Introduction aux graphes

Arbres

Mathilde Vernet
mathilde.vernet@univ-avignon.fr

Licence Informatique, CERI
Licence Mathématiques, Institut AgES
Avignon Université

Automne 2025

On considère dans tout ce chapitre des graphes simples non-orientés

Plan

1. Un graphe particulier
   - Définitions
   - Propriétés
2. Arbre couvrant
   - Définitions et premier algorithme
   - Arbre couvrant de poids minimum
3. Arbres et Parcours

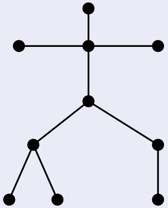
Définition : Arbre

Un graphe non orienté $G = (V, E)$ est un arbre si et seulement si G est connexe et acyclique

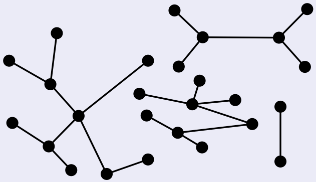
Définition : Forêt

Une forêt est un graphe non connexe dont chaque composante connexe est un arbre

Exemple : Un arbre



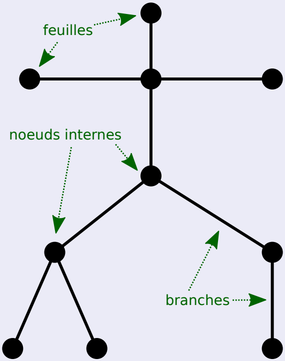
Exemple : Une forêt



Vocabulaire dans les arbres

- Un nœud est un sommet de l'arbre
- Une feuille est un sommet de degré 1
- Un nœud interne est un sommet de degré > 1
- Une branche est une arête de l'arbre

Exemple



Quelques propriétés

- Dans un arbre, $m = n - 1$
- Un arbre est un graphe biparti

Pourquoi un arbre possède-t-il n-1 arêtes ?

1. *Lemme : Si un graphe est connexe alors il possède au moins n-1 arêtes*
2. *Lemme : Si un graphe n'a pas de cycles alors il possède au plus n-1 arêtes*

Un arbre étant connexe et sans cycle, en appliquant 1 et 2, on a qu'un arbre possède $n - 1$ arêtes.

Pourquoi un arbre est-il biparti ?

On sait que :

- Un graphe est biparti si et seulement si il ne contient pas de cycle de longueur impaire
- Un arbre ne possède pas de cycles

Puisqu'un arbre n'a pas de cycles, alors il n'a pas de cycles de longueur impaire, donc il est biparti.

Caractérisation des arbres

Les propriétés suivantes sont équivalentes et caractérisent un arbre :

1. G est connexe et acyclique
2. G est connexe et possède $n - 1$ arêtes
3. G est acyclique et possède $n - 1$ arêtes
4. G est connexe et minimal au sens des arêtes pour cette propriété (retirer une arête déconnecte G)
5. G est acyclique et maximal au sens des arêtes pour cette propriété (ajouter une arête dans G crée un cycle)
6. Dans G il existe une unique chaîne entre chaque paire de sommets

Autres propriétés

- L'ajout d'une arête à un arbre crée exactement un cycle
- Dans un arbre d'au moins 2 sommets, il existe au moins 2 sommets de degré 1

Pourquoi ajouter une arête crée un unique cycle ?

- Un arbre est acyclique et maximal au sens des arêtes pour cette propriété donc ajouter une arête crée un cycle
- Dans un arbre il existe une unique chaîne entre chaque paire de sommets, donc ajouter une arête ajoutera un unique cycle

Pourquoi un arbre possède-t-il au moins deux feuilles ?

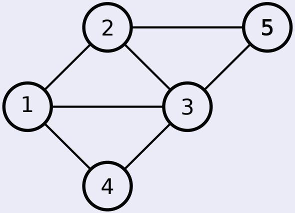
- On sait que $\sum d(u) = 2 \cdot m$ donc dans un arbre on a $\sum d(u) = 2 \cdot (n-1)$
  Si tous les sommets sont de degré au moins 2 alors $\sum d(u) \geqslant 2 \cdot n$
  Impossible donc il existe au moins un sommet de degré 1
- S'il y a exactement un sommet de degré 1 alors $\sum d(u) \geqslant 2 \cdot (n-1) + 1$ (car les autres sommets sont de degré au moins 2)
  Mais $\sum d(u) = 2 \cdot (n-1)$
  Donc c'est impossible et il y a au moins deux sommets de degré 1

