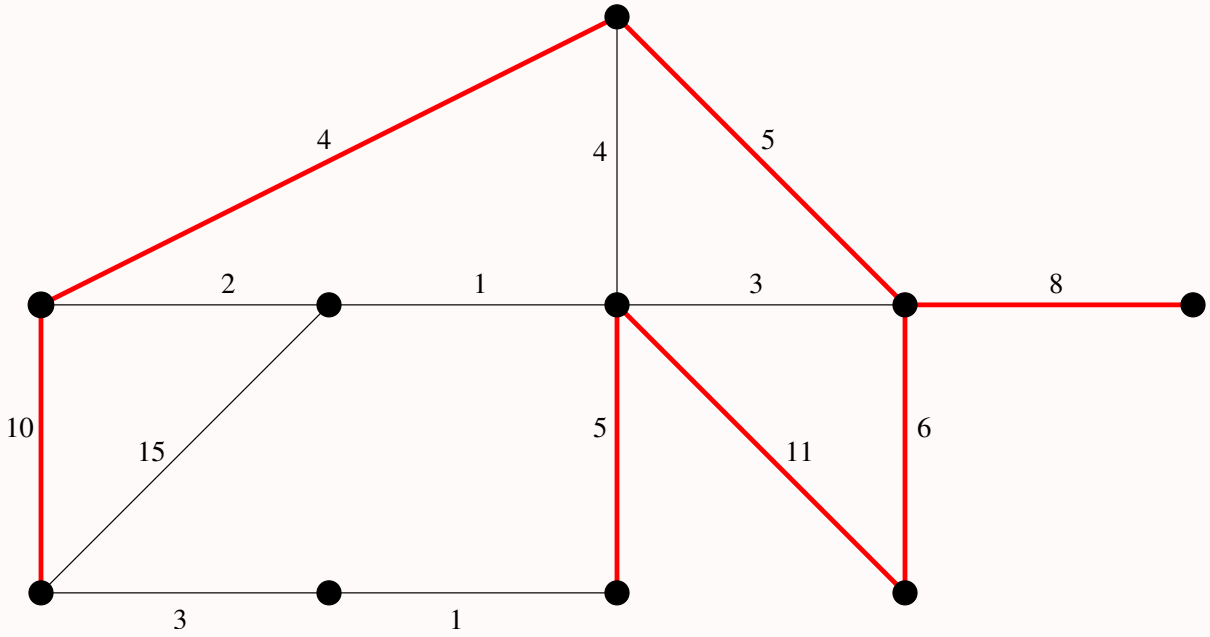
Algorithme de Prim



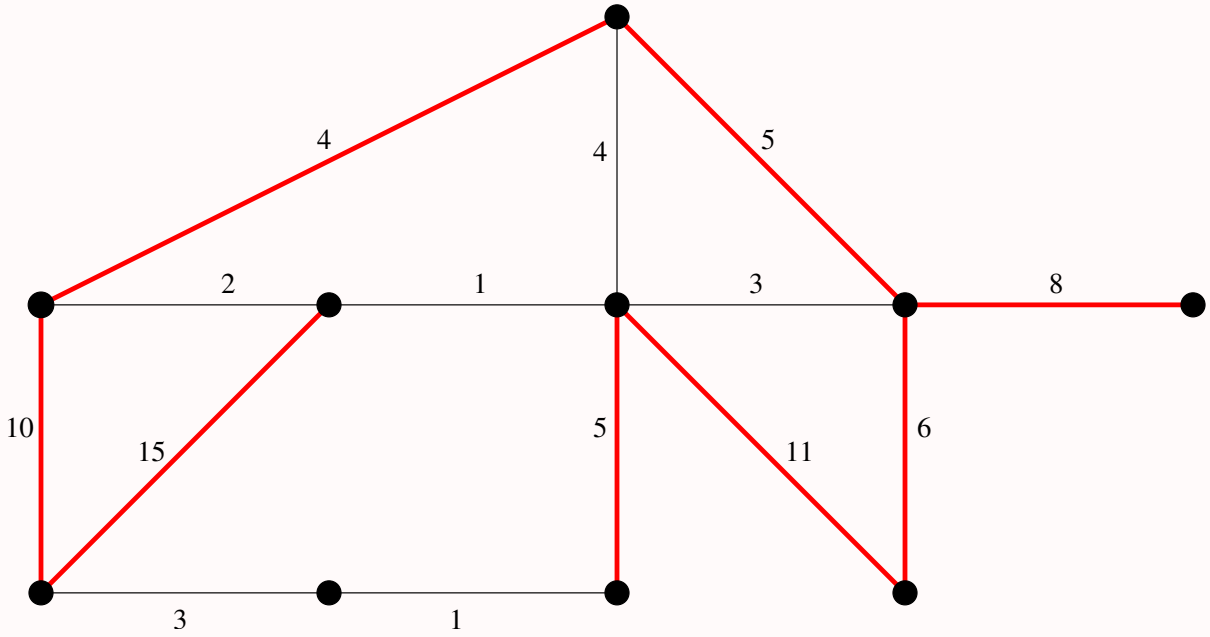
Algorithme de Prim



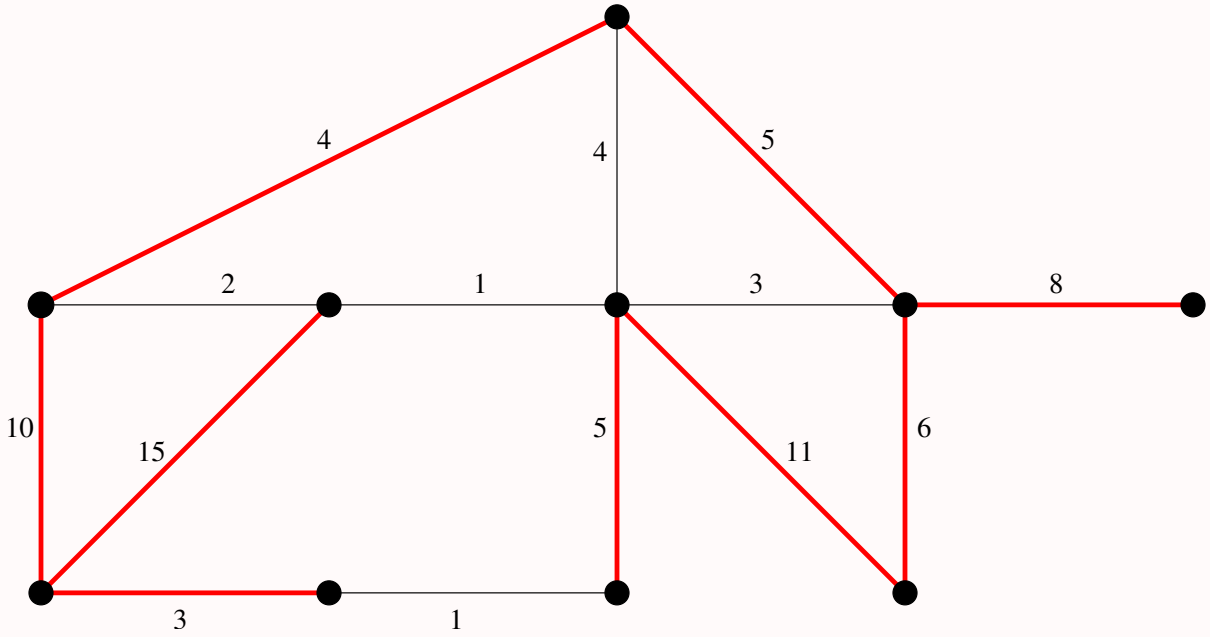
Algorithme de Prim



Algorithme de Prim



Algorithme de Prim



Remarque : importance de la « racine » v_0 : voir section 4 (gloutonoïde)

Complexité : $\mathscr{O}(a \log s)$ et même un peu mieux ... (voir biblio)

Optimalité : voir section 4 (gloutonoïde)

3. Matroïdes

Cadre général de la suite de l'exposé :

Définition : Un *système ensembliste* $\mathscr{S} = (E, \mathscr{F})$ est la donnée d'un ensemble fini E et d'une collection $\mathscr{F}$ de sous-ensembles de E.

