Chapitre 4 : Arbre couvrant à poids minimal

Cours Graphes & Applications

Unité Pédagogiques de Mathématiques,
Ecole Supérieure Privée d'Ingénierie et de Technologies (ESPRIT)

Année Universitaire : 2024-2025

Plan

Objectifs du chapitre

A l'issu du chapitre 4, l'étudiant sera capable de :

- Identifier des problèmes dont les solutions sont données via la détermination d'un arbre couvrant à poids minimal / maximal.
- Modéliser des situations réelles par des graphes.
- Connaitre les propriétés et définitions sur les arbres / arbres couvrants / co-cycles...
- Application de l'algorithme de Kruskal.
- Application de l'algorithme de Prim.

Plan

Historique

- C'est un problème qui est plus vieux que l'informatique elle même. En effet, en 1926, Otakar était le premier à fournir un algorithme pour trouver efficacement un arbre couvrant de poids minimum. La motivation au début était de concevoir un réseau de distribution électrique pour la Moravie du Sud (République tchèque) avec la contrainte que cela coûte le moins possible (pour plus de détails voir `https://w.wiki/BfN7`).
- Ce n'est qu'au milieu des années 50 et l'avènement de l'informatique que ce problème est revenu à lamode.C'est à Joseph B.Kruskal (1956) et à Robert Prim (1957) que nous devons les algorithmes que nous allons étudier(ce sont les plus connus).
- Les deux algorithmes(Kruskal et Prim) sont des algorithmes gloutons.

Applications

- Application1: Ce problème se pose, par exemple, lorsqu'on désirere lier n villes par un réseau routier de coût minimum. Les sommets du graphe représentent les villes, les arêtes sont les tronçons qu'il est possible de construire et les poids des arêtes correspondent aux coûts de construction du tronçon correspondant.
- Application2: Imaginons que dans un graphe chaque nœud soit un ordinateur et que le poids de l'arête entre x et y indique la longueur de câble nécessaire pour relier les ordinateurs x et y. Alors, l'arbre couvrant minimal donne la plus courte longueur de câble nécessaire pour relier tous les ordinateurs.

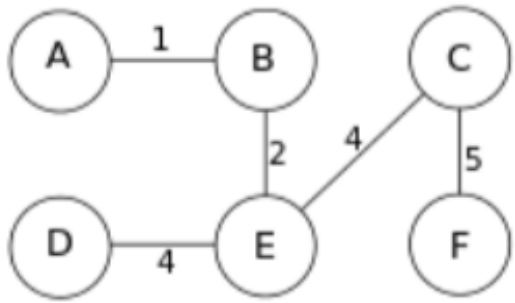
Plan

Définition et propriétés

Dans la suite de ce chapitre, on va travailler avec des graphes non orientés et valués.

Définition

- Un arbre T est un graphe tel que: T est connexe et T ne contient pas de cycles.

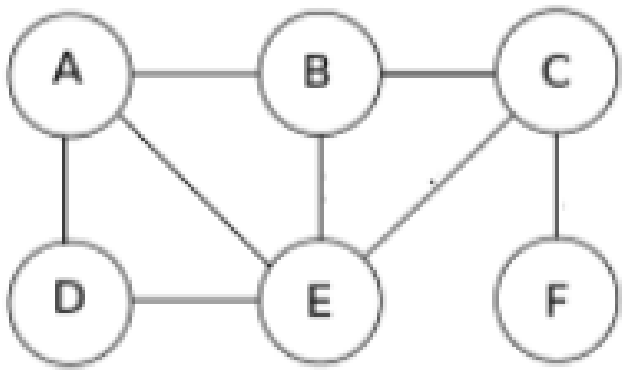


- Un arbre de n sommets est composé de $n-1$ arêtes.

