Théorie des graphes et applications

18 novembre 2013

Plan

Seront abordés les thèmes suivants :

- Généralités
- Coloration - Planarité
- Problème de l'arbre couvrant minimal
- Problème du plus court chemin
- Ordonnancement

Chapitre I – Généralités

Chapitre I – Généralités

- Définitions
- Chaînes et chemins
- Connexité (graphe non orienté)
- Forte connexité
- Représentation des graphes
- Distance dans un graphe non orienté
- Les 7 ponts de Königsberg

Chapitre I – Généralités

- Définitions
- Chaînes et chemins
- Connexité (graphe non orienté)
- Forte connexité
- Représentation des graphes
- Distance dans un graphe non orienté
- Les 7 ponts de Königsberg

Chapitre I – Généralités

- Définitions
- Chaînes et chemins
- Connexité (graphe non orienté)
- Forte connexité
- Représentation des graphes
- Distance dans un graphe non orienté
- Les 7 ponts de Königsberg

Chapitre I – Généralités

- Définitions
- Chaînes et chemins
- Connexité (graphe non orienté)
- Forte connexité
- Représentation des graphes
- Distance dans un graphe non orienté
- Les 7 ponts de Königsberg

Chapitre I – Généralités

- Définitions
- Chaînes et chemins
- Connexité (graphe non orienté)
- Forte connexité
- Représentation des graphes
- Distance dans un graphe non orienté
- Les 7 ponts de Königsberg

Chapitre I – Généralités

- Définitions
- Chaînes et chemins
- Connexité (graphe non orienté)
- Forte connexité
- Représentation des graphes
- Distance dans un graphe non orienté
- Les 7 ponts de Königsberg

Définitions

On ne considérera que des graphes *finis*

Graphe non orienté

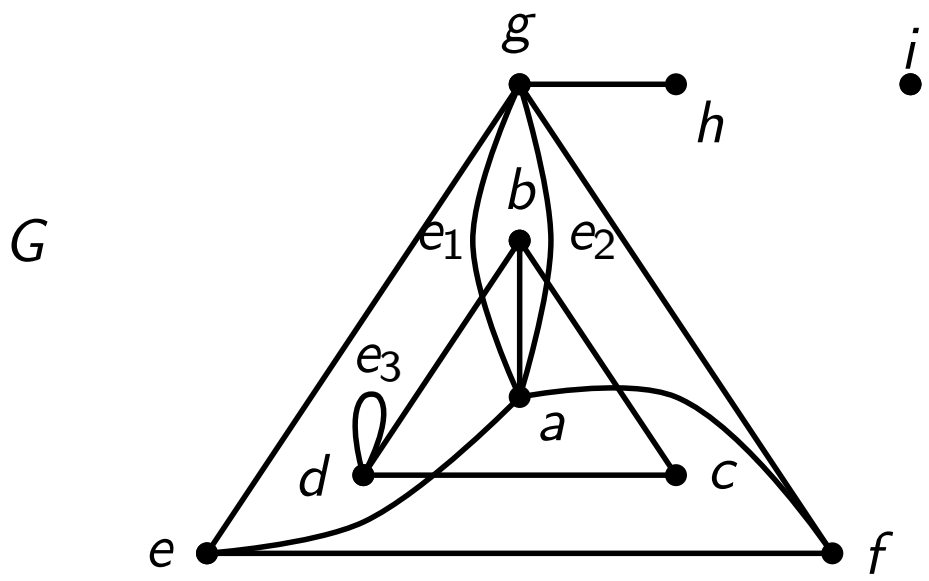
Définition

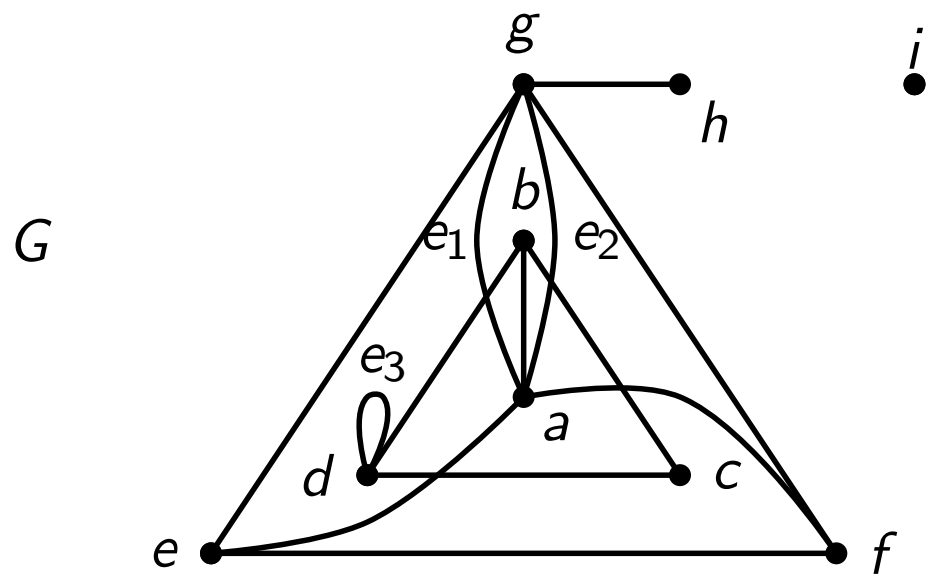
Un graphe non orienté est défini par la donnée

