Algorithmes gloutons

Quentin Fortier

February 10, 2022

Algorithme glouton

La stratégie d'un **algorithme glouton** est d'effectuer à chaque étape l'action qui semble la plus intéressante localement (sans revenir sur sa décision plus tard).

Algorithme glouton

La stratégie d'un **algorithme glouton** est d'effectuer à chaque étape l'action qui semble la plus intéressante localement (sans revenir sur sa décision 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)

Sac à dos

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 ?

Sac à dos

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

Sac à dos

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

Sac à dos

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

Sac à dos

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

Ces méthodes gloutonnes ne sont pas optimales.

Sac à dos

On considère maintenant le problème du **sac à dos fractionnaire**, où il est possible de prendre une fraction d'un objet.

Sac à dos

On considère maintenant le problème du **sac à dos fractionnaire**, où il est possible de prendre une fraction d'un objet.

Théorème

L'algorithme glouton ajoutant les objets dans l'ordre décroissant de valeur/poids, et ajoutant une fraction du dernier objet (qui ne rentre pas en entier dans le sac) est optimal pour le problème du sac à dos fractionnaire.

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

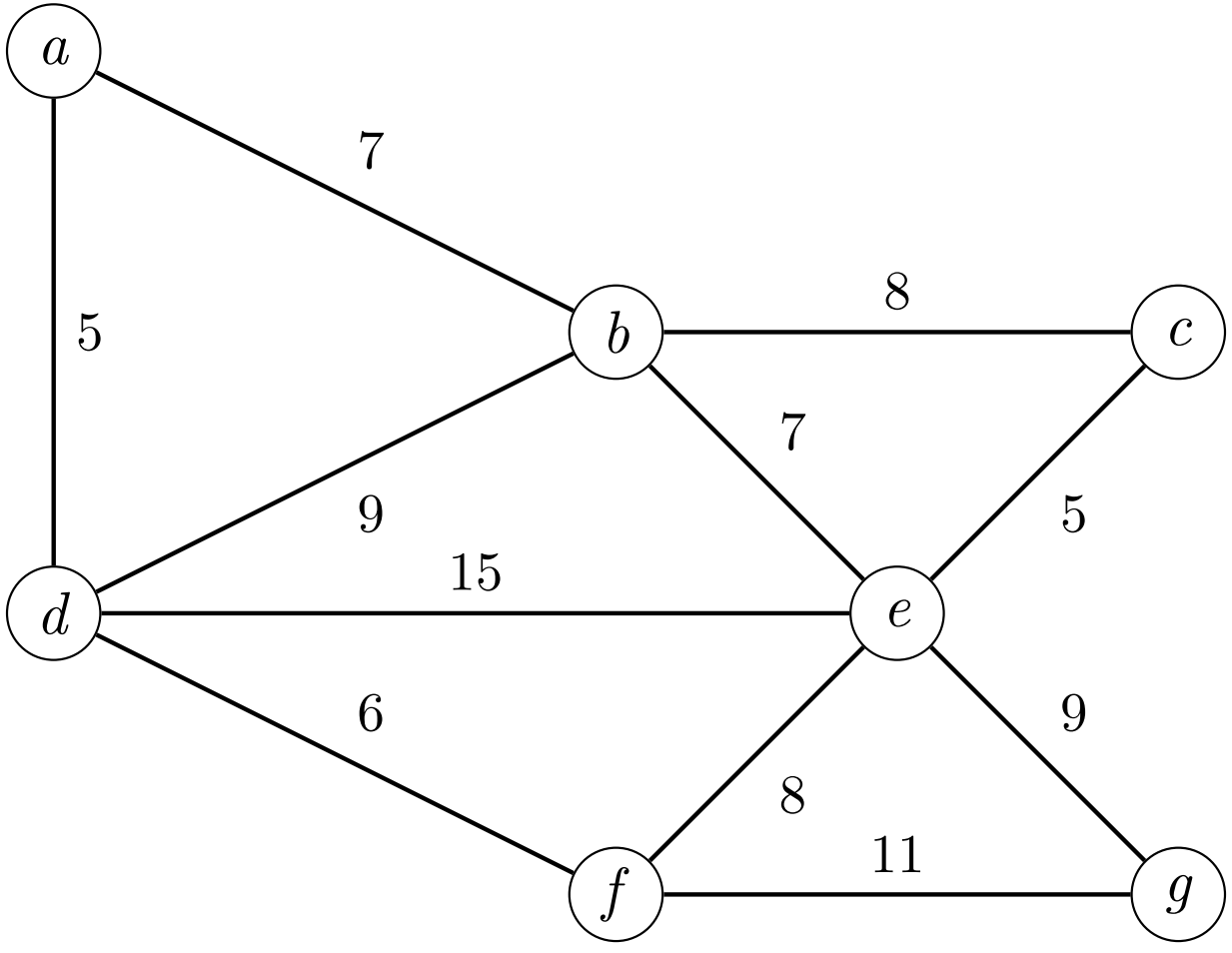
Arbre couvrant

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.

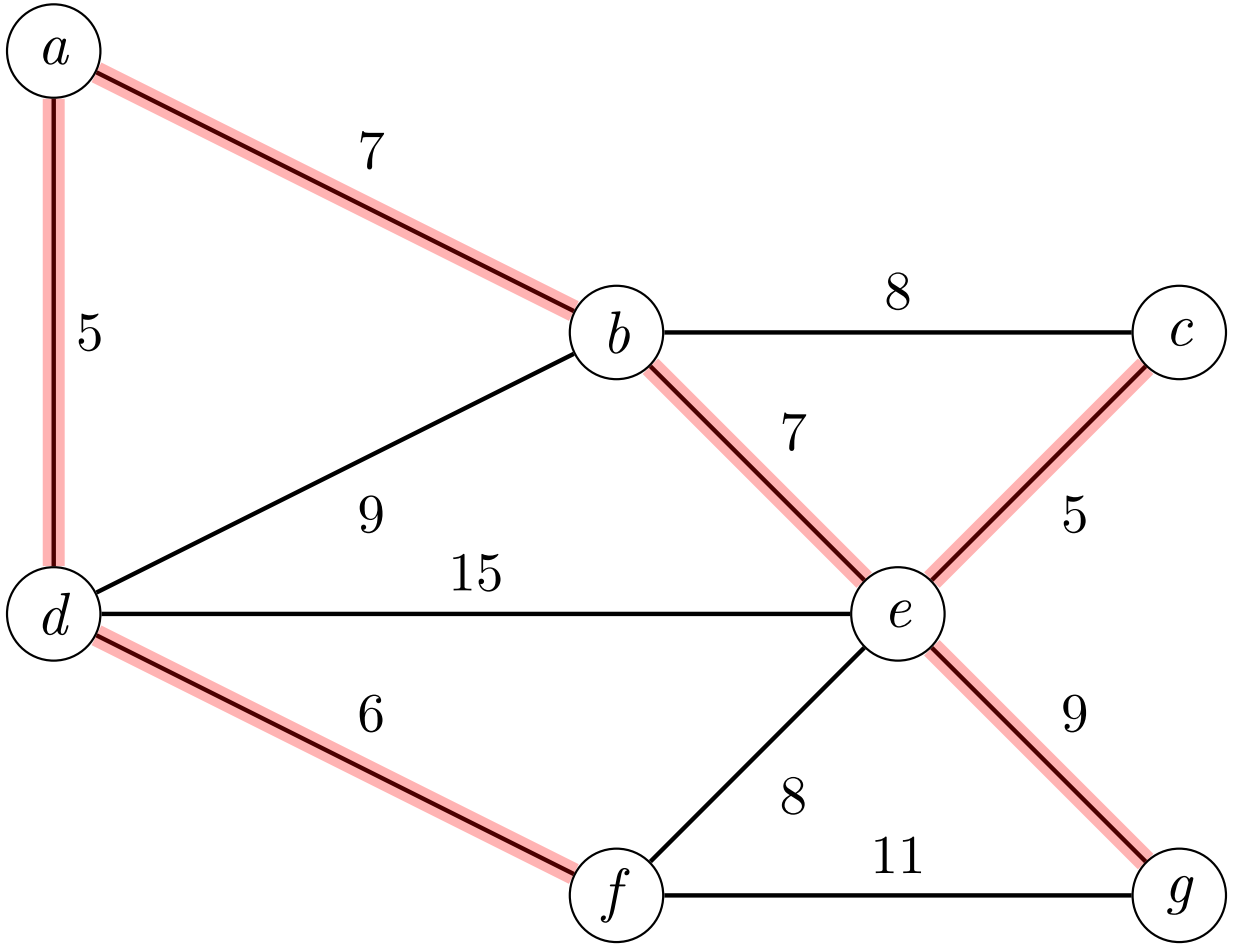
Arbre couvrant de poids minimal

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

Arbre couvrant de poids minimal



Arbre couvrant de poids minimal



Arbre couvrant de poids minimal

Deux algorithmes gloutons très connus permettent de trouver un arbre couvrant de poids minimum dans un graphe.
Ils ajoutent des arêtes une par une jusqu'à former un arbre couvrant de poids minimum.

