Algorithmes gloutons

1. Optimisation combinatoire
2. Enoncé de l'algorithme glouton, exemples
3. Arbres recouvrants
4. Caractérisation des situations où glouton donne l'optimum: Matroïdes
5. Jeu de Shannon

Optimisation Combinatoire

Soit E un ensemble fini

Une valuation v des éléments de E, $v : E \rightarrow \mathcal{R}^+$

Une famille de parties $\mathcal{F}$ de E

Problème: Trouver $F \in \mathcal{F}$ tel que

$$\sum_{f \in F} v(f)$$

soit maximal.

Remarques

1. Le nombre d'éléments de $\mathcal{F}$ est le plus souvent une fonction exponentielle de n.

2. Vérifier si $F \in \mathcal{F}$ prend un nombre d'opérations de l'ordre de $|F|$ (nombre d'éléments de F), ou bien est polynomial en $|F|$.

3. Le plus souvent la famille $\mathcal{F}$ est close par passage aux sous-ensembles :

$$F \in \mathcal{F}, \quad G \subset F \quad \Rightarrow \quad G \in \mathcal{F}$$

Exemples

1. **Sac a dos:** On a des objets $1, 2, 3, \ldots, n$ de poids $p_1, p_2, p_3, \ldots, p_n$ et on veut en mettre dans un sac de façon telle que le poids du sac soit inférieur ou égal à P. Il s'agit de trouver le poids maximum que l'on peut mettre dans le sac.

$$\mathcal{F} = \{F \mid p(F) = \sum_{i \in F} p_i \leq P\}, \text{ et } \forall i,\ 1 \leq i \leq n \ \ v(i) = p_i$$

2. Formation d'une liste de candidats: On se propose de constituer une liste de personnes aussi large que possible à prendre parmi $x_1, x_2, \ldots, x_n$, en tenant compte d'incompabilités correspondant à des couples qui refusent energiquement de se retrouver ensemble sur la liste.

$$\mathcal{F} = \{F \mid \forall x, y \in F,\ (x, y) \notin U\}, \text{ et } \forall x \quad v(x) = 1$$

Exemples (suite).

- Construire un réseau de fiabilité maximum: On a un certain nombre de points à relier par un réseau et chaque liaison entre deux points a une certaine fiabilité. Il s'agit de trouver l'ensemble des liens à retenir de façon telle que tous les points soient reliés entre eux, qu'il n'y ait pas de circuit et que la somme des fiabilités soit maximale.
- Ordonnancement On a des tâches à réaliser sur des machines et il s'agit de déterminer pour chaque machine les tâches qui doivent y être accomplies et leur ordre de façon à maximiser une certaine fonction.

Enoncé de l'algorithme glouton

1. Classer les éléments de E par valuations décroissantes :

$$v(e_1) \geq v(e_2) \ldots \geq v(e_n)$$

2. Poser $F = \emptyset$

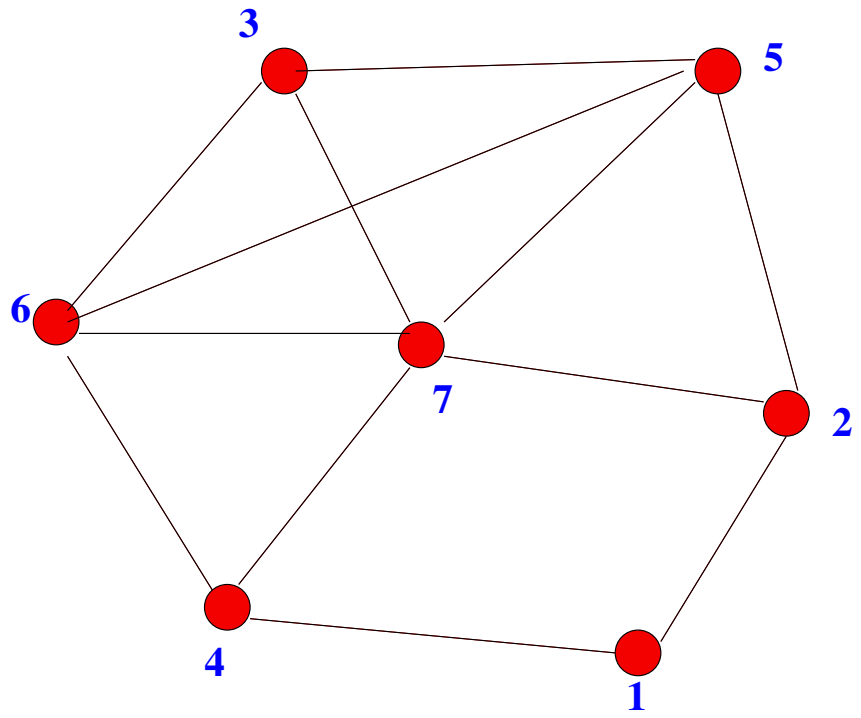
3. Pour $i = 1$ jusqu'à n faire Si $F \cup \{e_i\} \in \mathcal{F}$ Alors $F := F \cup \{e_i\}$

Ne marche pas toujours !!!

Liste sans incompatibilité Exemple :

L'idée est de choisir les personnes par ordre du nombre d'incompatibilités croissantes

On n'obtient que deux personnes, alors qu'on peut faire 3.



Graphe des incompatibilités

Un problème simple

- Donnée : Demandes $\{e_1, e_2, \ldots e_n\}$ d'affectation d'une ressource

- Pour chaque demande e_i :

  La date de début $d(e_i)$ et la date de fin $f(e_i)$

- On cherche un sous ensemble F qui satisfait :

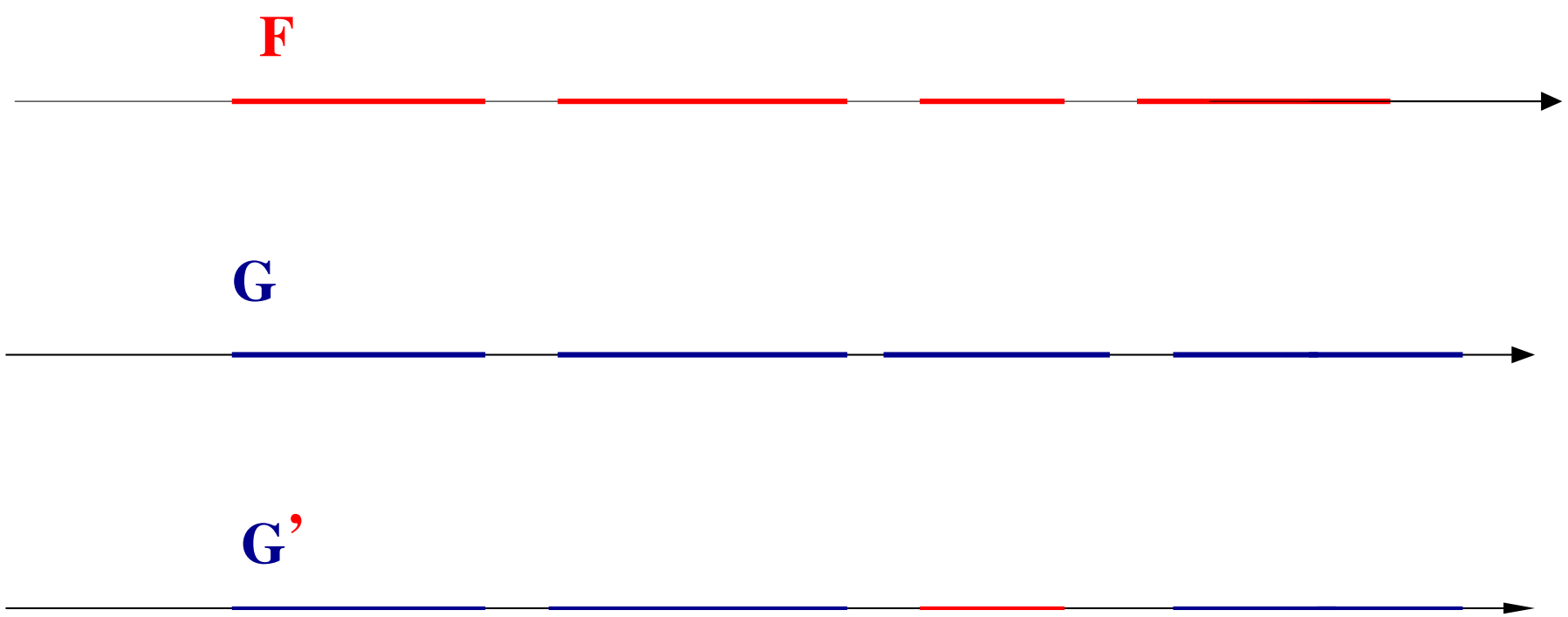
  1. $e_i, e_j \in F \;\Rightarrow\;]d(e_i), f(e_i)[\;\cap\;]d(e_j), f(e_j)[\;= \emptyset$
  2. $|F|$ maximum

Algorithme glouton

- Classer les e_i par dates de fins croissantes
- $F = \{e_1\}$
- Pour $i = 2, 3, \ldots, n$ faire
- Si e_i n'intersecte pas le dernier élément de F Alors
- $F := F \cup \{e_i\}$

Preuve de validité

- F donné par l'algorithme glouton
- G maximum
- Le premier élément g de G qui n'est pas dans F peut être remplacé par un élément f de F qui n'est pas dans G
- on trouve $G' = G - \{g\} + \{f\}$
- tel que : $|G'| = |G|, \quad |F \cap G'| > |F \cap G|$


F
G
G’

Graphes

Définition Un graphe $G = (X, E)$ est donné par un ensemble X de sommets et un ensemble E d'arêtes. Chaque arête est une paire de sommets.

