Algorithmes gloutons

Quentin Fortier

September 21, 2021

Algorithmes gloutons

La stratégie d'un **algorithme glouton** est d'effectuer à chaque étape l'action qui semble la plus intéressante localement (sans regarder ce qui va se passer plus tard).

Algorithmes gloutons

La stratégie d'un **algorithme glouton** est d'effectuer à chaque étape l'action qui semble la plus intéressante localement (sans regarder ce qui va se passer plus tard).

Exemples :

- Algorithme glouton pour le sac à dos (non optimal)
- Algorithmes de Prim, Kruskal pour trouver un arbre couvrant de poids minimum
- Coloriage de graphe (non optimal a priori)

Algorithmes gloutons

Problème du sac à dos

On considère un sac à dos de capacité 10kg et les objets suivants :

poids (kg)	2	2	2	3	5	5	8
valeur (€)	1	1	1	7	10	10	13

Quelle est la valeur maximum que l'on peut mettre dans le sac ?

Algorithmes gloutons

Problème du sac à dos

On considère un sac à dos de capacité 10kg et les objets suivants :

poids (kg)	2	2	2	3	5	5	8
valeur (€)	1	1	1	7	10	10	13

Quelle est la valeur maximum que l'on peut mettre dans le sac ?

Algorithme glouton n°1 : ajouter les éléments dans l'ordre croissant de poids, tant que c'est possible

Algorithmes gloutons

Problème du sac à dos

On considère un sac à dos de capacité 10kg et les objets suivants :

poids (kg)	2	2	2	3	5	5	8
valeur (€)	1	1	1	7	10	10	13

Quelle est la valeur maximum que l'on peut mettre dans le sac ?

Algorithme glouton n°2 : ajouter les éléments dans l'ordre décroissant de valeur, tant que c'est possible

Algorithmes gloutons

Problème du sac à dos

On considère un sac à dos de capacité 10kg et les objets suivants :

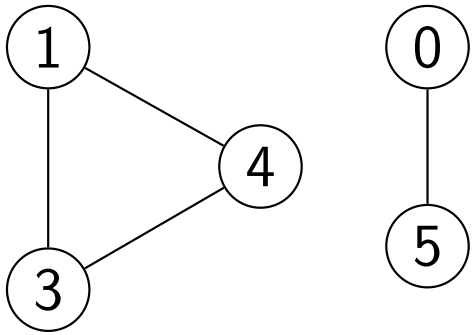
poids (kg)	2	2	2	3	5	5	8
valeur (€)	1	1	1	7	10	10	13
valeur/poids	0.5	0.5	0.5	2.3	2	2	1.6

Quelle est la valeur maximum que l'on peut mettre dans le sac ?

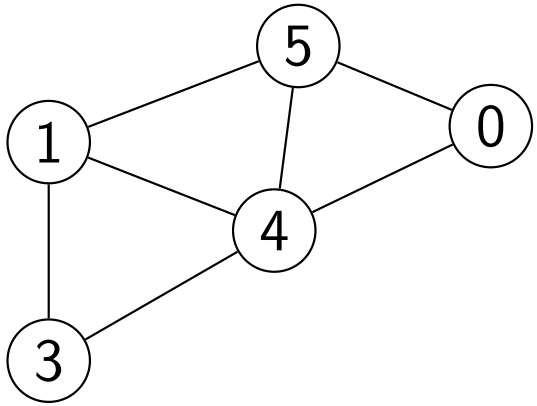
Algorithme glouton n°3 : ajouter les éléments dans l'ordre décroissant de valeur/poids, tant que c'est possible

Connexité

Un graphe non orienté est **connexe** s'il possède un chemin de n'importe quel sommet à n'importe quel autre.



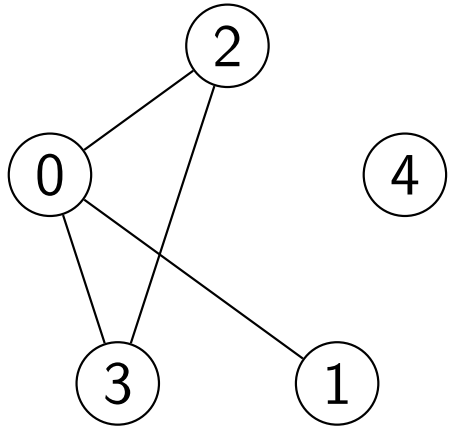
Graphe non connexe



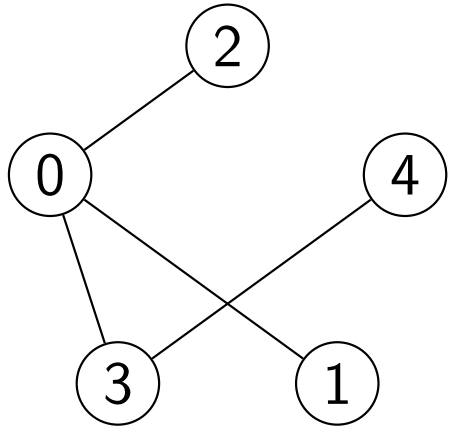
Graphe connexe

Graphe acyclique

Un graphe est **acyclique** (ou: sans cycle) s'il ne contient pas de cycle.



Graphe contenant un cycle



Graphe acyclique

Arbre

Définition

Un graphe est un **arbre** s'il est **connexe** et **sans cycle**

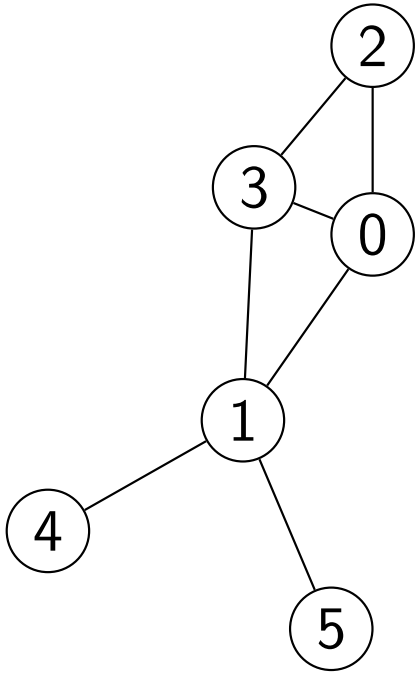
Arbre

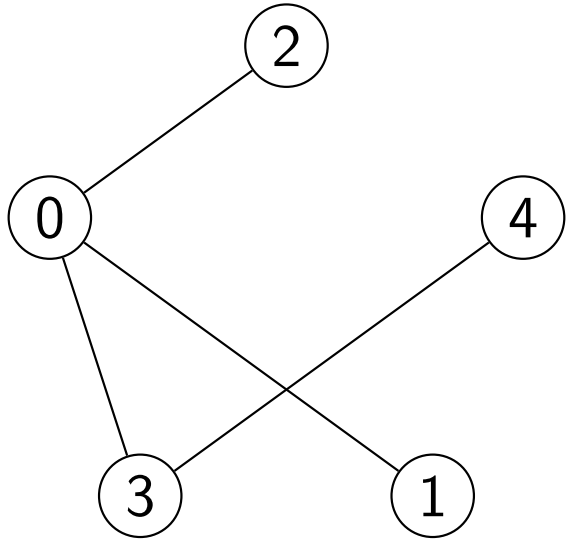
Définition

Un graphe est un **arbre** s'il est **connexe** et **sans cycle**

Question

Les graphes ci-dessous sont-ils des arbres?





Arbre

Théorème

T est un arbre (connexe et sans cycle)

$\Longleftrightarrow$

T est connexe et a $n-1$ arêtes

$\Longleftrightarrow$

T est sans cycle et a $n-1$ arêtes

Arbre couvrant de poids minimal

Arbre couvrant

Soit G un graphe pondéré (chaque arête e possède un poids $w(e)$).
Un arbre couvrant T de G est un ensemble d'arêtes de G qui forme un arbre et qui contient tous les sommets. Son poids $w(T)$ est la somme des poids des arêtes de l'arbre.