Arbre couvrant de poids minimal

Deux algorithmes gloutons très connus permettent de trouver un arbre couvrant de poids minimum dans un graphe.
Ils ajoutent des arêtes une par une jusqu'à former un arbre couvrant de poids minimum.
Ils différent par le choix de l'arête à ajouter à chaque itération :

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

Arbre couvrant de poids minimal

Deux algorithmes gloutons très connus permettent de trouver un arbre couvrant de poids minimum dans un graphe.
Ils ajoutent des arêtes une par une jusqu'à former un arbre couvrant de poids minimum.
Ils différent 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

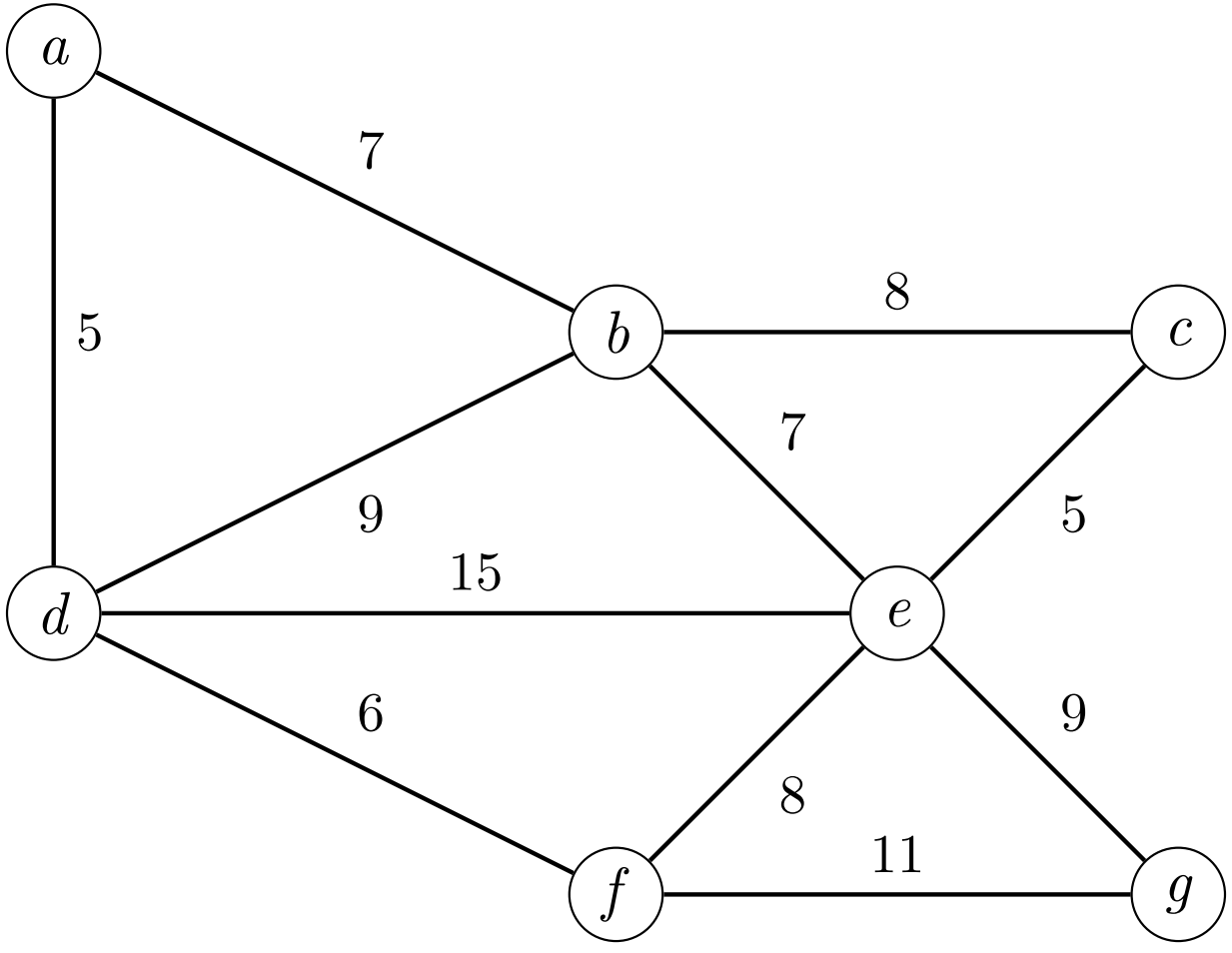
Arbre couvrant de poids minimal : Kruskal

Algorithme de Kruskal :