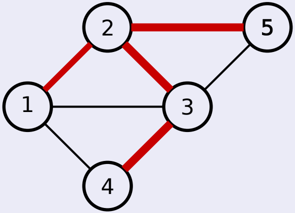
Définition : Arbre couvrant

Un arbre couvrant d'un graphe non-orienté connexe G est un sous-graphe de G, de même ordre que G, acyclique maximal

En d'autres termes, un arbre couvrant d'un graphe G est un arbre inclus dans G qui connecte tous les sommet du graphe

Exemple





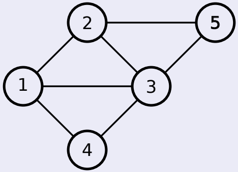
Remarques

- Un graphe connexe quelconque possède plusieurs arbres couvrants
- Un arbre possède un unique arbre couvrant

Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe
        que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

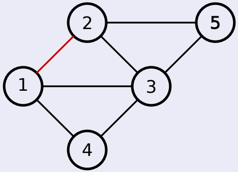
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

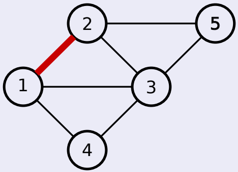
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

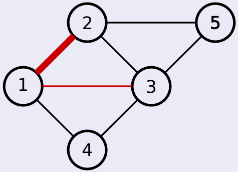
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe
        que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

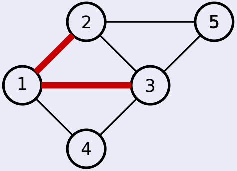
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

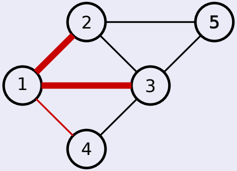
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

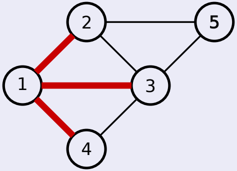
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe
        que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

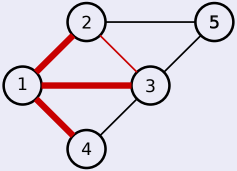
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

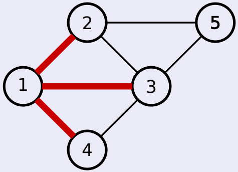
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

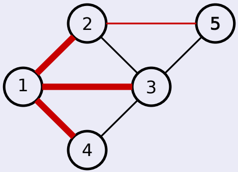
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

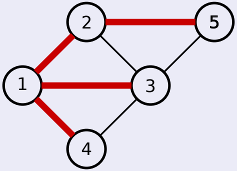
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

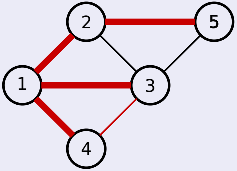
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

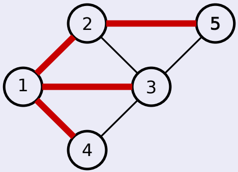
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

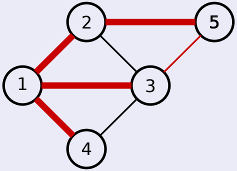
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

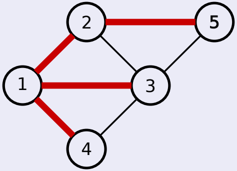
Exemple



Comment trouver un arbre couvrant de $G = (V, E)$?

Procedure `arbreCouvrant`(G)
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

Exemple

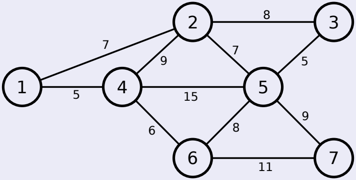


Définition : Graphe Pondéré

On dit qu'un graphe $G = (V, E)$ est pondéré s'il possède une fonction de pondération

$$w : E \longrightarrow \mathbb{R}$$

Exemple



Remarques

- On peut aussi considérer une fonction de pondération sur les sommets du graphe :

$$w : V \longrightarrow \mathbb{R}$$

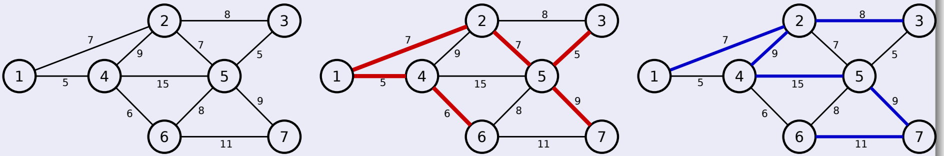
- Un graphe pondéré peut être multiple, orienté,...
- Un même graphe peut avoir plusieurs fonctions de pondération

