Arbre Recouvrant de Poids Minimal

Philippe Lac
(philippe.lac@ac-clermont.fr)
Malika More
(malika.more@u-clermont1.fr)

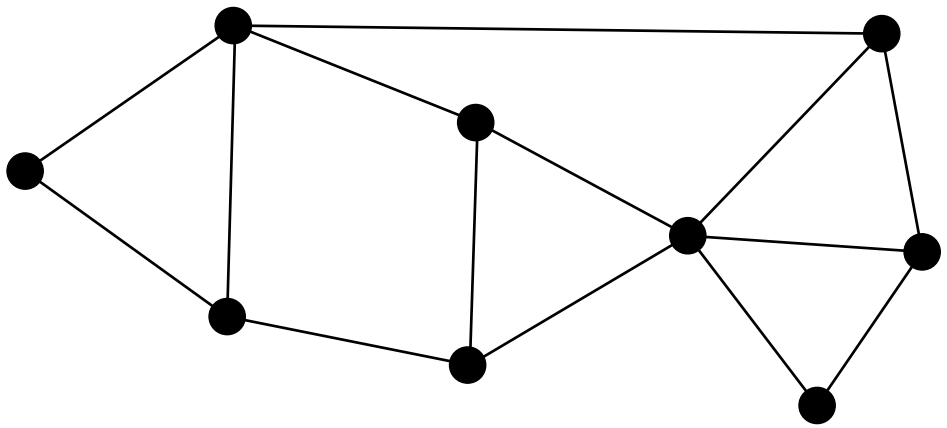
IREM Clermont-Ferrand

Stage Algorithmique

Année 2010-2011

Définition

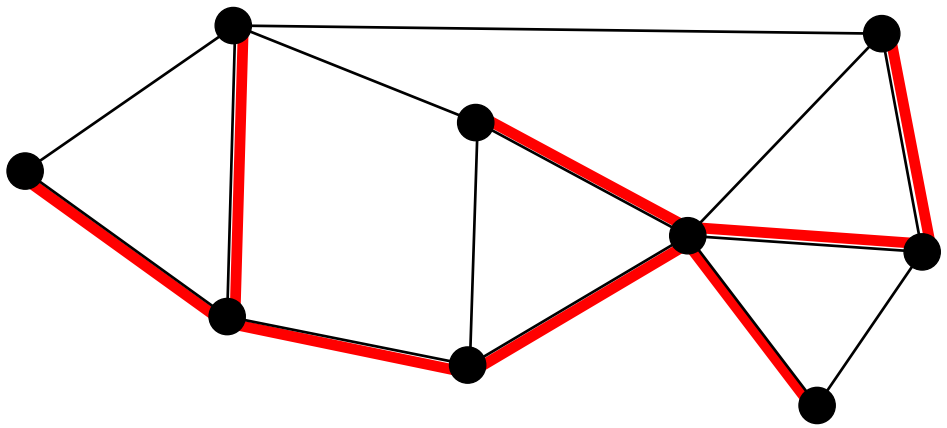
- graphe non orienté G
- arbre T $\begin{cases} \text{sommets : tous les sommets de } G \\ \text{arêtes : certaines arêtes de } G \end{cases}$



T arbre recouvrant de G

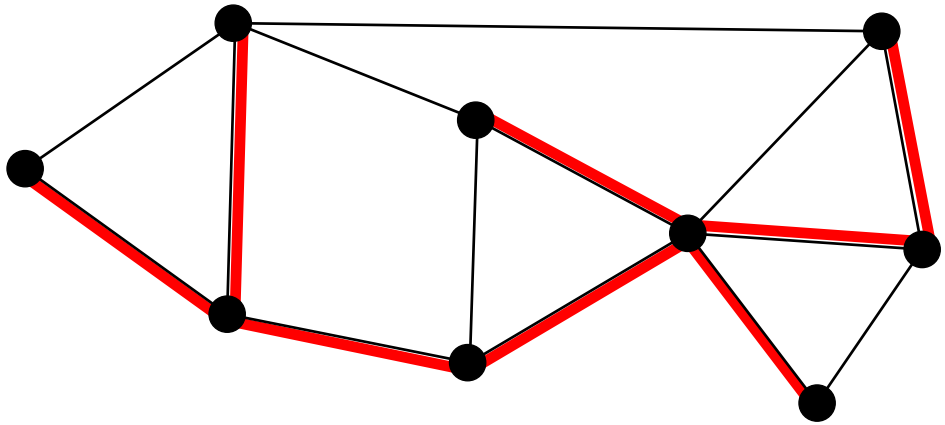
Définition

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



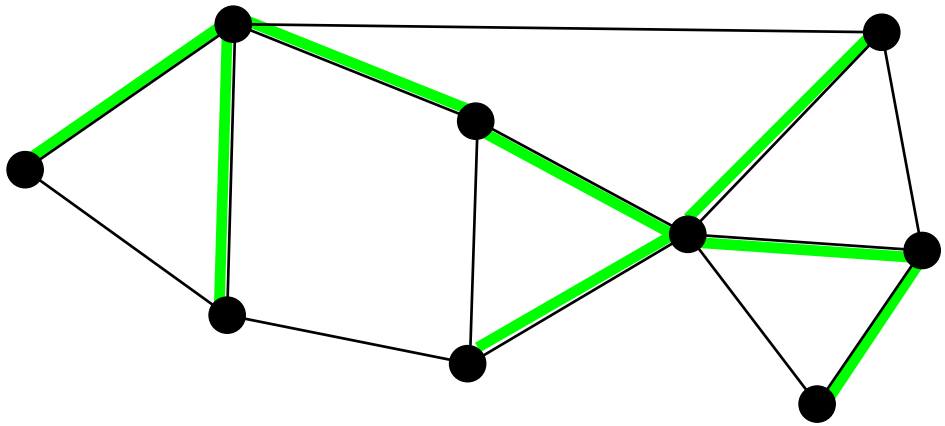
T arbre recouvrant de G

Remarques



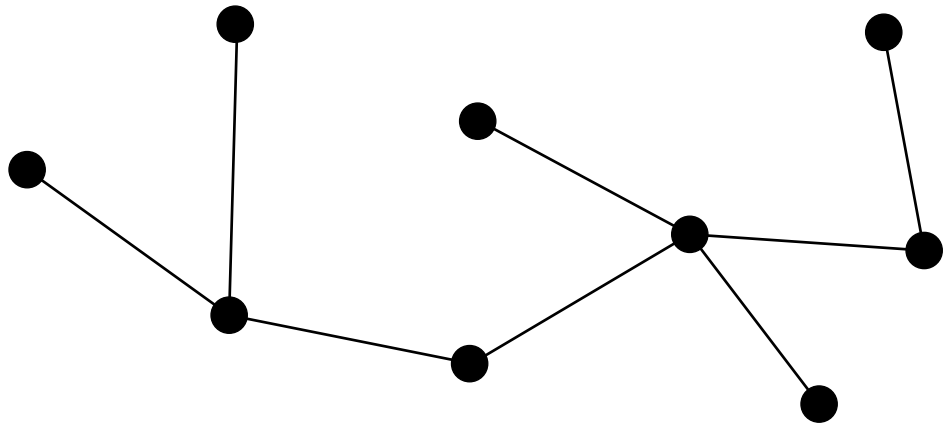
1. Un graphe peut avoir plusieurs arbres recouvrants.
2. Un arbre n'a qu'un seul arbre recouvrant, lui-même.
3. Un graphe non connexe n'a aucun arbre recouvrant.
   (Autrement dit, un graphe qui a un arbre recouvrant est forcément connexe.)
4. Un graphe connexe a forcément (au moins) un arbre recouvrant.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

Remarques



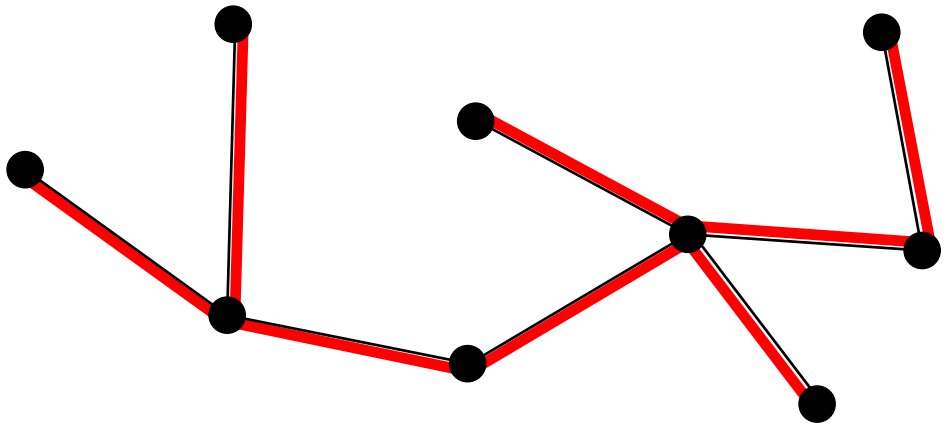
1. Un graphe peut avoir plusieurs arbres recouvrants.
2. Un arbre n'a qu'un seul arbre recouvrant, lui-même.
3. Un graphe non connexe n'a aucun arbre recouvrant.
   (Autrement dit, un graphe qui a un arbre recouvrant est forcément connexe.)
4. Un graphe connexe a forcément (au moins) un arbre recouvrant.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

Remarques



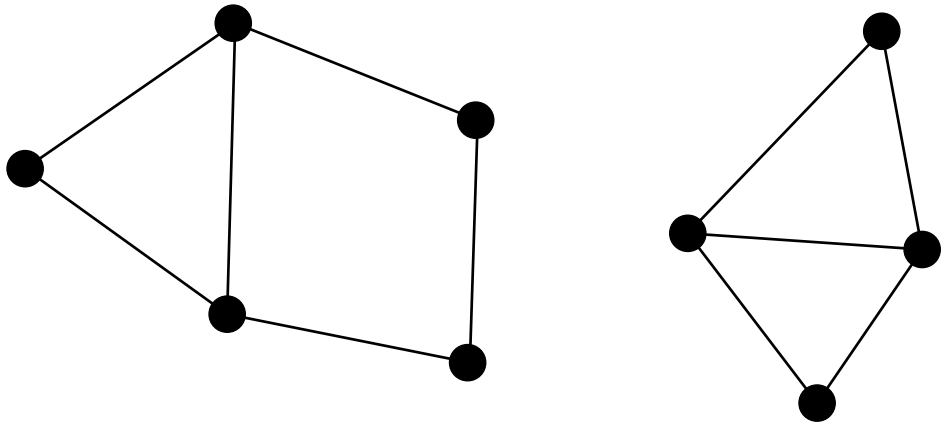
1. Un graphe peut avoir plusieurs arbres recouvrants.
2. Un arbre n'a qu'un seul arbre recouvrant, lui-même.
3. Un graphe non connexe n'a aucun arbre recouvrant.
   (Autrement dit, un graphe qui a un arbre recouvrant est forcément connexe.)
4. Un graphe connexe a forcément (au moins) un arbre recouvrant.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

Remarques



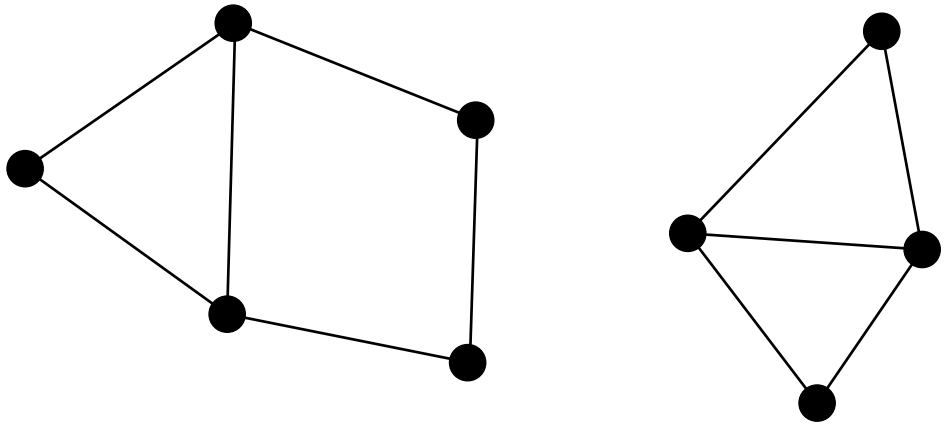
1. Un graphe peut avoir plusieurs arbres recouvrants.
2. Un arbre n'a qu'un seul arbre recouvrant, lui-même.
3. Un graphe non connexe n'a aucun arbre recouvrant.
   (Autrement dit, un graphe qui a un arbre recouvrant est forcément connexe.)
4. Un graphe connexe a forcément (au moins) un arbre recouvrant.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