- d'un ensemble fini X de *sommets*
- d'un ensemble fini E d'*arêtes*
- une application qui à chaque arête associe deux sommets non nécessairement distincts

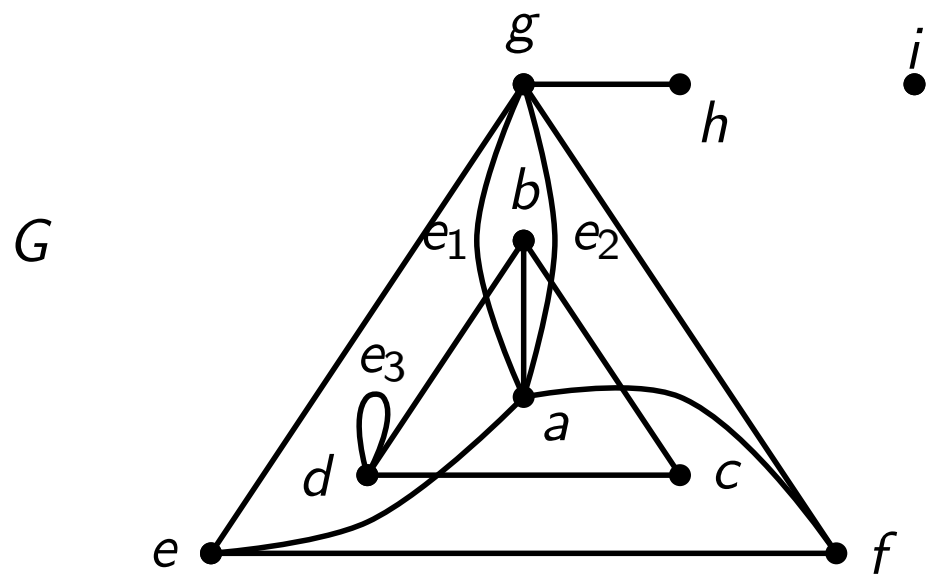
Définition

L'ordre d'un graphe est le nombre de sommet du graphe



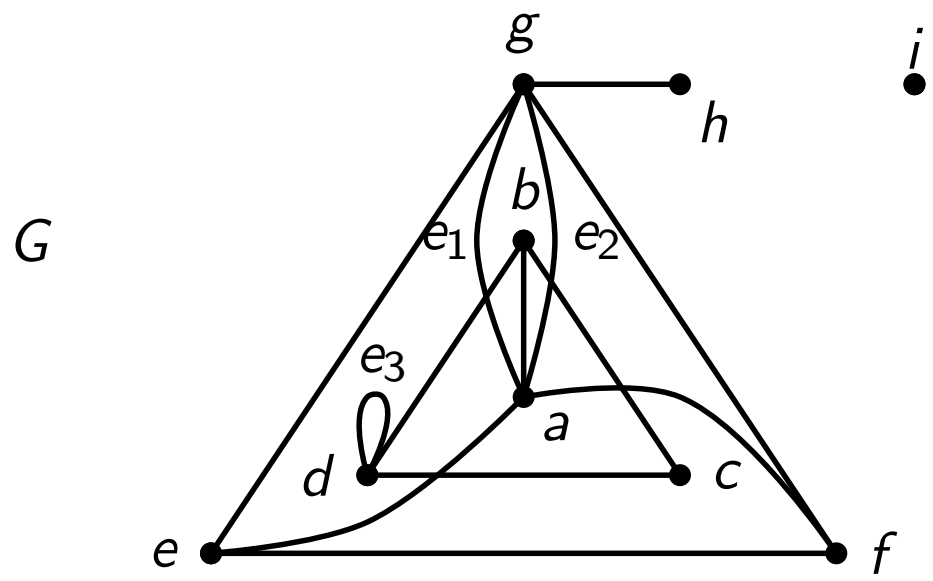


Toutes les arêtes n'ont pas été nommées

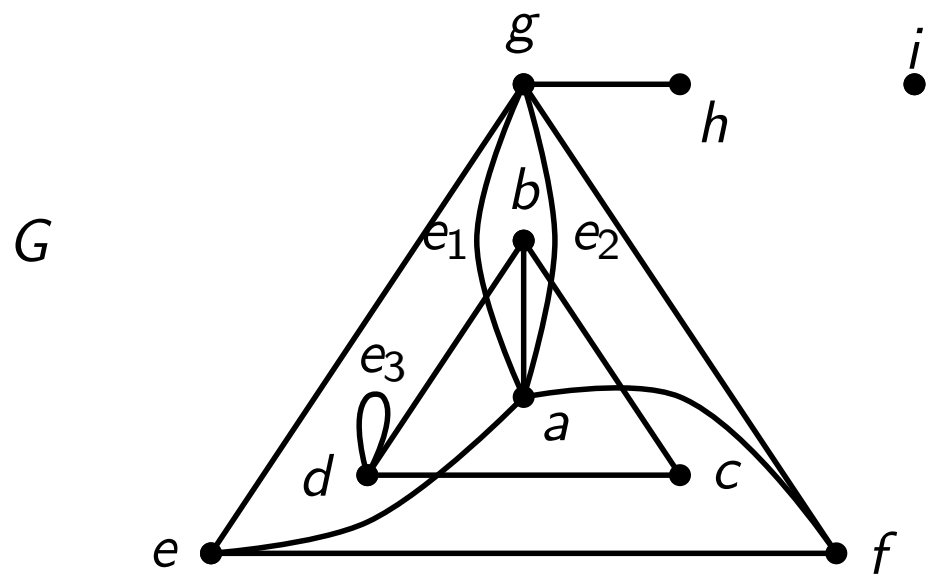


Toutes les arêtes n'ont pas été nommées

a et b sont *adjacents* ou voisins



Toutes les arêtes n'ont pas été nommées
a et b sont *adjacents* ou voisins
l'arête $\{eg\}$ est *incidente* à e et à g