Arbre couvrant de poids minimal

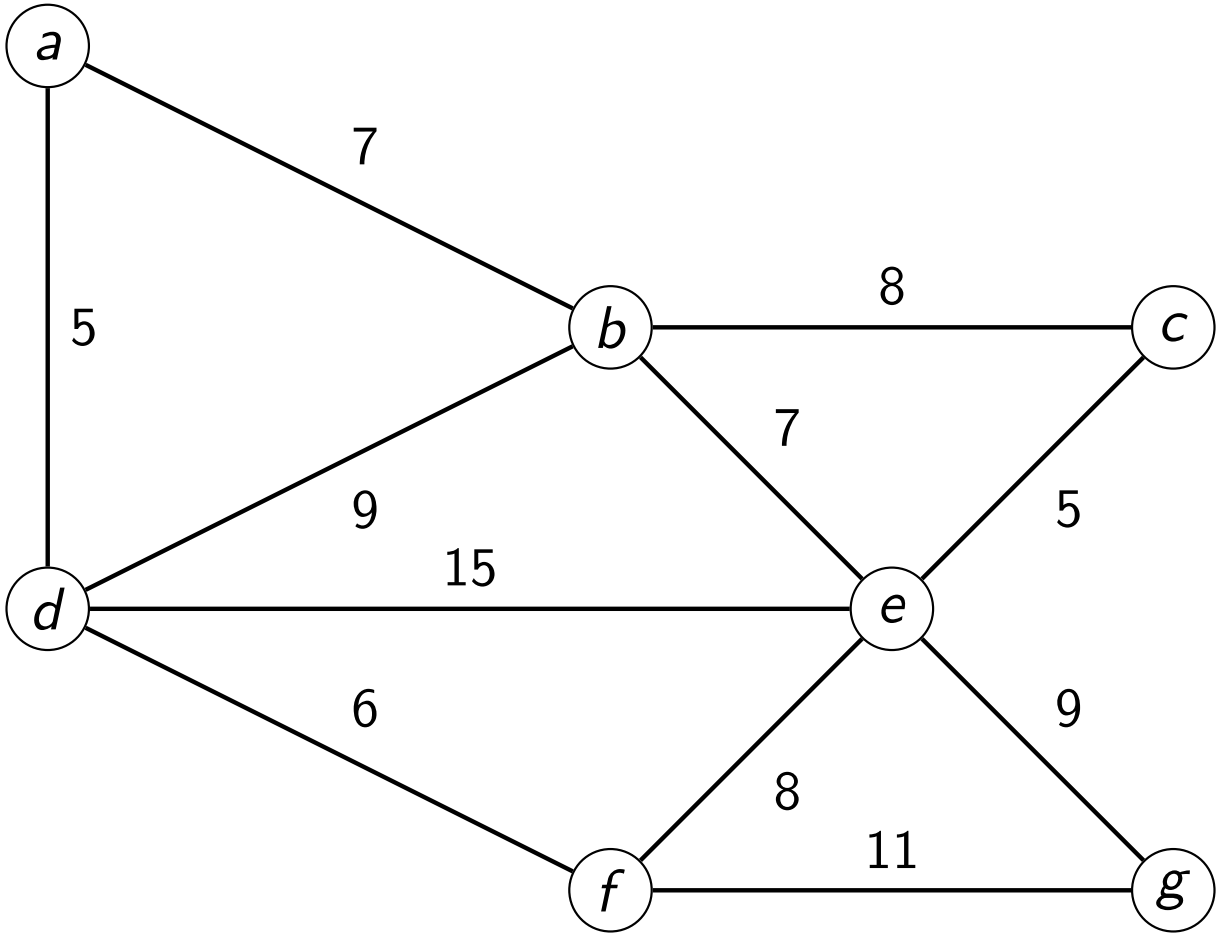
Arbre couvrant

Soit G un graphe pondéré (chaque arête e possède un poids $w(e)$). Un arbre couvrant T de G est un ensemble d'arêtes de G qui forme un arbre et qui contient tous les sommets. Son poids $w(T)$ est la somme des poids des arêtes de l'arbre.

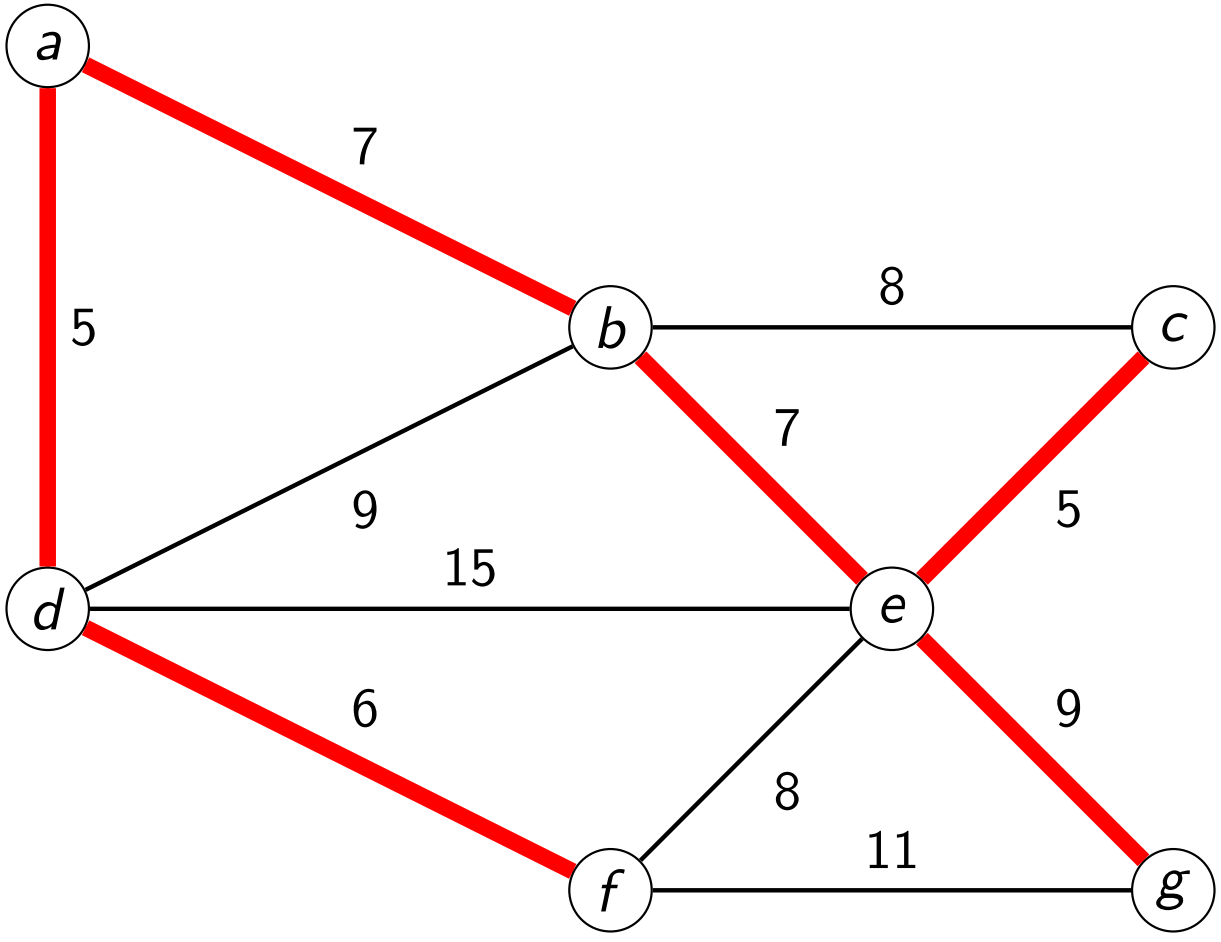
Arbre couvrant de poids minimal

Un arbre couvrant dont le poids est le plus petit possible est appelé un **arbre couvrant de poids minimal**.

Arbre couvrant de poids minimal : exemple



Arbre couvrant de poids minimal : exemple



Arbre couvrant de poids minimal : algorithmes

Il existe deux algorithmes gloutons très connus, qui ajoutent des arêtes une par une jusqu'à former un arbre couvrant de poids minimum.
Ils différent juste par le choix de l'arête à ajouter à chaque itération :

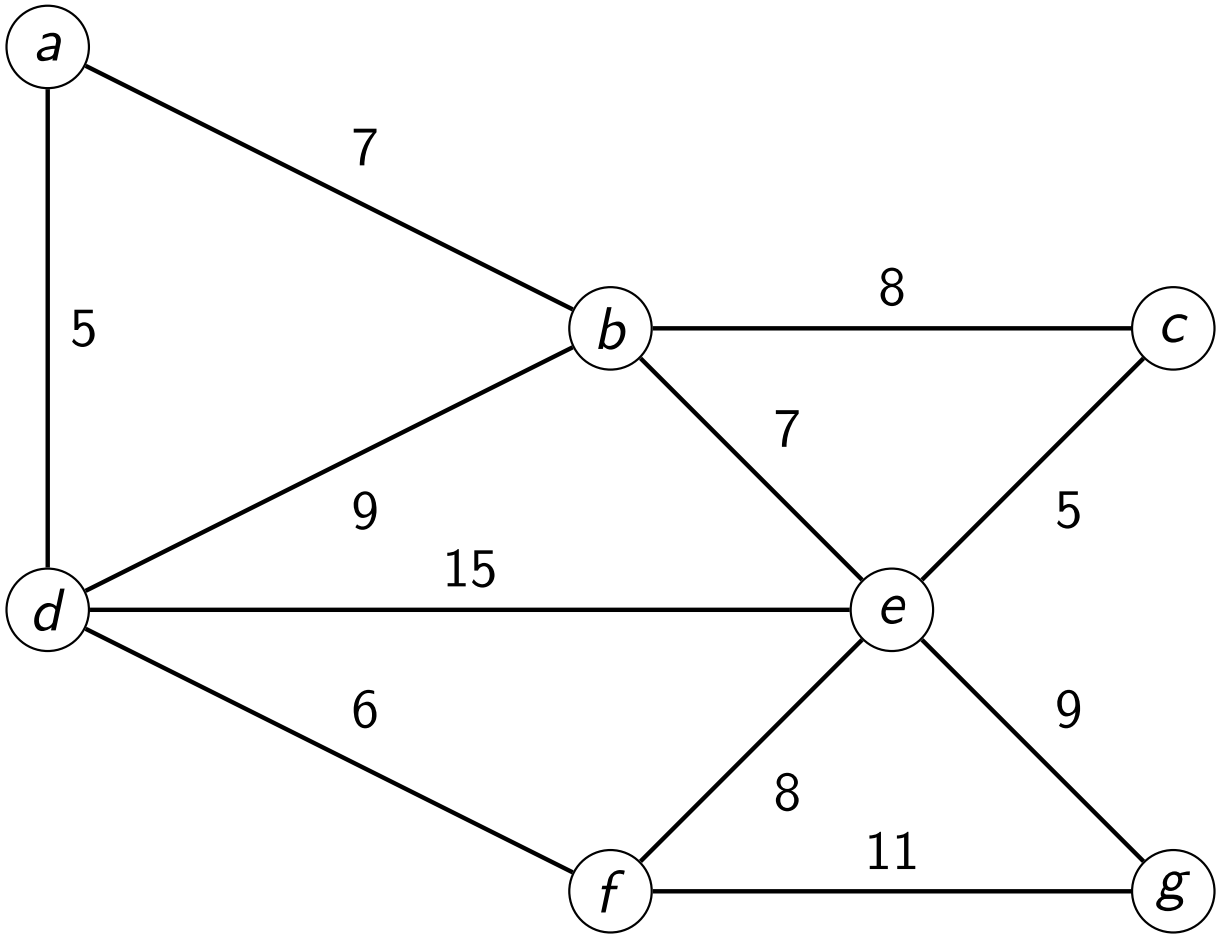
- **Kruskal** : ajoute la plus petite arête qui ne créé pas de cycle
- **Prim** : ajoute la plus petite arête qui conserve la connexité