Définition : Poids d'un arbre couvrant

Soit $G = (V, E)$ un graphe muni d'une fonction de pondération $w : E \longrightarrow \mathbb{R}^+$.
Le poids de $G' = (V, E')$, un arbre couvrant de G est :

$$\sum_{e \in E'} w(e)$$

Exemple



Arbre rouge de poids 39. Arbre Bleu de poids 59.

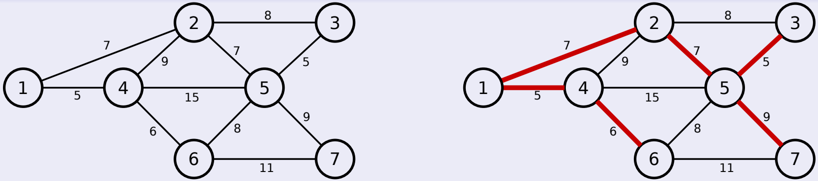
Remarques

- Il peut y avoir plusieurs arbres couvrants de même poids
- Si toutes les arêtes ont le même poids, tous les arbres couvrants ont le même poids
- Un arbre possède un unique arbre couvrant

Problème de l'arbre couvrant de poids minimum

Sachant un graphe G pondéré sur les arêtes, trouver l'arbre couvrant de poids min

Exemple



L'arbre de poids minimum de ce graphe est de poids 39

Remarques

- Le problème de l'arbre couvrant de poids minimum peut avoir plusieurs solutions dans un graphe donné
  - *Puisqu'il peut exister plusieurs arbres couvrants de même poids, il peut aussi exister plusieurs arbres couvrants de poids minimum*
- Le problème de l'arbre couvrant de poids minimum a une unique solution dans un arbre
  - *Un arbre possède un unique arbre couvrant qui est donc la solution au problème de l'arbre couvrant de poids minimum*

Comment trouver un arbre couvrant de poids minimum ?

- Algorithme de Kruskal
  - En 1956 par Kruskal
- Algorithme de Prim
  - En 1930 par Jarník
  - En 1959 par Prim et Dijkstra

Idée de l'algorithme de Kruskal

- Trier les arêtes selon leur poids
- Appliquer l'algorithme classique pour trouver un arbre couvrant en considérant les arêtes par ordre croissant des poids

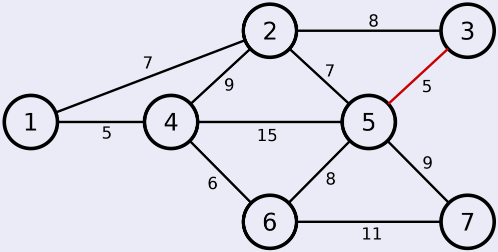
Idée de l'algorithme de Prim

- Commencer avec un arbre contenant un sommet du graphe
- Tant qu'on n'a pas couvert tous les sommets : parmi toutes les arêtes sortant de l'arbre en cours de construction, ajouter celle de poids minimum

Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

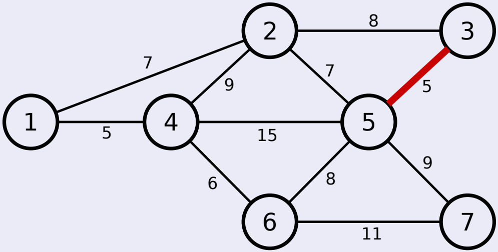
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

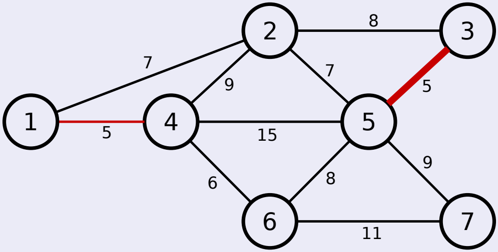
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