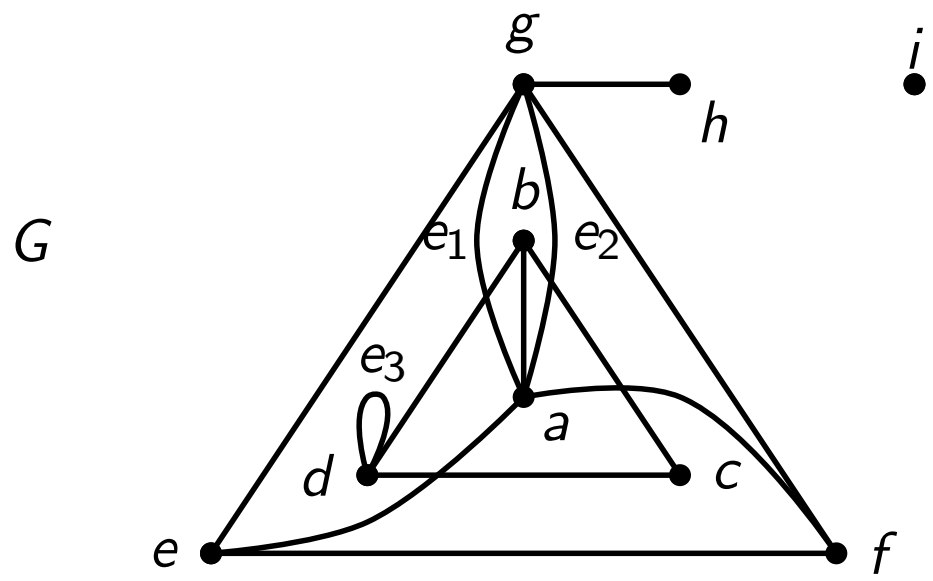


Toutes les arêtes n'ont pas été nommées

a et b sont *adjacents* ou voisins

l'arête $\{eg\}$ est *incidente* à e et à g

le sommet h est *pendant*, le sommet i est *isolé*



Toutes les arêtes n'ont pas été nommées
a et b sont *adjacents* ou voisins
l'arête $\{eg\}$ est *incidente* à e et à g
le sommet h est *pendant*, le sommet i est *isolé*
e_1 et e_2 sont *parallèles*, e_3 est une *boucle*

Graphe non orienté

Définition

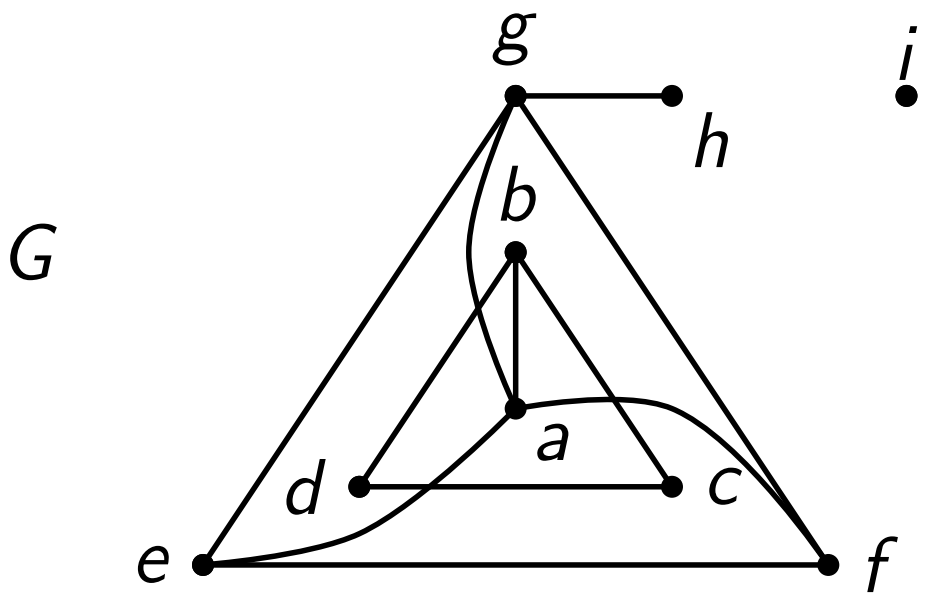
Un graphe non orienté est simple s'il ne contient