Chemin, cycle Un *chemin* est une suite d'arêtes telles que deux consécutives ont un sommet en commun

$$e_1 = \{x_1, x_2\}, e_2 = \{x_2, x_3\}, \ldots e_p = \{x_p, x_{p+1}\}$$

C'est un cycle si $x_1 = x_{p+1}$

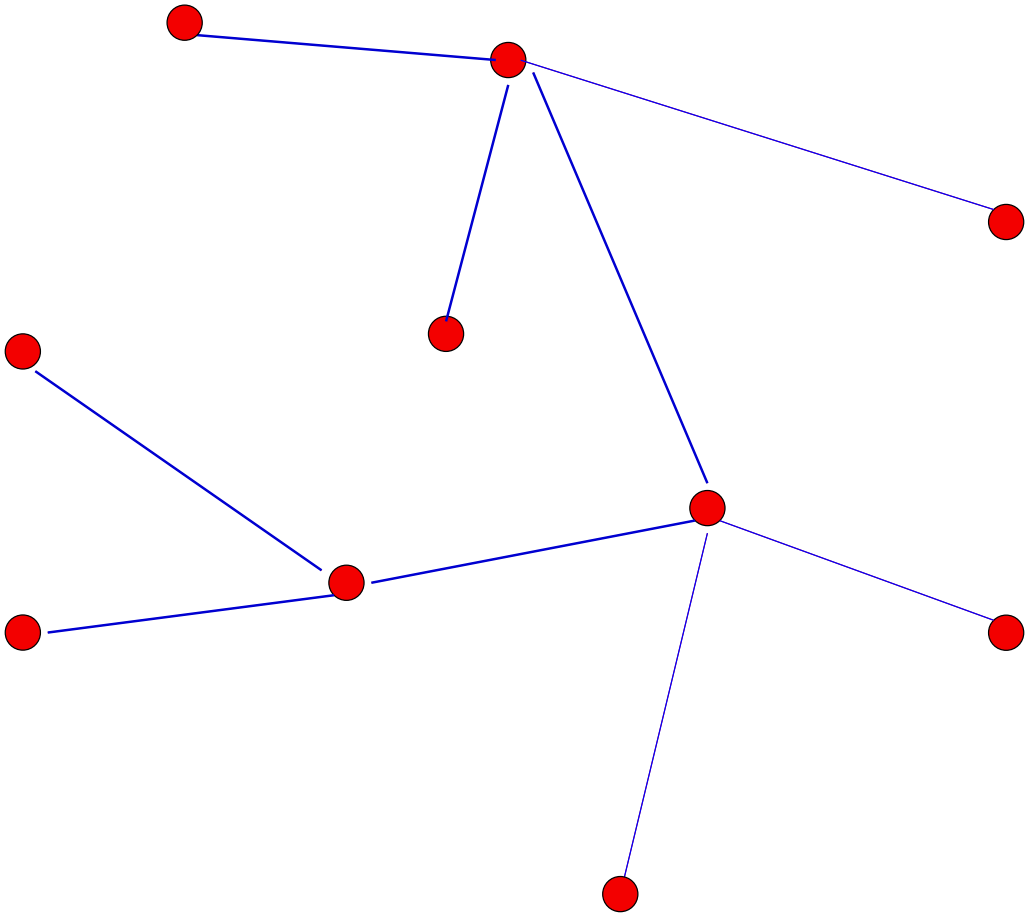
Graphe connexe: Pour tout couple de sommets il y a un chemin qui les joint.

Arbre: C'est un graphe connexe sans cycle.

Arbres

Les propriétés suivantes sont équivalentes pour un graphe ayant n sommets :

1. G est connexe et en supprimant une arête il ne l'est plus
2. G est sans cycle et en ajoutant une arête on en crée
3. G est connexe et possède $n-1$ arêtes
4. G est sans cycle et possède $n-1$ arêtes
5. Entre deux sommets de G il existe un unique chemin



Un arbre à 10 sommets et donc 9 arêtes

Arbres recouvrants d'un graphe $G = (X, E)$

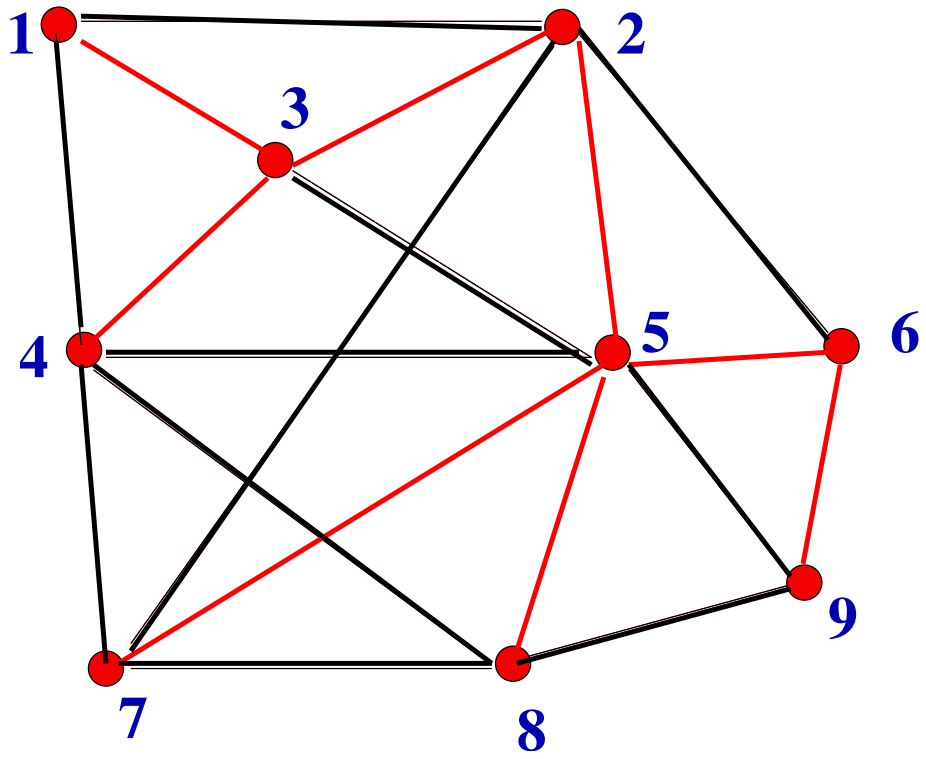
Définition Un arbre qui a pour ensemble de sommets X et dont les arêtes appartiennent à E.

Notations

- $G = (X, E)$
- n nombre de sommets, m nombre d'arêtes (X, T) arbre recouvrant

Propriétés

1. $|T| = n - 1$
2. Chaque arête e qui n'est pas dans T crée un cycle C_e unique.
3. Le nombre de ces cycles est $m - n + 1$, on l'appelle le nombre cyclomatique
4. Tout cycle du graphe s'écrit comme une différence symétrique des C_e



Un graphe et un arbre recouvrant

Arbre recouvrant de fiabilité maximum

On se donne un graphe où chaque arête e a une fiabilité $p(e)$, on cherche l'arbre recouvrant dont la somme des fiabilités des arêtes est maximum

Problème d'optimisation combinatoire avec :

$\mathcal{F}$ est la famille des partie de E qui ne contiennent pas de cycle

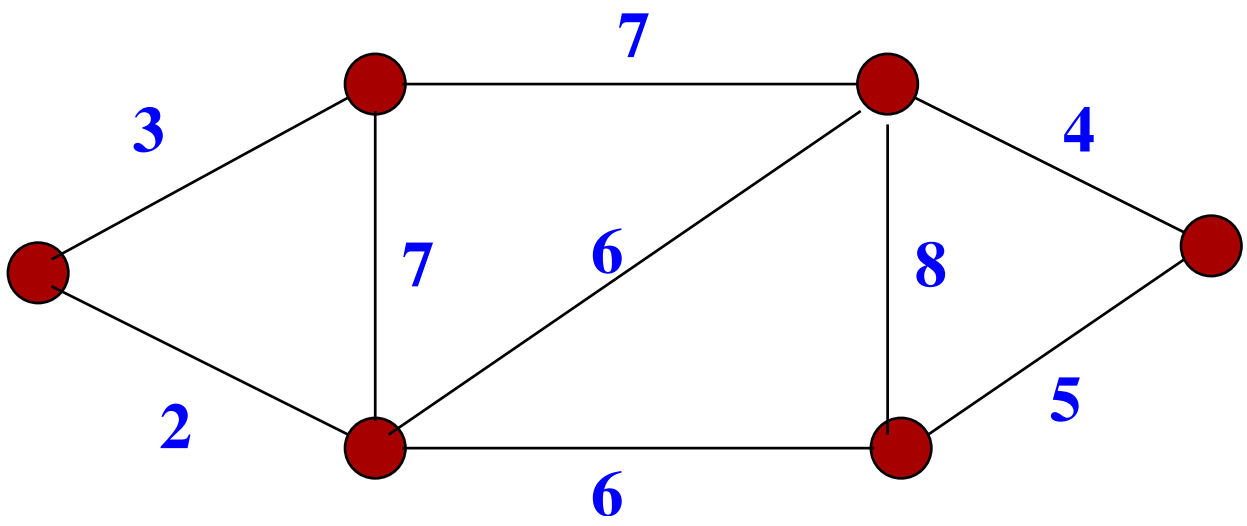
Algorithme de Kruskal

1. Classer les éléments de E par fiabilités décroissantes :

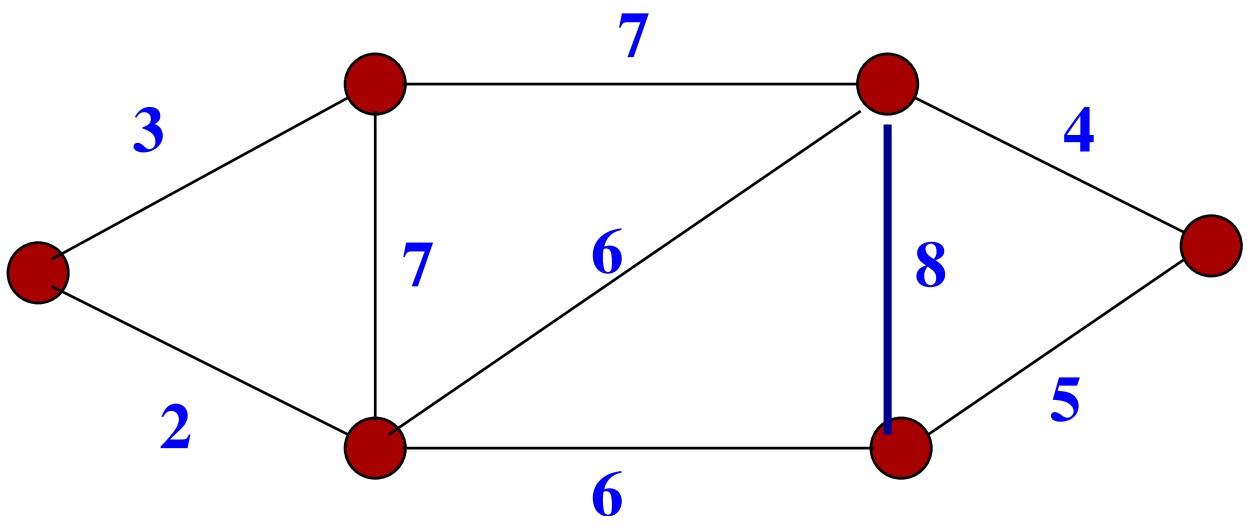
$$p(e_1) \geq p(e_2) \ldots \geq p(e_n)$$