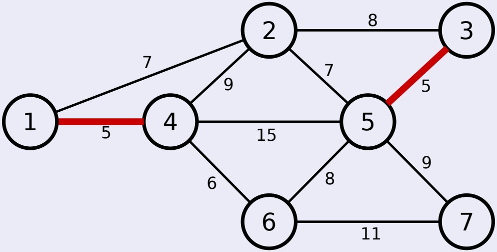
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

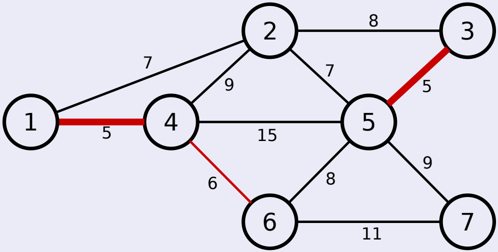
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

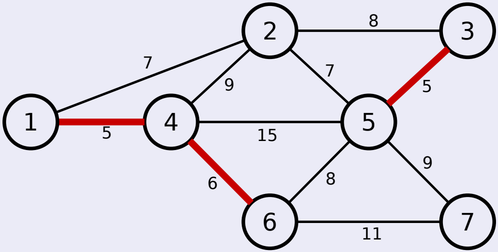
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

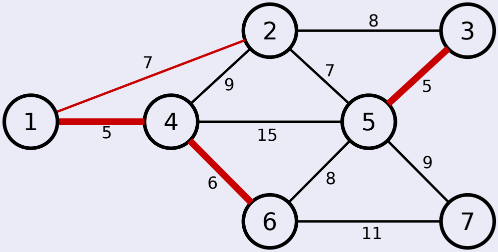
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

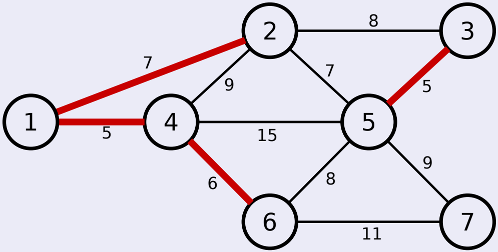
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

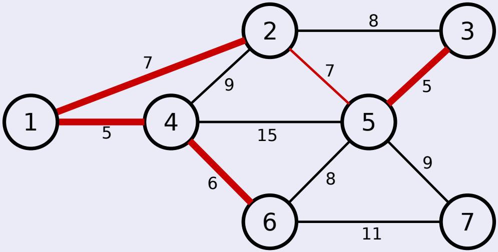
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

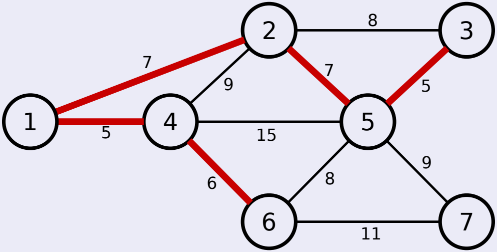
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

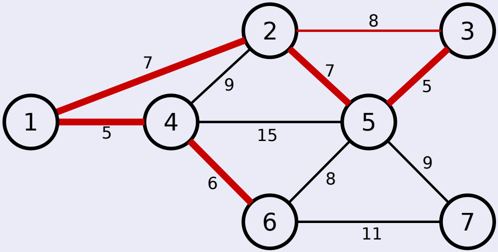
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

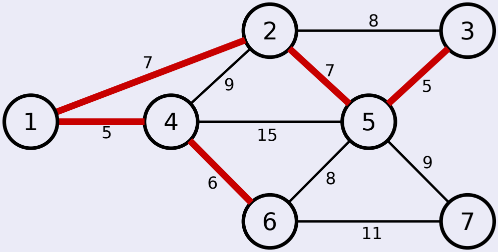
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

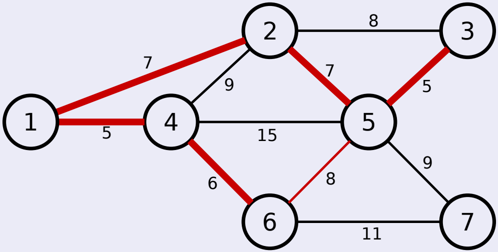
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

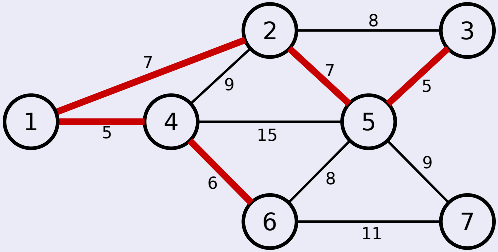
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