Définition : On appelle *extension* de $F \in \mathscr{F}$ tout élément $x \notin F$ tel que $F \cup \{x\} \in \mathscr{F}$. On notera $ext(F)$ l'ensemble des extensions de F.

Définition : Un ensemble de la collection $\mathscr{F}$ est dit *indépendant*. Un ensemble indépendant maximal pour l'inclusion est une *base*. On notera $\mathscr{B}$ l'ensemble des bases.

On considère les axiomes suivants :

(trivial)	$\varnothing \in \mathscr{F}$
(hérédité)	$\forall X \in \mathscr{F}, \forall Y \subset E, Y \subset X \Rightarrow Y \in \mathscr{F}$
(augmentation)	$\forall X, Y \in \mathscr{F}, \lvert X\rvert = \lvert Y\rvert + 1 \Rightarrow$ $\exists x \in X \setminus Y,\ Y \cup \{x\} \in \mathscr{F}$
(échange)	$\forall X, Y \in \mathscr{F}, \lvert Y\rvert < \lvert X\rvert \Rightarrow$ $\exists x \in X \setminus Y,\ Y \cup \{x\} \in \mathscr{F}$

Définition : Un système est *héréditaire* s'il vérifie l'axiome trivial et l'axiome d'hérédité. Un système héréditaire est un *matroïde* s'il vérifie en plus l'axiome d'augmentation.

Lemme : Un matroïde vérifie l'axiome d'échange.

Lemme : Soit $\mathscr{S}$ un système vérifiant l'axiome d'échange (par exemple un matroïde). Alors les bases de $\mathscr{S}$ ont toutes le même cardinal (appelé rang de $\mathscr{S}$).

Exemple 1 :

Dans le problème de la location d'un camion, notons E l'ensemble des demandes et $\mathscr{F}$ la collection formée des ensembles de demandes compatibles.
Il est clair que $(E, \mathscr{F})$ est un système héréditaire.
Considérons maintenant l'exemple suivant :

	e_1	e_2	e_3
d	0	0	1
f	2	1	2

On obtient alors

$$\mathscr{F} = \{\{e_1\}, \{e_2\}, \{e_3\}, \{e_2, e_3\}\}$$

$$\mathscr{B} = \{\{e_1\}, \{e_2, e_3\}\}$$

Le système $(E, \mathscr{F})$ possède des bases de cardinalité différente : ce n'est pas un matroïde.

Exemple 2 :
Soit E un ensemble fini de vecteurs de $\mathbb{R}^n$ (par exemple les colonnes d'une matrice).
Soit $\mathscr{F}$ la collection des ensembles formés de vecteurs linéairement indépendants.
Alors $(E, \mathscr{F})$ est un matroïde.
Historique : Whitney 1935
Le vocabulaire vient de l'algèbre linéaire : *indépendant*, *base*, *rang*, ...

Exemple 3 :
Soit $G = (S, A)$ un graphe.
Posons $E_G = A$ et soit $\mathscr{F}_G$ la collection des ensemble acycliques de A (i.e. des forêts de G).
Le système $\mathscr{M}_G = (E_G, \mathscr{F}_G)$ est un matroïde appelé *matroïde graphique* associé à G.

On considère un système héréditaire $\mathscr{S} = (E, \mathscr{F})$ et une fonction (pondération) $w : E \to \mathbb{R}$. Pour tout $F \in \mathscr{F}$ le poids de F est

$$f(X) = \sum_{x \in X} w(x).$$

Le but est de trouver une *base* de $\mathscr{S}$ de poids maximum. On propose l'algorithme glouton suivant :

GLOUTON1($\mathscr{S}$,w)
  Trier les élements de E par ordre de poids décroissant
    $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_n)$
  $F \leftarrow \varnothing$
  pour $i = 1$ à n **faire**
    si $F \cup \{e_i\} \in \mathscr{F}$ **alors**
      $F \leftarrow F \cup \{e_i\}$
    fin si
  fin pour
  retourner F

Complexité :

Elle dépend essentiellement du coût du test d'indépendance. Si celui-ci se fait en $\mathscr{O}(T(n))$, le temps de calcul total est en $\mathscr{O}(n(\log n + T(n)))$.