2. Poser $T = \emptyset$

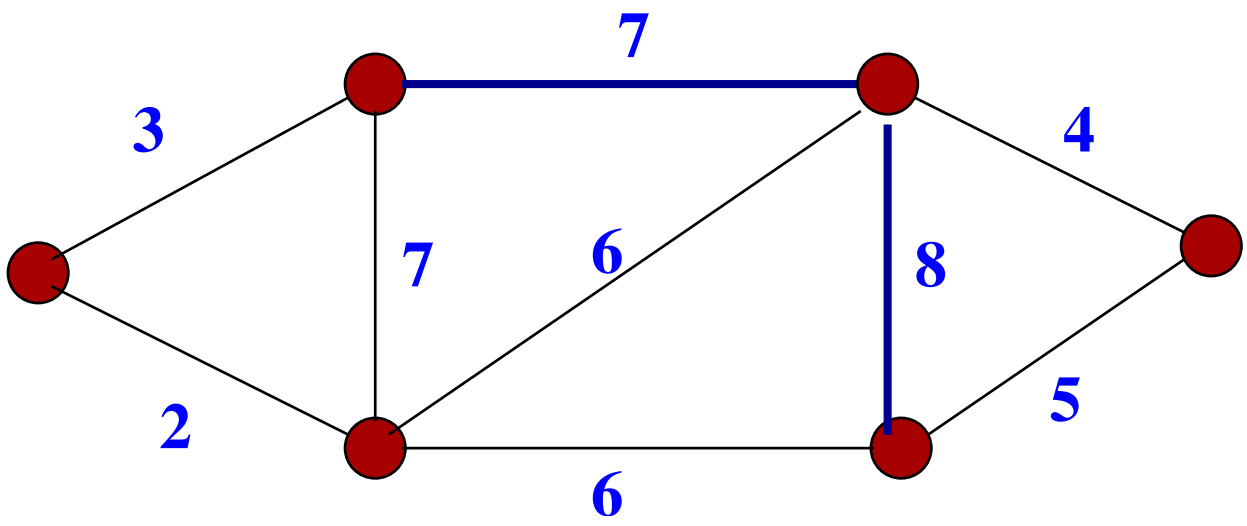
3. Pour $i = 1$ jusqu'à n faire Si $T \cup \{e_i\}$ n'a pas de cycle Alors $T := T \cup \{e_i\}$