Remarques



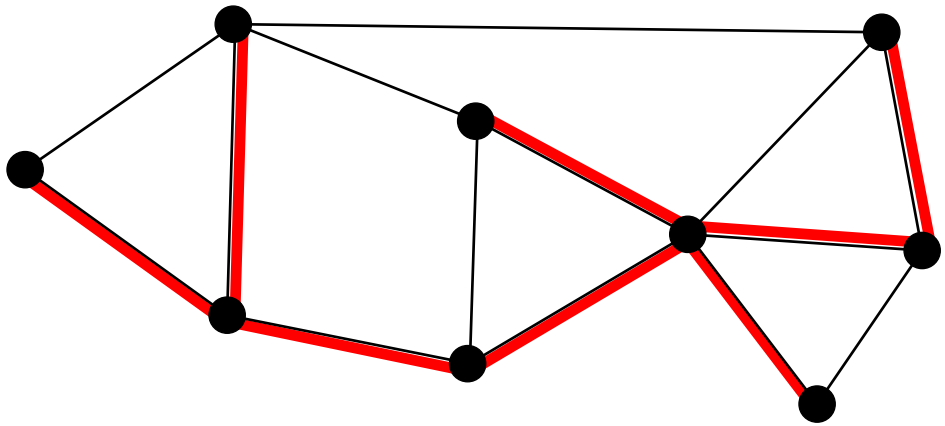
1. Un graphe peut avoir plusieurs arbres recouvrants.
2. Un arbre n'a qu'un seul arbre recouvrant, lui-même.
3. Un graphe non connexe n'a aucun arbre recouvrant.
   (Autrement dit, un graphe qui a un arbre recouvrant est forcément connexe.)
4. Un graphe connexe a forcément (au moins) un arbre recouvrant.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

Remarques



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

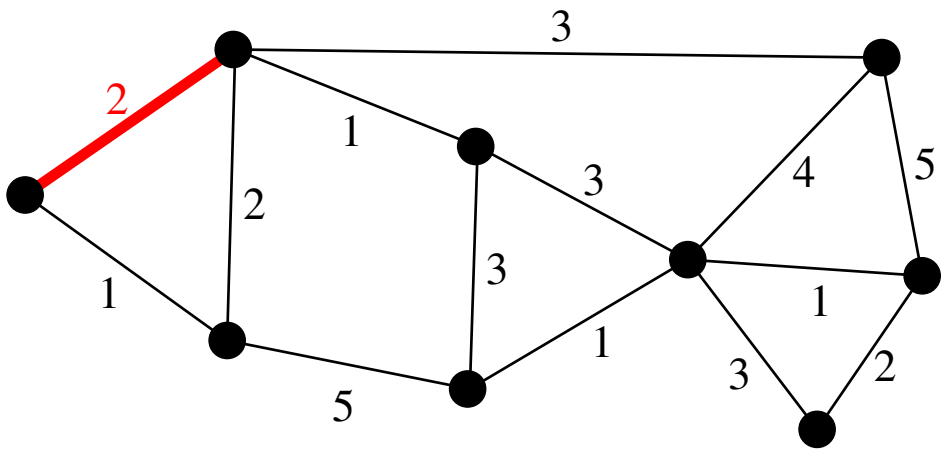
Remarques



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

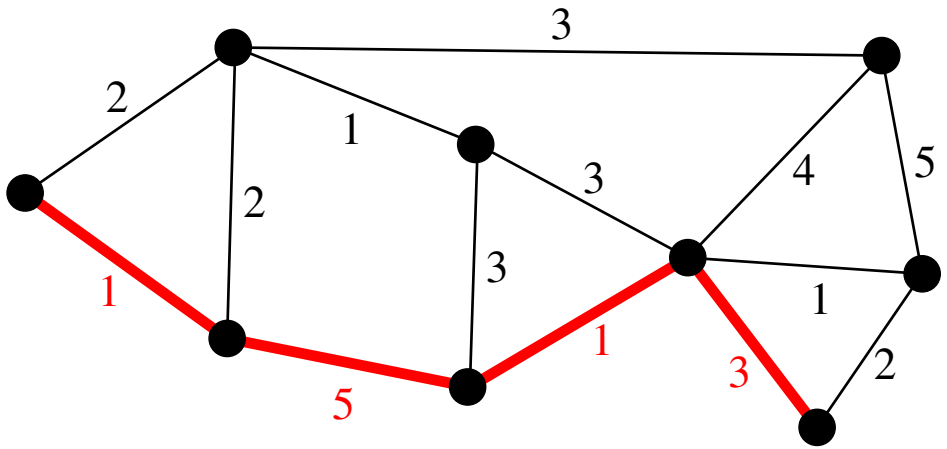
Définition

- Chaque arête a un poids (> 0)
- Le poids d'une chaîne est la somme des poids des arêtes qui la composent.



Définition

- Chaque arête a un poids (> 0)
- Le poids d'une chaîne est la somme des poids des arêtes qui la composent.

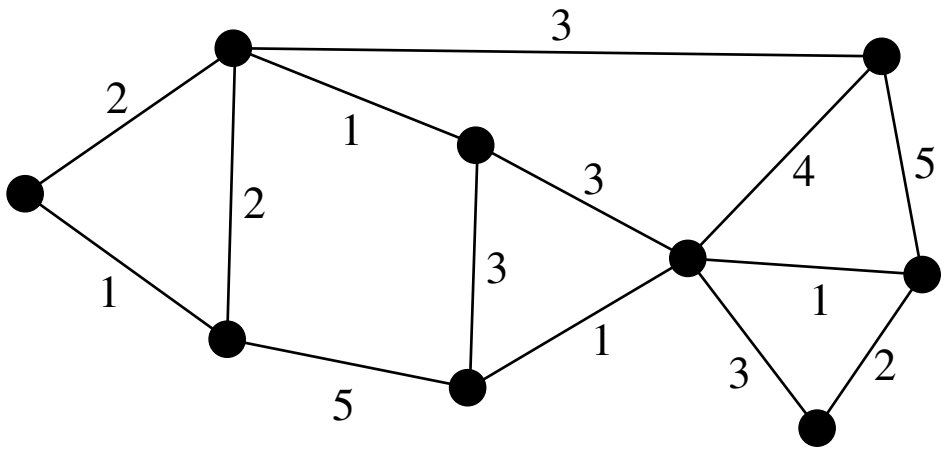


$$1 + 5 + 1 + 3 = 10$$

Question

Étant donné un graphe valué connexe

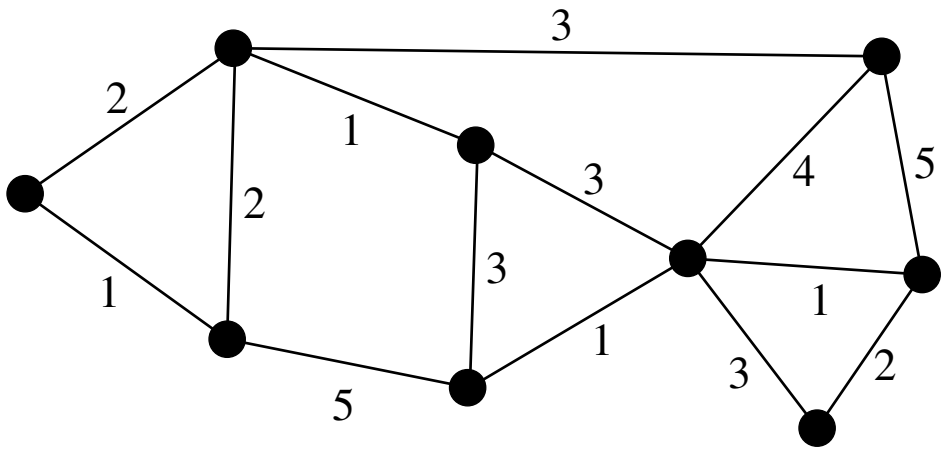
On veut construire un arbre recouvrant de poids total le plus petit possible



Question

Étant donné un graphe valué connexe

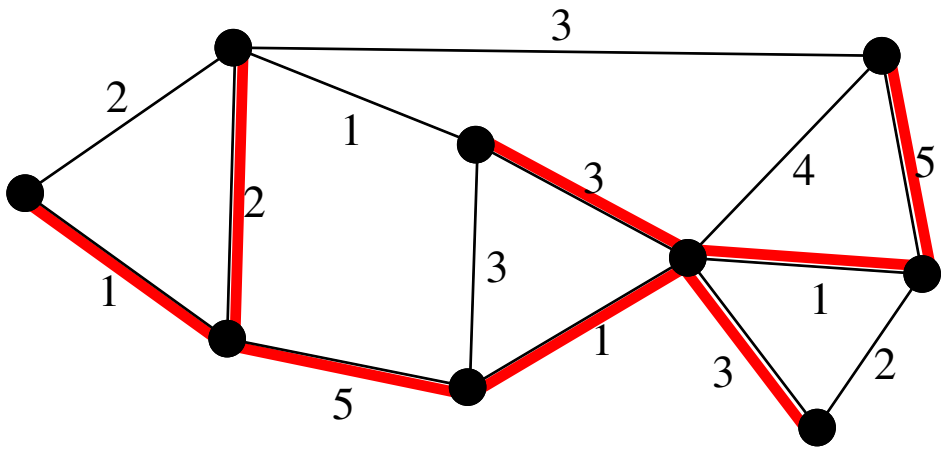
On veut construire un arbre recouvrant de poids total le plus petit possible



Question

Étant donné un graphe valué connexe

On veut construire un arbre recouvrant de poids total le plus petit possible



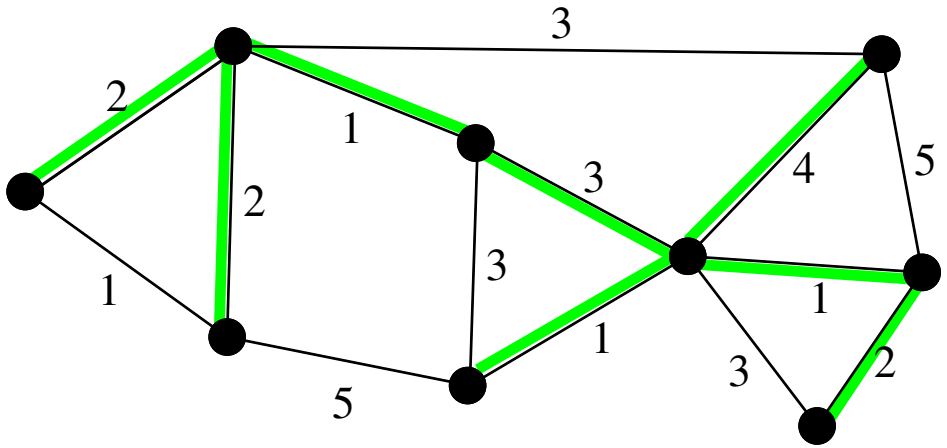
$1 + 2 + 5 + 1 + 3 + 5 + 1 + 3 = 21$

Peut-on faire mieux ?

Question

Étant donné un graphe valué connexe

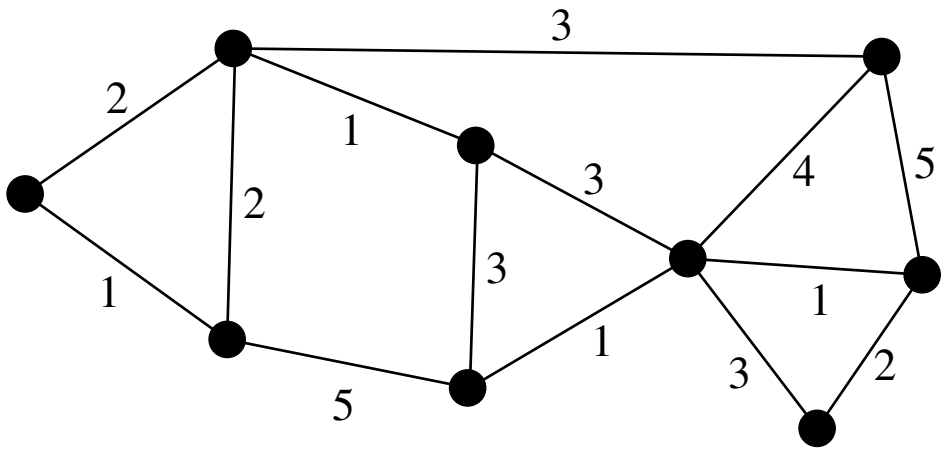
On veut construire un arbre recouvrant de poids total le plus petit possible



$2 + 2 + 1 + 3 + 1 + 4 + 1 + 2 = 16$

Peut-on faire mieux ?