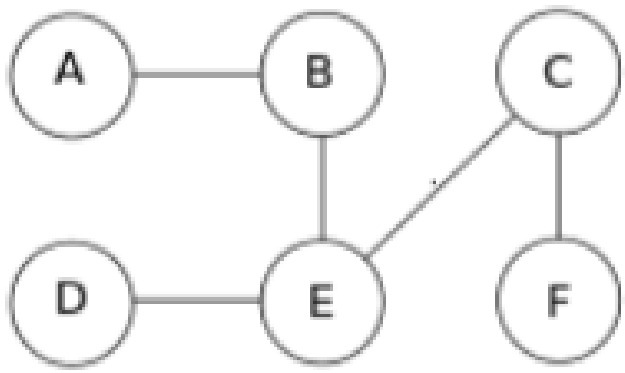
Rappels

- Un graphe $G = (V, E)$ avec $|V| = n$, est dit connexe si pour tout couple de noeud (v_i, v_j) il existe un chemin reliant v_i et $v_j, 1 \leq i, j \leq n$.
- Un graphe partiel $G' = (V, E')$ de G est un graphe qui a le même ensemble de sommets que G et l'ensemble des arêtes $E' \subseteq E$.

Exemple:



G



G'

Arbre couvrant

- Un arbre couvrant d'un graphe $G = (V, E)$ est un graphe partiel de G : $T = (V, E')$ avec $E' \subseteq E$ qui est connexe et sans cycle.

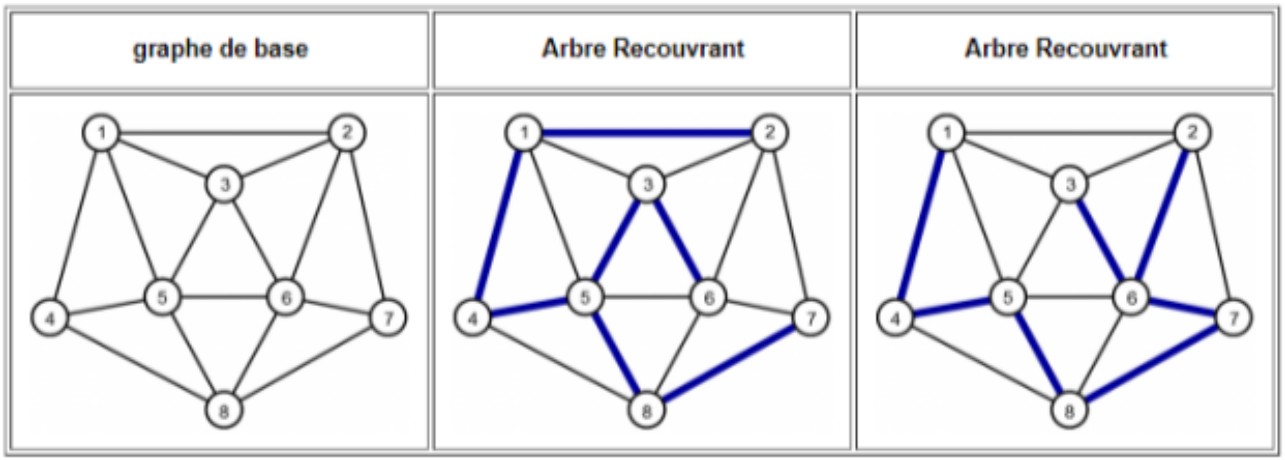
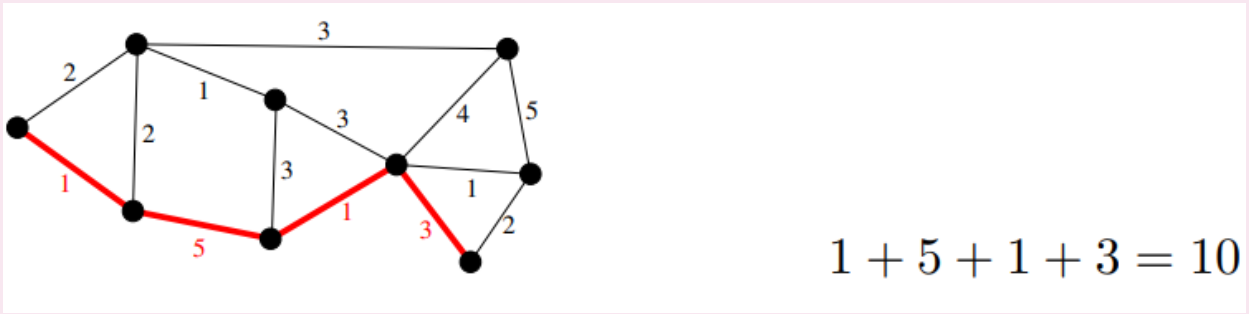