Algorithme de Kruskal

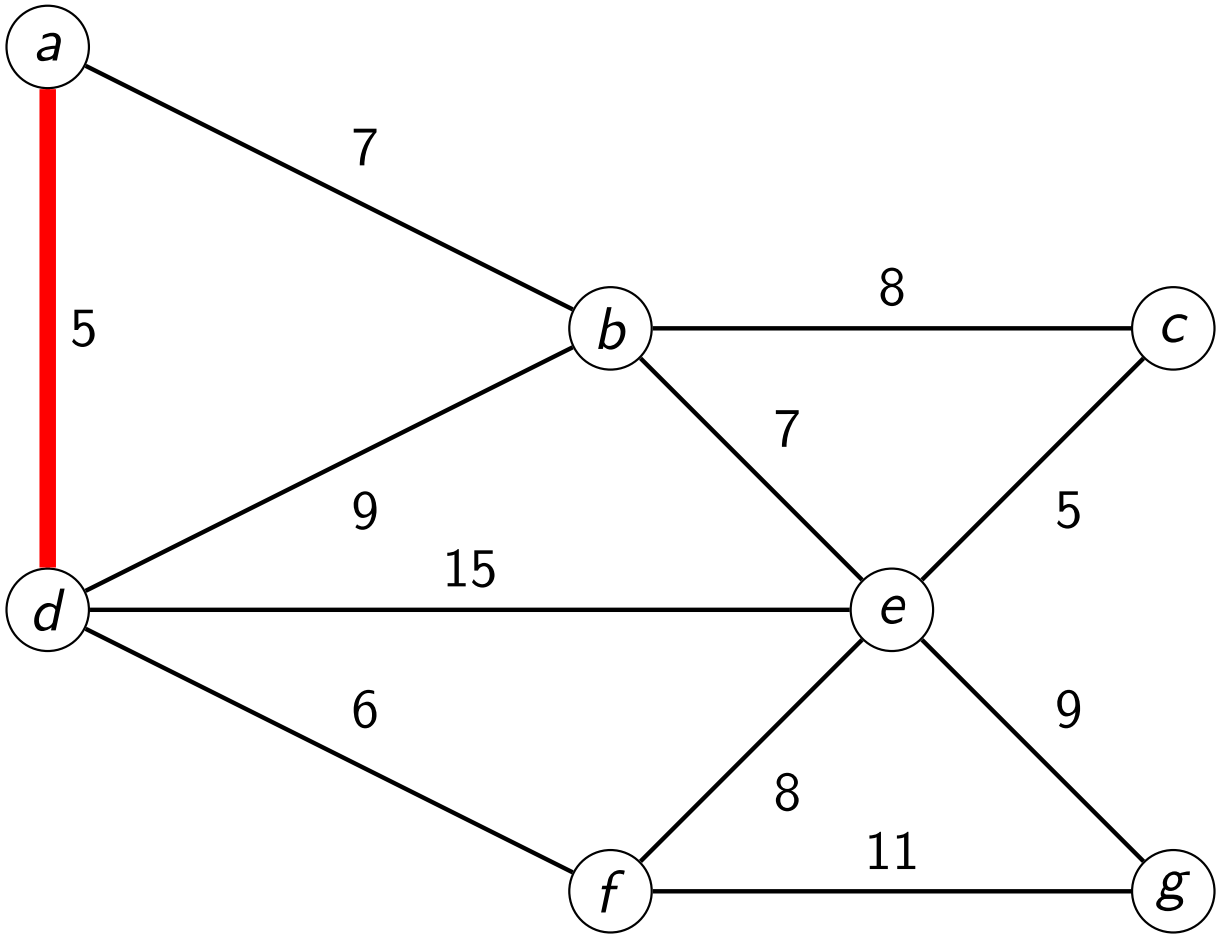
Trier les arêtes par poids croissant.
Commencer avec un arbre T vide (aucune arête).

Pour chaque arête *a* par poids croissant:
Si l'ajout de *a* ne créé pas de cycle dans T :
Ajouter *a* à T