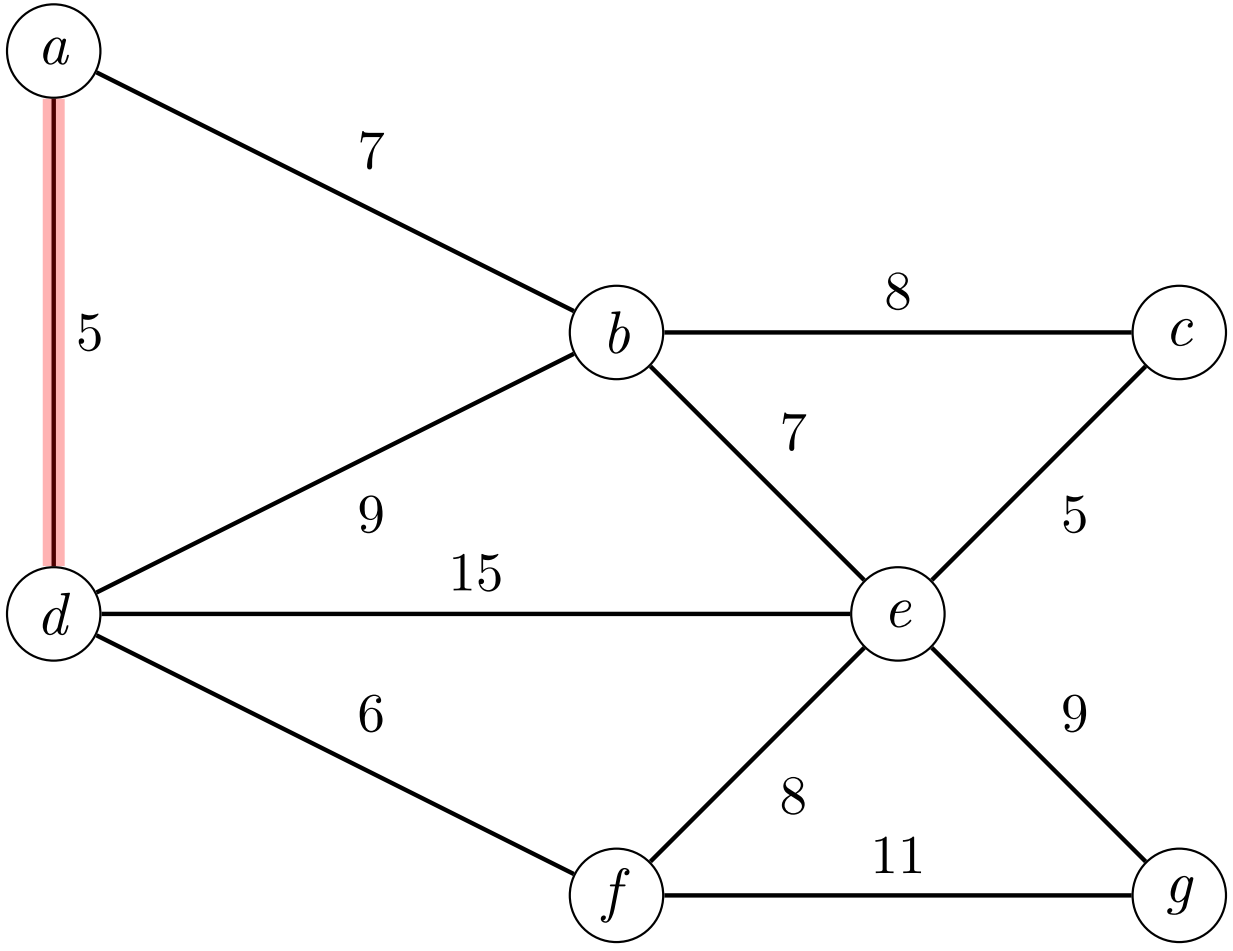
```
Trier les arêtes par poids croissant.
Commencer avec un arbre T vide (aucune arête).

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

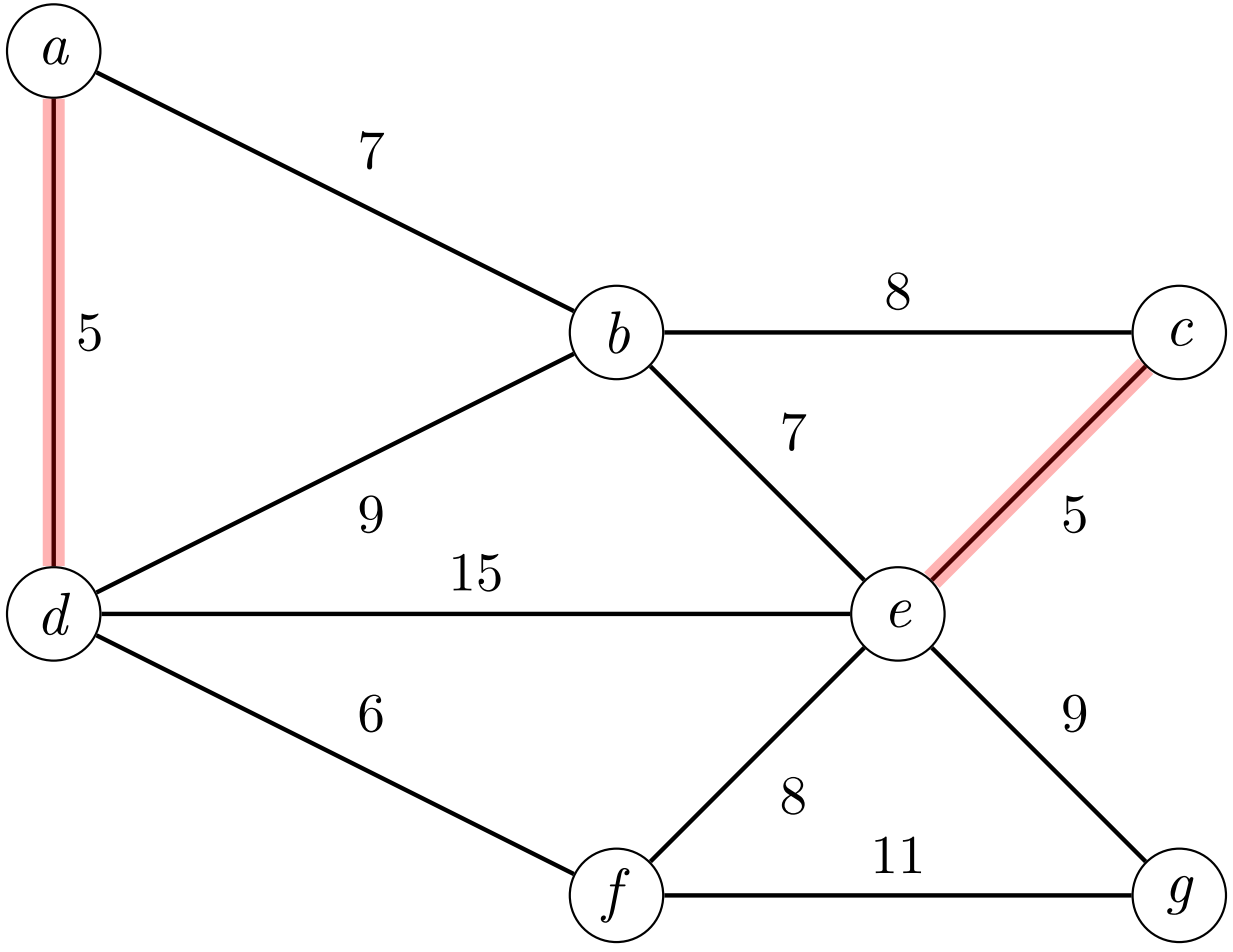
Arbre couvrant de poids minimal : Kruskal



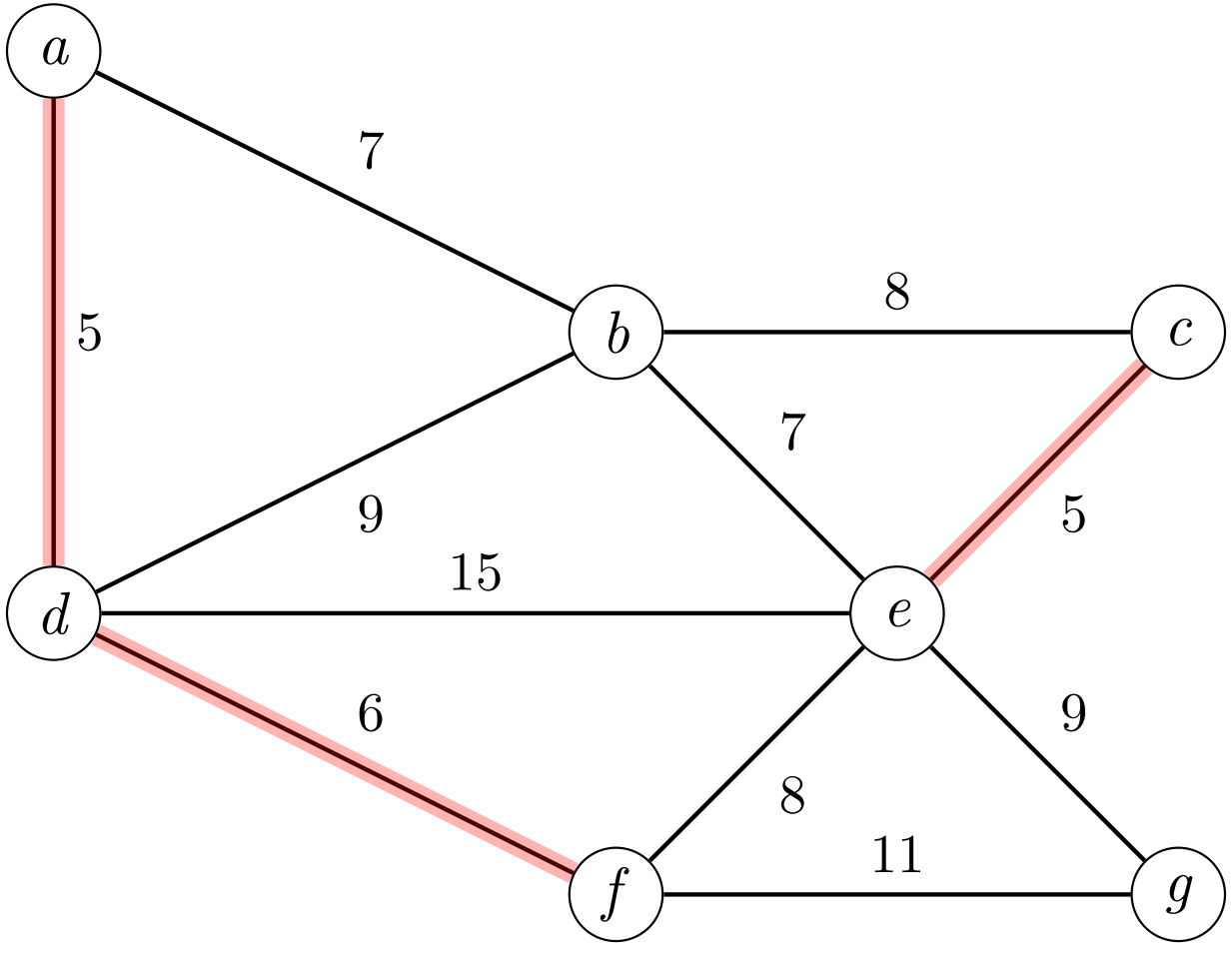
Arbre couvrant de poids minimal : Kruskal



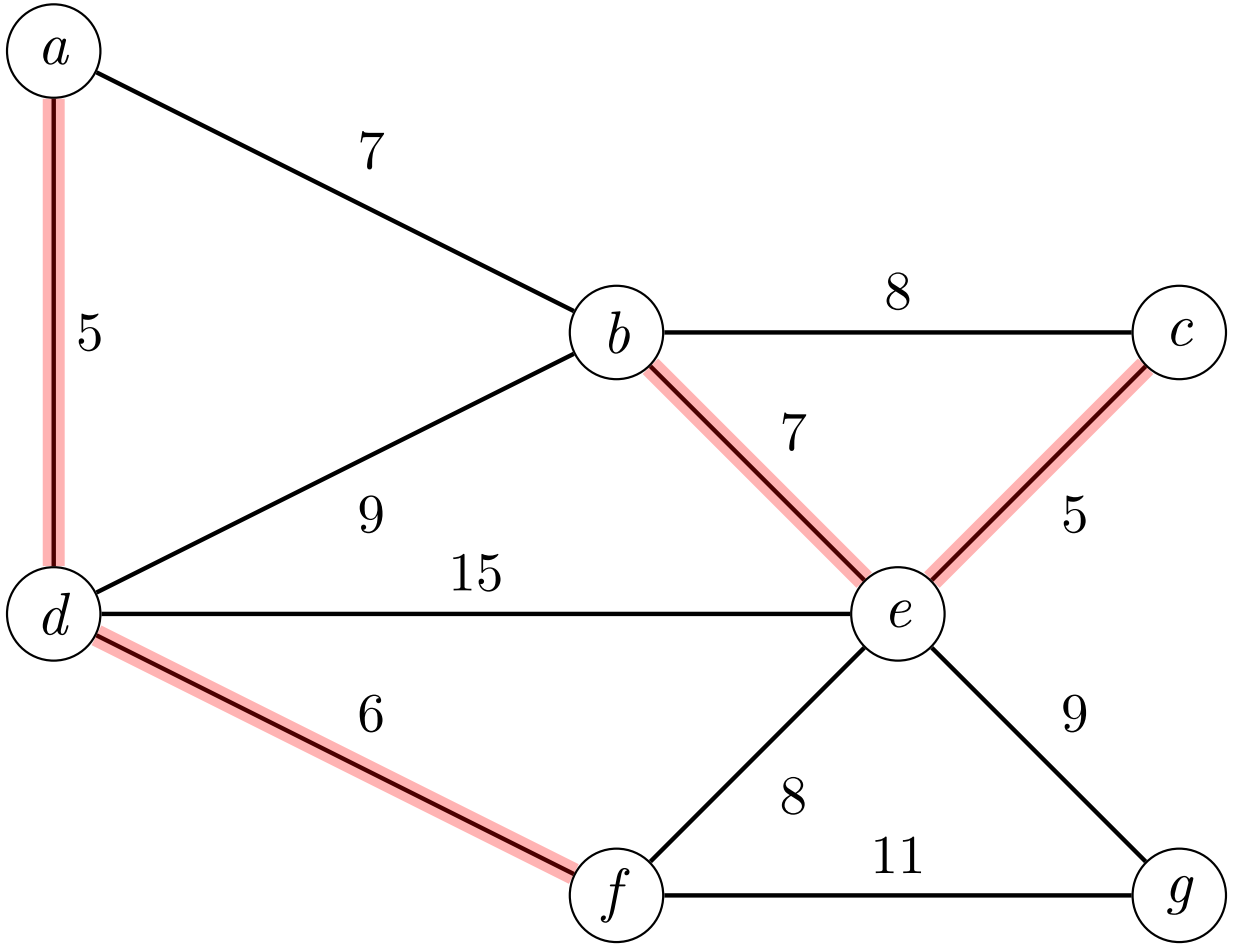
Arbre couvrant de poids minimal : Kruskal



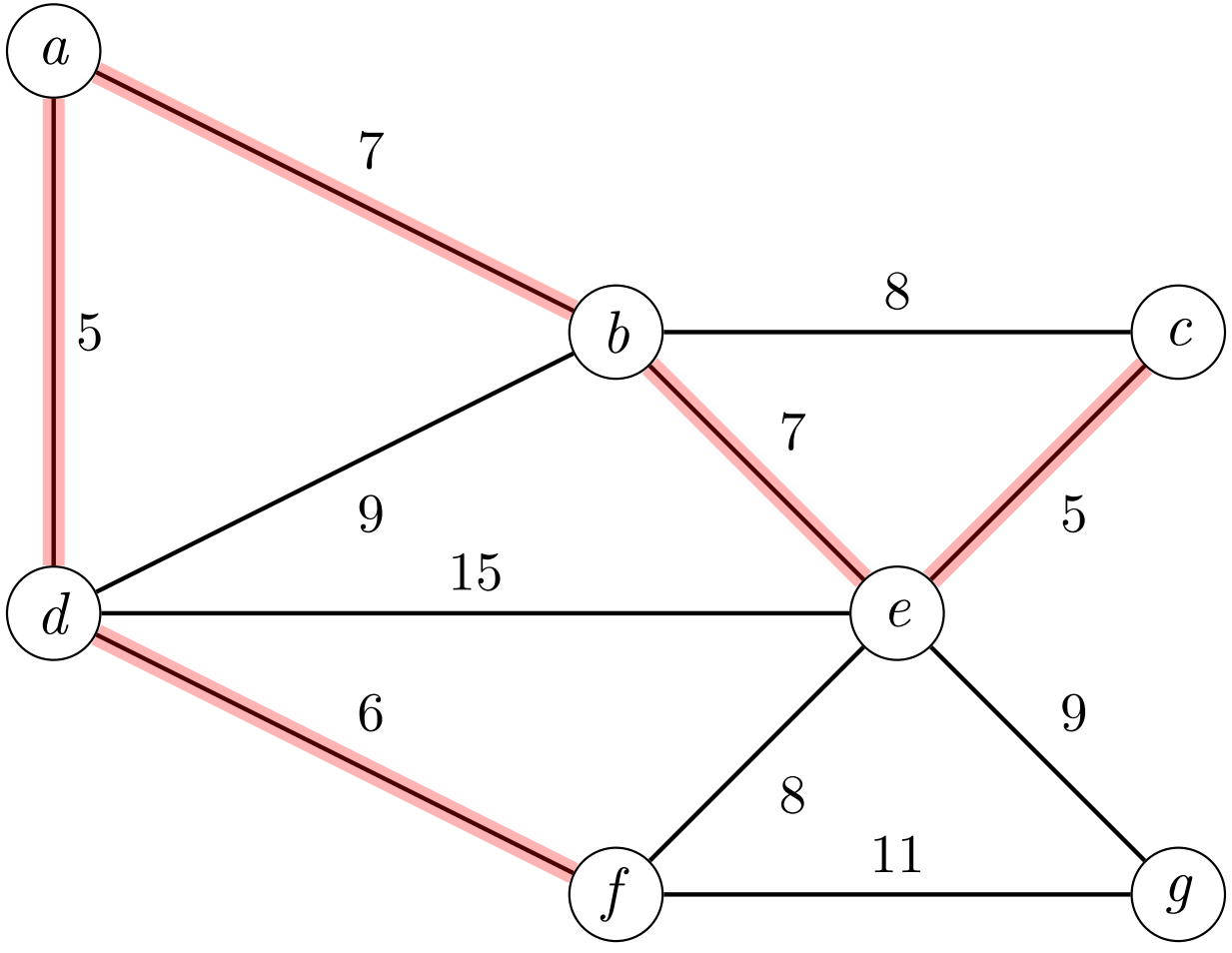
Arbre couvrant de poids minimal : Kruskal



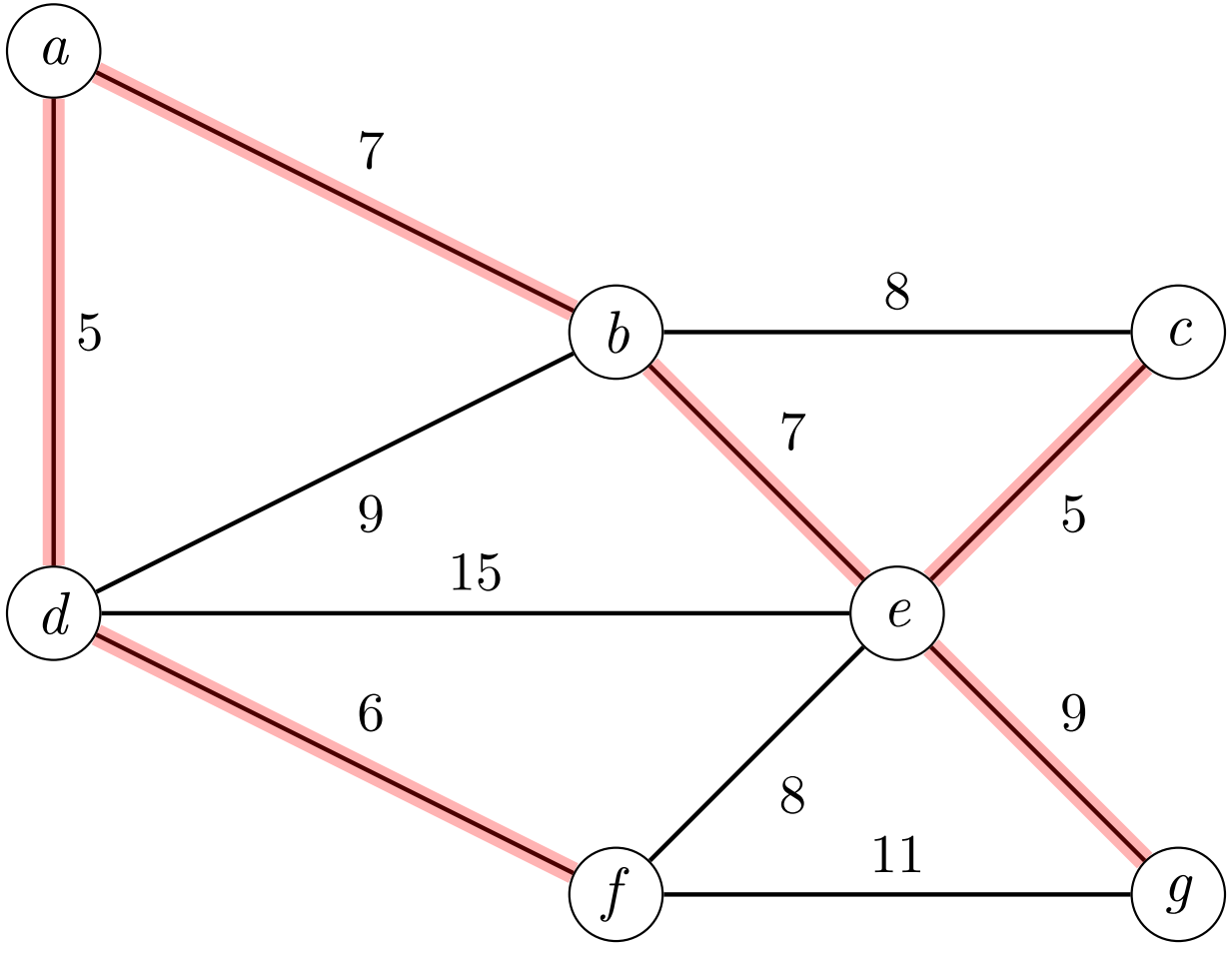
Arbre couvrant de poids minimal : Kruskal



Arbre couvrant de poids minimal : Kruskal



Arbre couvrant de poids minimal : Kruskal



Arbre couvrant de poids minimal : 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.

Arbre couvrant de poids minimal : Kruskal

T est un arbre couvrant :

1. T est sans cycle :

Arbre couvrant de poids minimal : Kruskal

T est un arbre couvrant :

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

Arbre couvrant de poids minimal : Kruskal

T est un arbre couvrant :

1. T est sans cycle : car l'algorithme ne 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.

Arbre couvrant de poids minimal : Kruskal

T est un arbre couvrant :

1. T est sans cycle : car l'algorithme ne 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 $V \setminus U$.

Arbre couvrant de poids minimal : Kruskal

T est un arbre couvrant :

1. T est sans cycle : car l'algorithme ne 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.

Arbre couvrant de poids minimal : Kruskal

Théorème

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

Preuve :

Arbre couvrant de poids minimal : 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.

Arbre couvrant de poids minimal : 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.

Arbre couvrant de poids minimal : 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^*.

Arbre couvrant de poids minimal : 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^*

Arbre couvrant de poids minimal : 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)

Arbre couvrant de poids minimal : Kruskal

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

Arbre couvrant de poids minimal : Kruskal

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

- T_2 est un arbre couvrant

Arbre couvrant de poids minimal : Kruskal

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

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

Arbre couvrant de poids minimal : 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**.

Arbre couvrant de poids minimal : Kruskal

On va avoir besoin d'un graphe pondéré :