- ni boucle
- ni arête parallèle

Propriété

Soit G un graphe simple d'ordre n. Le nombre d'arêtes m de G vérifie

$$m \leq \frac{n(n-1)}{2}$$



G est simple

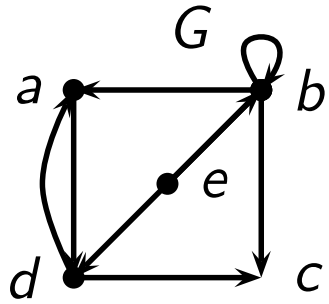
Graphe orienté

Définition

Un graphe orienté est défini par la donnée

- d'un ensemble fini X de *sommets*
- d'un ensemble fini E d'*arcs*
- une application qui à chaque arc associe un couple de sommets (x, y) non nécessairement distincts ; x est l'origine de l'arc et y est le but de l'arc.

Dans G d est un *prédécesseur* de c, c est un *successeur* de d. Il y a une boucle sur le sommet b donc G n'est pas simple. Le sommet e est une *source* car il n'a aucun prédécesseur, c est un *puits* car il n'a aucun successeur.



Isomorphismes de graphes

Définition

les graphes H et G sont isomorphes s'il existe une bijection entre les ensembles de sommets des deux graphes qui respectent les relations de voisinage.

Isomorphismes de graphes

Définition

les graphes H et G sont isomorphes s'il existe une bijection entre les ensembles de sommets des deux graphes qui respectent les relations de voisinage.

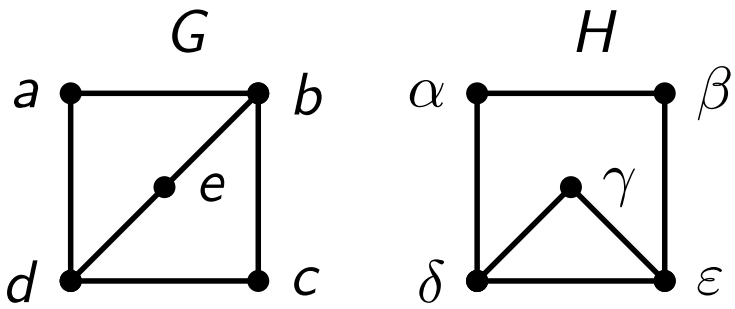
Autrement dit, si le sommet a de G correspond au sommet α de H, alors les correspondants des voisins de a dans G doivent être les voisins de α dans H et réciproquement.

Isomorphismes de graphes

Définition

les graphes H et G sont isomorphes s'il existe une bijection entre les ensembles de sommets des deux graphes qui respectent les relations de voisinage.

Autrement dit, si le sommet a de G correspond au sommet α de H, alors les correspondants des voisins de a dans G doivent être les voisins de α dans H et réciproquement.
Les deux graphes ci-dessous sont-ils isomorphes ?

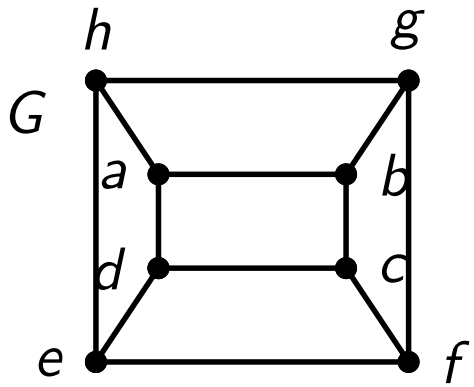


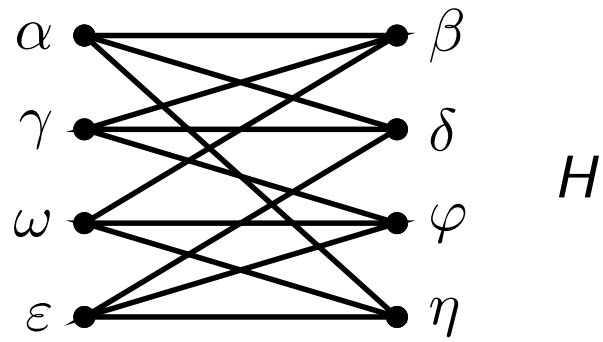
Isomorphismes de graphes

Définition

les graphes H et G sont isomorphes s'il existe une bijection entre les ensembles de sommets des deux graphes qui respectent les relations de voisinage.