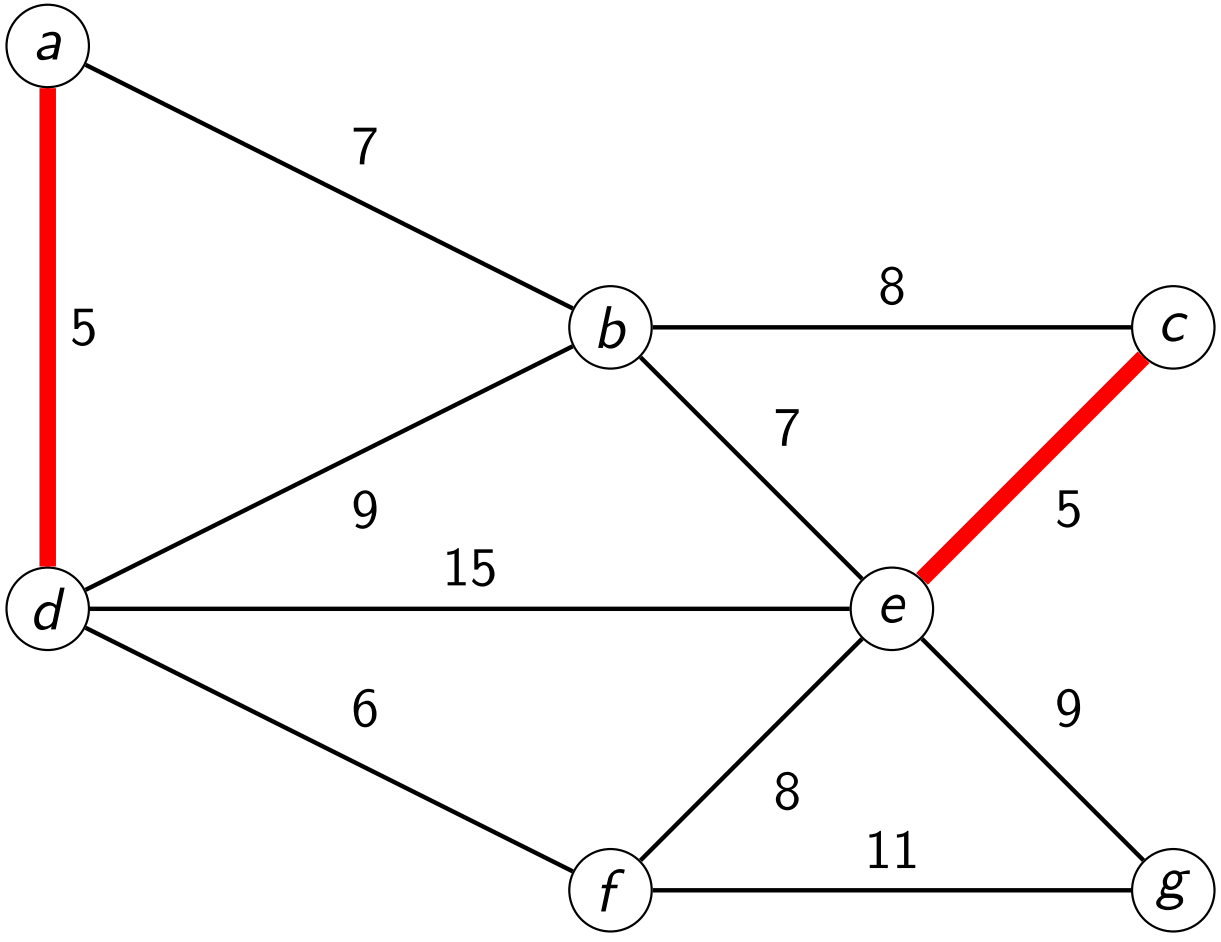
Algorithme de Kruskal



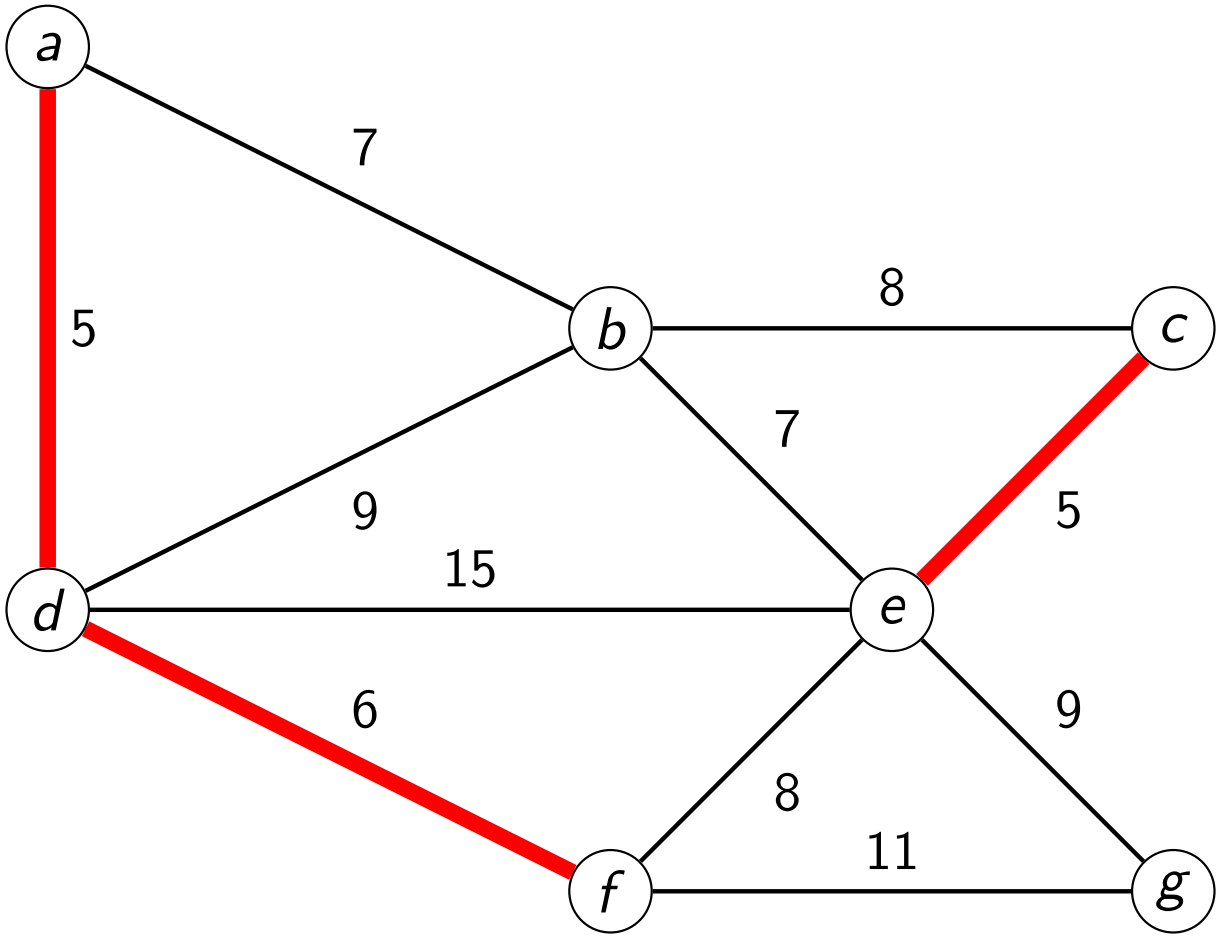
Algorithme de Kruskal



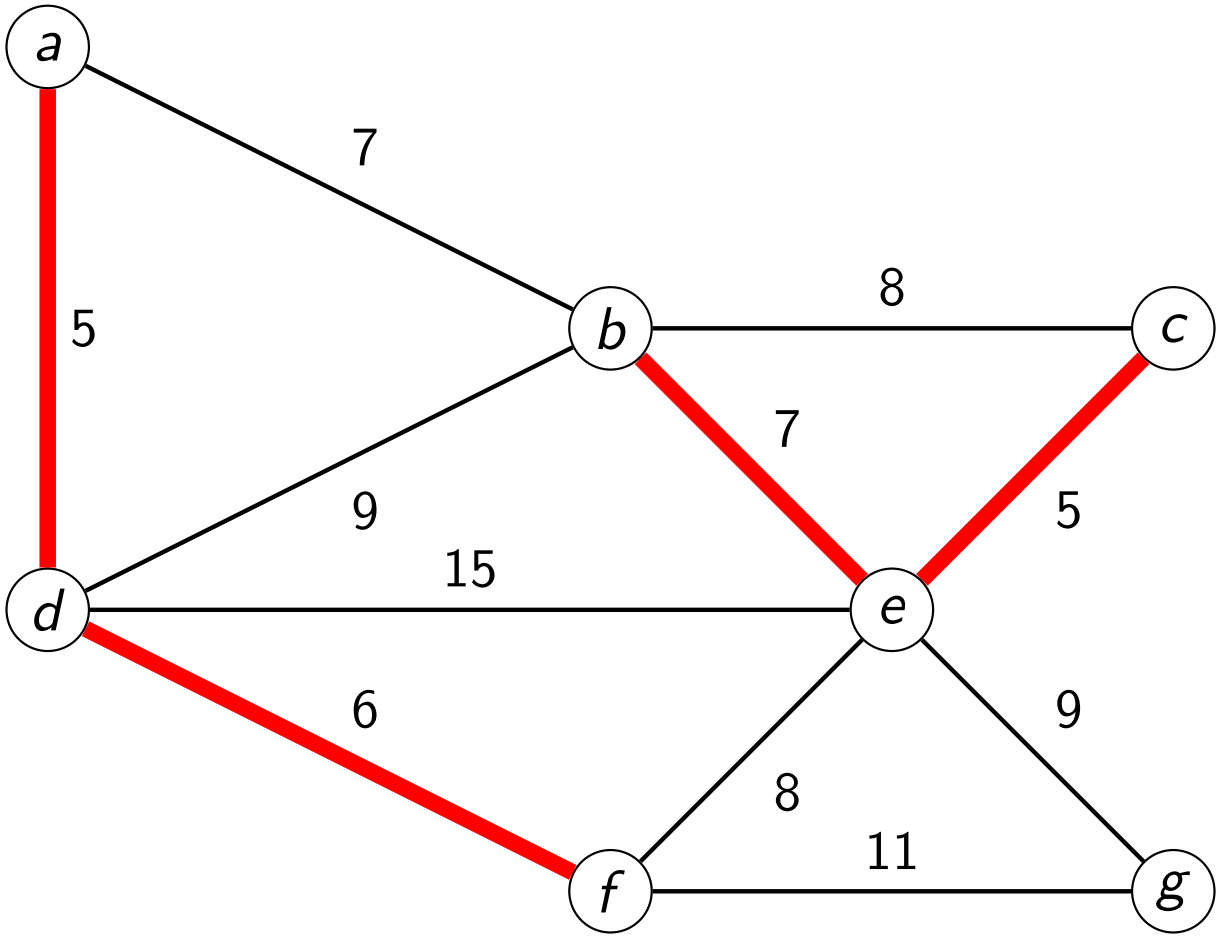
Algorithme de Kruskal



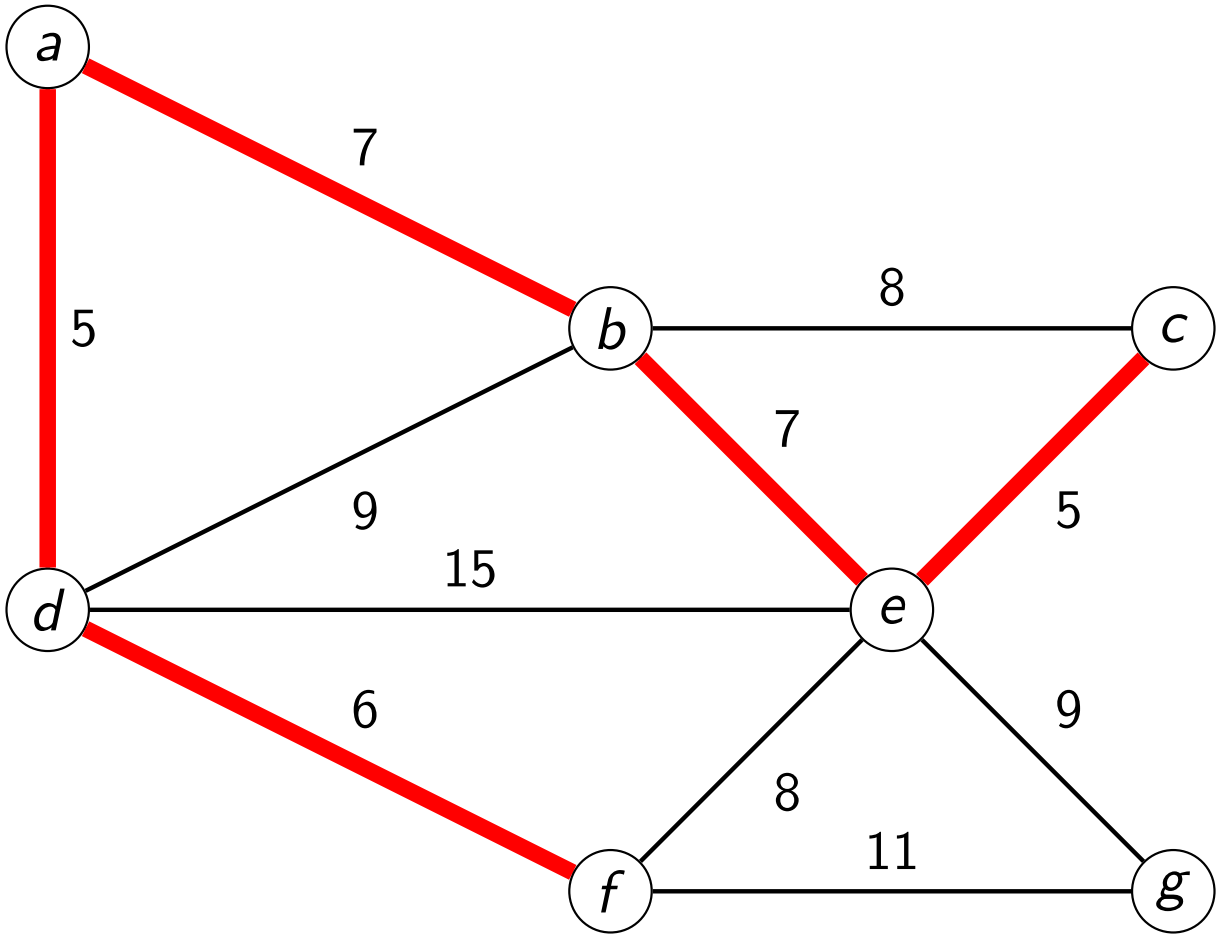
Algorithme de Kruskal



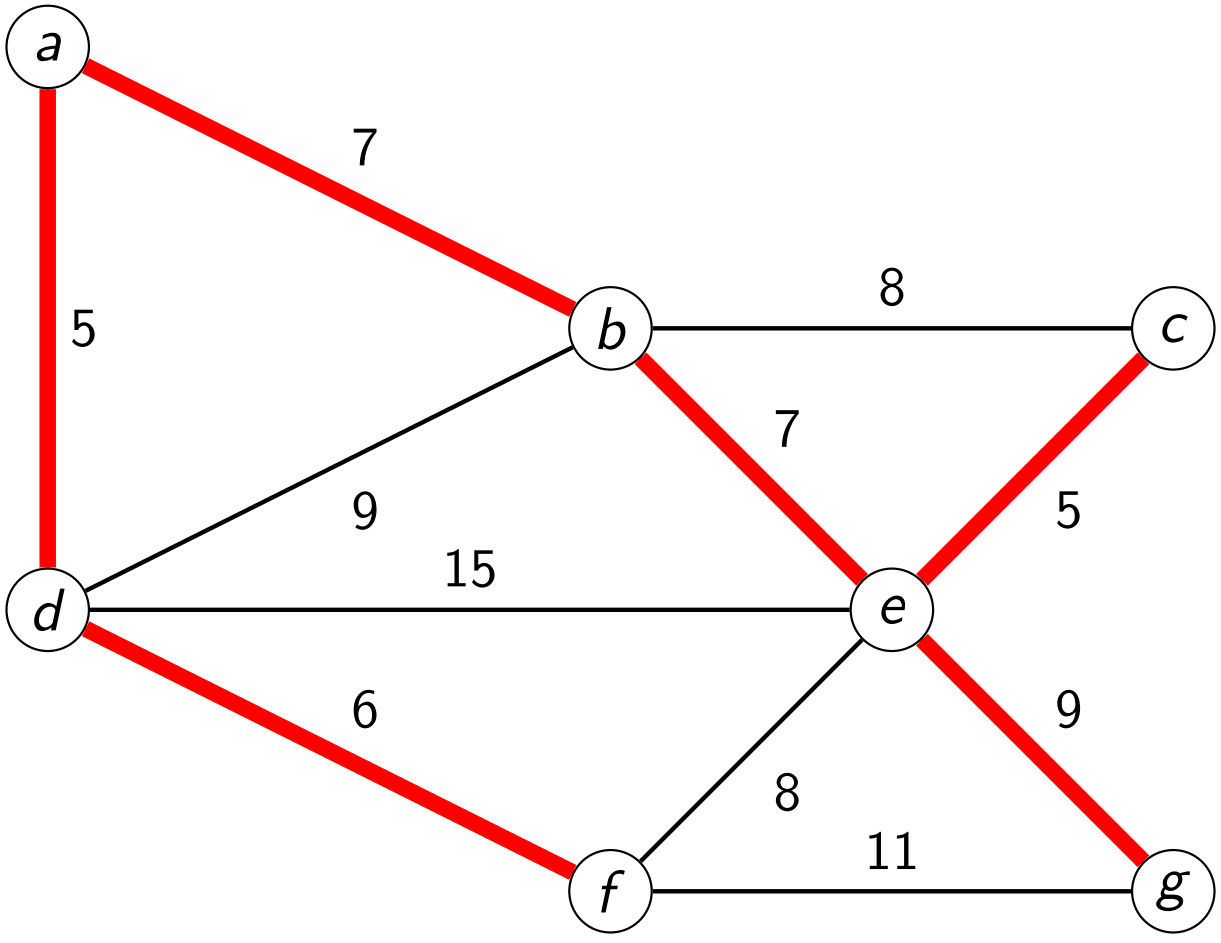
Algorithme de Kruskal



Algorithme de Kruskal



Algorithme de Kruskal



Algorithme de Kruskal

Théorème

L'algorithme de Kruskal sur un graphe G donne bien un arbre couvrant de poids minimum.

Preuve : Soit T l'arbre obtenu par Kruskal.

1. Montrer que T est un arbre couvrant.
2. Montrer que T est de poids minimum.

Algorithme de Kruskal

T est un arbre couvrant :

1. T est sans cycle :

Algorithme de Kruskal

T est un arbre couvrant :

1. T est sans cycle : car l'algorithme de créé pas de cycle
2. T est connexe (et couvrant) :

Algorithme de Kruskal

T est un arbre couvrant :

1. T est sans cycle : car l'algorithme de créé pas de cycle
2. T est connexe (et couvrant) :
   Soit u et v deux sommets de G. Soit U l'ensemble des sommets accessibles depuis u dans T.

Algorithme de Kruskal

T est un arbre couvrant :

1. T est sans cycle : car l'algorithme de créé pas de cycle

2. T est connexe (et couvrant) :
   Soit u et v deux sommets de G. Soit U l'ensemble des sommets accessibles depuis u dans T.
   Supposons $v \notin U$. Comme G est connexe, il existe une arête de G entre U et $V \setminus U$.

Algorithme de Kruskal

T est un arbre couvrant :

1. T est sans cycle : car l'algorithme de créé pas de cycle

2. T est connexe (et couvrant) :
   Soit u et v deux sommets de G. Soit U l'ensemble des sommets accessibles depuis u dans T.
   Supposons $v \notin U$. Comme G est connexe, il existe une arête de G entre U et $V \setminus U$.
   Cette arête aurait dû être ajoutée à T, puisqu'elle ne créée pas de cycle.
   Contradiction : v est donc accessible depuis u dans T.