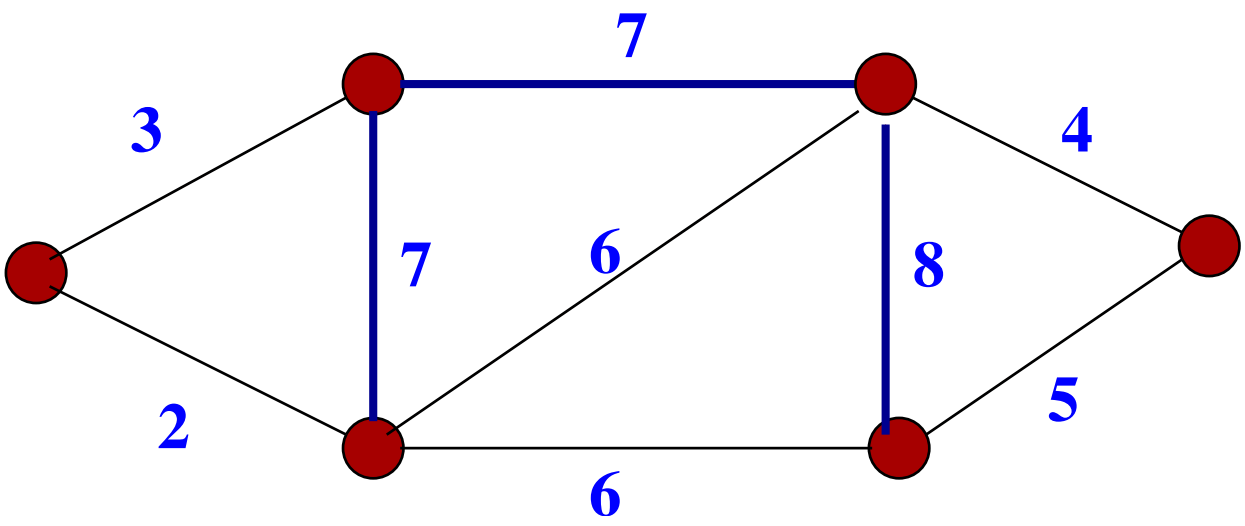
Exemple d’exécution de Kruskal



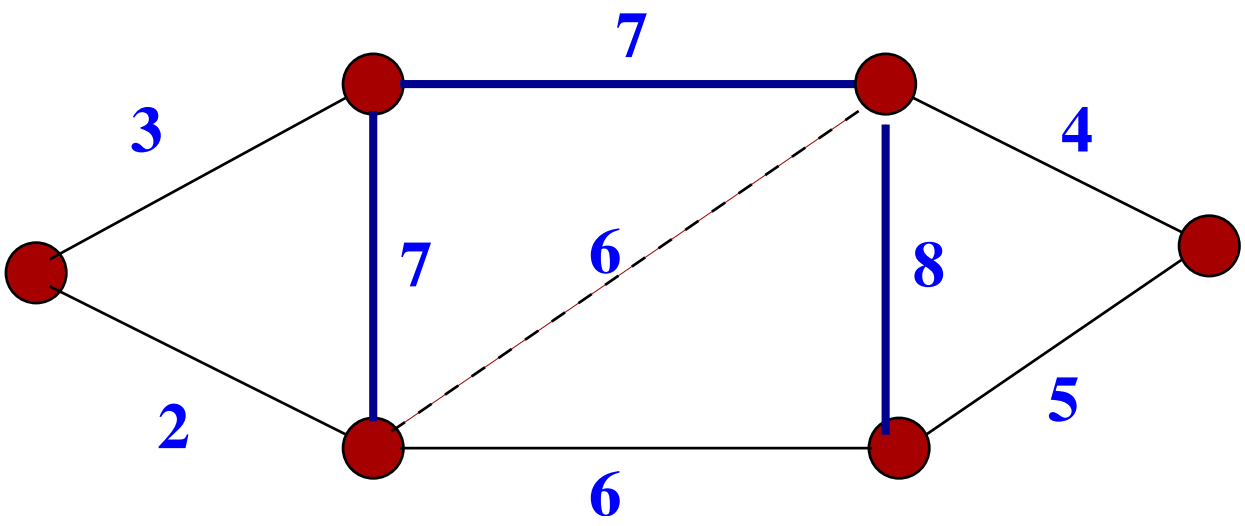
Exemple d’exécution de Kruskal



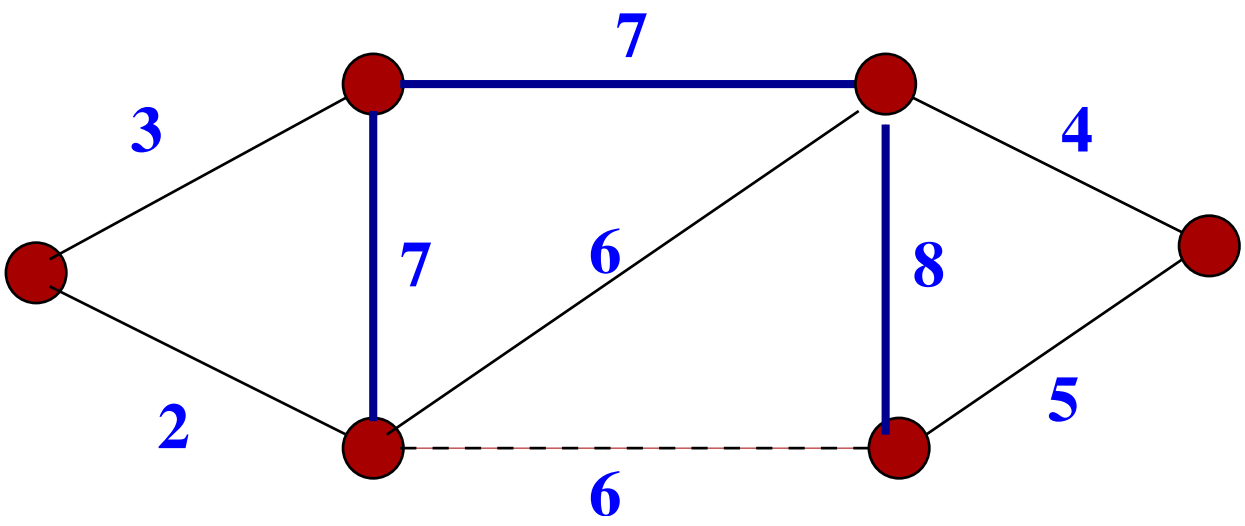
Exemple d’exécution de Kruskal



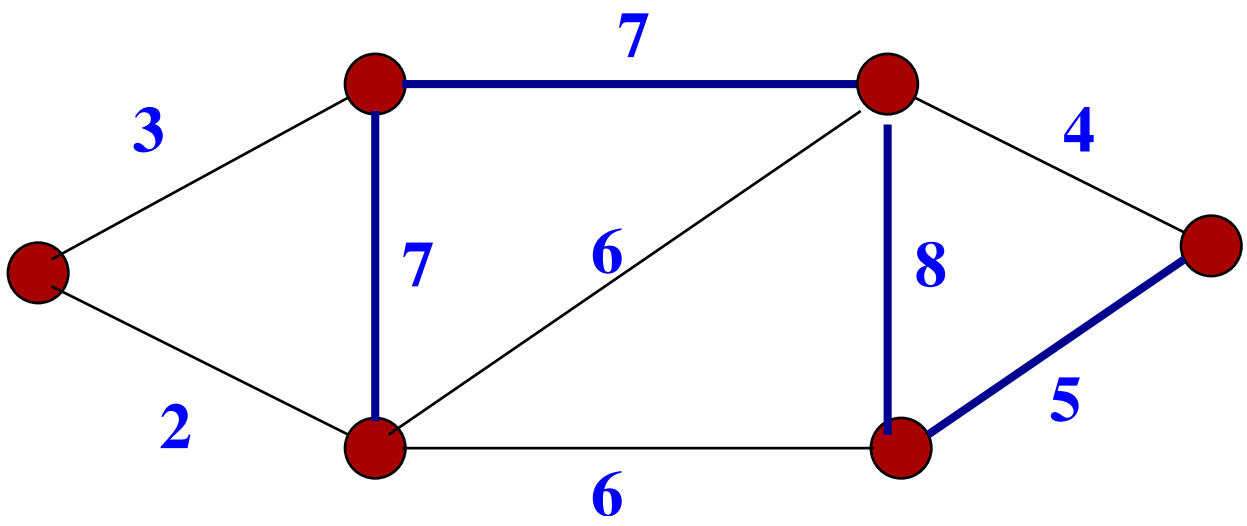
Exemple d’exécution de Kruskal



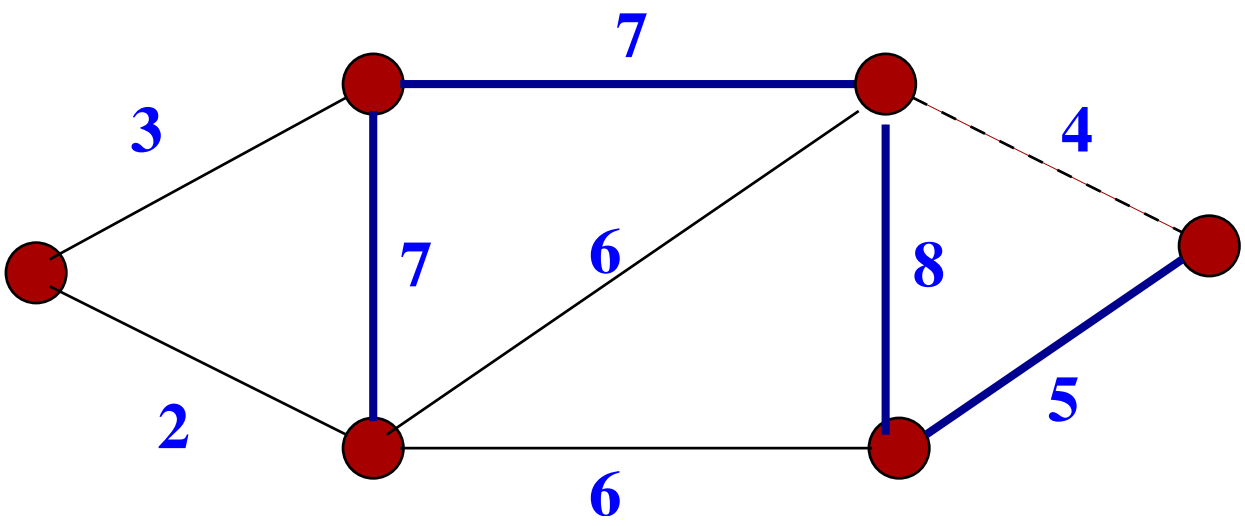
Exemple d’exécution de Kruskal



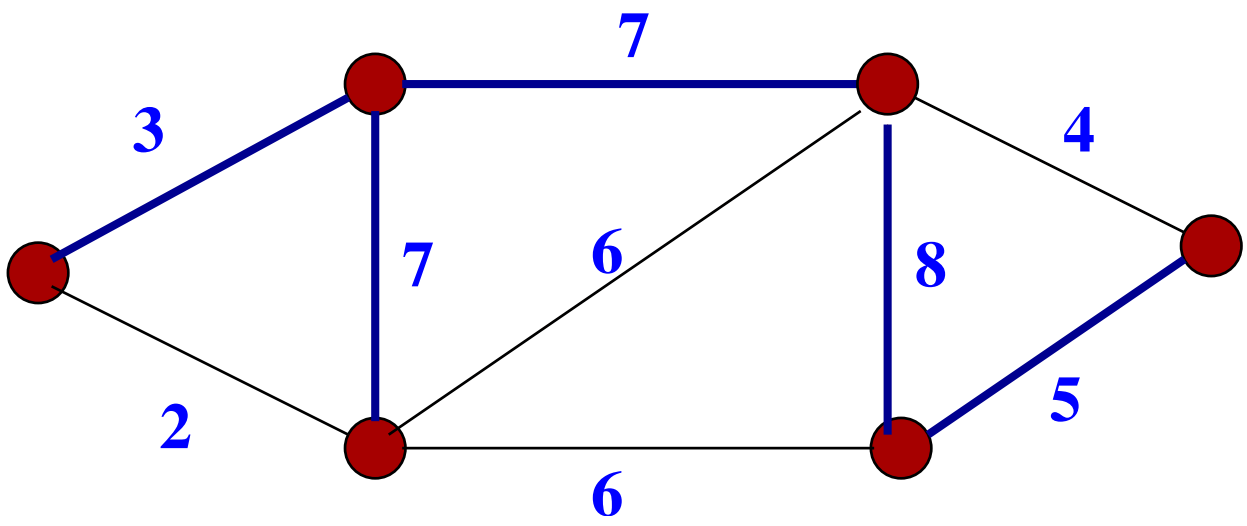
Exemple d'exécution de Kruskal



Exemple d’exécution de Kruskal



Exemple d’exécution de Kruskal



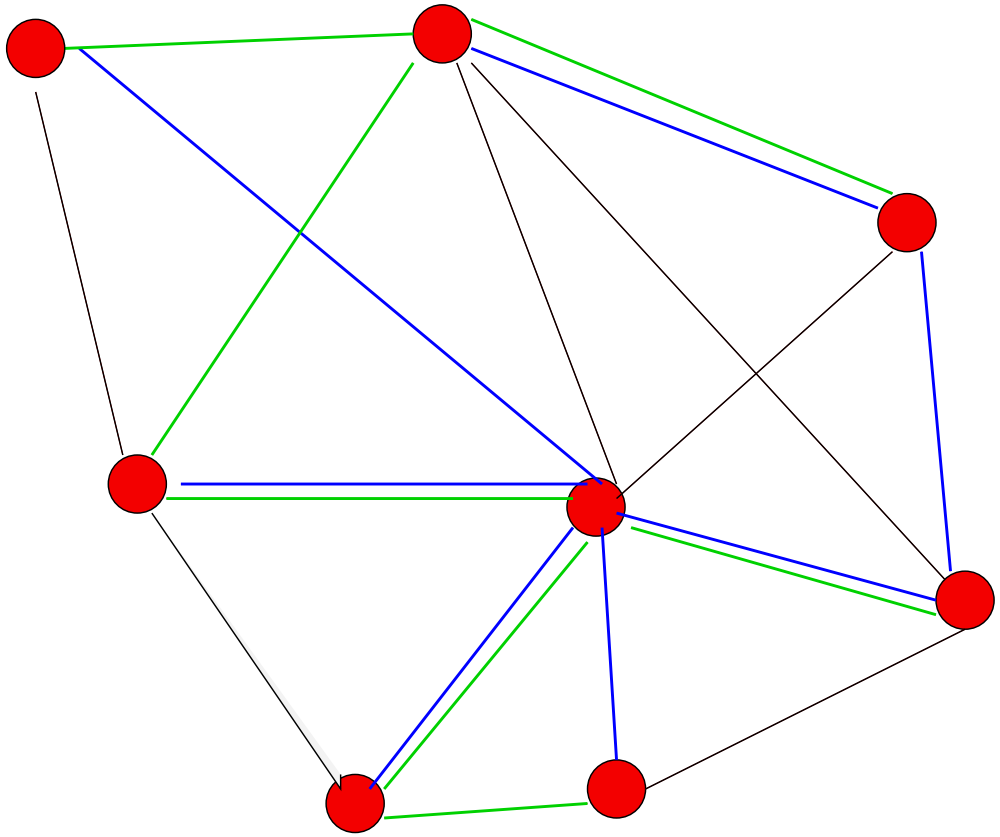
Exemple d’exécution de Kruskal

Preuve de validité de Kruskal

Lemme

Soient T et U deux arbres recouvrants de G, soit $a \in U,\ a \notin T$ alors il existe $b \in T$ tel que $U - a + b$ soit un arbre recouvrant.

De plus b peut être choisi dans le cycle formé par des arêtes de T et a.



Preuve (suite et fin)

- Soit T l'arbre obtenu par l'algo glouton U le maximal
- $\exists a \in U,\ a \notin T$ soit b donné par le lemme
- $p(b) \geq p(a)$ car le choix de b est glouton
- $p(b) = p(a)$ car U maximal
- $U' = U + a - b$ est plus proche de T.

Caractérisation de l'algorithme glouton

L'algorithme glouton donne une solution optimale quelle que soit la fonction de fiabilité si et seulement si la famille $\mathcal{F}$ vérifie la propriété suivante :

Si F et G sont deux ensembles de $\mathcal{F}$ tels que G a plus d'éléments que F, alors il existe un élément x de G qui n'est pas dans F et tel que $F \cup \{x\} \in \mathcal{F}$.

Preuve

glouton optimal pour toutes les valuations $\Rightarrow$ $\mathcal{F}$ satisfait la condition:

- Soient F et G donnés tels que $|G| > |F|$, on construit v par:
- On pose $p = |F \cap G|$, $q = |F| - p$
  1. Si $e \notin F \cup G$ $\quad v(e) = 0$
  2. Si $e \in F$ $\quad v(e) = 2q + 2$
  3. Si $e \in G \setminus F$ $\quad v(e) = 2q + 1$

L'algorithme glouton donne comme résultat F' contenant F et peut être d'autres éléments :

- $v(F) = (p+q)(2q+2) = 2pq + 2q^2 + 2p + 2q$
- $v(G) \geq p(2q+2) + (q+1)(2q+1) = 2pq + 2q^2 + 2p + 3q + 1$

Comme $v(G) > v(F)$ il y a d'autres éléments dans F', un quelconque de ces éléments est le x cherché

Réciproque

On suppose que la condition est vérifiée, on montre que glouton donne l'optimum

- Soit F obtenu par l'algo glouton G appartenant à $\mathcal{F}$ on montre que $v(F) \geq v(G)$ par récurrence sur $p = |G \setminus F|$.
- Si $p = 0$ alors trivial
- $\exists x \in F \setminus G$ sinon glouton construit G
- Soit x le premier $x \in F \setminus G$ choisi par glouton
- $y \in G \setminus F \ \Rightarrow \ v(y) \leq v(x)$ sinon y retenu par glouton.
- Soit $F' = (F \cap G) + \{x\}$ on ajoute des éléments de G à F' pour obtenir un ensemble G' tel que $|G'| = |G|$
- On $G' = G - y + x$ et $v(G) \leq v(G') \leq v(F)$

Matroïdes

La famille $\mathcal{F}$ de sous-ensembles de E forme un matroïde si les trois conditions suivantes sont satisfaites :

1. La famille $\mathcal{F}$ n'est pas vide
2. Si $F \in \mathcal{F}$ et $G \subset F$ alors $G \in \mathcal{F}$
3. Si $F, G \in \mathcal{F}$ et $|G| > |F|$ alors $\exists x \in G, x \notin F$ tel que $F \cup \{x\} \in \mathcal{F}$

Les éléments de $\mathcal{F}$ sont appelés les *ensembles indépendants* du matroïde.