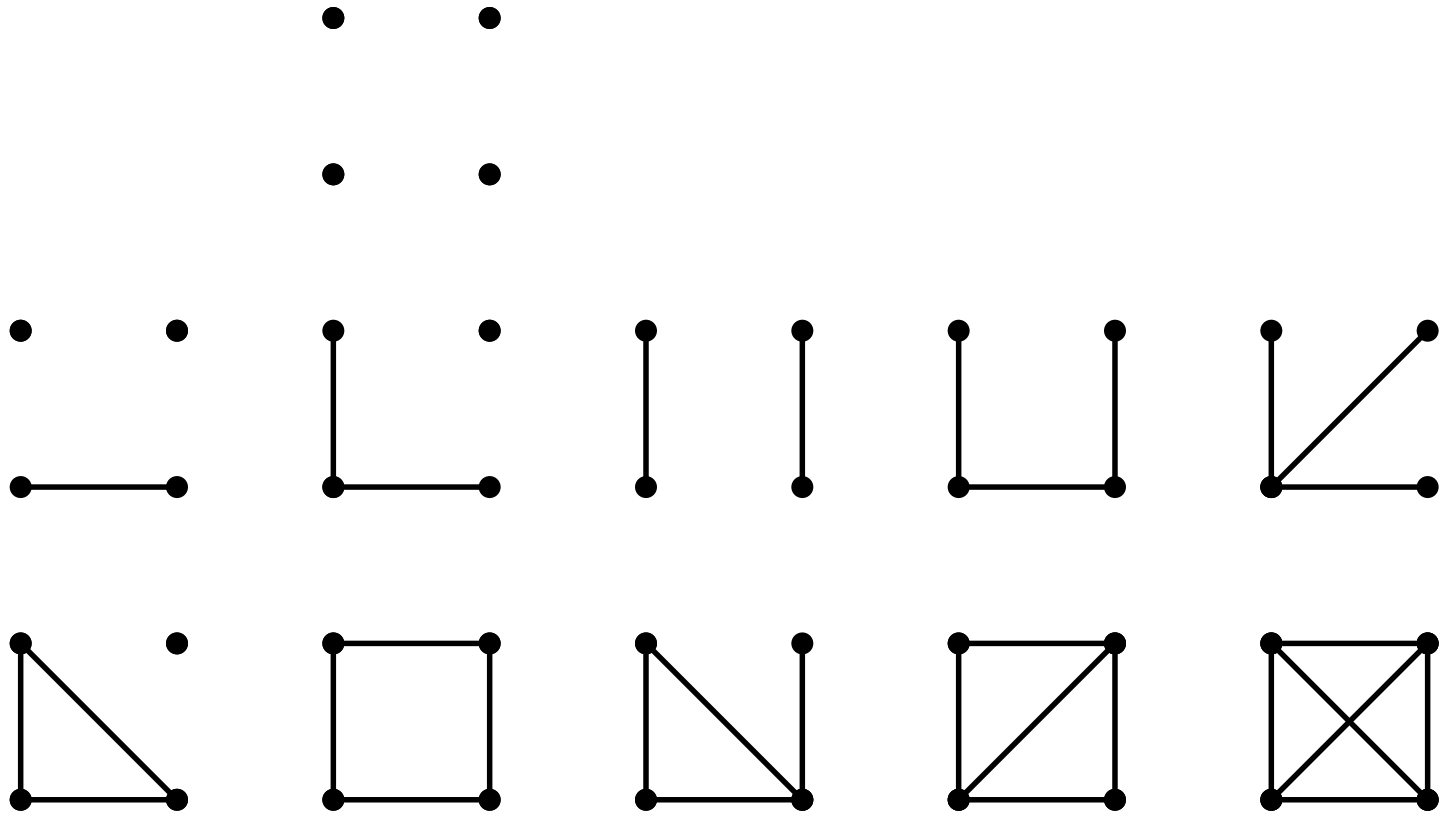
Autrement dit, si le sommet a de G correspond au sommet α de H, alors les correspondants des voisins de a dans G doivent être les voisins de α dans H et réciproquement.
Les deux graphes ci-dessous sont isomorphes.





Trouver les 11 graphes simples d'ordre 4 non isomorphes deux à deux.

Trouver les 11 graphes simples d'ordre 4 non isomorphes deux à deux.