Idée générale

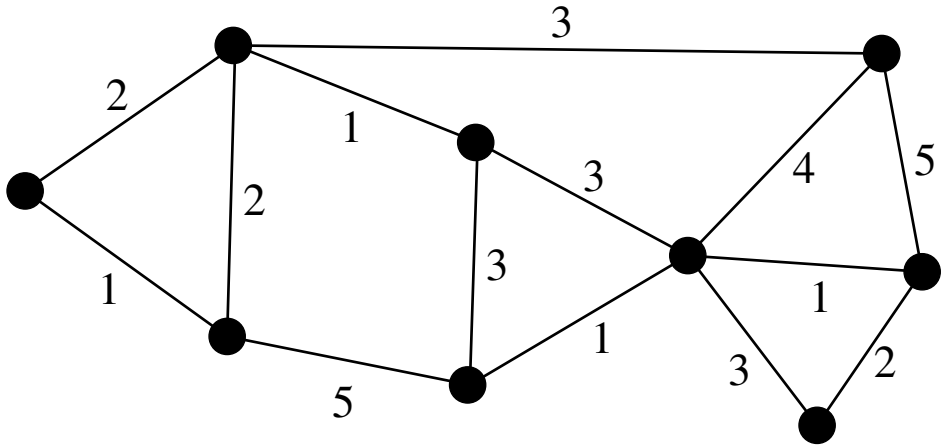


- On part d'une forêt d'arbres constitués de chacun des sommets isolés du graphe.
- À chaque itération, on ajoute à cette forêt l'arête de poids le plus faible ne créant pas de cycle avec les arêtes déjà choisies.
- On stoppe quand on a examiné toutes les arêtes.

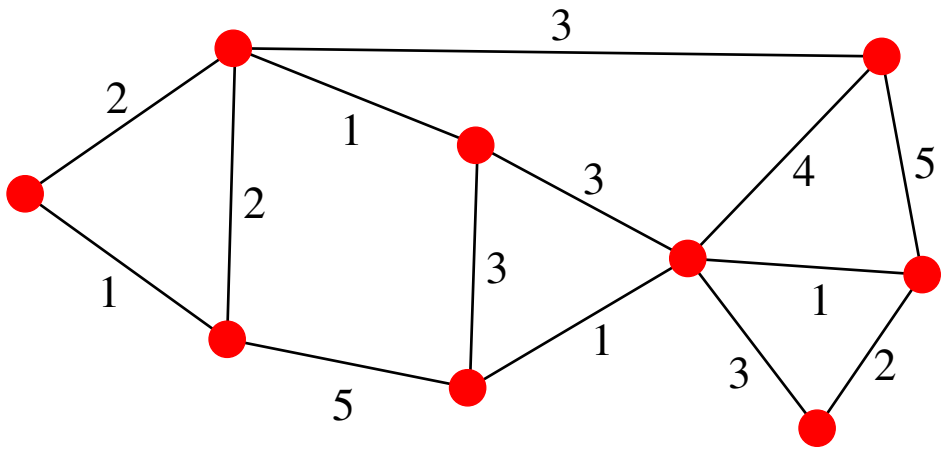
Méthode

- Initialiser T avec
  - sommets : tous les sommets de G
  - arêtes : aucune
- 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 quand il n'y a plus d'arêtes
- Retourner T