Figure : Un graphe connexe non valué et deux arbres recouvrants possibles.

Arbre couvrant à poids minimum

Poids d'un graphe

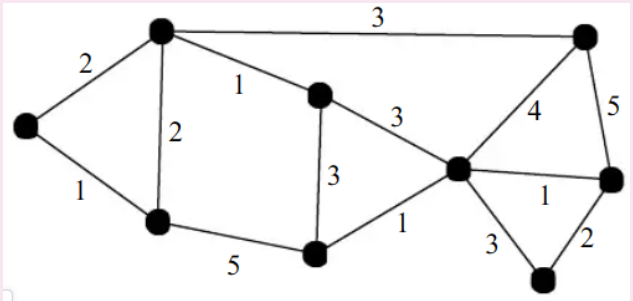
Le poids d'un arbre est la somme des poids des arêtes qui le composent.



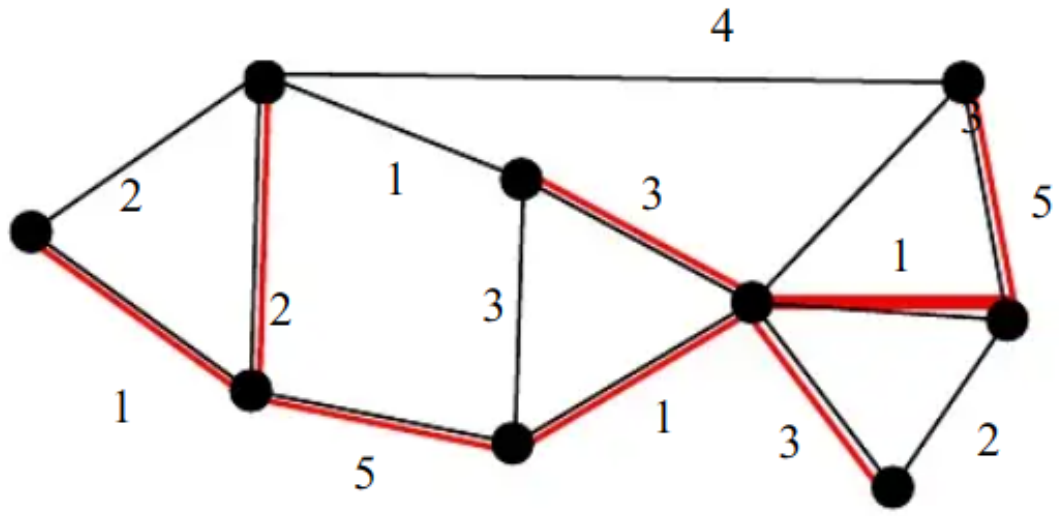
$$1 + 5 + 1 + 3 = 10$$

Question

Étant donné un graphe valué connexe, on veut construire un arbre couvrant de poids total le plus petit possible.