Sous-graphe

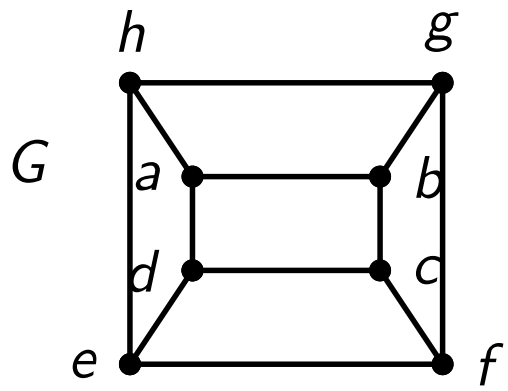
Définition

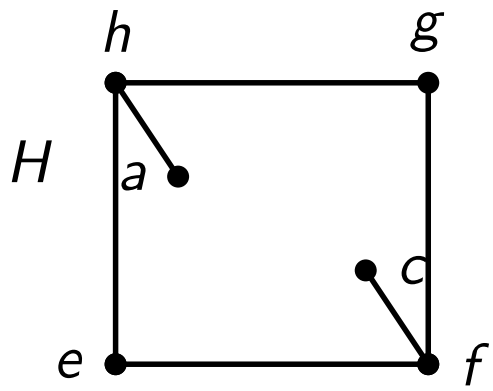
Un sous- graphe est obtenu en supprimant des sommets et les arêtes (ou arcs) qui lui sont incidentes.

Remarque

dans le cas général, un sous-graphe a moins de sommets et moins d'arêtes(ou arcs).

H est un sous-graphe de G, $H = G \setminus \{b, d\}$





Graphe partiel

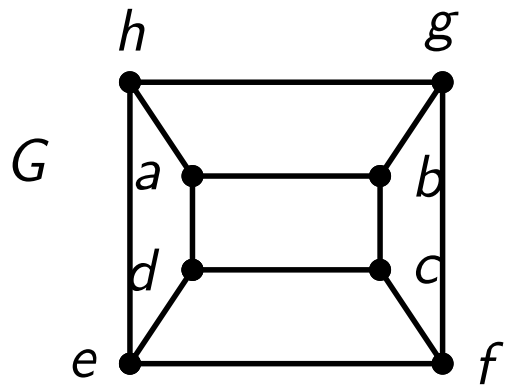
Définition

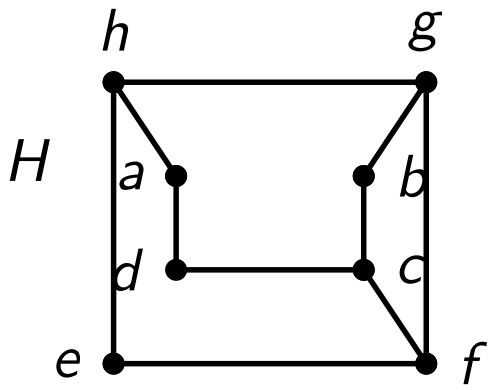
Un graphe partiel est obtenu en supprimant des arêtes (ou arcs)

Remarque

un graphe partiel a exactement les mêmes sommets

H est un graphe partiel de G, $H = G \setminus \{\{a, b\}, \{d, e\}\}$





Degrés dans un graphe non orienté

Définition

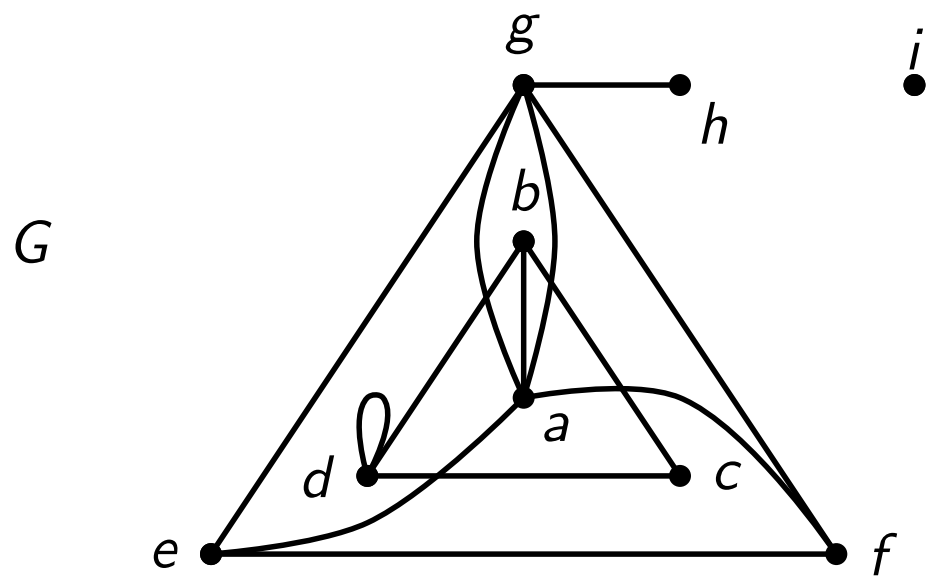
Dans un graphe non orienté, le degré d'un sommet x, noté $d(x)$, est égal au nombre d'arêtes incidents à x (une boucle étant 2 fois incidente)

Degrés dans un graphe non orienté

Définition

Dans un graphe non orienté, le degré d'un sommet x, noté $d(x)$, est égal au nombre d'arêtes incidents à x (une boucle étant 2 fois incidente)

a est de degré 5, i est de degré 0, h est de degré 1, d est de degré 4



Degrés dans un graphe non orienté

Propriéte

Soit G un graphe non orienté et m son nombre d'arêtes.

$$\sum_{x \in X} d(x) = 2m$$

Degrés dans un graphe non orienté

Propriéte

Soit G un graphe non orienté et m son nombre d'arêtes.

$$\sum_{x \in X} d(x) = 2m$$

En effet chaque arête est comptée exactement deux fois : une fois pour chaque extrêmité. C'est aussi valable pour une boucle qui est deux fois incidente à un sommet.