   Comme c'est vrai pour tout u, v, T est connexe.

Preuve de Kruskal

Théorème

L'algorithme de Kruskal sur un graphe G donne un arbre couvrant de poids minimum.

Preuve :

Preuve de Kruskal

Théorème

L'algorithme de Kruskal sur un graphe G donne un arbre couvrant de poids minimum.

Preuve : Soient T l'arbre obtenu par Kruskal et T^* un arbre de poids minimum.
Si $T = T^*$, le théorème est démontré.

Preuve de Kruskal

Théorème

L'algorithme de Kruskal sur un graphe G donne un arbre couvrant de poids minimum.

Preuve : Soient T l'arbre obtenu par Kruskal et T^* un arbre de poids minimum.
Si $T = T^*$, le théorème est démontré.
Sinon, soit $e^* \in T^*$ une arête de poids min n'appartenant pas à T.

Preuve de Kruskal

Théorème

L'algorithme de Kruskal sur un graphe G donne un arbre couvrant de poids minimum.

Preuve : Soient T l'arbre obtenu par Kruskal et T^* un arbre de poids minimum.
Si $T = T^*$, le théorème est démontré.
Sinon, soit $e^* \in T^*$ une arête de poids min n'appartenant pas à T.
Comme T est connexe, il existe un chemin C dans T reliant les extrémités de e^*.

Preuve de Kruskal

Théorème

L'algorithme de Kruskal sur un graphe G donne un arbre couvrant de poids minimum.

Preuve : Soient T l'arbre obtenu par Kruskal et T^* un arbre de poids minimum.
Si $T = T^*$, le théorème est démontré.
Sinon, soit $e^* \in T^*$ une arête de poids min n'appartenant pas à T.
Comme T est connexe, il existe un chemin C dans T reliant les extrémités de e^*.

- Il existe une arête e de C qui n'est pas dans T^*

Preuve de Kruskal

Théorème

L'algorithme de Kruskal sur un graphe G donne un arbre couvrant de poids minimum.

Preuve : Soient T l'arbre obtenu par Kruskal et T^* un arbre de poids minimum.
Si $T = T^*$, le théorème est démontré.
Sinon, soit $e^* \in T^*$ une arête de poids min n'appartenant pas à T.
Comme T est connexe, il existe un chemin C dans T reliant les extrémités de e^*.