```
type ('v, 'w) graph = {
    n : int; (* nombre de sommets *)
    add_edge : 'v -> 'v -> 'w -> unit;
    del_edge : 'v -> 'v -> unit;
    edges : unit -> 'v list; (* toutes les arêtes *)
    w : 'v -> 'v -> 'w option; (* poids d'une arête *)
    adj : 'v -> 'v list (* liste des sommets adjacents *)
}
```

Arbre couvrant de poids minimal : Kruskal

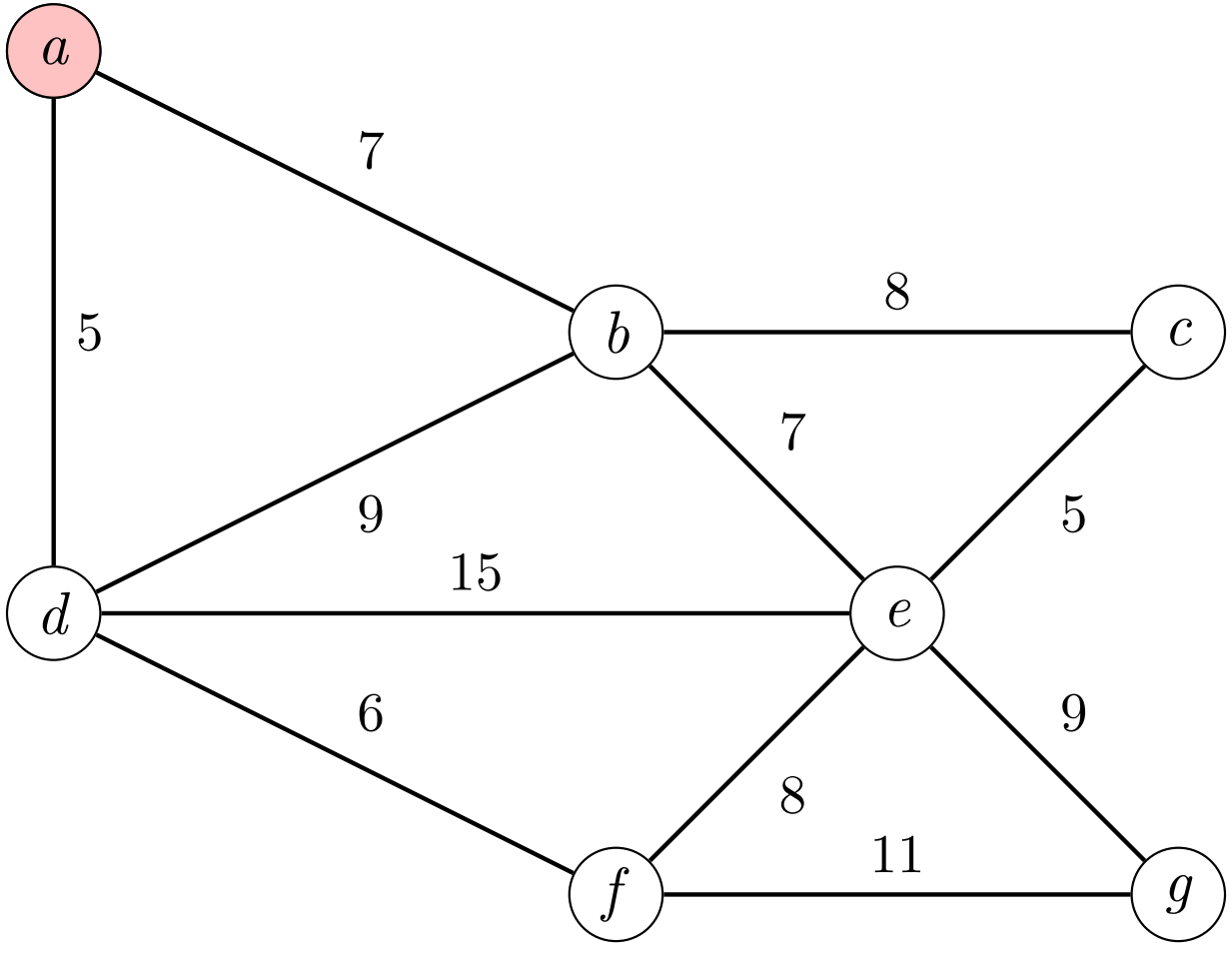
```
let kruskal g =
```

Prim

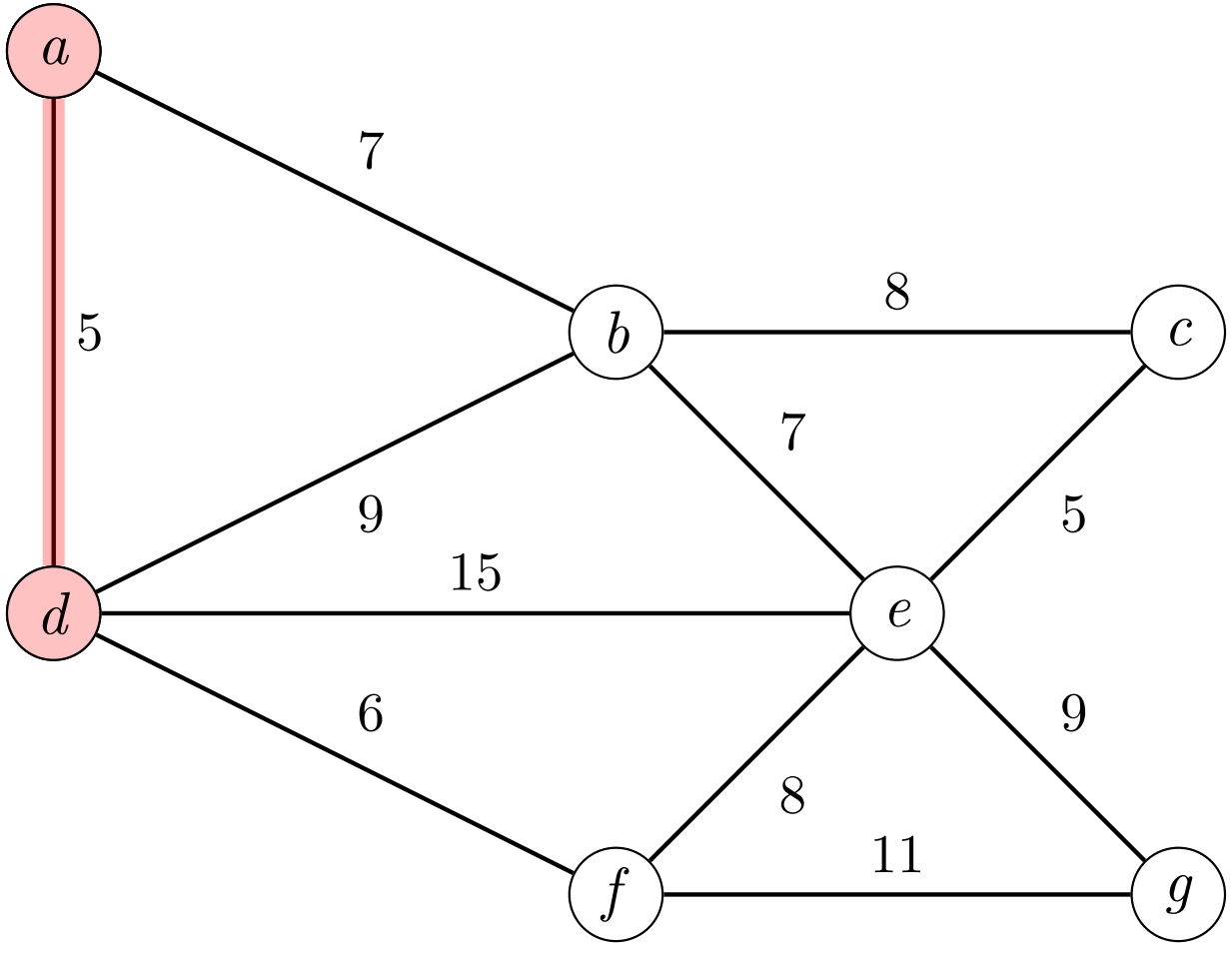
Commencer avec un arbre T contenant un seul sommet.

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

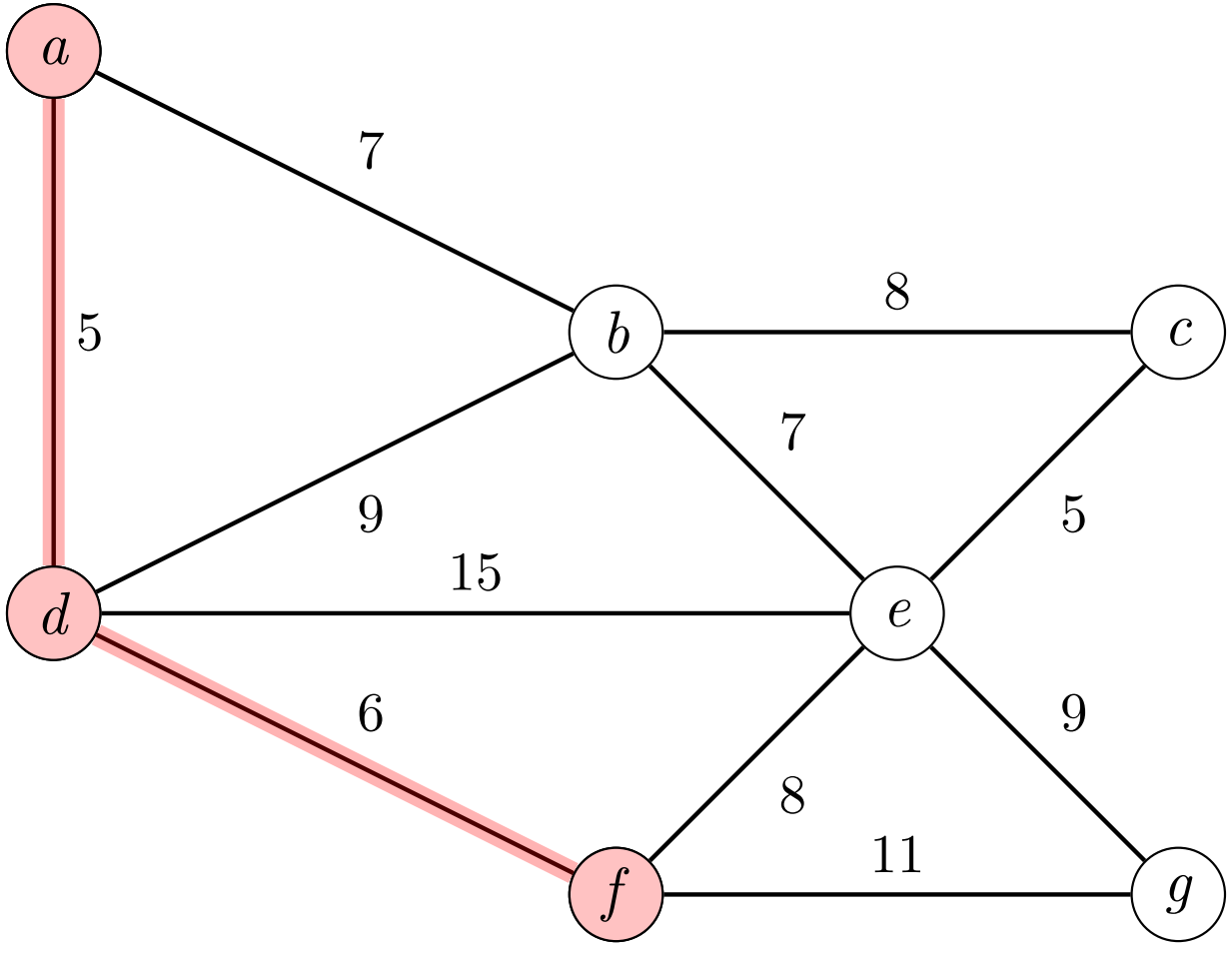
Prim



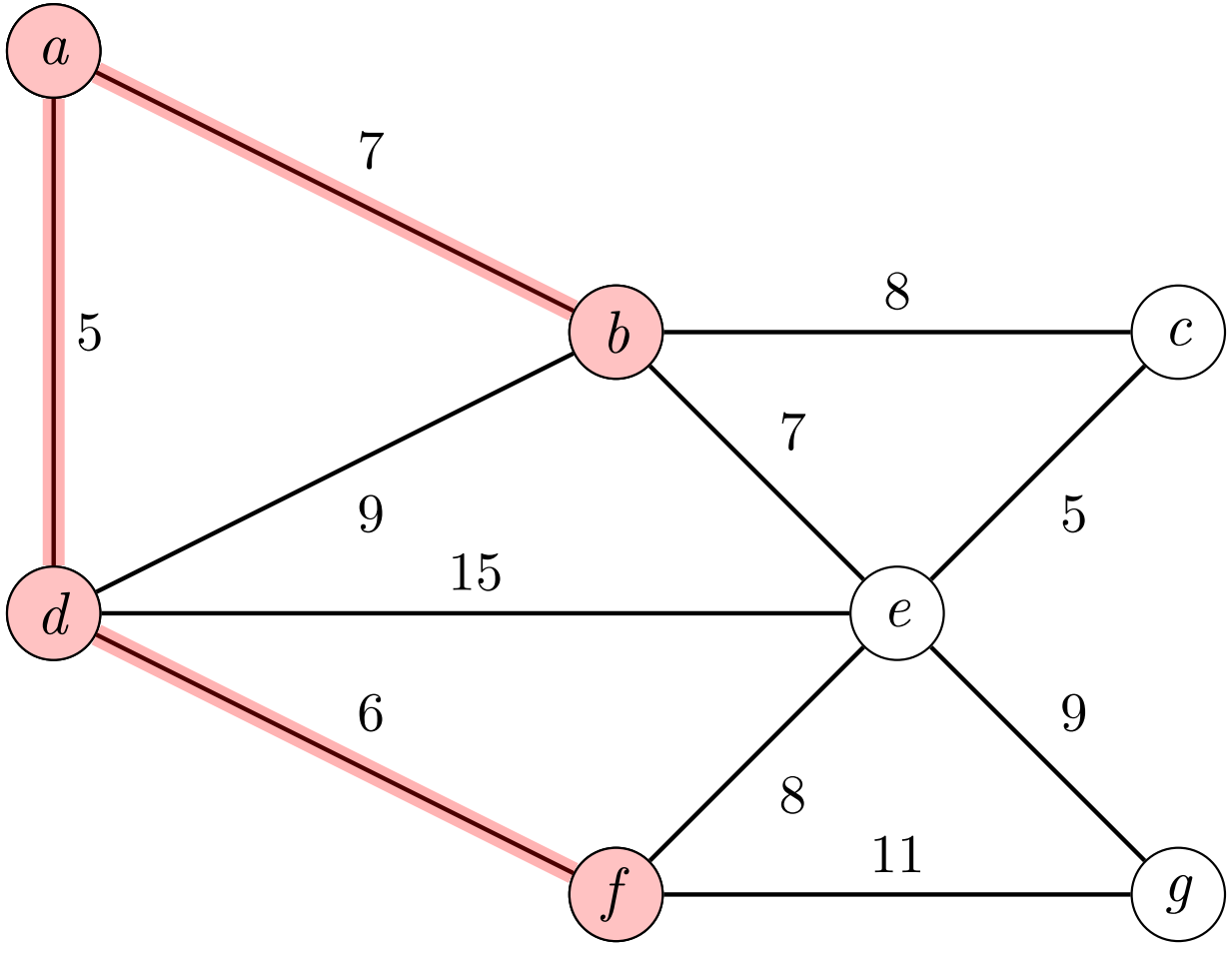