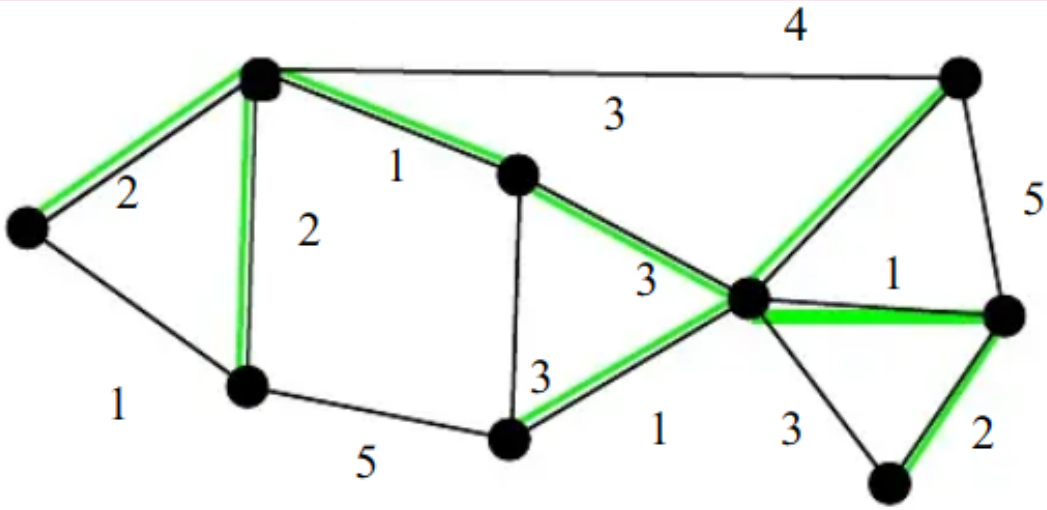
Solution 1:



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

Peut-on faire mieux ?

Solution 2:



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

Peut-on faire mieux ?

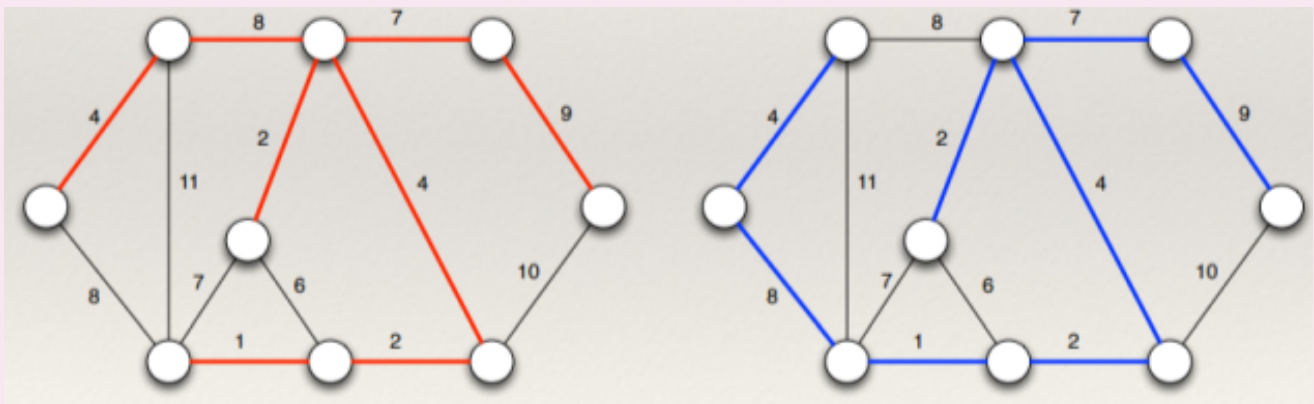
Arbre couvrant à poids minimum

Soit G un graphe valué non orienté connexe:

- Un Arbre Couvrant de poids Minimum (ACM) (en anglais Minimum Spanning Tree (MST)) est un arbre couvrant dont le poids est le plus petit possible parmi les arbres couvrants de G.

Propriété

Tout graphe non orienté, valué et connexe admet un ou plusieurs ACM.



Propriétés

Soit G un graphe à n sommets et m arêtes.
Les propriétés suivantes sont équivalentes:

- G connexe sans cycles.
- G connexe et $m = n - 1$.
- Entre 2 sommets il existe un unique chemin.
- Un graphe non connexe n'a aucun arbre recouvrant.
- Un graphe connexe a forcément (au moins) un arbre couvrant (par exemple un arbre de parcours).