Résultat principal

L'algorithme glouton donne une solution optimale quelle que soit la fonction de fiabilité si et seulement si la famille $\mathcal{F}$ est un matroïde

Exemples de matroïdes

- Ensemble de vecteurs libres d'un espace vectoriel
- Arcs sans origine commune dans un graphe orienté
- Ensemble de représentants distincts
- Ensemble d'arêtes sans origine commune dans un graphe biparti

Bases d'un matroïde

Une base est un ensemble indépendant maximal. Deux bases ont le même nombre d'éléments

La famille $\mathcal{B}$ des bases est caractérisée par:

Si $B, B' \in \mathcal{B}$ et $x \in B' \setminus B$ alors $\exists y \in B,\ y \notin B'$ tel que $B' \cup \{y\} \setminus \{x\} \in \mathcal{F}$

Rang dans un matroïde

Le rang $\rho(U)$ d'un sous-ensemble U de E est le nombre d'éléments du sous-ensemble X indépendant maximal inclus dans E

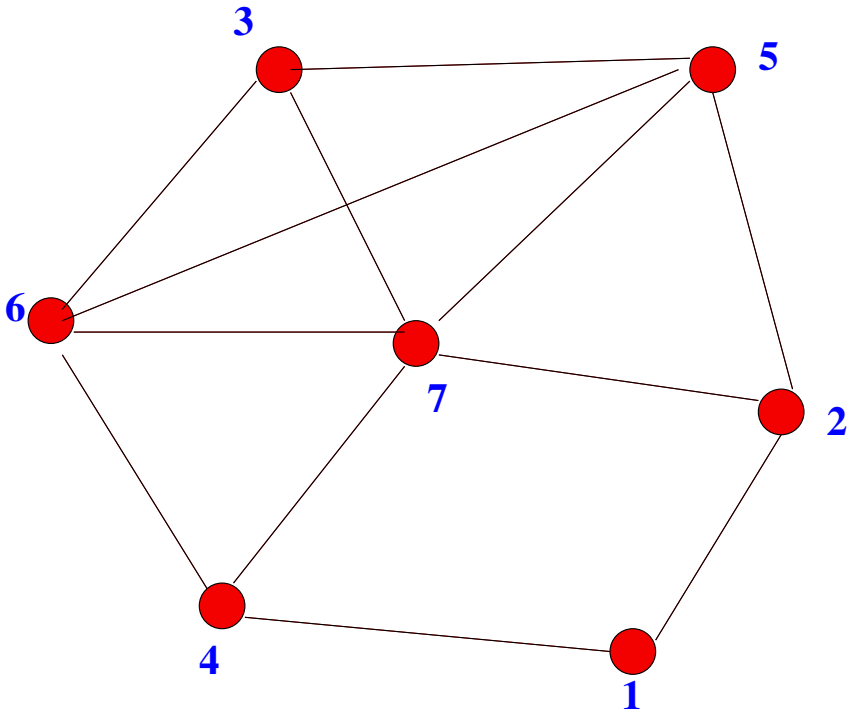
La fonction rang $\mathcal{B}$ des bases est caractérisée par:

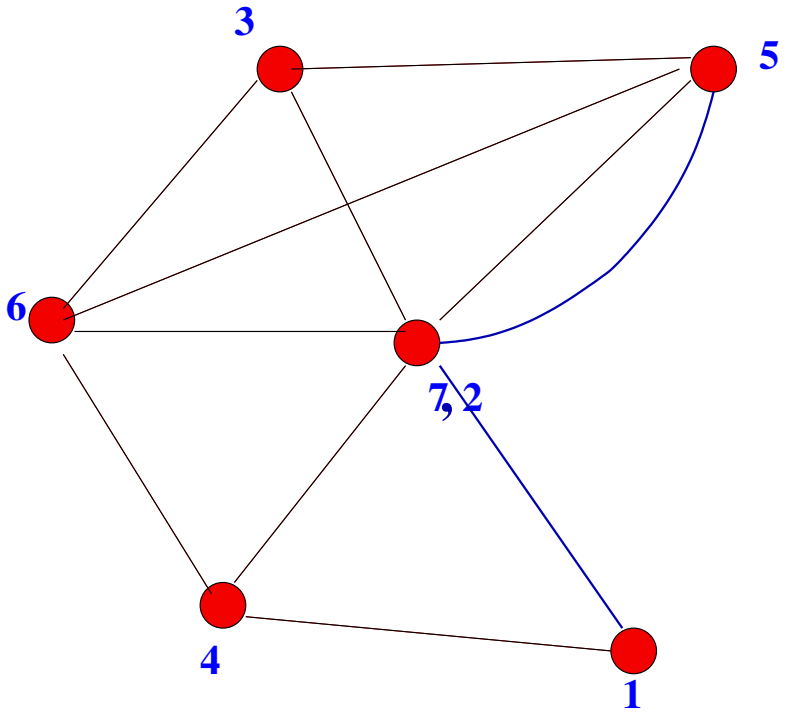
- $$V \subset W \Rightarrow \rho(V) \leq \rho(W) \leq |W|$$

- $$\rho(V) + \rho(W) \geq \rho(V \cap W) + \rho(V \cup W)$$

Jeu de Shannon

- Un graphe et deux sommets u et v
- Le joueur A doit relier u et v par des arêtes vertes
- Le joueur B coupe des arêtes en les coloriant en rouge
- Chaque joueur colorie une arête à son tour
- B commence





Contraction de l'arête 2,7

Stratégie

A est gagnant si et seulement si il existe deux arbres sans arête commune qui contiennent u et v

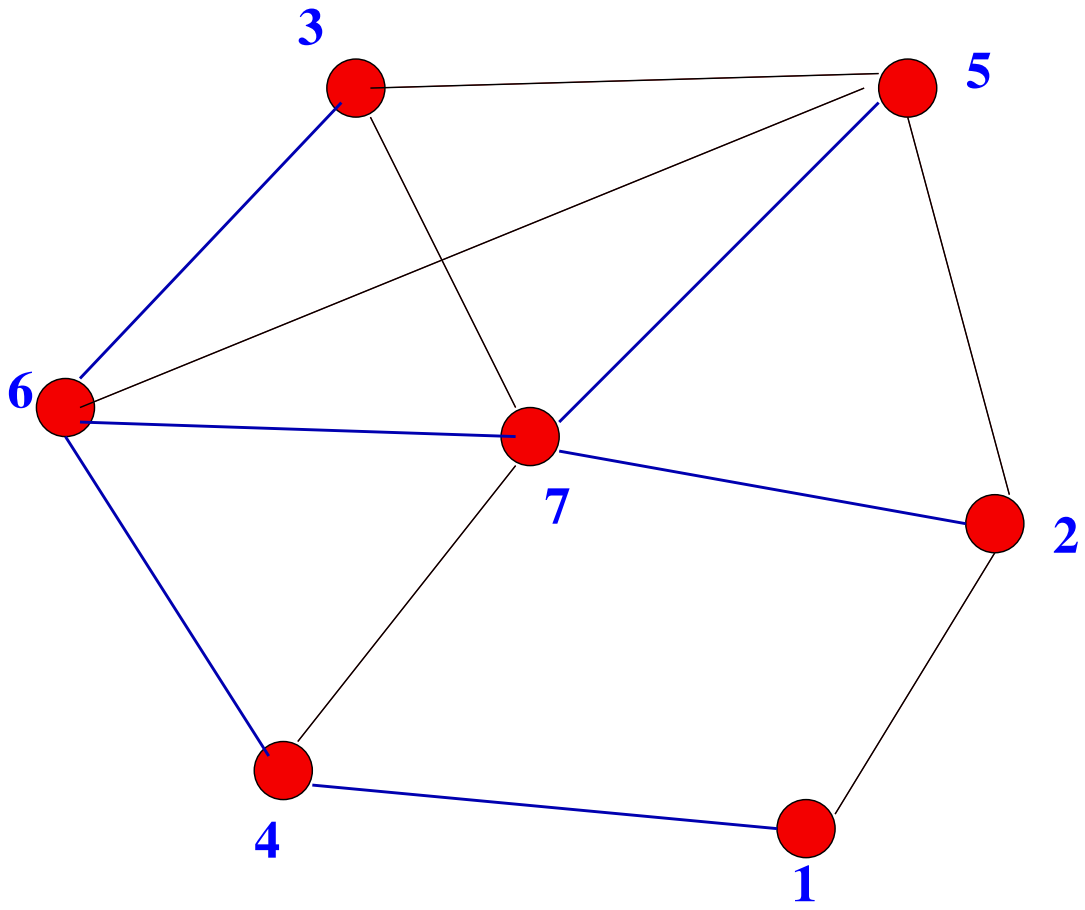
Règle1

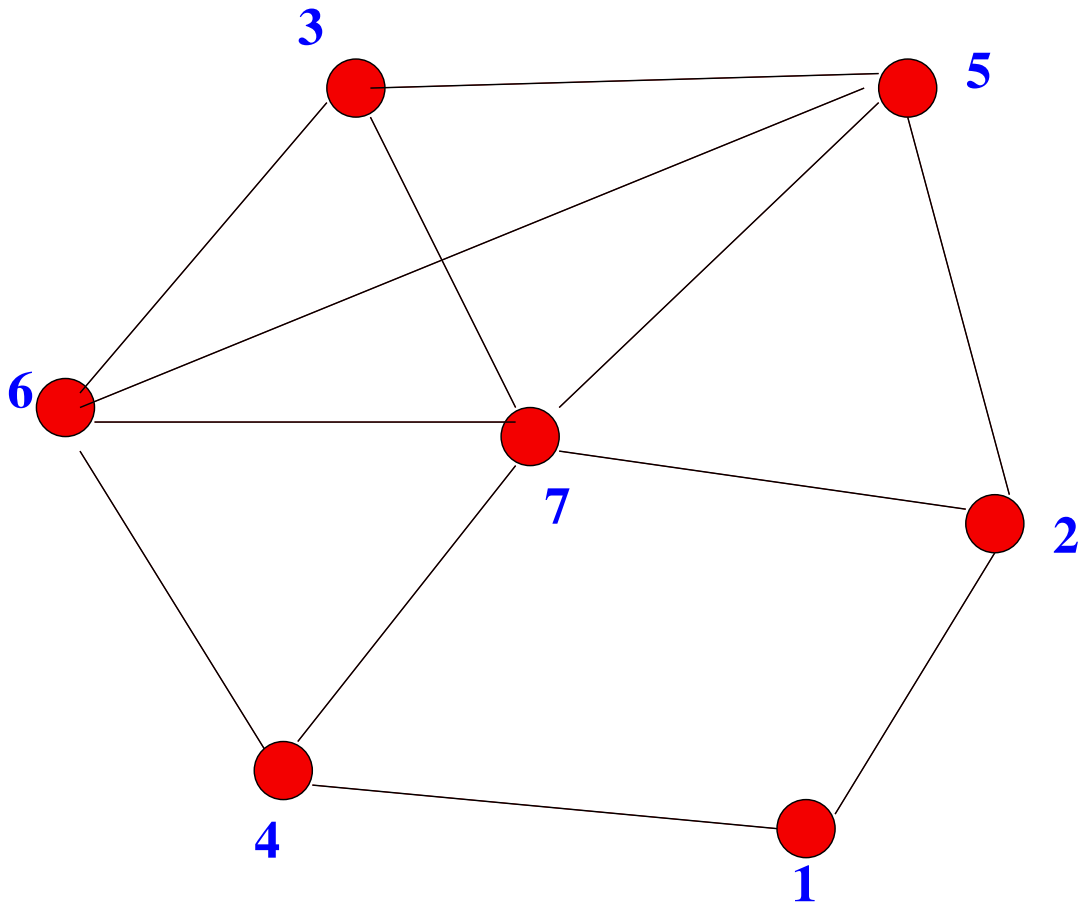
Lorsque B joue dans un des arbres il coupe celui-ci en deux.