Remarque :

Dans le cas d'un matroïde graphique on retrouve l'algorithme de Kruskal.

Lemme : Soit $\mathscr{S} = (E, \mathscr{F})$ un système héréditaire. Alors l'ensemble retourné par l'algorithme $\textsc{Glouton1}$ est une base de $\mathscr{S}$.

Théorème (Edmonds 1971)
Soit $\mathscr{S} = (E, \mathscr{F})$ un système héréditaire. Alors $\textsc{Glouton1}$ fournit une solution optimale pour toute pondération w si et seulement si $\mathscr{S}$ est un matroïde.

Ce théorème démontre donc la validité de l'algorithme glouton dans le cas des matroïdes.

En particulier on dispose d'une (nouvelle) preuve d'optimalité de l'algorithme de Kruskal.

Nouveaux axiomes :

(accessibilité) $\forall X \in \mathscr{F},\ X \neq \varnothing \Rightarrow \exists x \in X,\ X \setminus \{x\} \in \mathscr{F}$

(échange fort) $\forall X \in \mathscr{F}, \forall B \in \mathscr{B}$, si $x \notin B$ et $X \cup \{x\} \in \mathscr{F}$ alors $\exists y \in B \setminus X$ tel que $X \cup \{y\} \in \mathscr{F}$ et $B \cup \{x\} \setminus \{y\} \in \mathscr{F}$

Définition : Un système $\mathscr{S} = (E, \mathscr{F})$ est dit *accessible* s'il vérifie l'axiome trivial et l'axiome d'accessibilité. Un gloutonoïde est un système accessible vérifiant l'axiome d'augmentation.

Remarque : **(hérédité)** $\Rightarrow$ **(accessibilité)** donc un matroïde est un gloutonoïde.

Lemme : Si $\mathscr{S}$ vérifie l'axiome trivial, alors l'axiome d'échange est équivalent aux axiomes d'accessibilité et d'augmentation.

Corollaire : Les bases d'un gloutonoïde ont le même cardinal.

Exemple :

Soit $G = (S, A)$ un graphe.
Posons $E_G = A$ et soit $\mathscr{F}_{G,v_0}$ la collection des arbres contenant v_0.
Le système $\mathscr{M}_G = (E_G, \mathscr{F}_G)$ est un gloutonoïde appelé *gloutonoïde graphique* associé à G, enraciné au point v_0.

Remarque :

Sans la racine v_0 l'axiome d'augmentation n'est pas vérifié.
Il suffit pour le voir de prendre deux arbres n'ayant aucun sommet en commun.

Version modifiée de l'algorithme GLOUTON1 (qui ne marche que pour un sytème héréditaire) :

GLOUTON2($\mathscr{S}$,w)
  $F \leftarrow \varnothing$
  tant que $ext(F) \neq \varnothing$ **faire**
    Choisir $e \in ext(F)$ de poids maximum
    $F \leftarrow F \cup \{e\}$
  fin tant que
  retourner F

Théorème (Korte-Lovász 1984)
Soit $\mathscr{S} = (E, \mathscr{F})$ un gloutonoïde. Alors l'algorithme GLOUTON2 fournit une solution optimale pour toute pondération w si et seulement si $\mathscr{S}$ vérifie l'axiome d'échange fort.

Complexité : Sous cette forme générale l'algorithme a un coût très important puisqu'il calcule $ext(F)$ à chaque étape. Mais on ne peut pas en dire plus car la complexité précise dépend de l'implémentation.

Remarque 1 : L'accessibilité est une condition de base pour que l'algorithme glouton puisse contruire une solution séquentiellement.

Remarque 2 : Dans le cas d'un gloutonoïde graphique, on retrouve bien sûr l'algorithme de Prim. On peut d'ailleurs vérifier que l'axiome d'échange est alors vérifié.

Remarque 3 : Pour l'algorithme de Prim il est important d'imposer à chaque étape que l'arbre en construction contienne un sommet donné pour prouver que l'on a un gloutonoïde.
Mais si on s'affranchit de cette condition, on constate que l'algorithme fonctionne encore. En fait on obtient alors un *plongement matroïdique*, une autre généralisation possible des matroïdes.