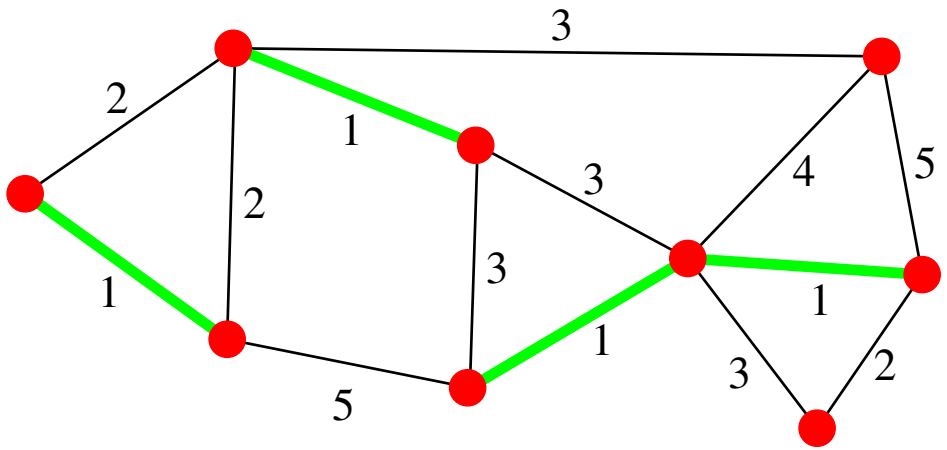
Exemple



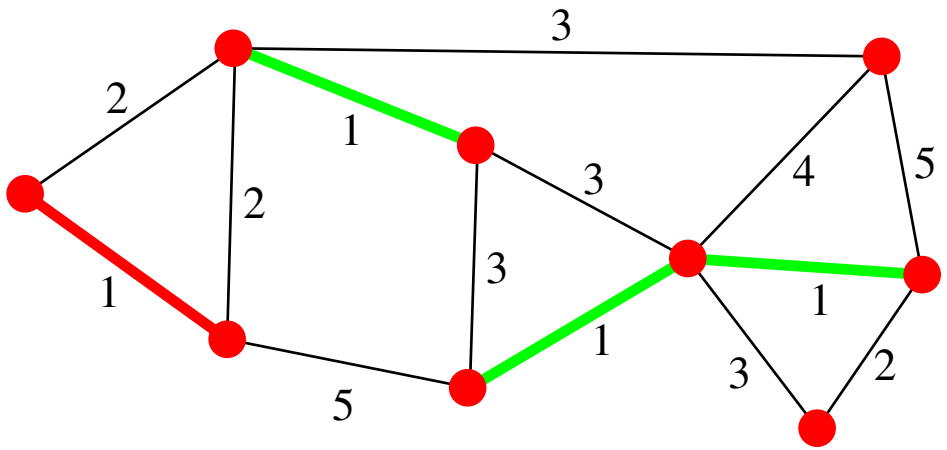
Exemple



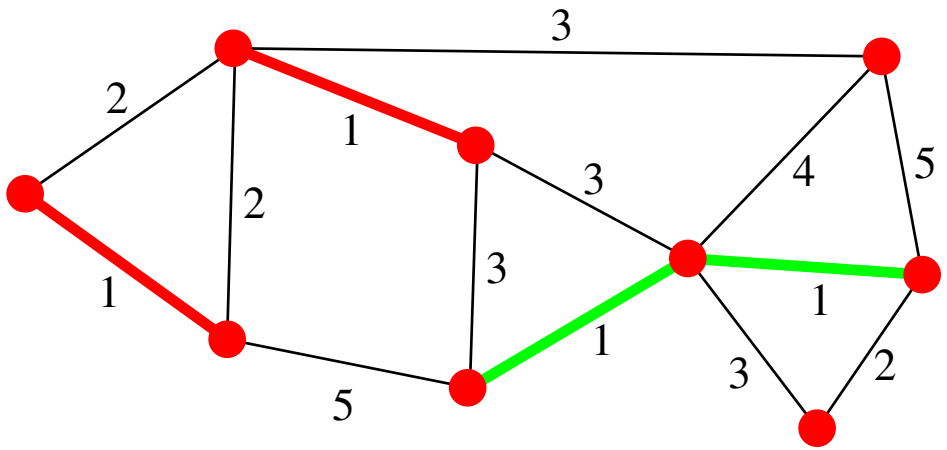
Exemple



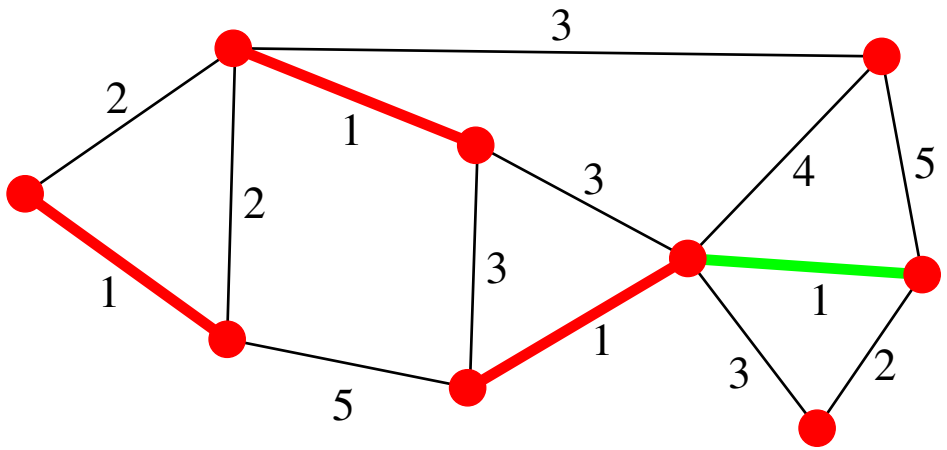
Exemple



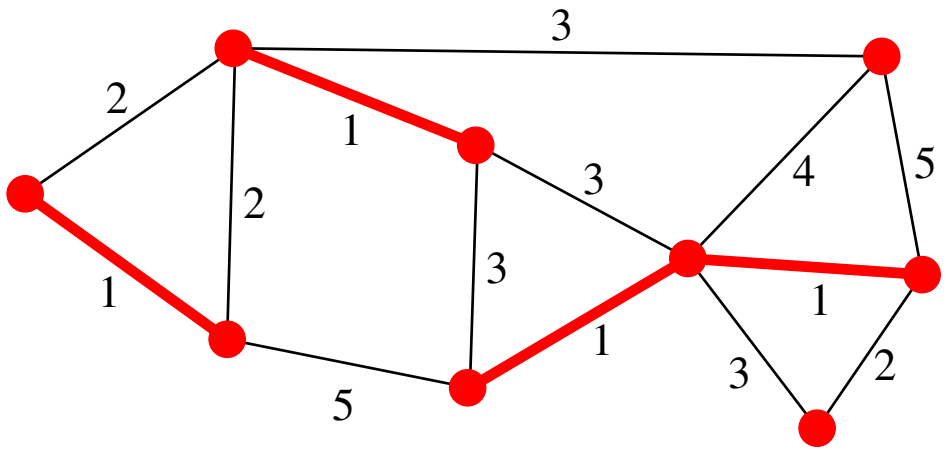
Exemple



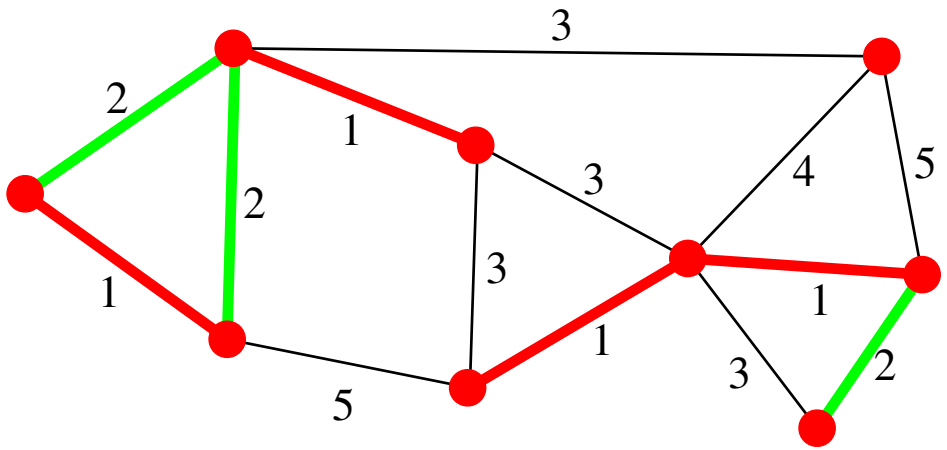
Exemple



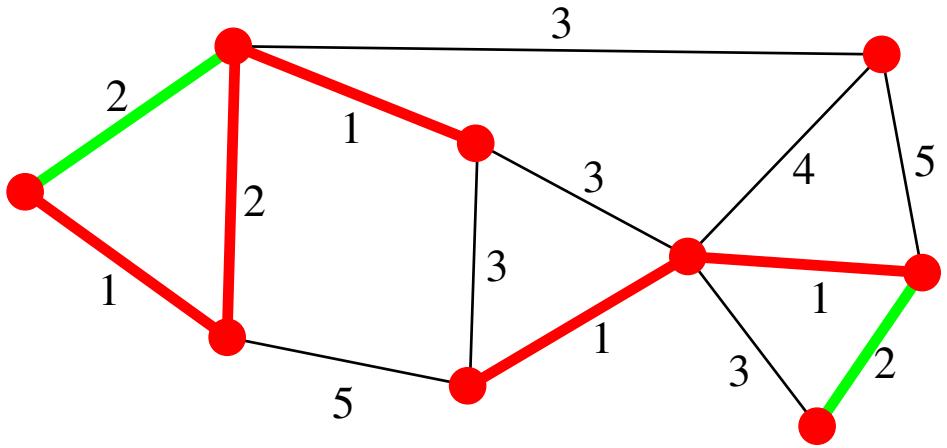
Exemple



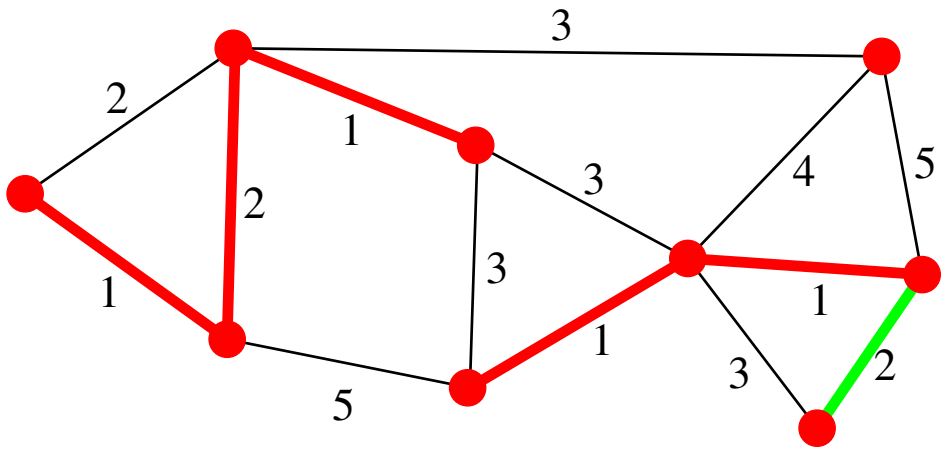
Exemple



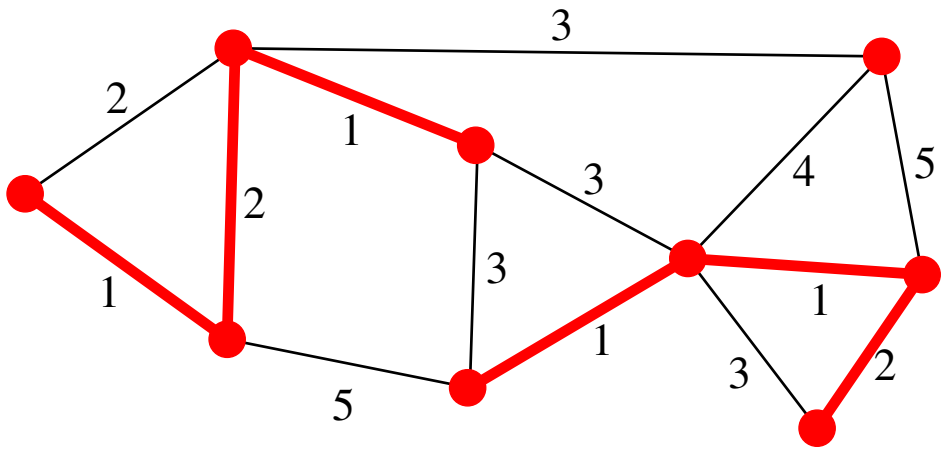
Exemple



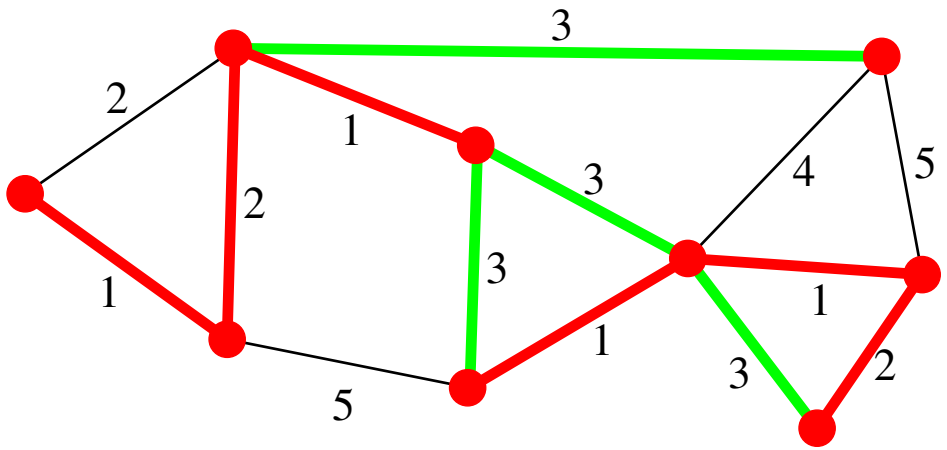
Exemple



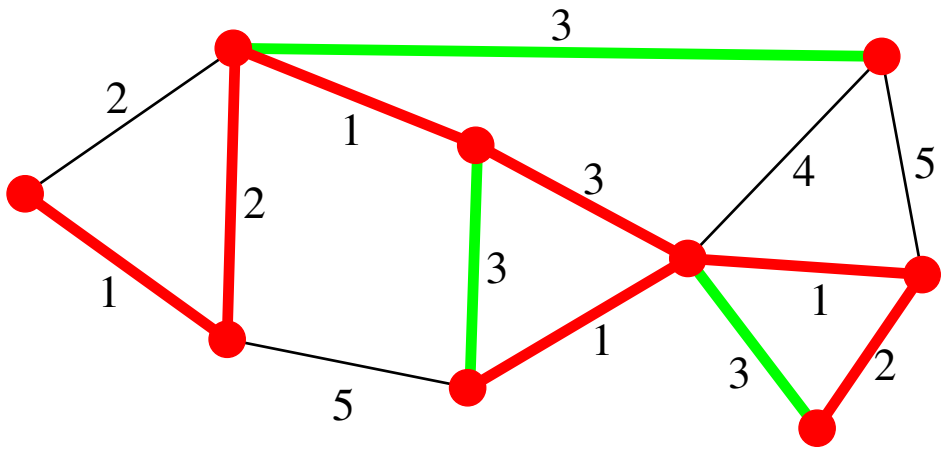
Exemple



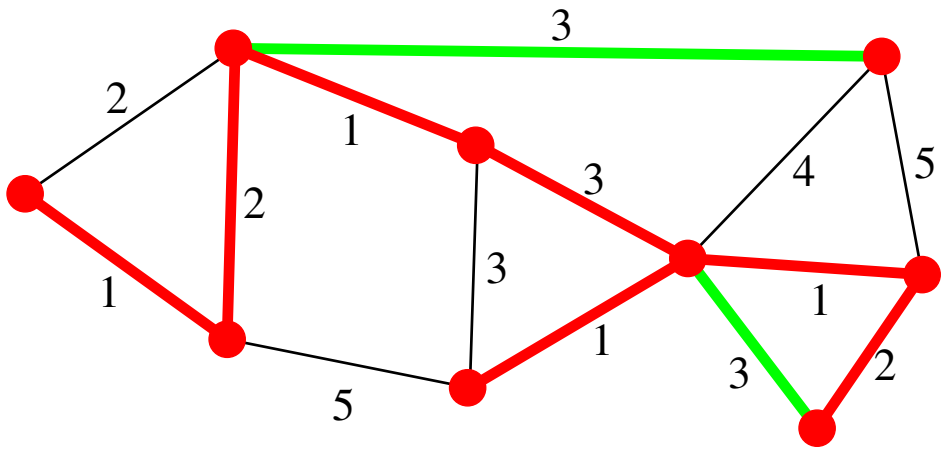
Exemple



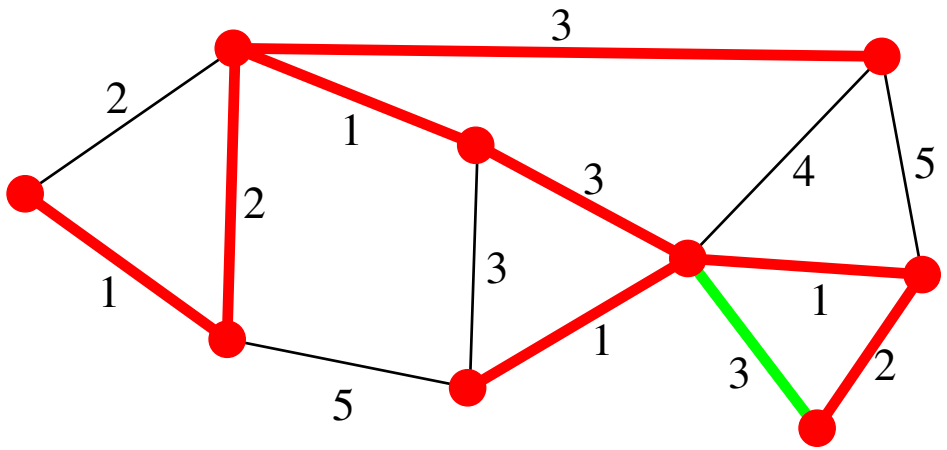
Exemple



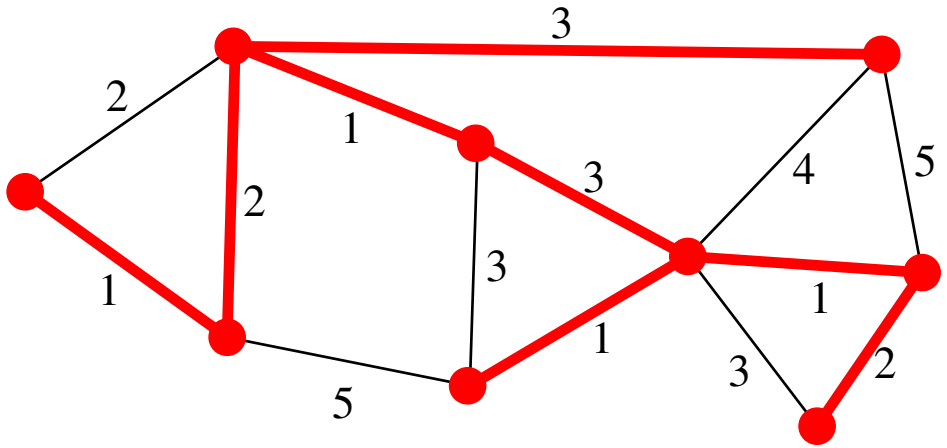
Exemple



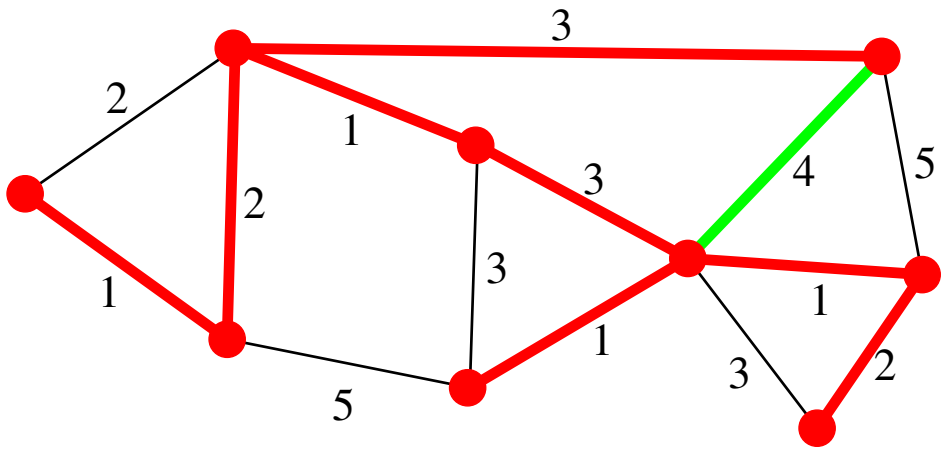
Exemple



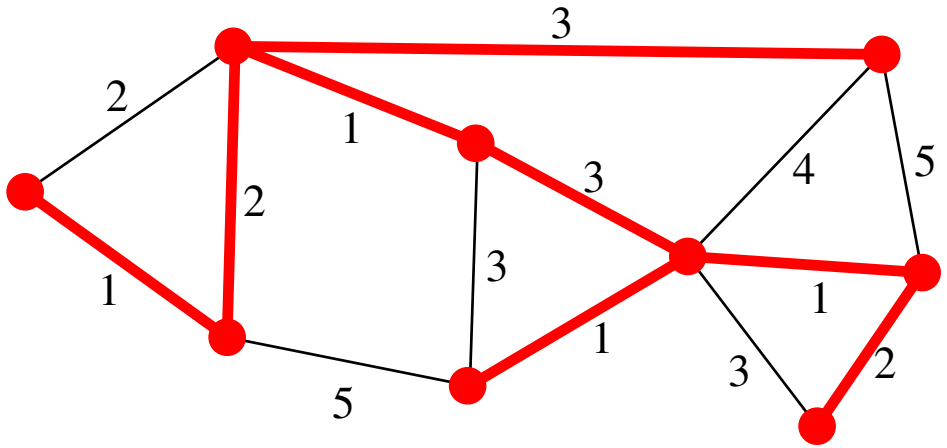
Exemple



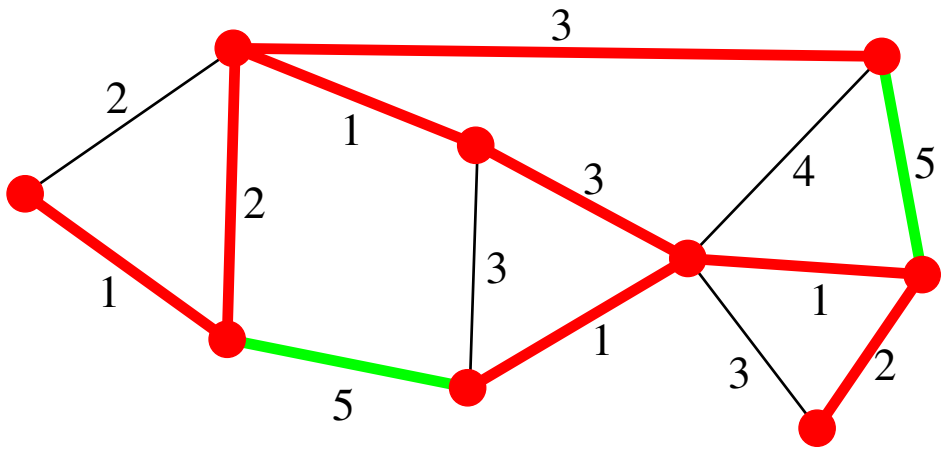
Exemple



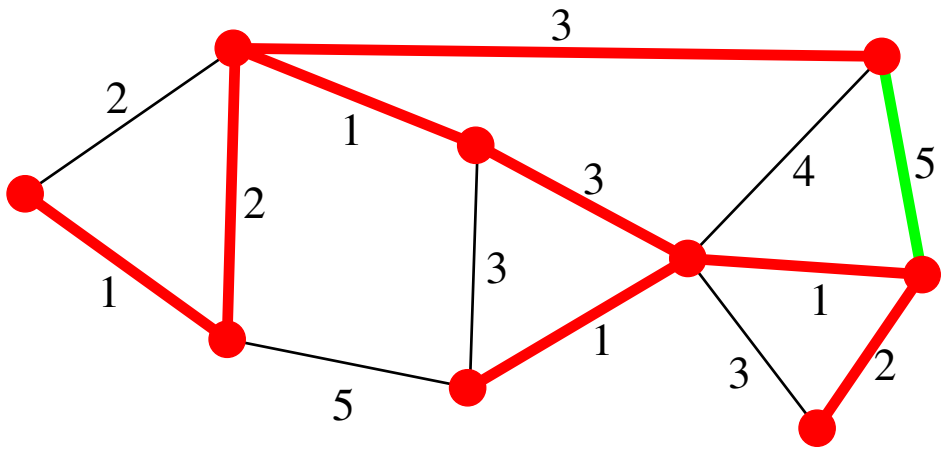
Exemple



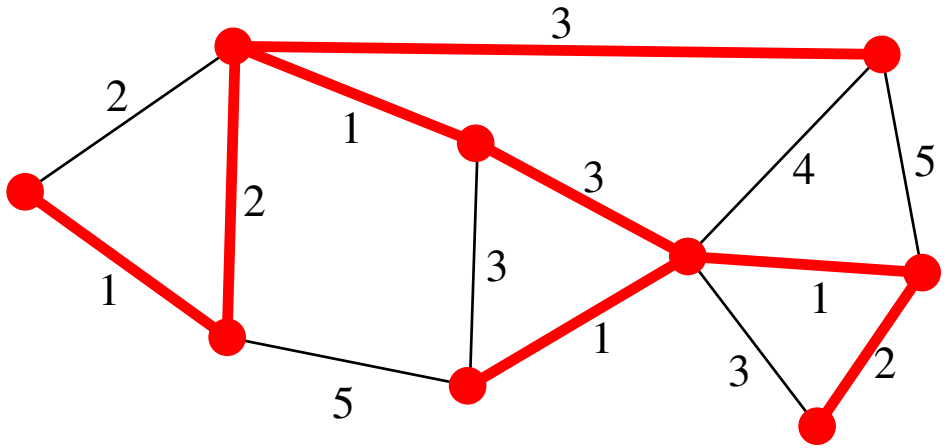
Exemple



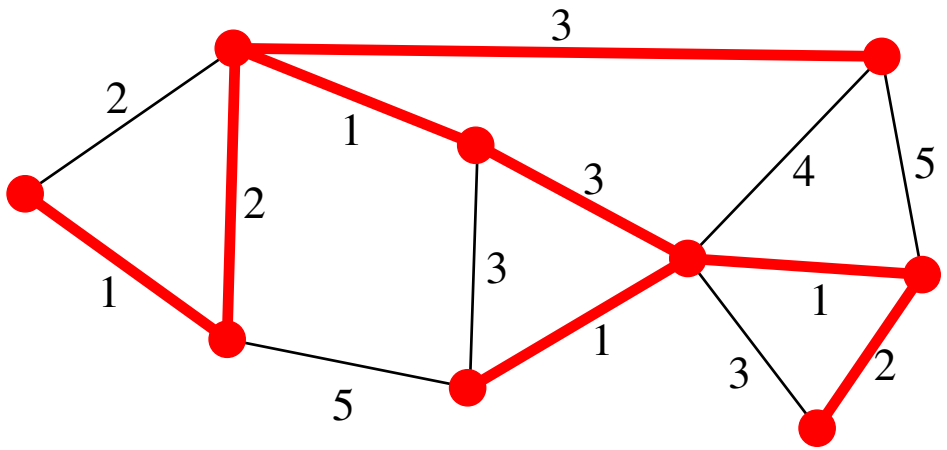
Exemple



Exemple



Exemple



$1 + 1 + 1 + 1 + 2 + 2 + 3 + 3 = 14$

Peut-on faire mieux ?

Rappel de l'algorithme de Kruskal

- Initialiser T avec
  - sommets : tous les sommets de G
  - arêtes : aucune
- 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 quand il n'y a plus d'arêtes
- Retourner T

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

Il n'y a qu'un nombre fini d'arêtes, donc l'algorithme finit par s'arrêter

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

Propriétés du graphe T retourné par Kruskal :

- Sommets de T = Sommets de G
- Arêtes de $T \subseteq$ Arêtes de G
- T 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.
- Reste à vérifier que T est connexe

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

- Supposons que T ne soit pas connexe, et notons T_1 et T_2 deux composantes connexes de T.
- Comme G est connexe, il existe des arêtes de G entre T_1 et T_2. Notons e la première de ces arêtes dans la liste.