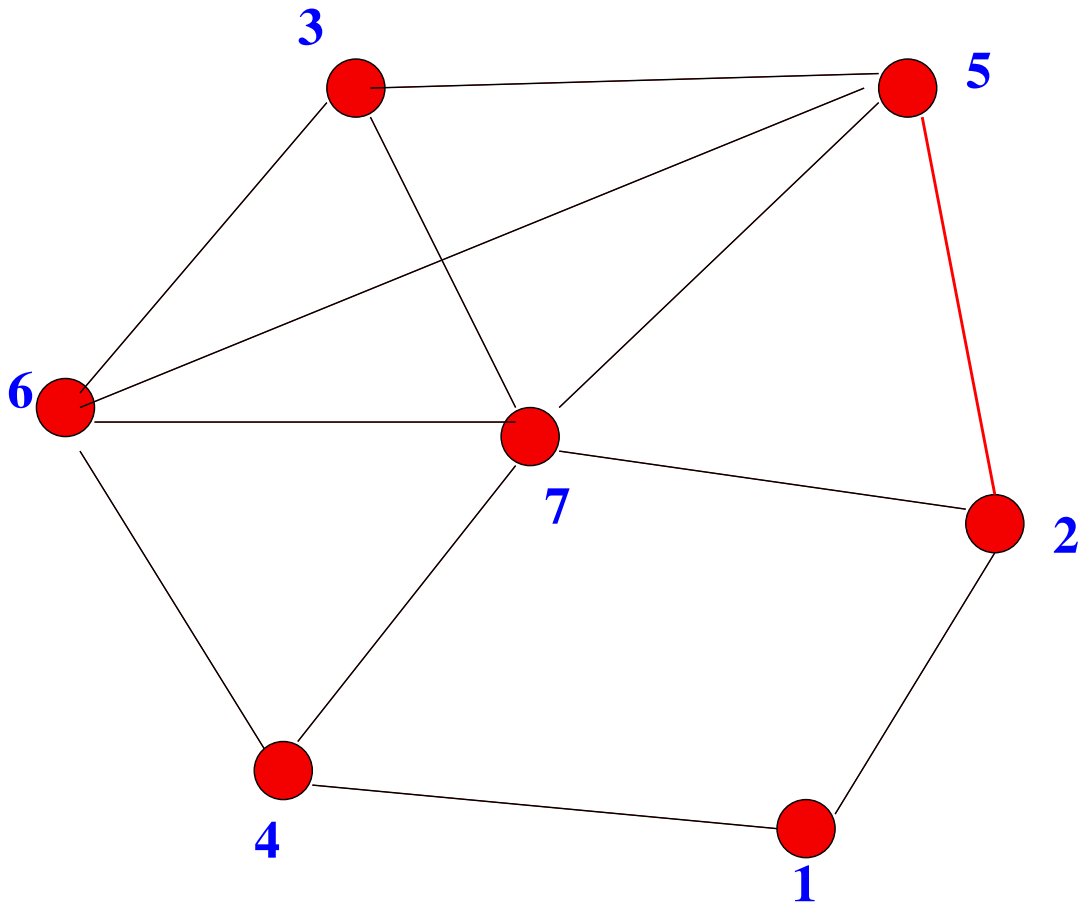
A doit jouer une arête de l'autre arbre qui relie les composantes ainsi formées

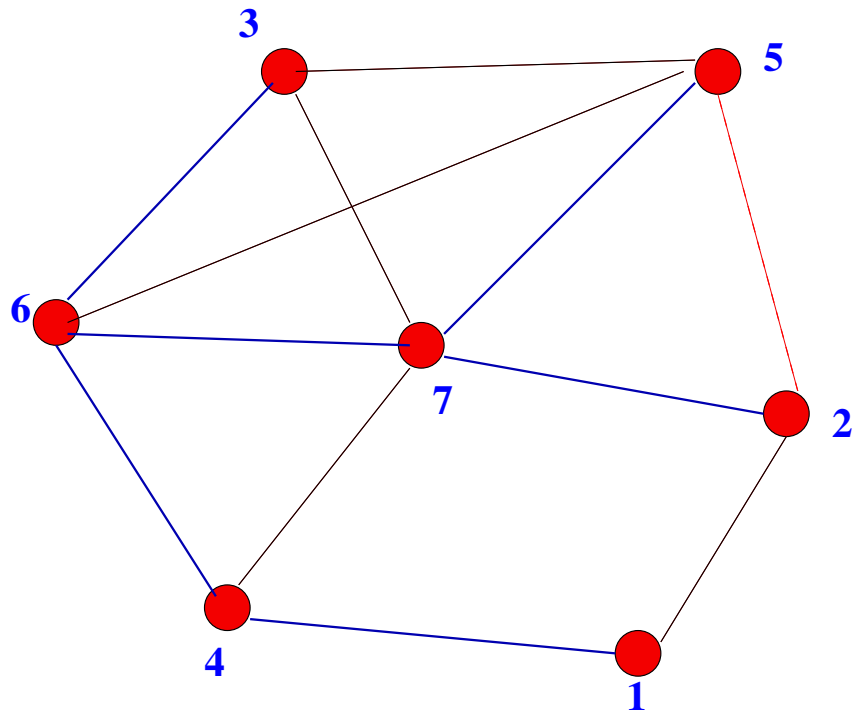
Règle2

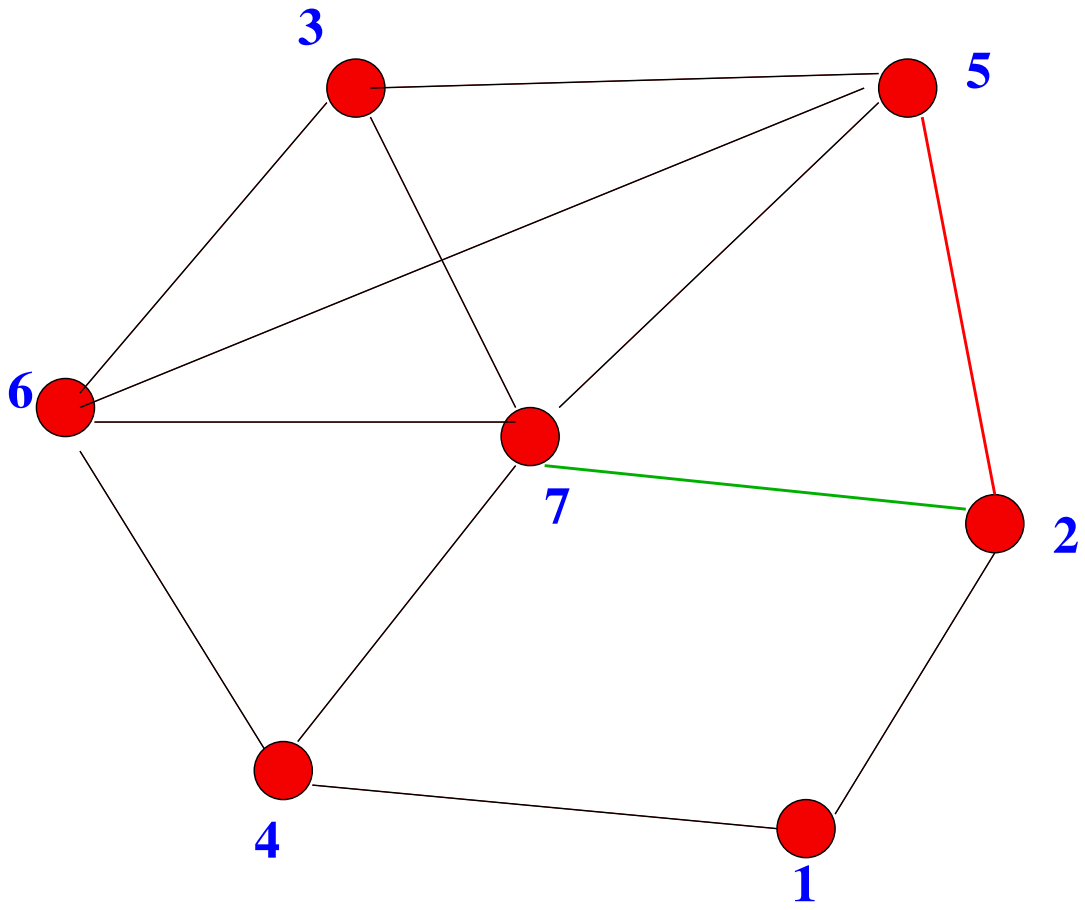
- Contracter les arêtes jouées par A
- Supprimer les arêtes jouées par B