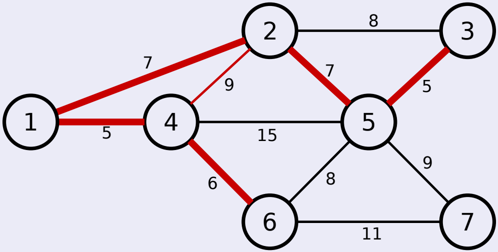
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

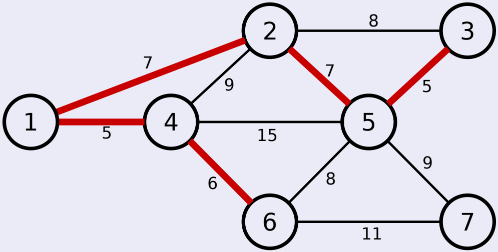
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

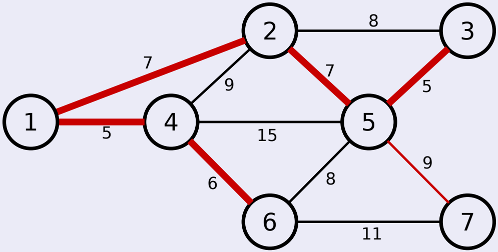
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

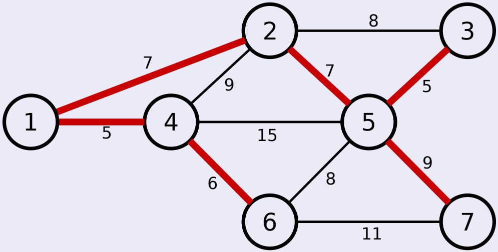
Exemple



Algorithme de Kruskal

Procedure `Kruskal`(G)
  Trier les arêtes dans E par ordre croissant des poids
  Soit $E' = \emptyset$
  Pour $(u, v) \in E$ par ordre croissant des poids
    Si $G' = (V, E' \cup (u, v))$ ne contient pas de cycle **Alors**
      $E' \leftarrow E' \cup (u, v)$
      //Si l'arête (u, v) ne crée par un cycle dans le graphe que l'on est en train de créer, on la garde
    FinSi
  FinPour
  Retourner $G' = (V, E')$
FinProcedure

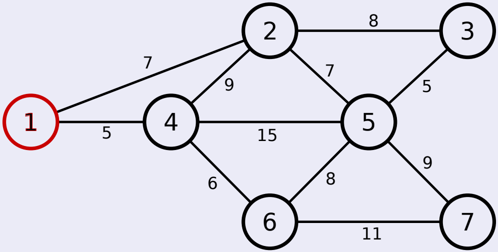
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

Exemple



Algorithme de Prim

Procedure `Prim`(G)

  Soit $E' = \emptyset$

  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*

  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*

    Soit (u, v) l'arête de poids minimum dans $\delta(V')$

    //u est dans V', v n'est pas dans V',

      (u, v) est l'arête sortante de V' dont le poids est minimum

    $E' \leftarrow E' \cup (u, v)$

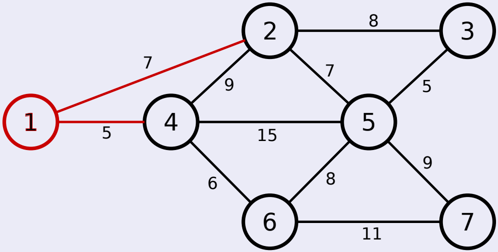
    $V' \leftarrow V' \cup v$

  FinTantQue

  Retourner $G' = (V', E')$

FinProcedure

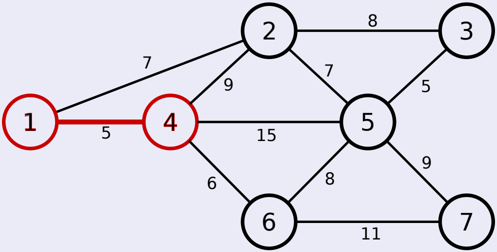
Exemple



Algorithme de Prim

Procedure `Prim`(G)
Soit $E' = \emptyset$
Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
Soit (u, v) l'arête de poids minimum dans $\delta(V')$
//u est dans V', v n'est pas dans V',
(u, v) est l'arête sortante de V' dont le poids est minimum
$E' \leftarrow E' \cup (u, v)$
$V' \leftarrow V' \cup v$
FinTantQue
Retourner $G' = (V', E')$
FinProcedure