- Lorsque l'algorithme a visité e, il aurait dû l'ajouter à T, puisqu'elle permettait de connecter deux composantes connexes de T (a priori plus petites que T_1 et T_2)
- Contradiction : T est bien connexe

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

Supposons que ce ne soit pas le cas.

- Soit T' un arbre recouvrant de poids minimal de G, qui diffère le moins possible de T.
- Soit e la première arête de T qui n'est pas une arête de T'.
- On ajoute e à T', et on crée un cycle.
- Ce cycle ne contient pas que des arêtes de T (car T n'a pas de cycle).

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

Soit e' une arête du cycle de $T' \cup \{e\}$ qui n'est pas une arête de T.

- On note $T'' = T' \cup \{e\} - \{e'\}$.
- T'' est un arbre recouvrant de G.
- On a $Poids(T'') \geq Poids(T')$ (car T' est de poids minimal) donc $Poids(e) \geq Poids(e')$.

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

Avant l'étape d'ajout de e, T , T' et T'' sont identiques.

- Or e est choisie de poids minimal en construisant T.
- Comme on n'a pas choisi e' à la place de e, c'est que en fait $Poids(e) = Poids(e')$.
- Donc on a aussi $Poids(T'') = Poids(T')$, càd T'' est de poids minimum.

Pourquoi l'algorithme de Kruskal construit-il un arbre recouvrant de poids minimal ?

- Kruskal s'arrête toujours
- Kruskal construit un arbre recouvrant
- l'arbre recouvrant construit par Kruskal est de poids minimal

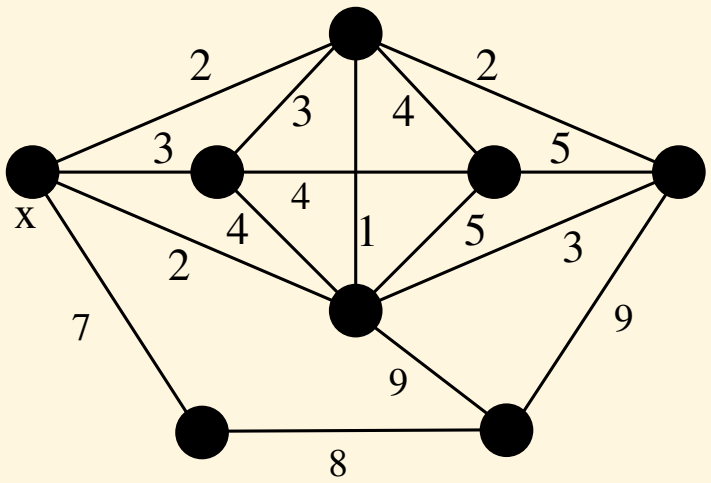
Mais T'' a plus d'arêtes communes avec T que T', ce qui est contraire à l'hypothèse de départ : "T' est un arbre recouvrant de poids minimal de G, différant le moins possible de T".

Conclusion : T' n'existe pas et par conséquent T est un arbre recouvrant de poids minimal de G.

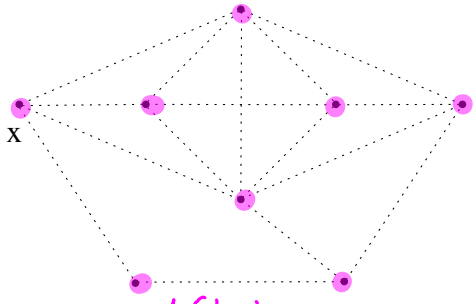
Mise en oeuvre

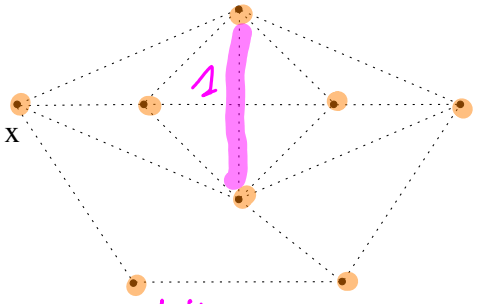
Exercice

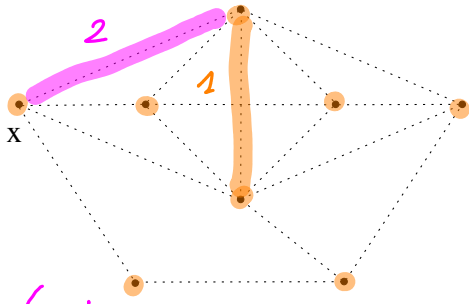
Appliquer l'algorithme de KRUSKAL au graphe G suivant.