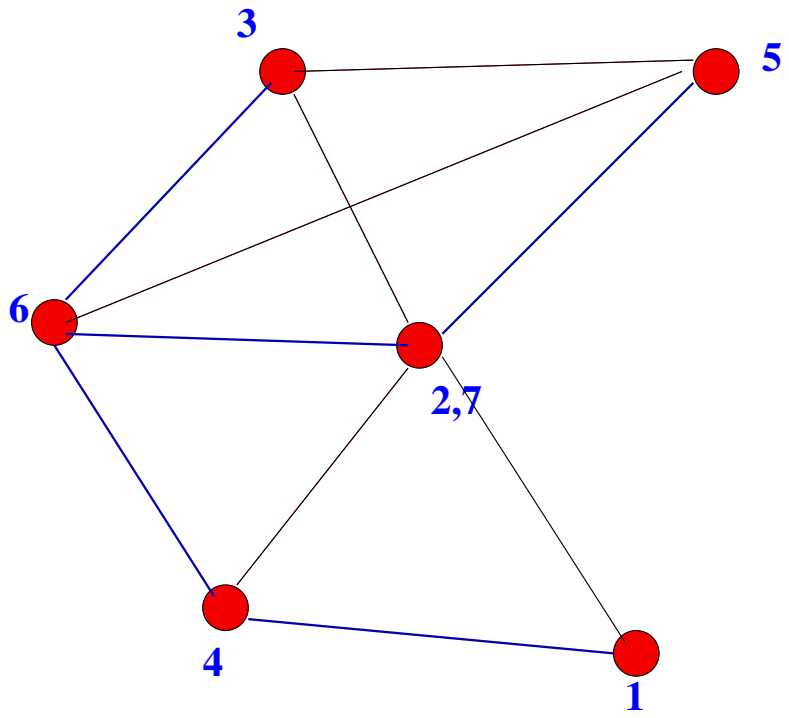

3
5
6
7
2
4
1

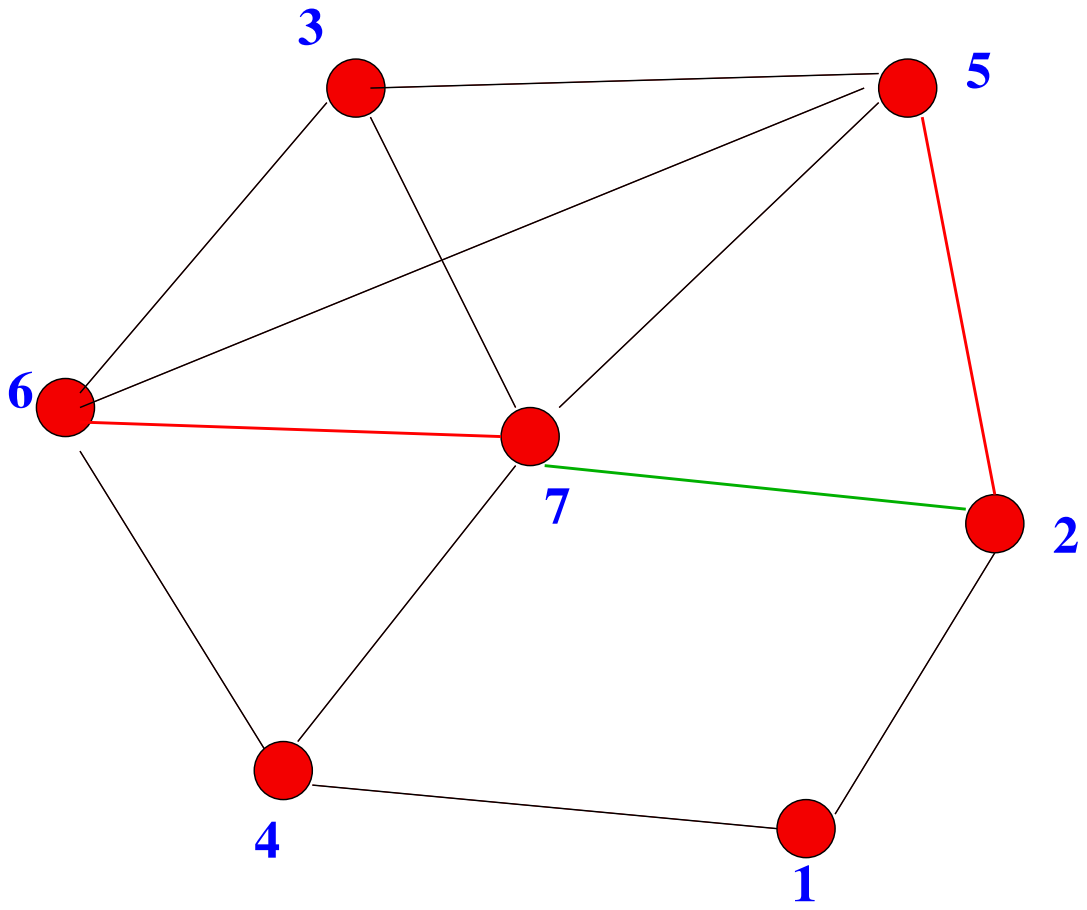

3
5
6
7
2
4
1

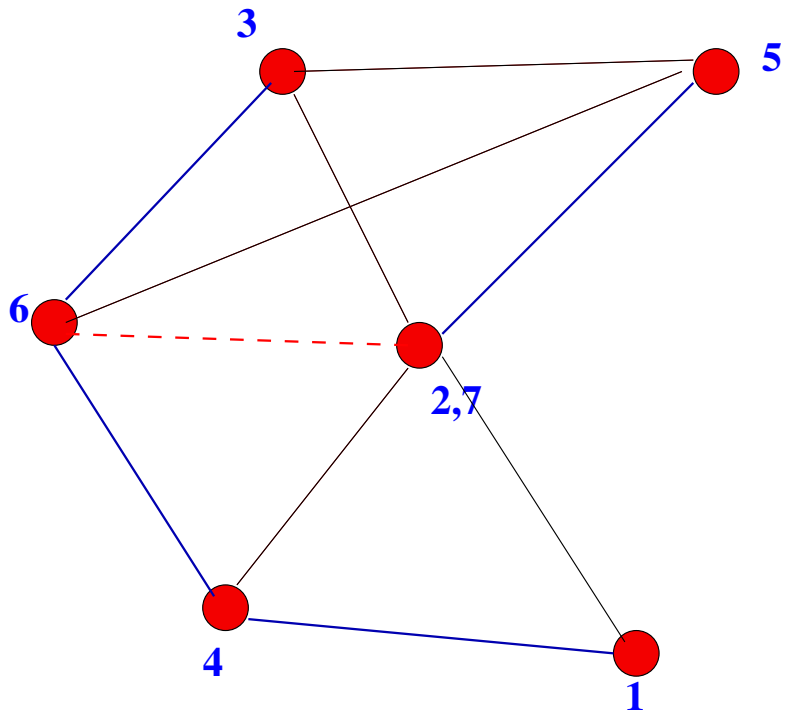

3
5
6
7
2
4
1

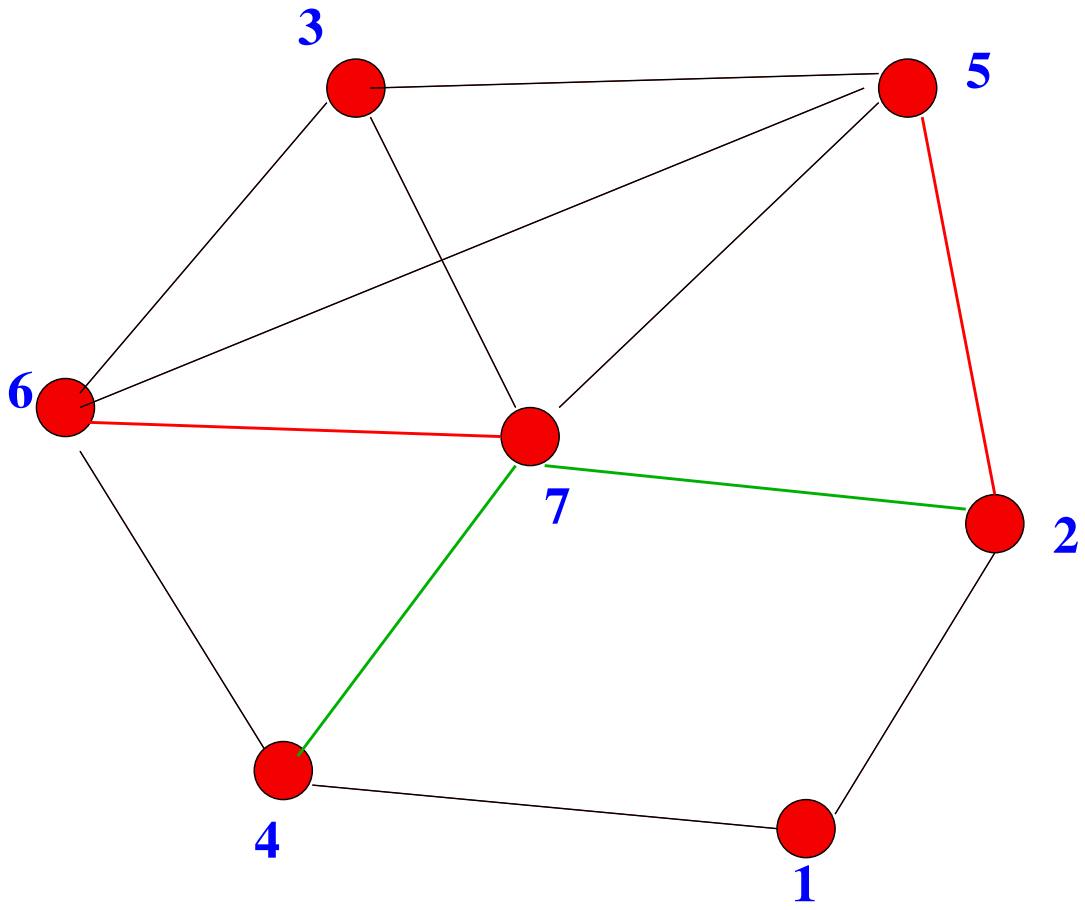

3
5
6
7
2
4
1

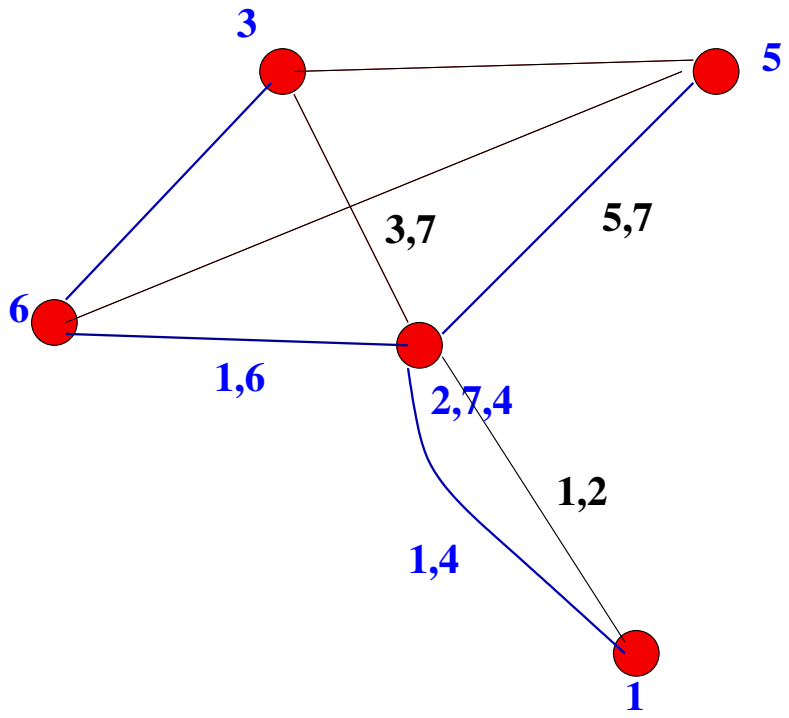

3
5
6
7
2
4
1