- Il existe une arête e de C qui n'est pas dans T^* car T^* ne peut pas contenir de cycle
- $w(e) < w(e^*)$ (sinon Kruskal aurait ajouté e^* à T)

Preuve de Kruskal

Considérons $T_2 = T \setminus \{e\} \cup \{e^*\}$.

Preuve de Kruskal

Considérons $T_2 = T \setminus \{e\} \cup \{e^*\}$.

- T_2 est un arbre couvrant

Preuve de Kruskal

Considérons $T_2 = T \setminus \{e\} \cup \{e^*\}$.

- T_2 est un arbre couvrant
- $w(T) \leq w(T_2)$

Preuve de Kruskal

Considérons $T_2 = T \setminus \{e\} \cup \{e^*\}$.

- T_2 est un arbre couvrant
- $w(T) \leq w(T_2)$

On répète le même processus sur T_2, ce qui nous donne T_3, T_4... jusqu'à obtenir T^* :

$$w(T) \leq w(T_2) \leq w(T_3) \leq ... \leq w(T^*)$$

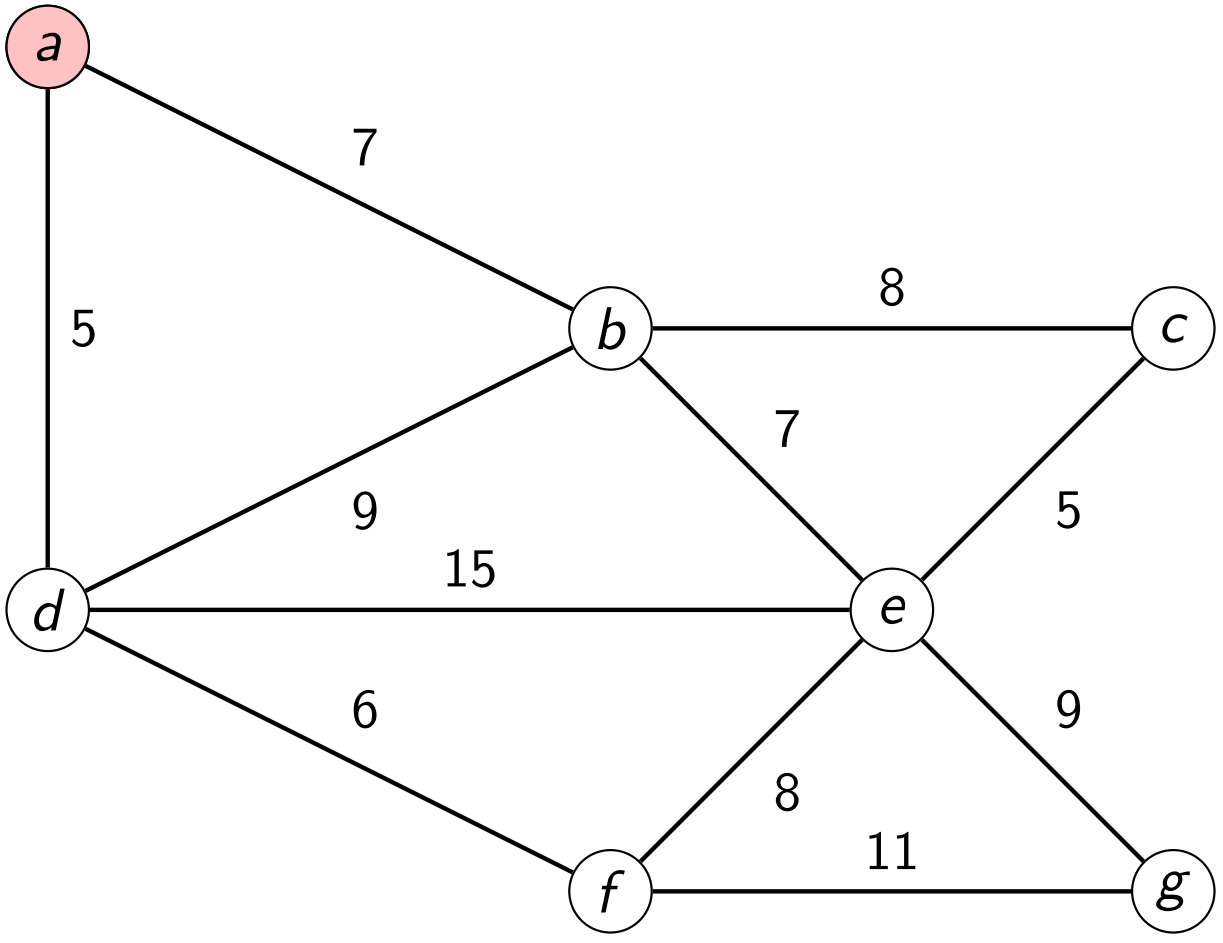
Comme T est un arbre couvrant et $w(T) \leq w(T^*)$, on a en fait $w(T) = w(T^*)$ et T **est un arbre couvrant de poids minimum**.

Algorithme de Prim

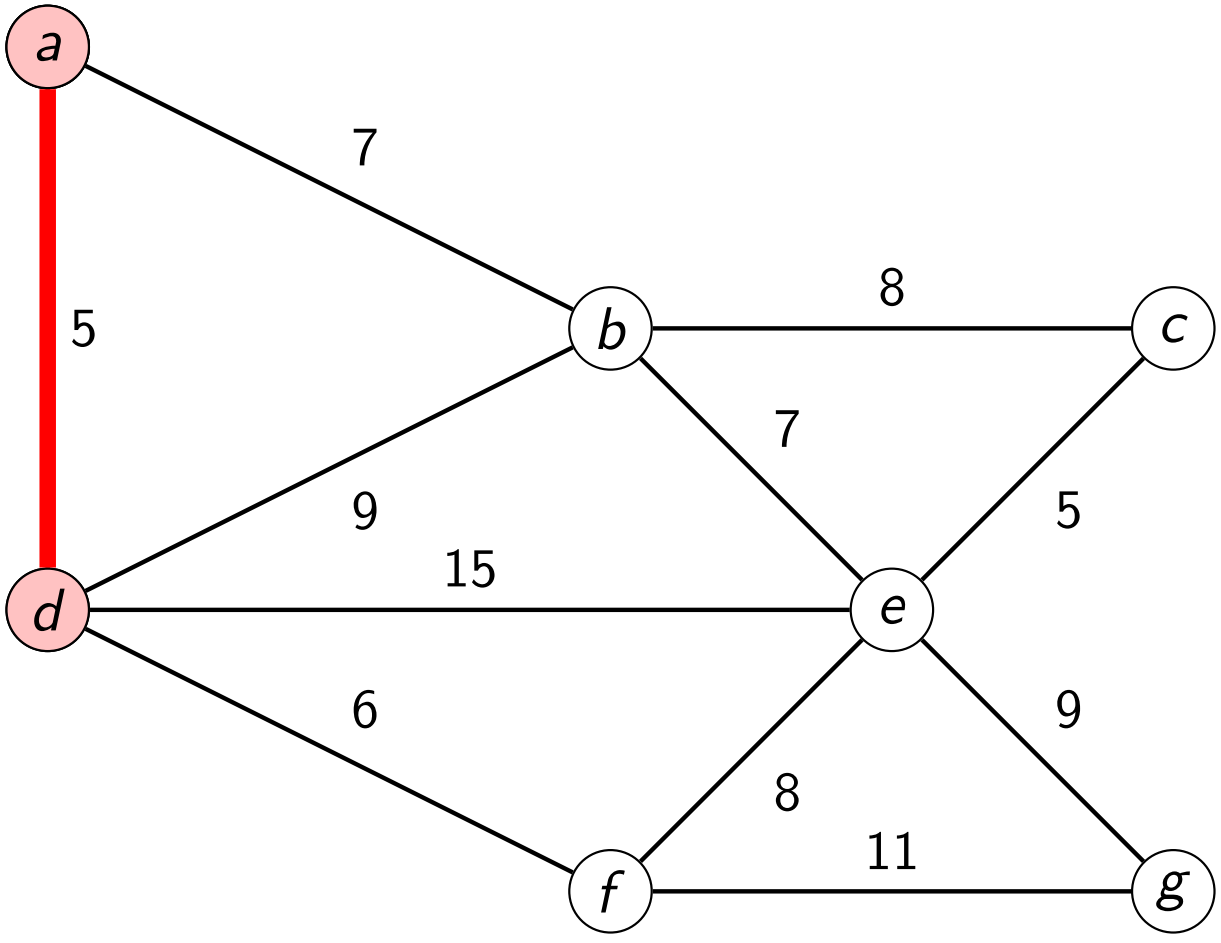
Commencer avec un arbre T contenant un seul sommet.