Prim



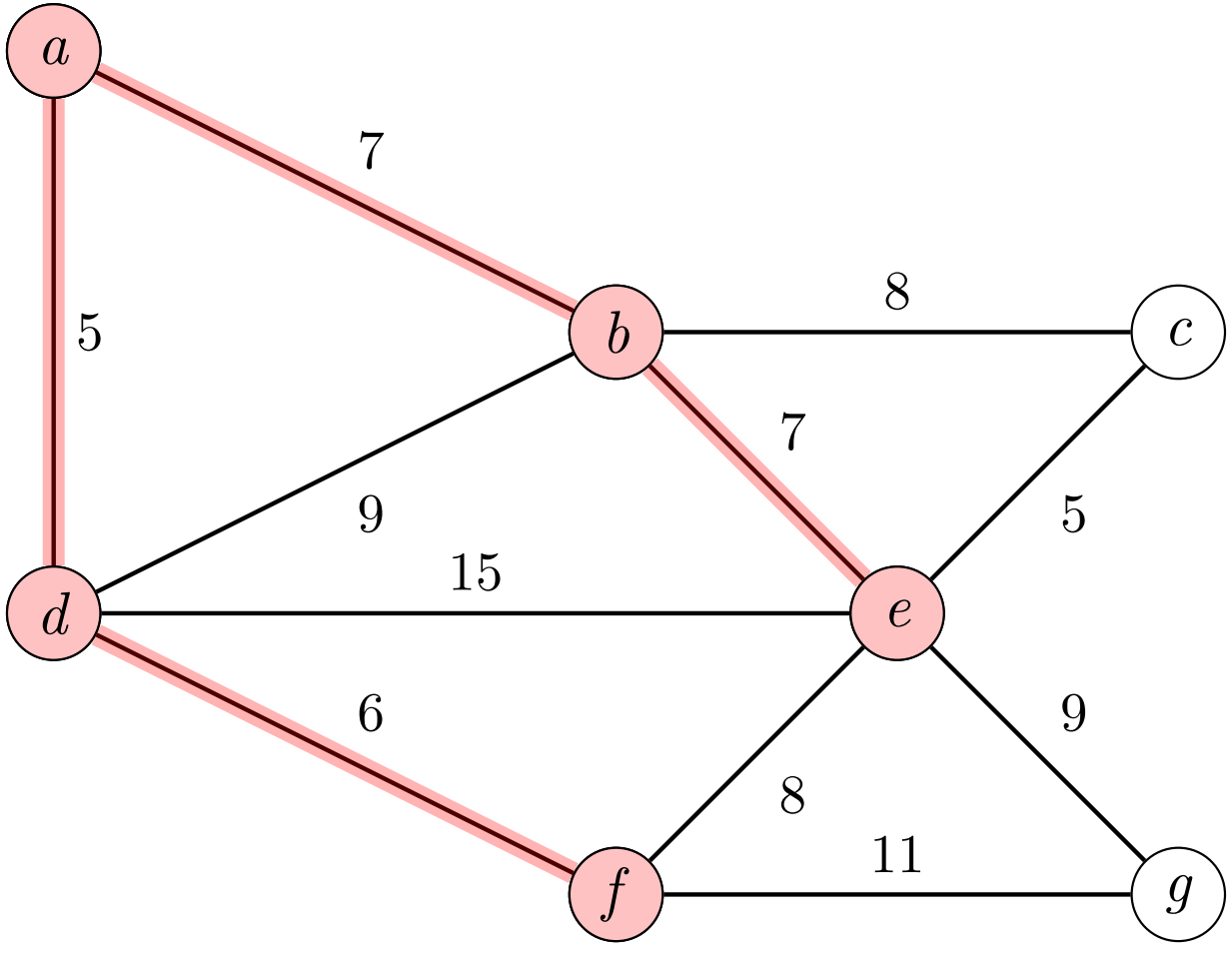
Prim



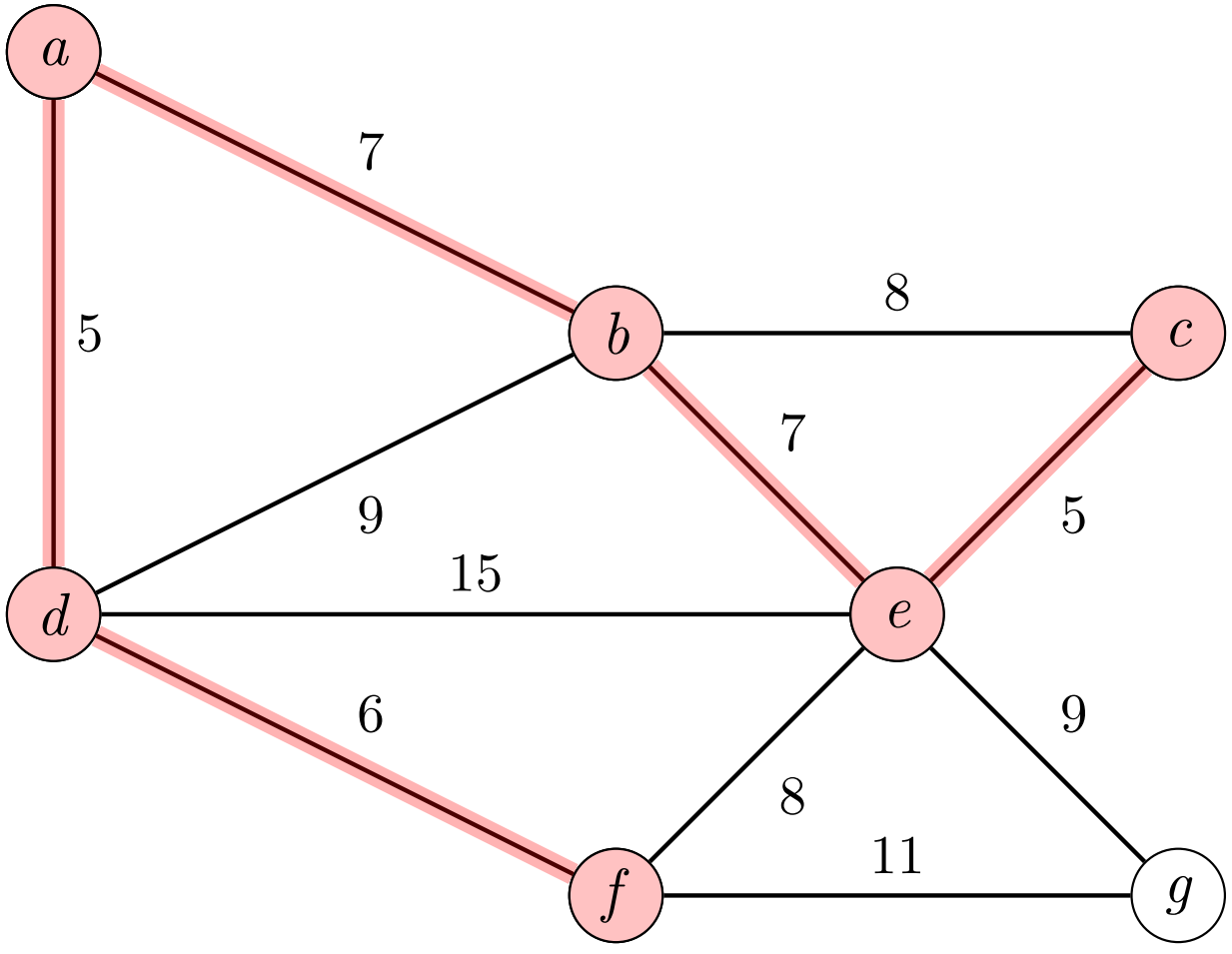
Prim



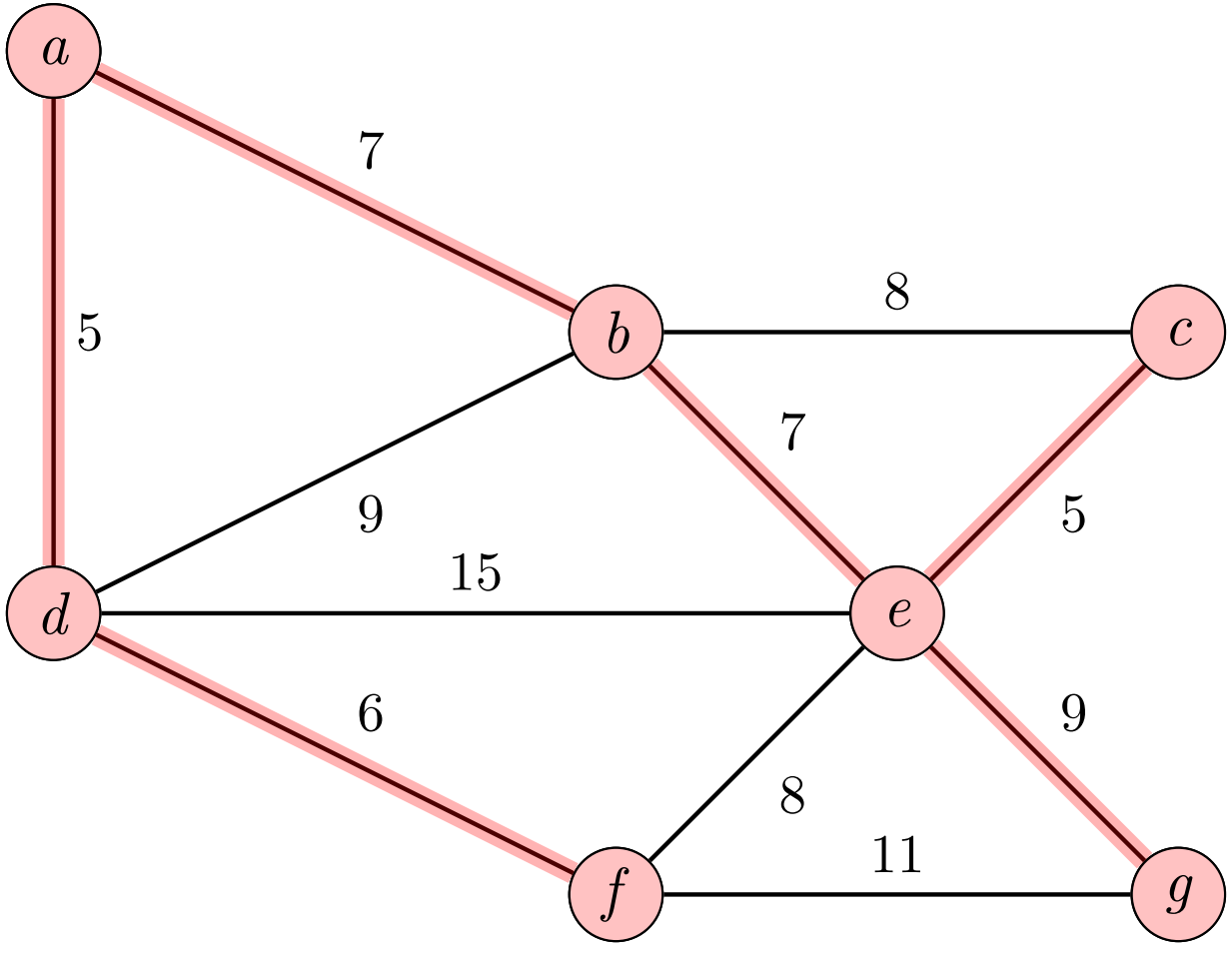
Prim



Prim



Prim



Prim

Théorème

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

Preuve :

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