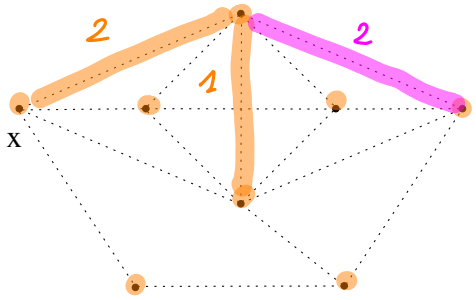


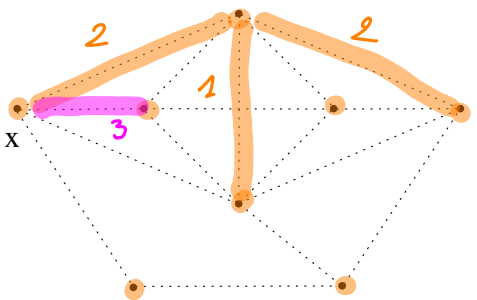
Correction pour Kruskal

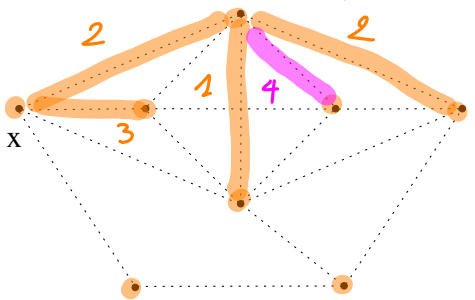


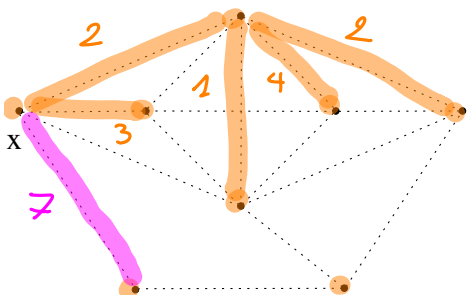


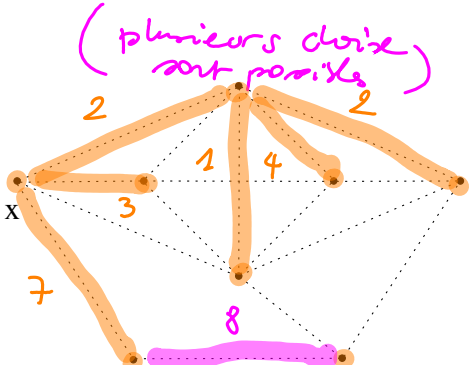










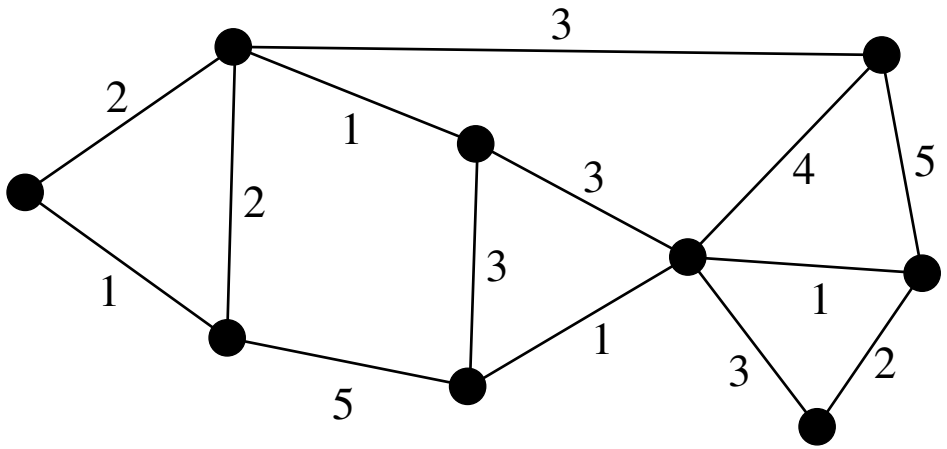


Poids Total :

1+2+2+3+4+7+8

= 27

Idée générale

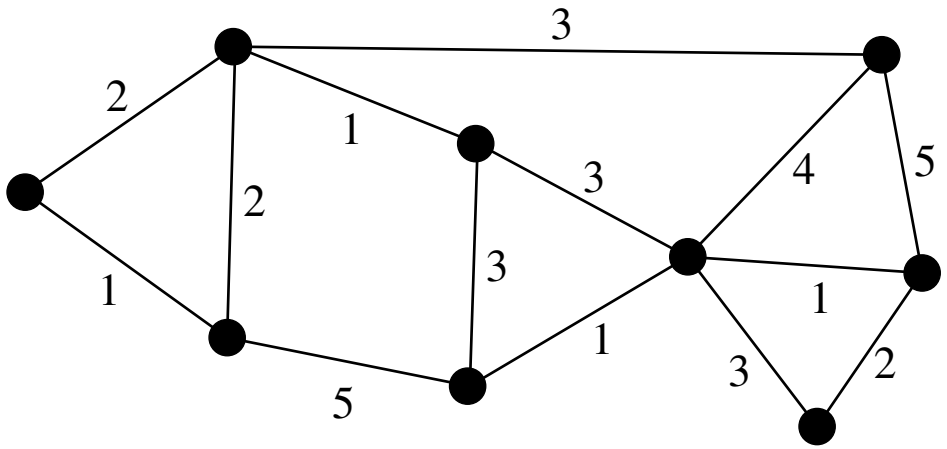


- On part d'un arbre initial réduit à un seul sommet du graphe.
- À chaque itération, on agrandit l'arbre en lui ajoutant le sommet libre accessible de plus petit poids possible.
- On stoppe quand l'arbre est recouvrant.