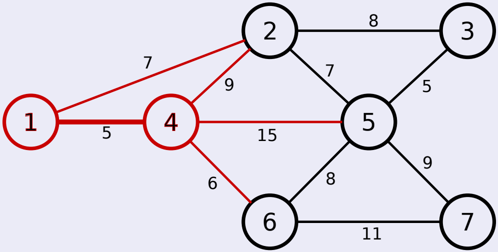
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

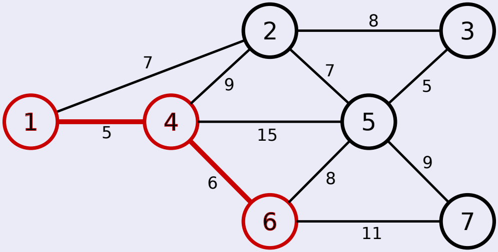
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

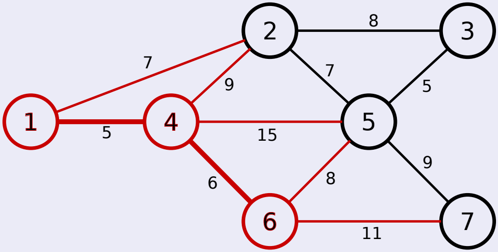
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

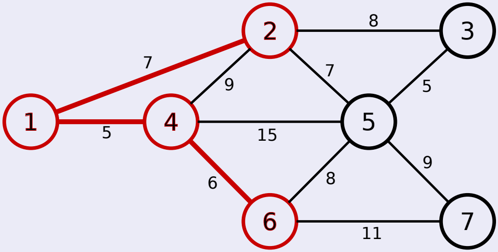
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

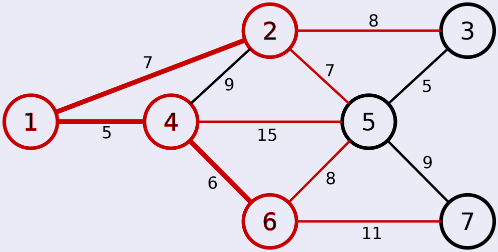
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

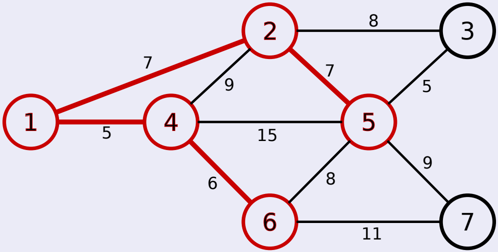
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

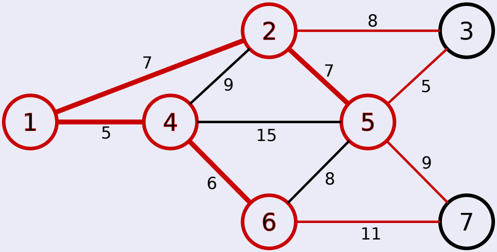
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

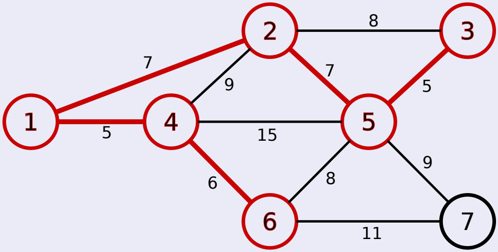
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

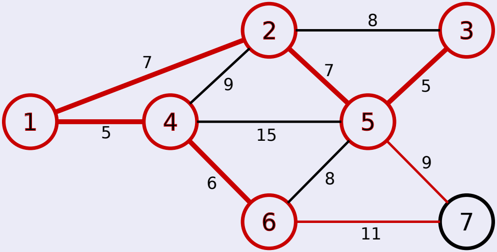
Exemple



Algorithme de Prim

Procedure `Prim`(G)
Soit $E' = \emptyset$
Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
Soit (u, v) l'arête de poids minimum dans $\delta(V')$
//u est dans V', v n'est pas dans V',
(u, v) est l'arête sortante de V' dont le poids est minimum
$E' \leftarrow E' \cup (u, v)$
$V' \leftarrow V' \cup v$
FinTantQue
Retourner $G' = (V', E')$
FinProcedure