Plan

Algorithme de KrusKal

Principe

L'algorithme de KrusKal consiste à d'abord ranger par ordre de poids croissant les arêtes d'un graphe, puis à retirer une à une les arêtes selon cet ordre et à les ajouter à l'ACM cherché tant que cet ajout ne fait pas apparaître un cycle dans l'ACM.

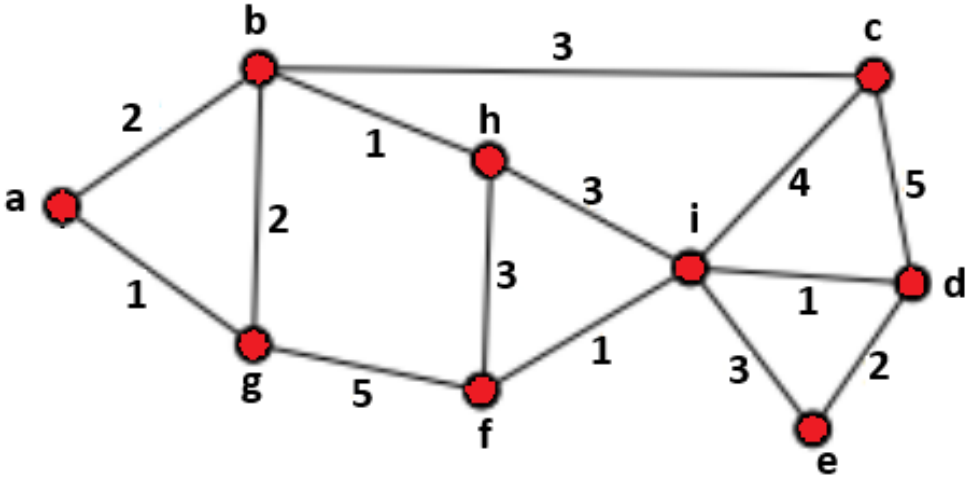
¿¿ L'algorithme s'arrête quand on aura incorporé $n - 1$ arêtes.

Algorithme de KrusKal

Méthode

- Initialiser T avec $T = \emptyset$
- On commence par une arête de valeur la plus faible.
- 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 dès que T est connexe.
- Retourner T.

Exemple: Algorithme de Kruskal

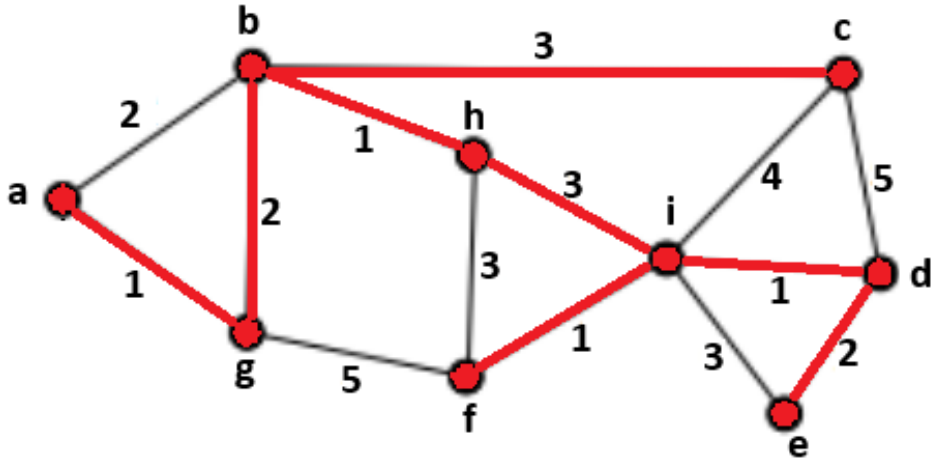


Pour trouver l'arbre couvrant à poids minimal, nous pouvons utiliser l'algorithme de Kruskal comme suit:

1. Trier les arêtes par poids croissant.
2. Ajouter les arêtes les moins coûteuses tout en évitant les cycles.
3. Vérifier que tous les sommets sont connectés et qu'il n'y a pas de cycles.