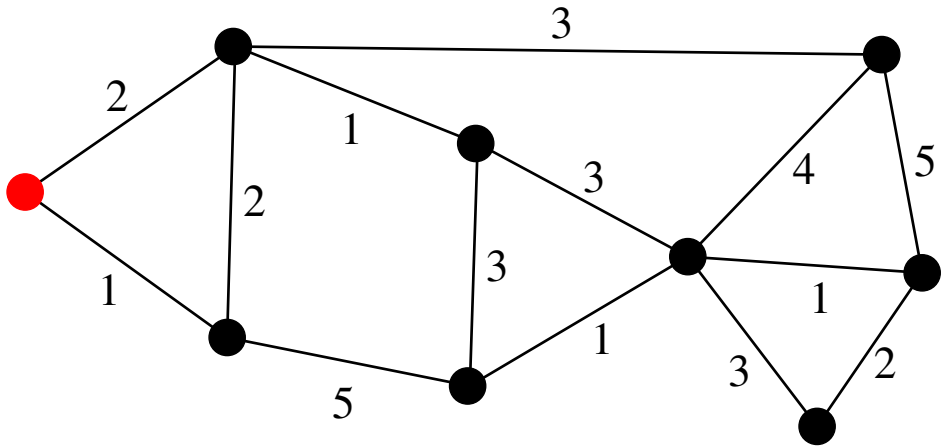
Méthode

- 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

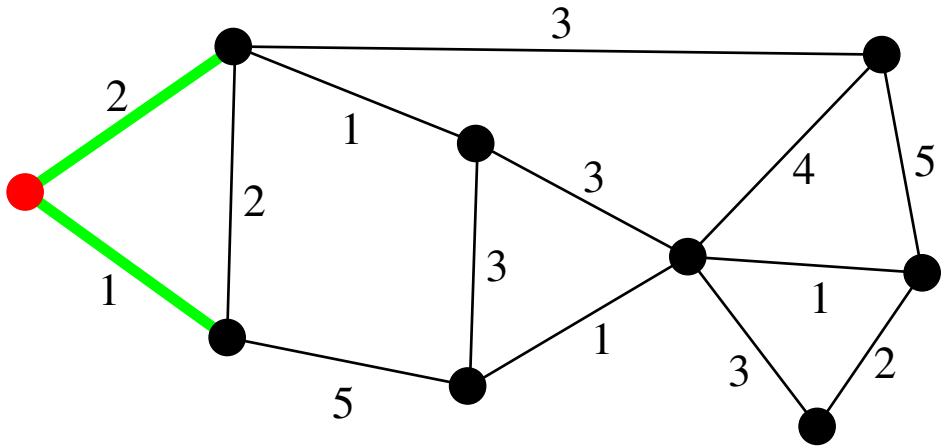
Exemple



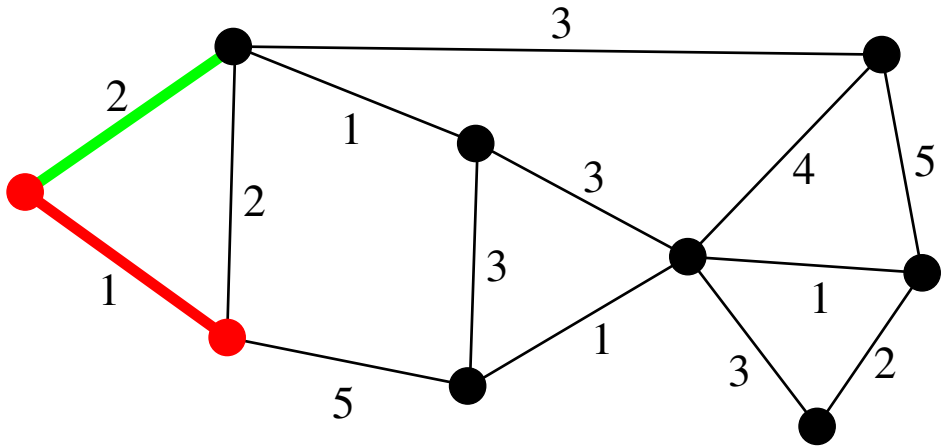
Exemple



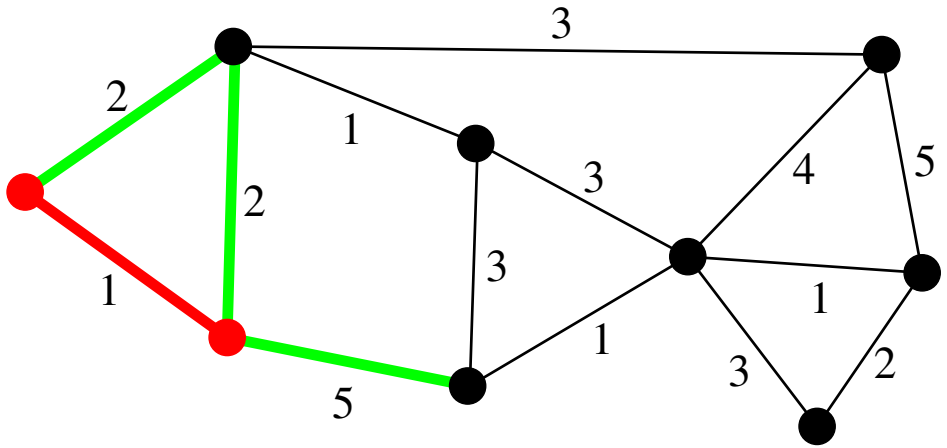
Exemple



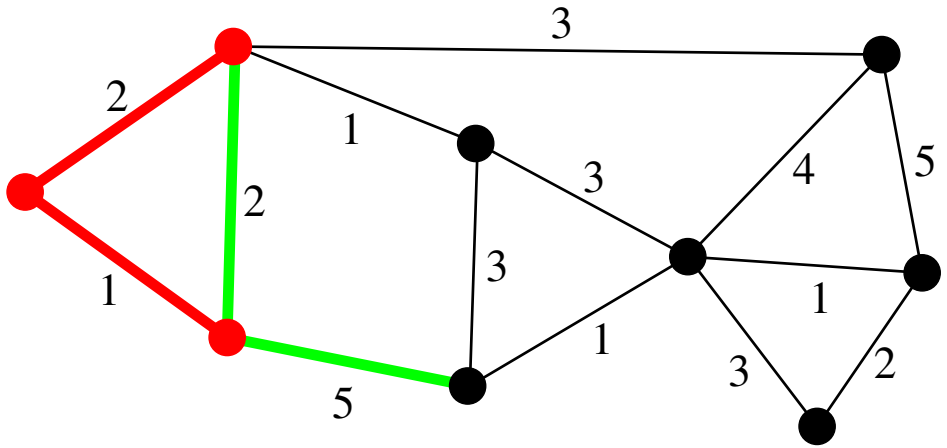
Exemple



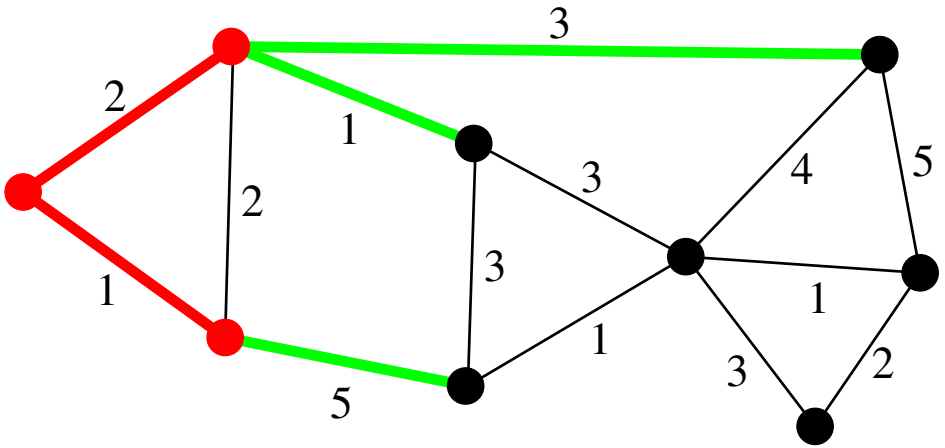
Exemple



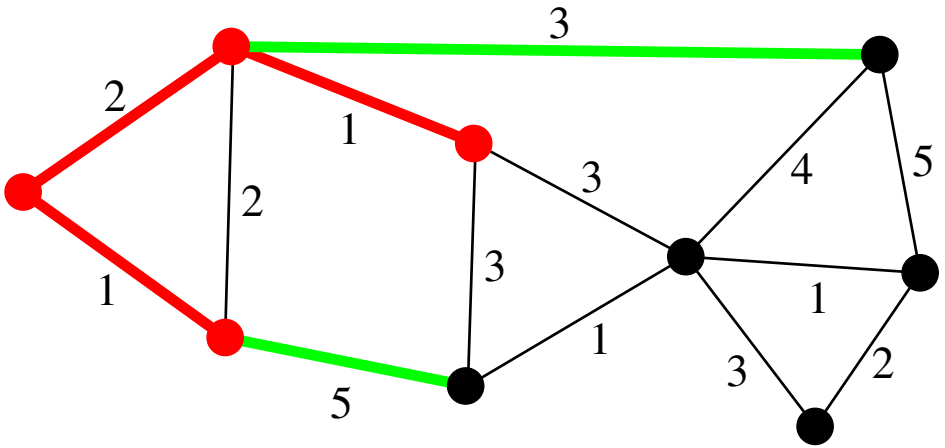
Exemple



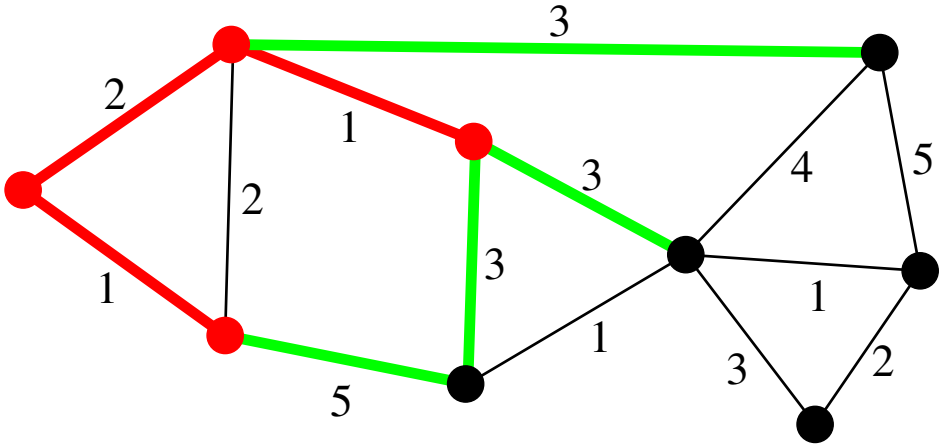
Exemple



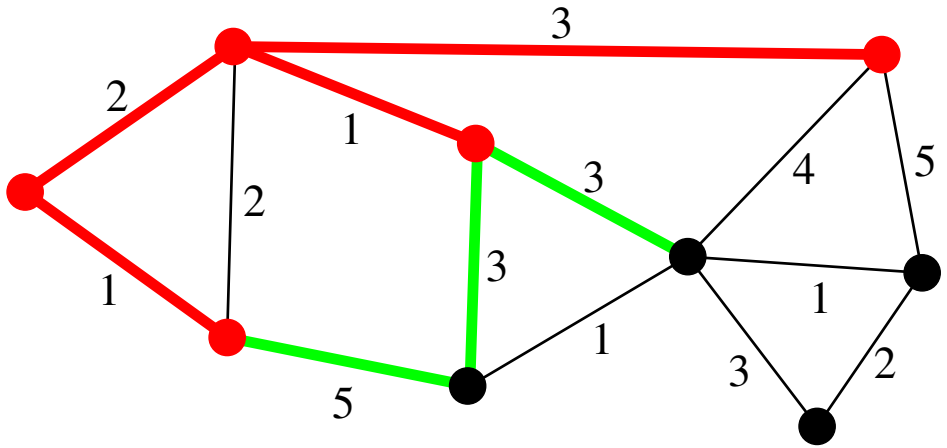
Exemple



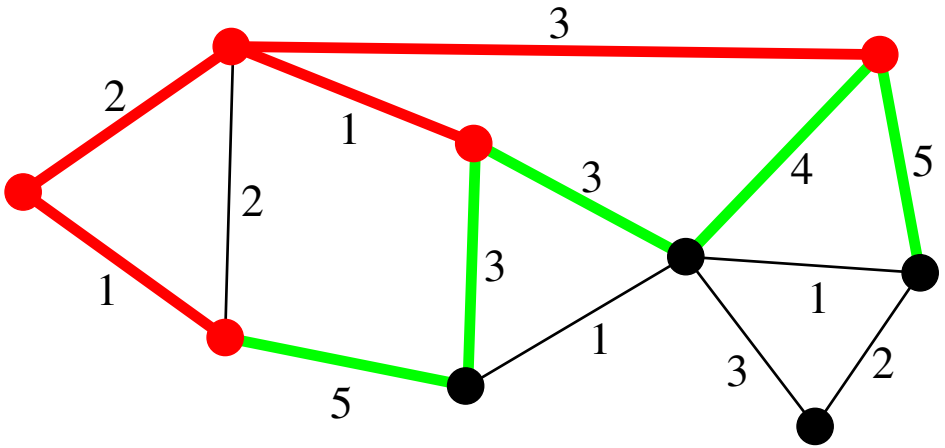
Exemple



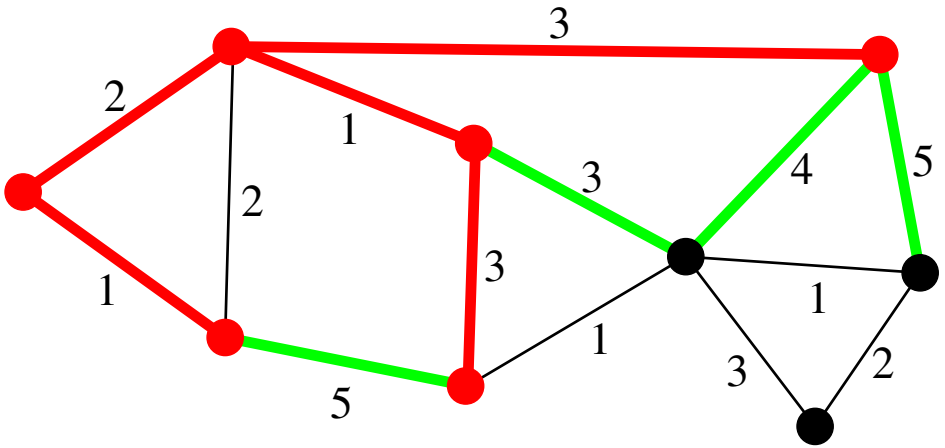
Exemple



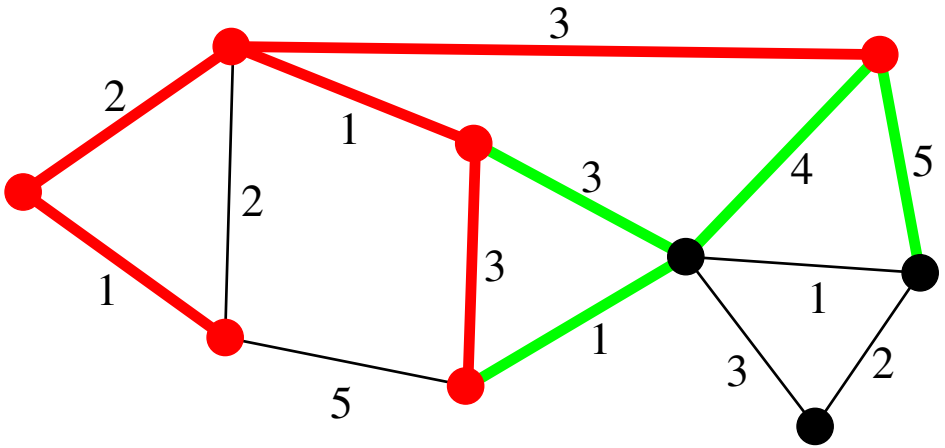
Exemple



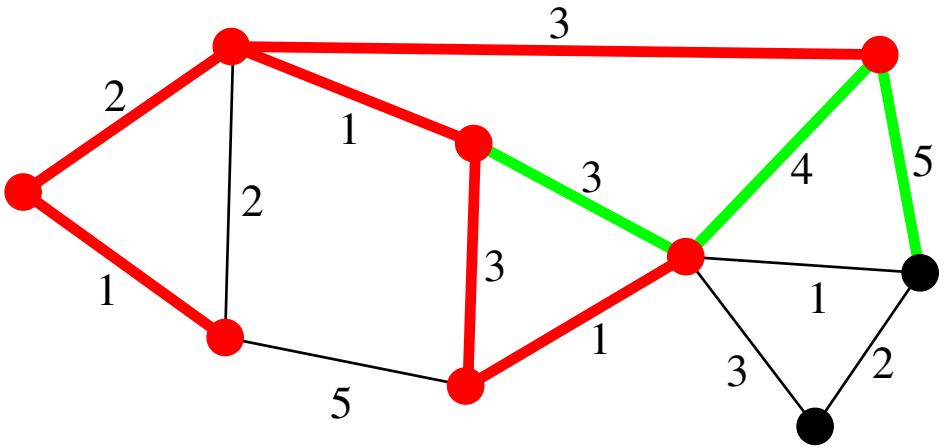
Exemple



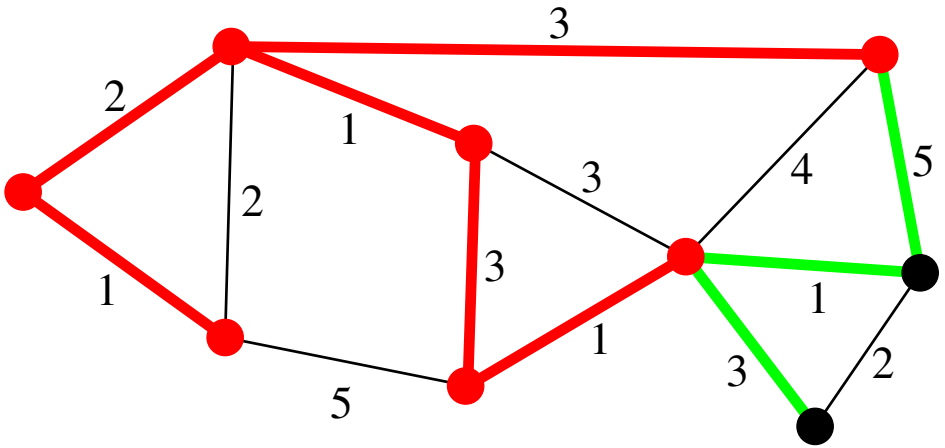
Exemple



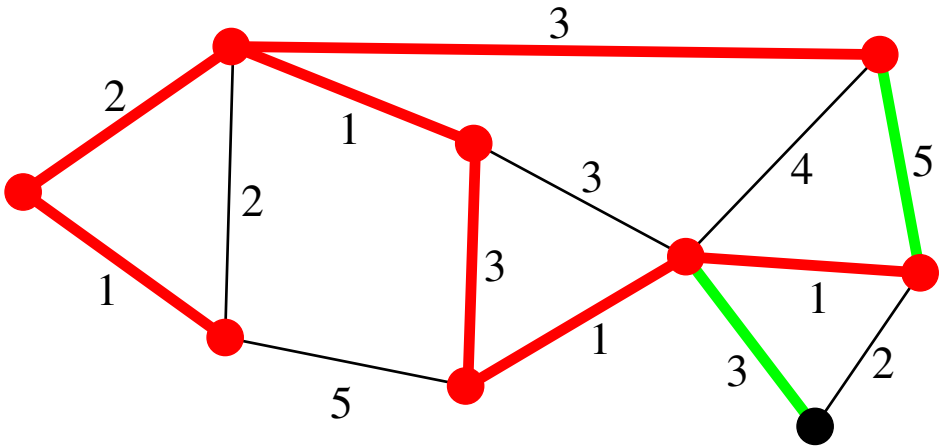
Exemple



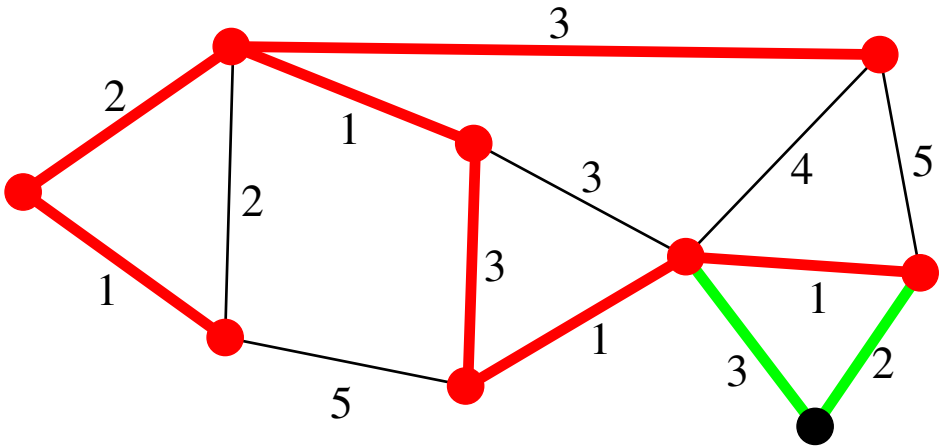
Exemple



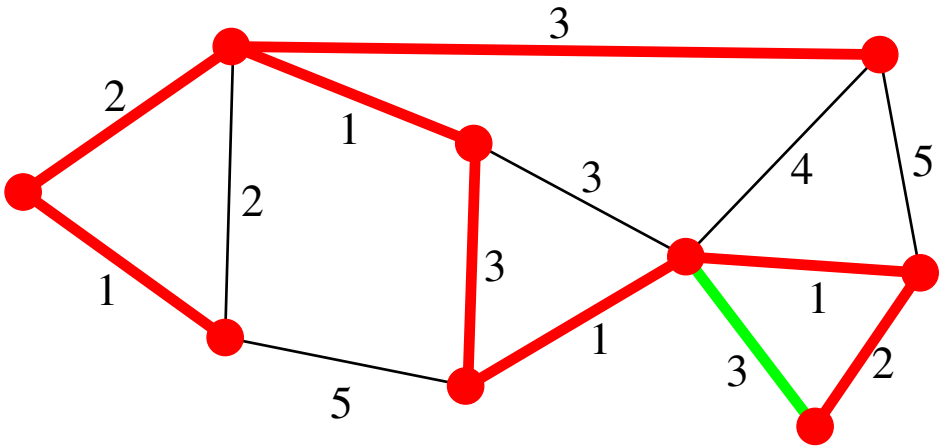
Exemple



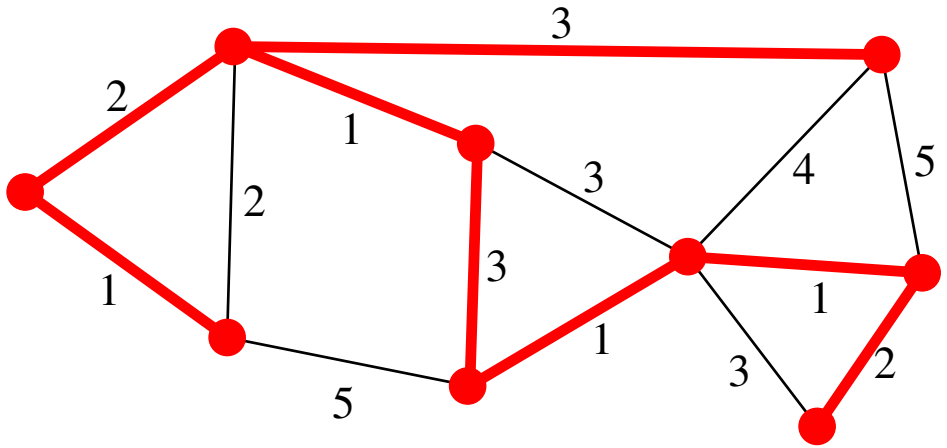
Exemple



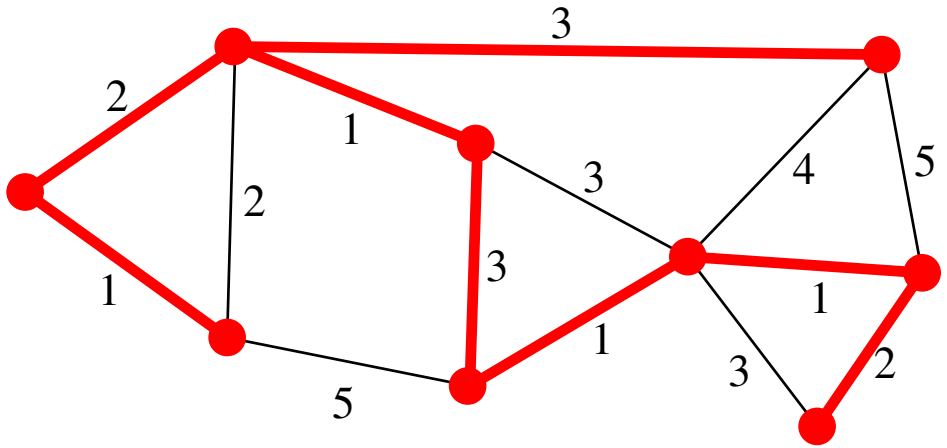
Exemple



Exemple



Exemple



$1 + 2 + 1 + 3 + 3 + 1 + 1 + 2 = 14$

On obtient un arbre recouvrant différent mais de même poids

Rappel de l'algorithme de Prim

- Initialiser T avec
  - sommets : un sommet de G qu'on choisit
  - arêtes : aucune
- 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

Pourquoi l'algorithme de Prim construit-il un arbre recouvrant de poids minimal ?

1. L'algorithme est bien défini car si la condition « *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).
2. L'algorithme termine car on ajoute à chaque passage un sommet et une arête.
3. Enfin, au départ, T est réduit à un sommet, c'est donc un arbre et dans la boucle, on passe d'un arbre à un autre par ajout d'une feuille. Finalement, quand on sort de la boucle on retourne un arbre.
   On obtient donc à l'issue de l'algorithme un arbre recouvrant.

Pourquoi l'algorithme de Prim construit-il un arbre recouvrant de poids minimal ?