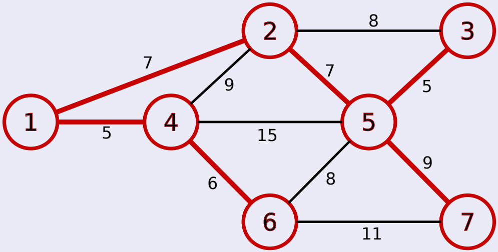
Exemple



Algorithme de Prim

Procedure `Prim`(G)
  Soit $E' = \emptyset$
  Soit $V' = \{s\}$ *//On commence arbitrairement avec le sommet s*
  TantQue $V' \neq V$ *//Tant qu'on n'a pas couvert tous les sommets du graphe*
    Soit (u, v) l'arête de poids minimum dans $\delta(V')$
    //u est dans V', v n'est pas dans V',
      (u, v) est l'arête sortante de V' dont le poids est minimum
    $E' \leftarrow E' \cup (u, v)$
    $V' \leftarrow V' \cup v$
  FinTantQue
  Retourner $G' = (V', E')$
FinProcedure

Exemple



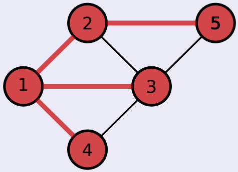
Remarques

- Les arêtes utilisées lors d'un parcours d'un graphe G connexe donne un arbre couvrant de G
- Les arêtes utilisées lors d'un parcours d'un graphe G non-connexe donne une forêt couvrante G

Pourquoi un parcours d'un graphe connexe donne un arbre couvrant ?

- Tous les sommets sont visités donc *couverts*
- On ne retourne jamais sur un sommet déjà visité donc on ne crée pas de cycle
- On obtient donc un graphe connexe acyclique autrement dit : un arbre

Exemple



Pourquoi un parcours d'un graphe non-connexe donne une forêt couvrante ?

- On obtient un arbre couvrant sur chaque composante connexe donc une forêt

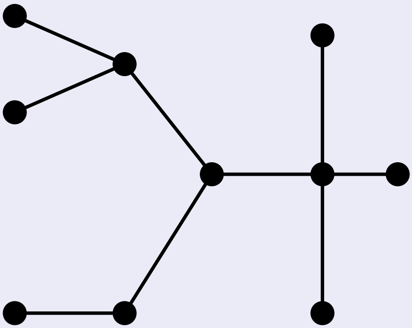
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



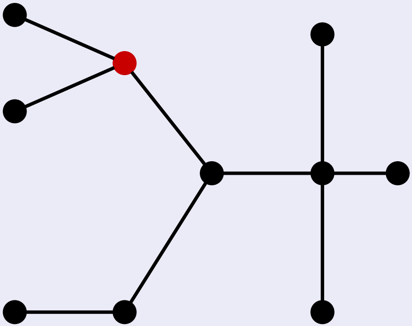
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



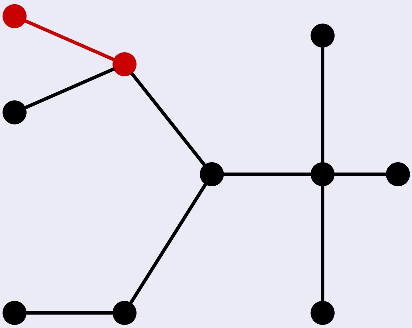
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



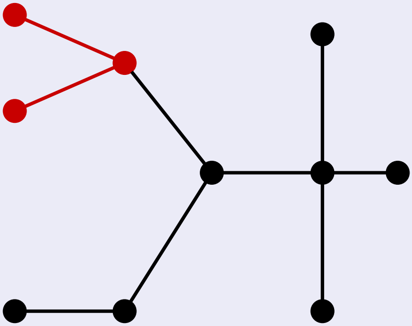
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



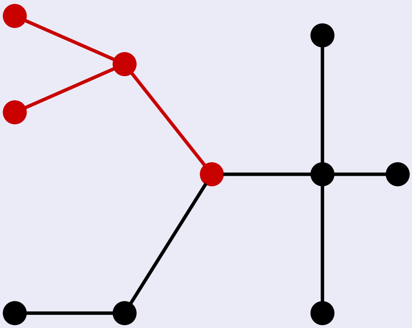
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