Exemple: Algorithme de Kruskal

Arêtes triées par ordre croissant	Poids	Etat (je prends ou pas)
a-g	1	Oui
b-h	1	Oui
f-i	1	Oui
i-d	1	Oui
g-b	2	Non
a-b	2	Oui
e-d	2	Oui
b-c	3	Oui
h-f	3	Non
i-e	3	Non
h-i	3	Oui
i-c	4	Non
g-f	5	Non
c-d	5	Non



L'arbre couvrant à poids minimal pour ce graphe est constitué des arêtes : a-g, b-h, f-i, i-d, a-b, e-d et b-c, avec un poids total de $1 + 1 + 1 + 1 + 2 + 2 + 3 + 3 + 3 = 14$.

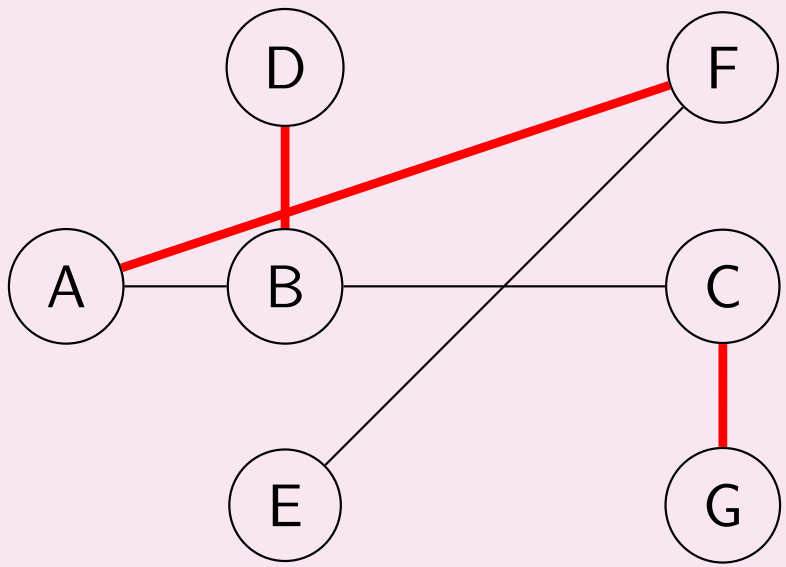
Plan

Algorithme de Prim

Définition d'un co-cycle

Soit $G = (V, E)$ un graphe simple non-orienté. Un co-cycle d'un ensemble $S \subset V$ de sommets de V est l'ensemble des arêtes (u, v) incidentes aux sommets de S telles que $u \in S$ et $v \in V/S$.

Exemple : Le sous-ensemble des arêtes $\{(A, F), (B, D), (C, G)\} \subset E$ est un co-cycle du sous-ensemble $S = \{A, B, C\}$ de V.



Algorithme de Prim

Principe

Le principe de l'algorithme de Prim est de partir d'un arbre initial réduit à un seul sommet, puis d'augmenter à chaque itération la taille de l'arbre en le connectant au plus proche voisin libre.

Pseudo-code de l'algorithme de Prim

Pseudo-code

Soit $G = (V, E)$ un graphe simple non-orienté avec $card(V) = n$ et $card(E) = m$. On désigne par $\mathcal{CC}(V_T)$ le co-cycle de V_T.

Objectif : On chercher à construire l'arbre couvrant $T = (V_T, E_T)$ avec $V_T = V$ et $E_T \subseteq E$.