Degrés dans un graphe non orienté

Propriéte

Soit G un graphe non orienté et m son nombre d'arêtes.

$$\sum_{x \in X} d(x) = 2m$$

En effet chaque arête est comptée exactement deux fois : une fois pour chaque extrêmité. C'est aussi valable pour une boucle qui est deux fois incidente à un sommet.

Conséquence

dans un graphe non orienté, le nombre de sommets de degré impair est *pair*

Degrés dans un graphe non orienté

Propriéte

Soit G un graphe non orienté et m son nombre d'arêtes.

$$\sum_{x \in X} d(x) = 2m$$

En effet chaque arête est comptée exactement deux fois : une fois pour chaque extrêmité. C'est aussi valable pour une boucle qui est deux fois incidente à un sommet.

Conséquence

dans un graphe non orienté, le nombre de sommets de degré impair est *pair*

Propriéte

Dans un graphe non orienté simple on a pour tout sommet x,

$$d(x) \leq (n-1)$$

Graphe complet

Définition

On appelle graphe complet d'ordre n, noté K_n, le graphe simple d'ordre n dont tout sommet est adjacent à tous les autres sommets (défini à un isomorphisme près)

Graphe complet

Définition

On appelle graphe complet d'ordre n, noté K_n, le graphe simple d'ordre n dont tout sommet est adjacent à tous les autres sommets (défini à un isomorphisme près)

K_n a $\frac{n(n-1)}{2}$ arêtes. Chacun de ses sommets est de degré $n-1$

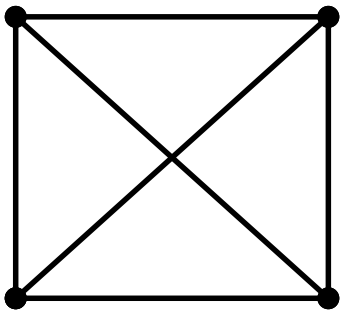
Graphe complet

Définition

On appelle graphe complet d'ordre n, noté K_n, le graphe simple d'ordre n dont tout sommet est adjacent à tous les autres sommets (défini à un isomorphisme près)

K_n a $\frac{n(n-1)}{2}$ arêtes. Chacun de ses sommets est de degré $n-1$

K_4



Degrés dans un graphe orienté

Définition

Dans un graphe orienté,

- on appelle degré entrant d'un sommet x, noté $d^-(x)$, le nombre d'arcs de but x.
- on appelle degré sortant d'un sommet x, noté $d^+(x)$, le nombre d'arcs d'origine x.

Degrés dans un graphe orienté

Définition

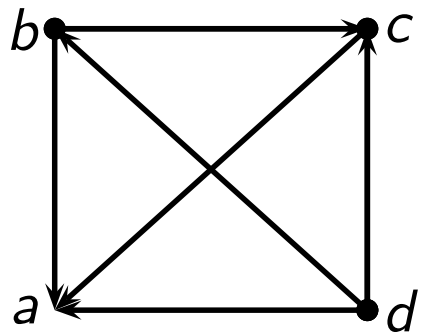
Dans un graphe orienté,

- on appelle degré entrant d'un sommet x, noté $d^-(x)$, le nombre d'arcs de but x.
- on appelle degré sortant d'un sommet x, noté $d^+(x)$, le nombre d'arcs d'origine x.

exemple :

$d^-(a) = 3, d^-(b) = 1, d^-(c) = 2, d^-(d) = 0$

$d^+(a) = 0, d^+(b) = 2, d^+(c) = 1, d^+(d) = 3$



Degrés dans un graphe orienté

Définition

Dans un graphe orienté,

- on appelle degré entrant d'un sommet x, noté $d^-(x)$, le nombre d'arcs de but x.
- on appelle degré sortant d'un sommet x, noté $d^+(x)$, le nombre d'arcs d'origine x.

Propriété

Dans un graphe orienté, à m arcs, on a

$$\sum_{x \in X} d^-(x) = \sum_{x \in X} d^+(x) = m$$

Définition

On dit qu'une suite d'entiers naturels $k_1, \cdots, k_n$ est graphique s'il existe un graphe non orienté **simple** d'ordre n tel que ses sommets $x_1, \cdots, x_n$ ont respectivement les degrés $k_1, \cdots, k_n$.

exemple : la suite $7, 7, 5, 5, 4, 4, 3, 2$ n'est pas graphique car ...