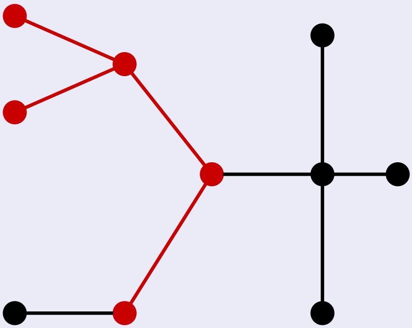
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



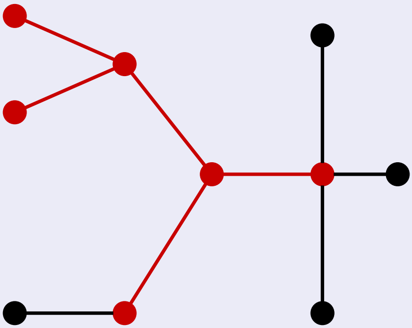
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



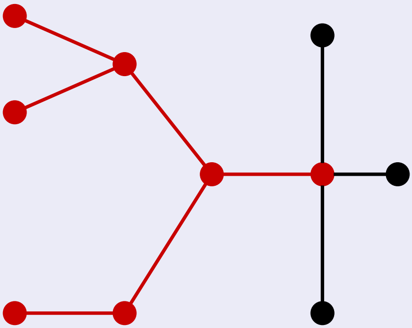
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



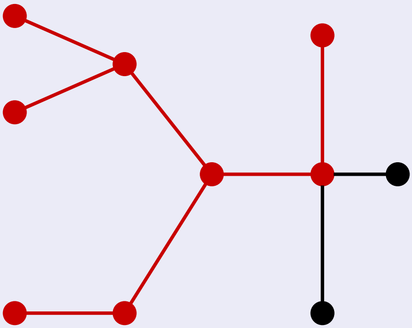
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



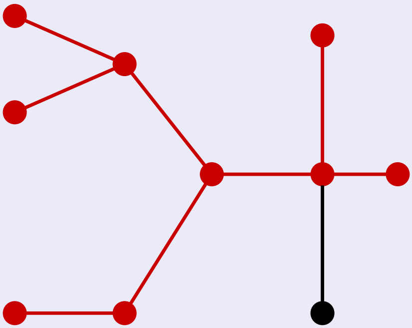
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple



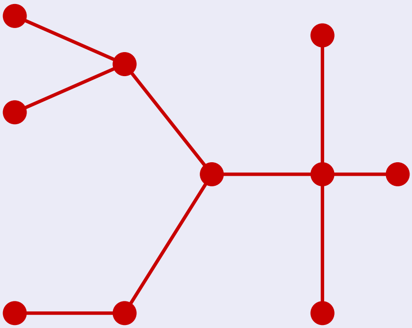
Parcourir un arbre

On sait que :

- un parcours de graphe donne un arbre couvrant
- un arbre ne possède qu'un seul arbre couvrant

Lors du parcours d'un arbre, toutes les arêtes du graphe seront utilisées

Exemple

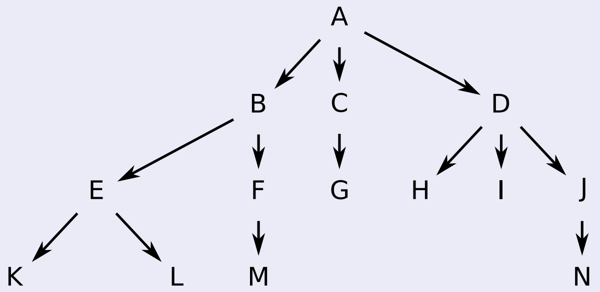


Rappel sur le parcours en largeur

- On visite d'abord un sommet s
- Puis tous les voisins (à distance 1 de s)
- Puis les voisins des voisins (à distance 2 de s)
- ...

Parcourir un arbre en largeur

Un parcours en largeur d'un arbre ressemble à :



Rappel sur le parcours en profondeur

- On visite d'abord un sommet s
- Puis on avance dans le graphe *le plus loin possible*
- Avant de regarder *les voisins directs*
- ...

Parcourir un arbre en profondeur

Un parcours en profondeur d'un arbre ressemble à :