Il reste à vérifier que cet arbre est de poids minimal.

- Sinon, soit T' un arbre de poids minimum recouvrant G et possédant en commun avec T un nombre maximum d'arêtes.
- On note $\{e_1, \ldots, e_{n-1}\}$) les arêtes de T (dans l'ordre obtenu par l'algorithme de Prim).
- Comme T et T' ne sont pas identiques, T a au moins une arête n'appartenant pas à T'.
- Soit e_i $(1 \leq i \leq n-1)$, la première arête de T qui n'appartient pas à T'.
- Notons S l'ensemble des sommets de G donné par les arêtes $e_1, \ldots, e_{i-1}$ (si $i = 1$, $S = \{A\}$, en notant A le sommet de départ).

Pourquoi l'algorithme de Prim construit-il un arbre recouvrant de poids minimal ?

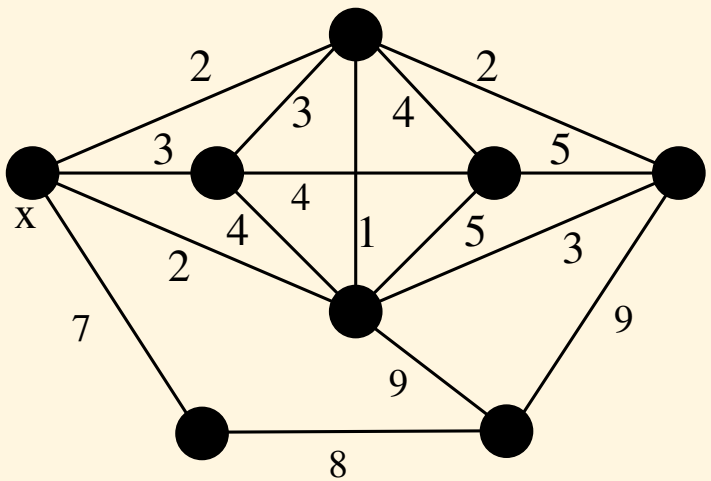
(suite)

- Le graphe $T' + e_i$ a exactement un cycle C.
- Bien sûr, e_i joint un sommet de S et un sommet de $T \setminus S$. En fait C contient une autre arête e_0 joignant un sommet de S et un sommet de $T \setminus S$.
- Alors, $T'' = T' + e_i - e_0$ est un arbre.
- L'ordre des arêtes de T étant choisi par l'algorithme de Prim, le poids de e_i est inférieur ou égal au poids de e_0 et le poids total de T'' est inférieur ou égal au poids de T'.
- Donc, par minimalité du poids de T', les poids de T' et de T'' sont égaux.
- Or, T'' a plus d'arêtes communes avec T que T' : absurde.

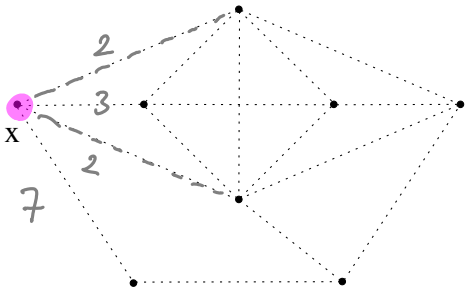
Mise en oeuvre

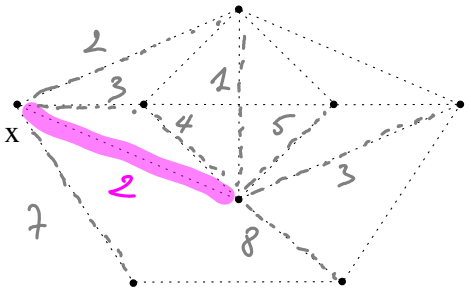
Exercice

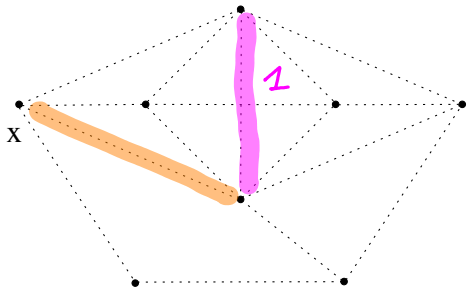
Appliquer l'algorithme de PRIM au graphe G suivant en commençant par le sommet x.



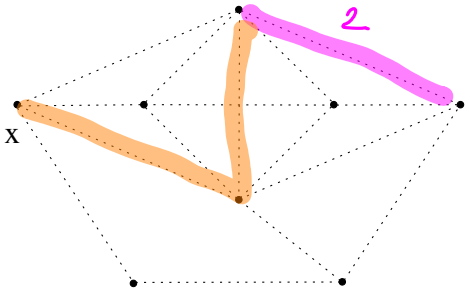
Correction pour Prim

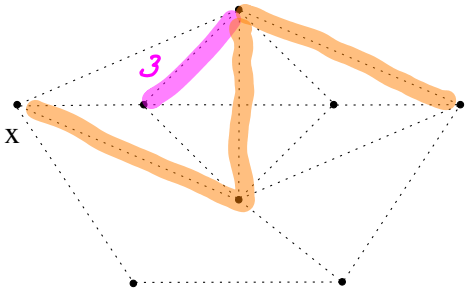


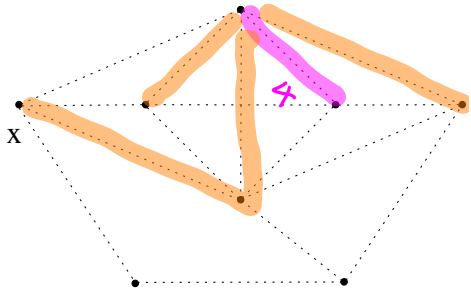




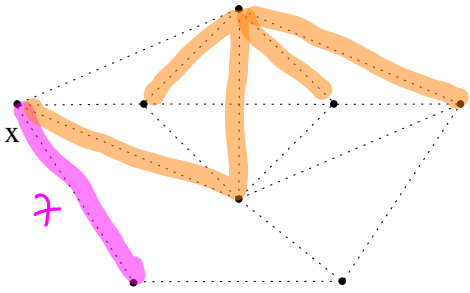
Prim : On choisit une arête de poids plus faible pour visiter un autre sommet.

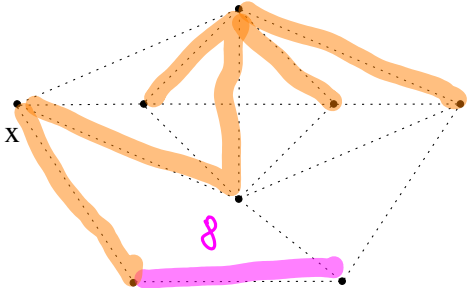






etc.





Poids total

$1+2+2+3+4+7+8$

$= 27$

FIN