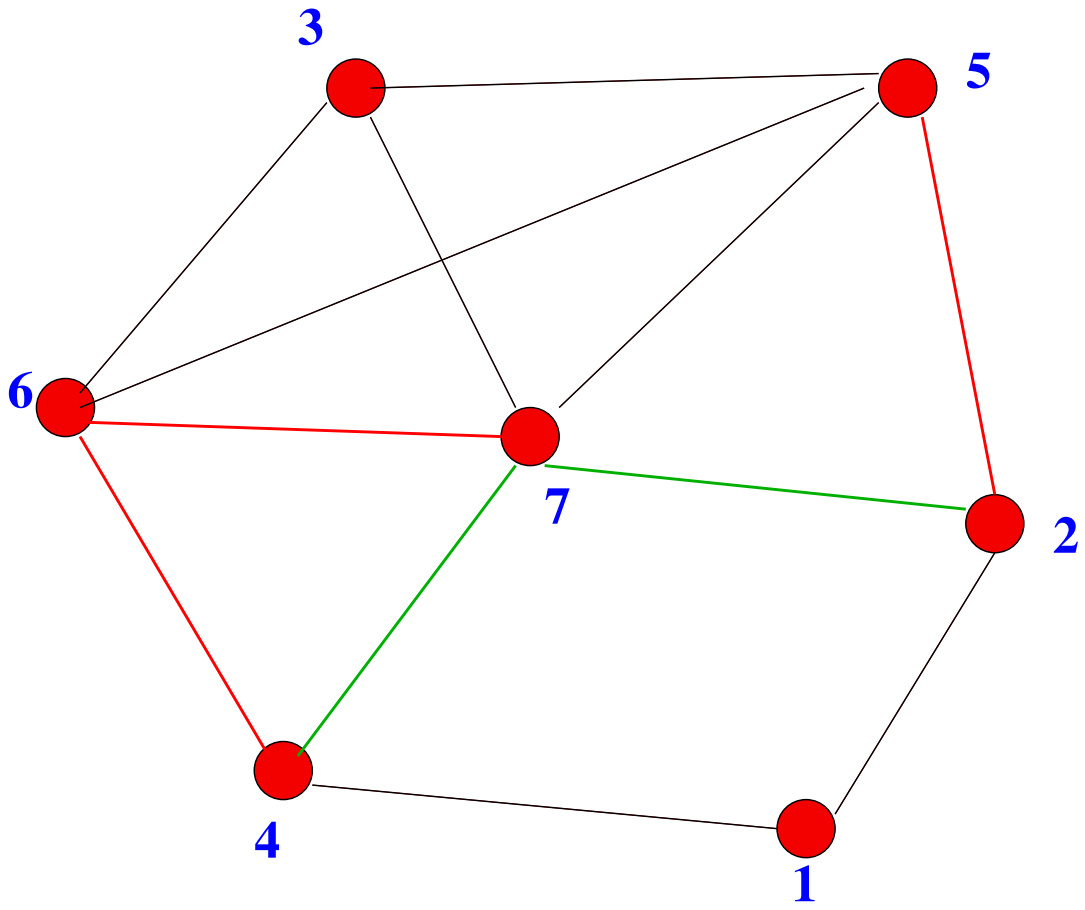

3
5
6
2,7
4
1

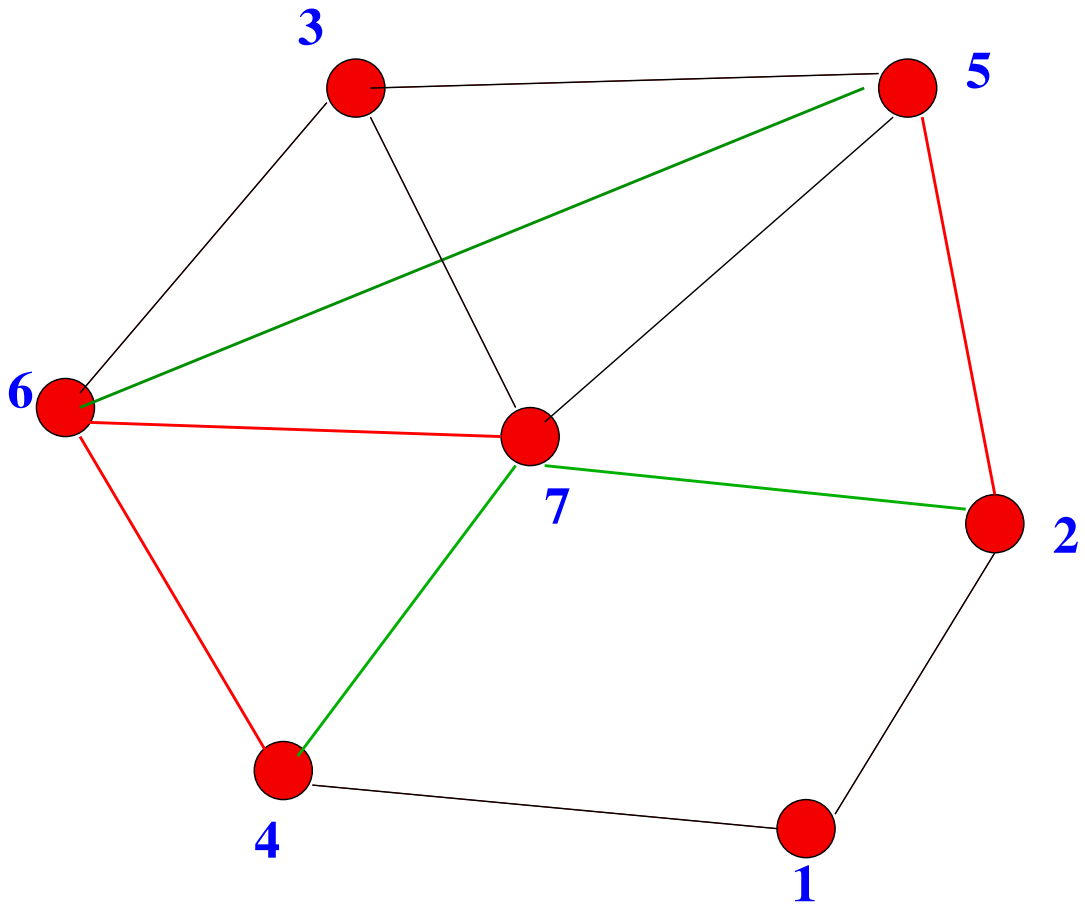

3
5
6
7
2
4
1

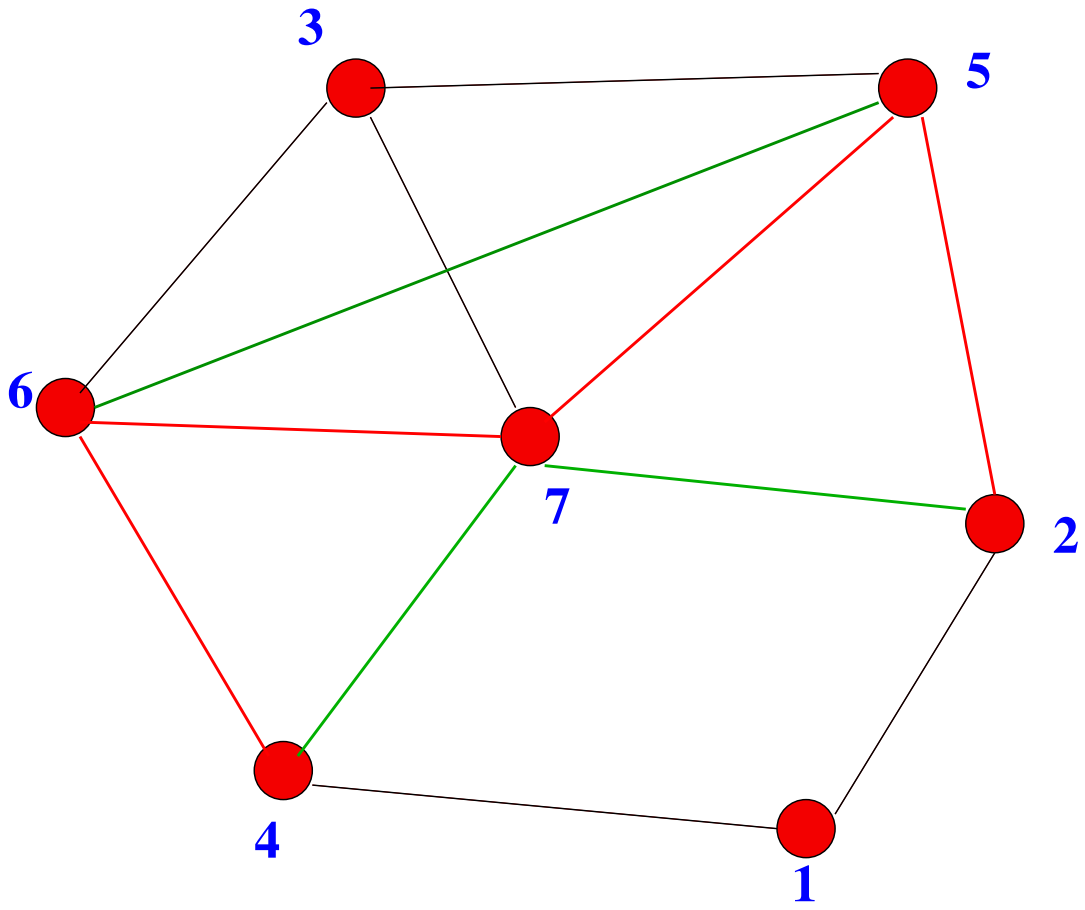

3
5
6
2,7
4
1

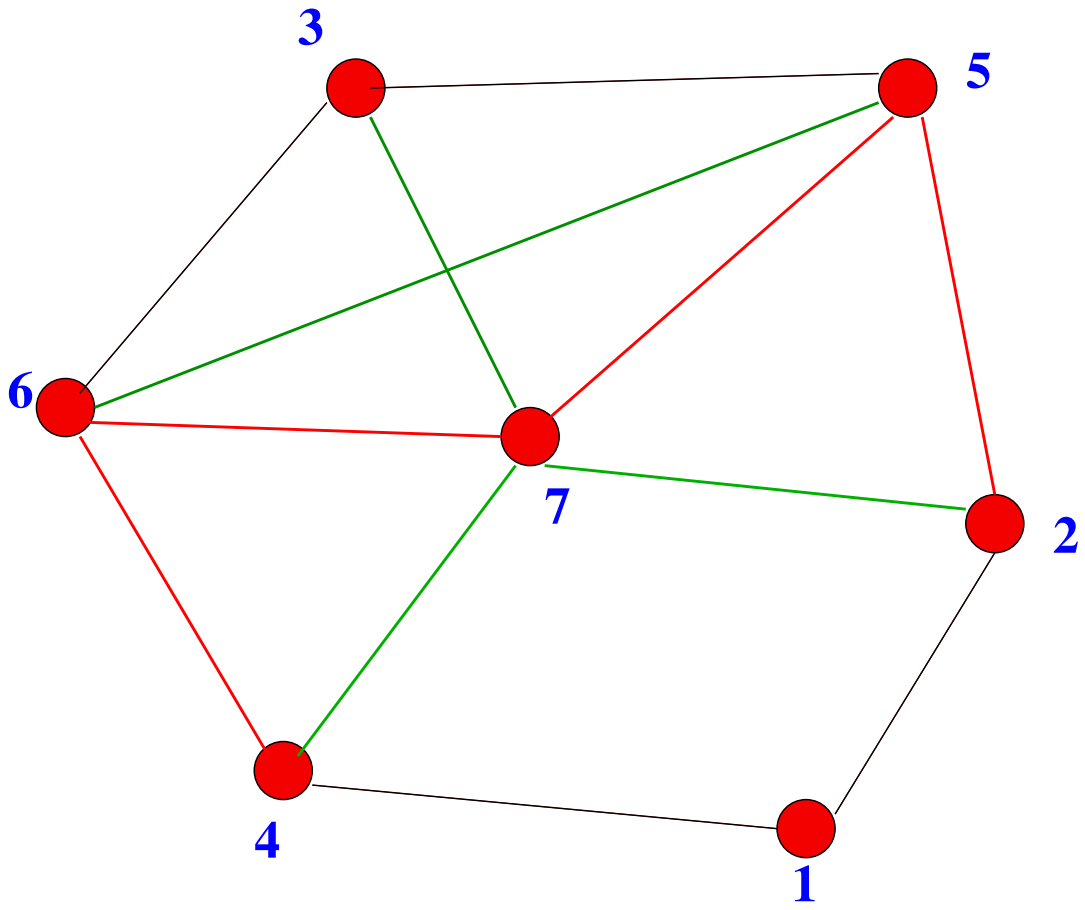

3
5
6
7
2
4
1


3
5
3,7
5,7
6
1,6
2,7,4
1,2
1,4
1


3
5
6
7
2
4
1


3
5
6
7
2
4
1


3
5
6
7
2
4
1


3
5
6
7
2
4
1