Tant que T ne contient pas tous les sommets:
Ajouter l'arête sortante de T de poids minimum

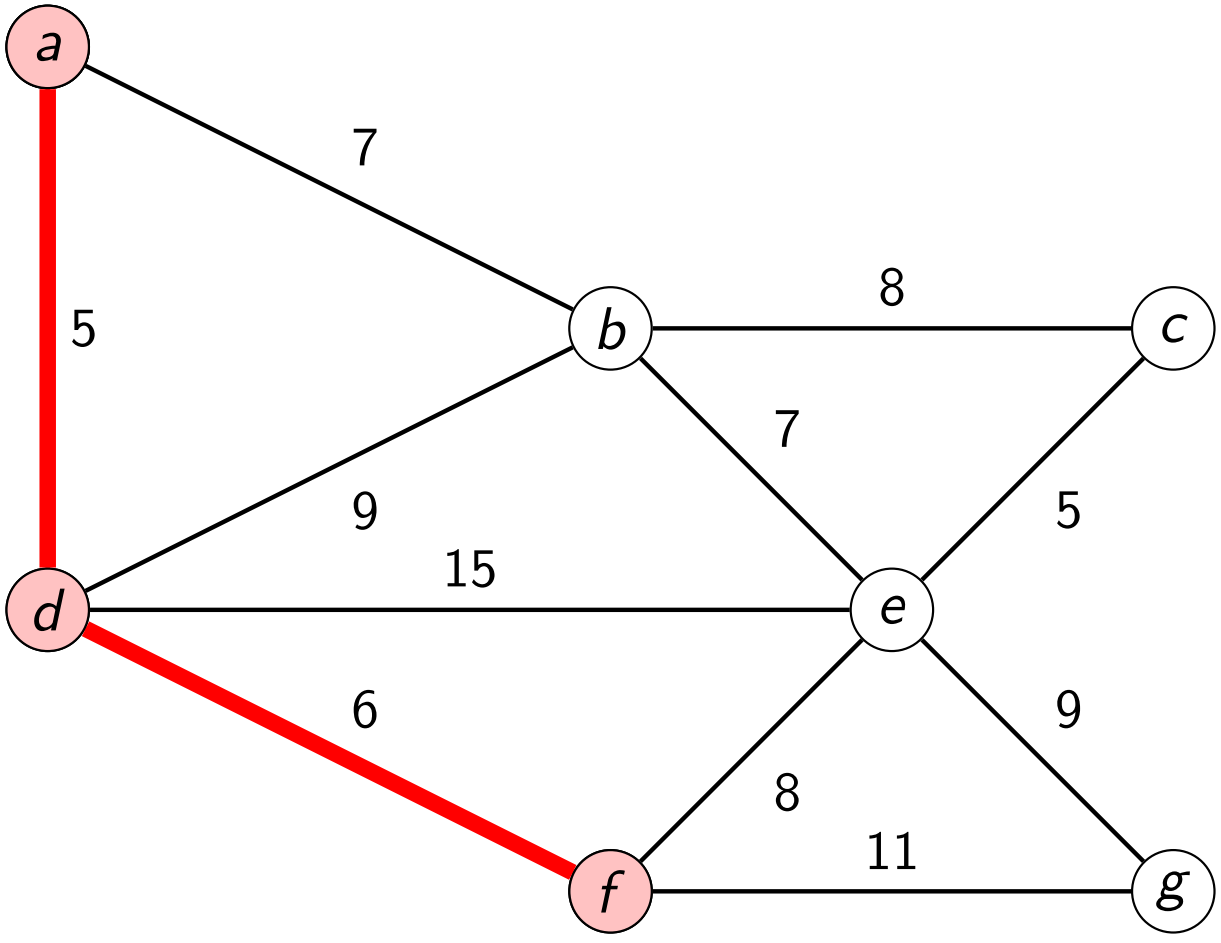
Algorithme de Prim



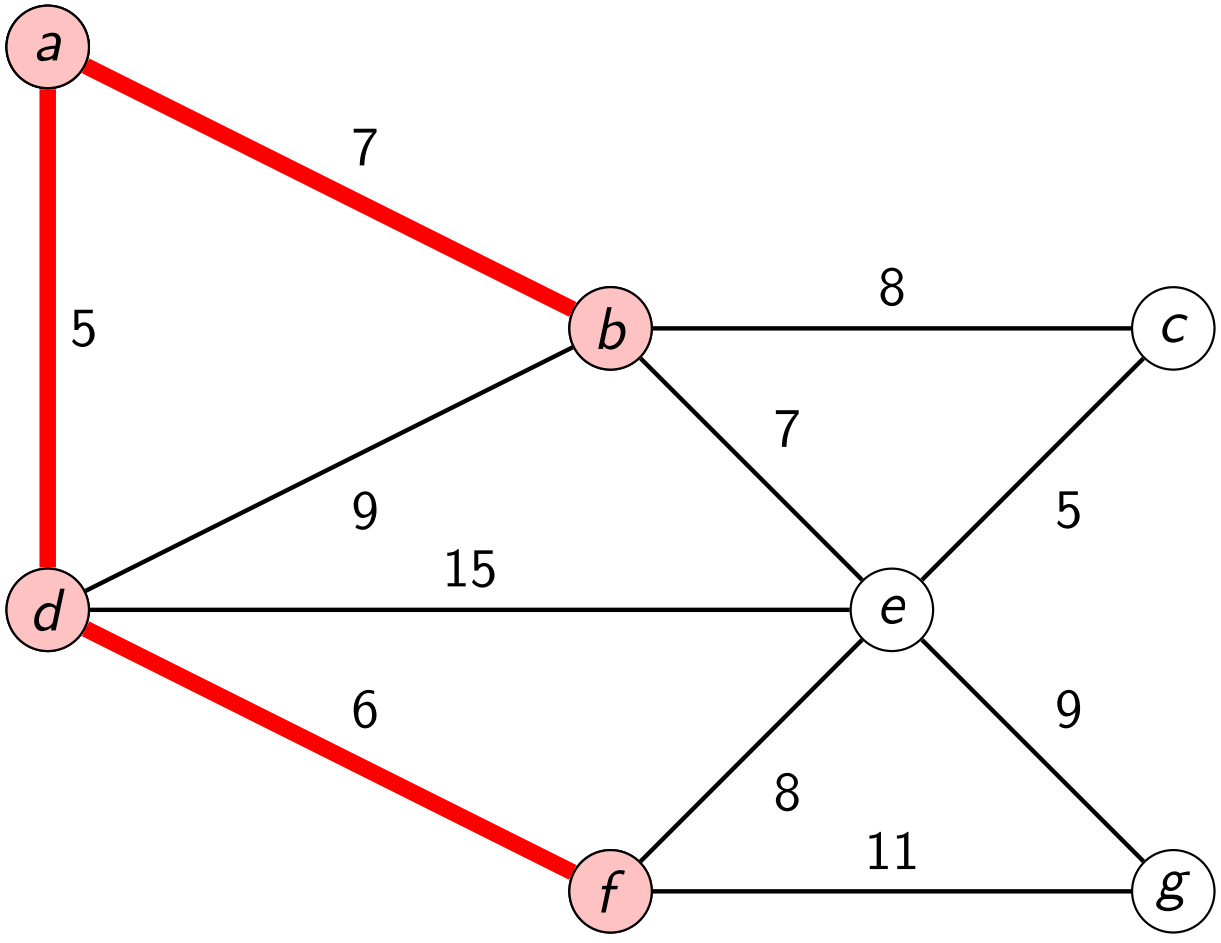
Algorithme de Prim



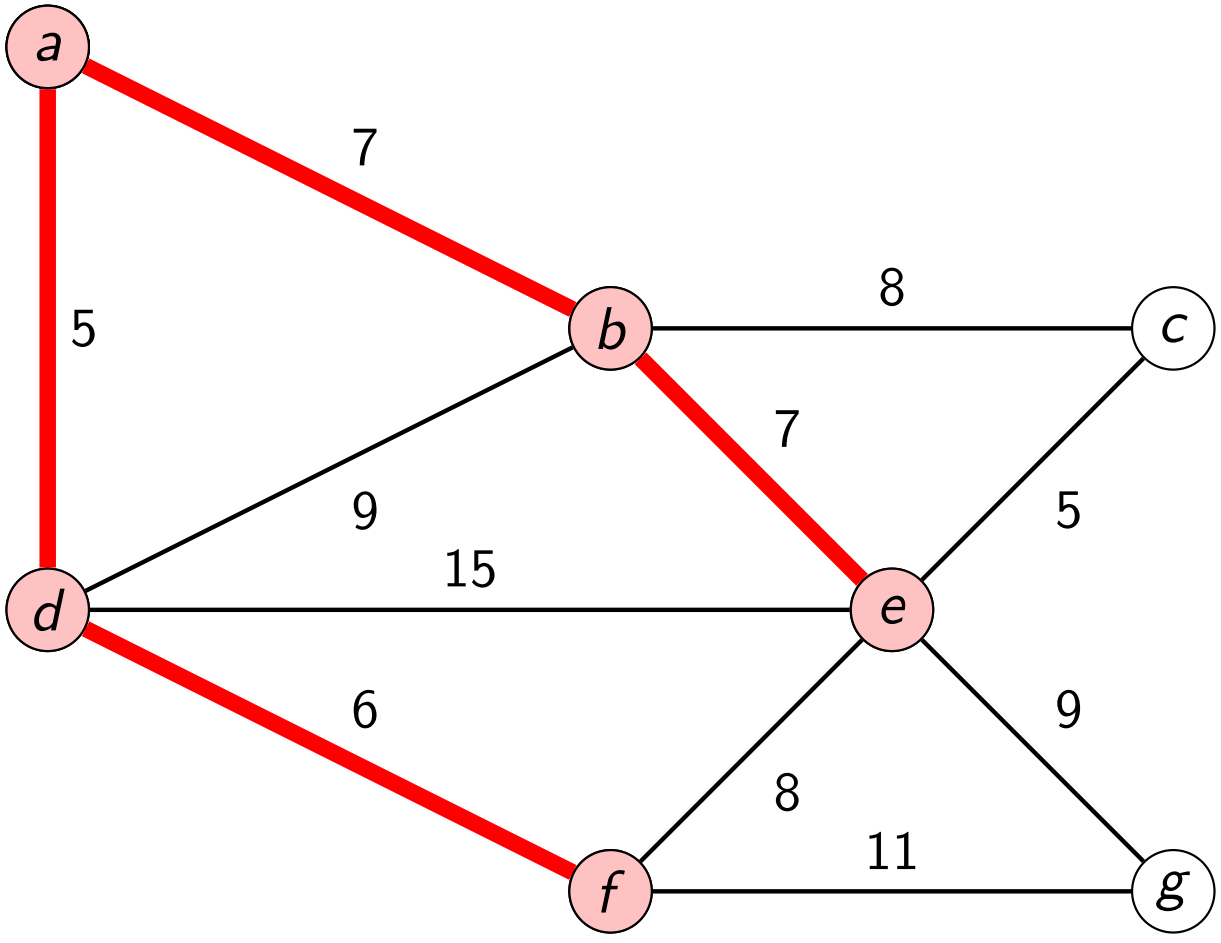
Algorithme de Prim



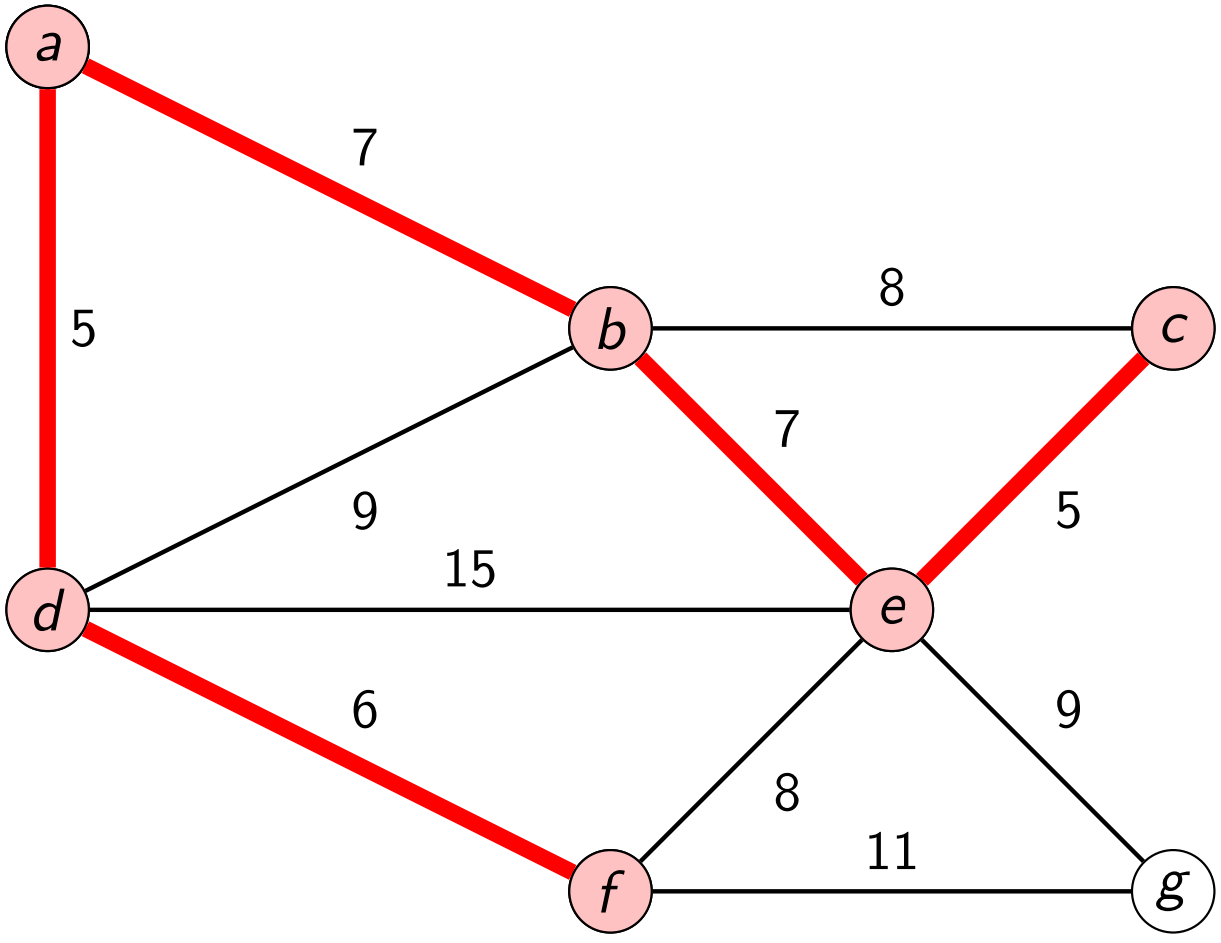
Algorithme de Prim



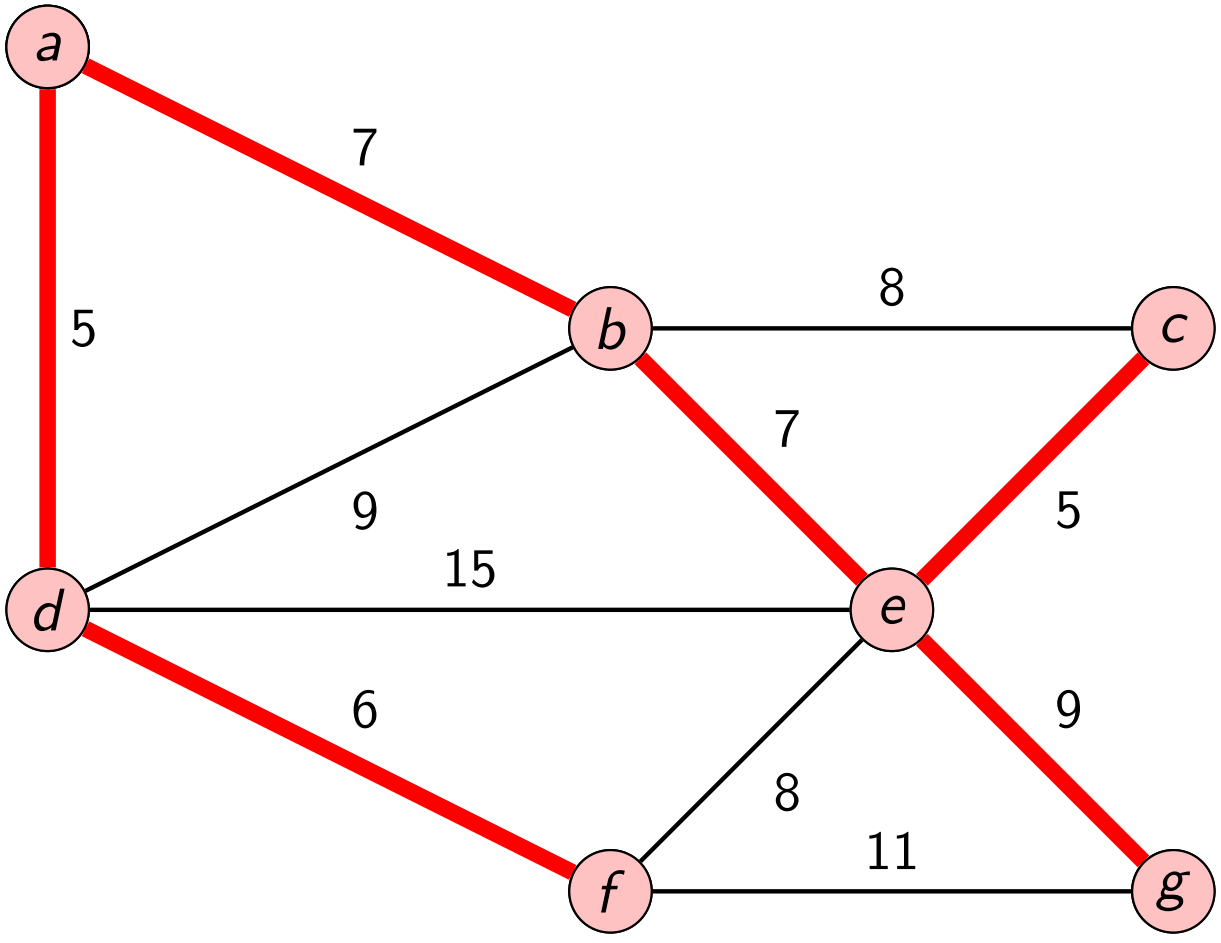
Algorithme de Prim



Algorithme de Prim



Algorithme de Prim



Algorithme de Prim

Théorème

L'algorithme de Prim sur un graphe G donne bien un arbre couvrant de poids minimum.

Preuve :

Algorithme de Prim

Théorème

L'algorithme de Prim sur un graphe G donne bien un arbre couvrant de poids minimum.

Preuve :

Soient T l'arbre obtenu par Prim et T^* un arbre de poids minimum.
Si T n'est pas optimal, il existe i tel que $e_i > e_i^*$. Considérons le plus petit tel i.
Comme Kruskal considère les arêtes par poids croissant, il aurait dû