Initialisation :

- $T = (V_T, E_T)$ avec $V_T = \{v_1\}$ et $E_T = \emptyset$

Procedure itérative :

- Tant que $|V_T| < |V|$ faire
  - Déterminer $e = (u, v) \in \mathcal{CC}(V_T)$ de poids minimal ($u \in V_T$ et $v \in V/V_T$).
  - $V_T \leftarrow V_T \cup v$: ajouter v à V_T
  - $E_T \leftarrow E_T \cup e$: ajouter e à E_T

Algorithme de Prim

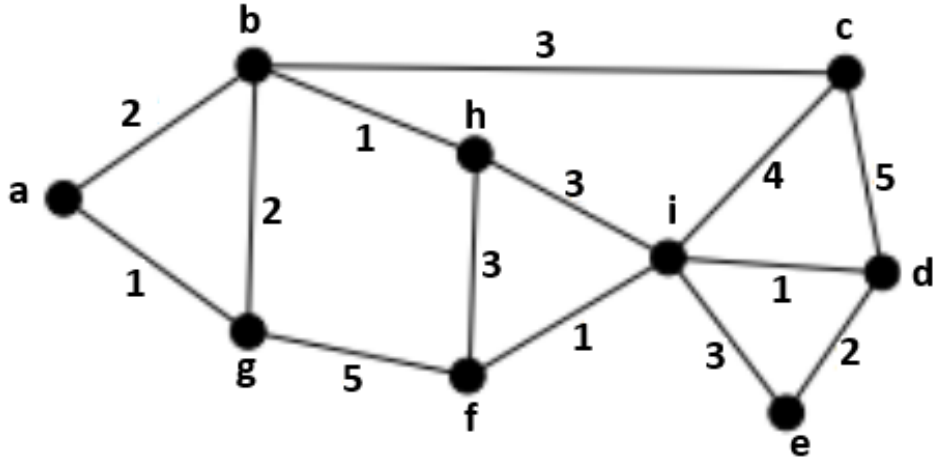
Méthode

- Initialiser T avec

$$\begin{cases} \textit{sommets : un sommet de } G \textit{ qu'on choisit}, \\ \textit{arêtes : aucune}. \end{cases}$$

- 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: Algorithme de Prim



Entrée de l'algorithme : $G = (V, E)$ un graphe simple non-orienté avec $card(V) = 9$ et $card(E) = 14$.

Sortie de l'algorithme : $T = (V_T, E_T)$ un arbre couvrant à poids minimal tels que $V_T = V$ et $E_T \subseteq E$ telque $card(E_T) = 8$.

Initialisation de l'algorithme : $T = (V_T, E_T)$ avec $V_T = \{a\}$ (choix quelconque) et $E_T = \emptyset$

Nombre d'itérations : exactement 8

Exemple: Algorithme de Prim

It	V_T	$CC(V_T)$	E_T
1	$\{a\}$	$\{(a,b)2; (a,g)1\}$	$\{(a,g)\}$
2	$\{a,g\}$	$\{(a,b)2; (g,b)2; (g,f)5\}$	$\{(a,g); (g,b)\}$
3	$\{a,g,b\}$	$\{(b,c)3; (b,h)1; (g,f)5\}$	$\{(a,g); (g,b); (b,h)\}$
4	$\{a,g,b,h\}$	$\{(b,c)3; (h,i)3; (h,f)3; (g,f)5\}$	$\{(a,g); (g,b); (b,h); (h,f)\}$
5	$\{a,g,b,h,f\}$	$\{(b,c)3; (h,i)3; (f,i)1\}$	$\{(a,g); (g,b); (b,h); (h,f); (f,i)\}$
6	$\{a,g,b,h,f,i\}$	$\{(b,c)3; (i,c)4; (i,d)1; (i,e)3\}$	$\{(a,g); (g,b); (b,h); (h,f); (f,i); (i,d)\}$
7	$\{a,g,b,h,f,i,d\}$	$\{(b,c)3; (i,c)4; (i,e)3; (d,e)2; (d,c)5\}$	$\{(a,g); (g,b); (b,h); (h,f); (f,i); (i,d); (d,e)\}$
8	$\{a,g,b,h,f,i,d,e\}$	$\{(b,c)3; (i,c)4; (d,c)5\}$	$\{(a,g); (g,b); (b,h); (h,f); (f,i); (i,d); (d,e); (b,c)\}$

Exemple: Algorithme de Prim