Définition

On dit qu'une suite d'entiers naturels $k_1, \cdots, k_n$ est graphique s'il existe un graphe non orienté **simple** d'ordre n tel que ses sommets $x_1, \cdots, x_n$ ont respectivement les degrés $k_1, \cdots, k_n$.

exemple : la suite $7, 7, 5, 5, 4, 4, 3, 2$ n'est pas graphique car ...

Théorème de Havel-Hakimi :

une suite **décroissante** d'entiers naturels $k_1 \geq \cdots \geq k_n$ avec $n \geq 2$ et $k_1 \geq 1$ est graphique si et seulement si la suite $k_2 - 1, k_3 - 1, \cdots, k_{k_1+1} - 1, k_{k_1+2}, \cdots k_n$ est graphique. (on a supprimé k_1 de la suite puis on a retranché 1 aux k_1 premiers termes)

Ce théorème fournit un algorithme pour déterminer si une suite est graphique ou non.

Chaînes et chemins

Définition

Dans un graphe non orienté.

- une *chaîne* est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = \{x_{i-1}, x_i\}$ pour $i = 1 \cdots k$. On dit que cette chaîne est de longueur k et relie x_0 à x_k.
- une chaîne est *simple* si elle ne contient pas deux fois la même arête.
- un *cycle* est une chaîne simple *fermée* c'est-à-dire $x_0 = x_k$.
- une chaîne est *élémentaire* si elle ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un cycle élémentaire).
- une chaîne est *eulérienne* si elle est simple et contient toutes les arêtes.
- une chaîne est *hamiltonienne* si elle est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe non orienté.

- une *chaîne* est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = \{x_{i-1}, x_i\}$ pour $i = 1 \cdots k$. On dit que cette chaîne est de longueur k et relie x_0 à x_k.
- une chaîne est *simple* si elle ne contient pas deux fois la même arête.
- un *cycle* est une chaîne simple *fermée* c'est-à-dire $x_0 = x_k$.
- une chaîne est *élémentaire* si elle ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un cycle élémentaire).
- une chaîne est *eulérienne* si elle est simple et contient toutes les arêtes.
- une chaîne est *hamiltonienne* si elle est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe non orienté.

- une *chaîne* est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = \{x_{i-1}, x_i\}$ pour $i = 1 \cdots k$. On dit que cette chaîne est de longueur k et relie x_0 à x_k.
- une chaîne est *simple* si elle ne contient pas deux fois la même arête.
- un *cycle* est une chaîne simple *fermée* c'est-à-dire $x_0 = x_k$.
- une chaîne est *élémentaire* si elle ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un cycle élémentaire).
- une chaîne est *eulérienne* si elle est simple et contient toutes les arêtes.
- une chaîne est *hamiltonienne* si elle est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe non orienté.

- une *chaîne* est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = \{x_{i-1}, x_i\}$ pour $i = 1 \cdots k$. On dit que cette chaîne est de longueur k et relie x_0 à x_k.
- une chaîne est *simple* si elle ne contient pas deux fois la même arête.
- un *cycle* est une chaîne simple *fermée* c'est-à-dire $x_0 = x_k$.
- une chaîne est *élémentaire* si elle ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un cycle élémentaire).
- une chaîne est *eulérienne* si elle est simple et contient toutes les arêtes.
- une chaîne est *hamiltonienne* si elle est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe non orienté.

- une *chaîne* est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = \{x_{i-1}, x_i\}$ pour $i = 1 \cdots k$. On dit que cette chaîne est de longueur k et relie x_0 à x_k.
- une chaîne est *simple* si elle ne contient pas deux fois la même arête.
- un *cycle* est une chaîne simple *fermée* c'est-à-dire $x_0 = x_k$.
- une chaîne est *élémentaire* si elle ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un cycle élémentaire).
- une chaîne est *eulérienne* si elle est simple et contient toutes les arêtes.
- une chaîne est *hamiltonienne* si elle est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe non orienté.

- une *chaîne* est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = \{x_{i-1}, x_i\}$ pour $i = 1 \cdots k$. On dit que cette chaîne est de longueur k et relie x_0 à x_k.
- une chaîne est *simple* si elle ne contient pas deux fois la même arête.
- un *cycle* est une chaîne simple *fermée* c'est-à-dire $x_0 = x_k$.
- une chaîne est *élémentaire* si elle ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un cycle élémentaire).
- une chaîne est *eulérienne* si elle est simple et contient toutes les arêtes.
- une chaîne est *hamiltonienne* si elle est élémentaire et contient tous les sommets.

Remarque

une chaîne élémentaire est simple mais le contraire n'est pas nécessairement vrai.

Propriété

si x et y sont reliées par une chaîne alors il existe une chaîne élémentaire qui relie x à y.

Chaînes et chemins

Définition

Dans un graphe orienté.

- un chemin est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = (x_{i-1}, x_i)$ pour $i = 1 \cdots k$. On dit que ce chemin est de longueur k et relie x_0 à x_k.
- un chemin est *simple* s'il ne contient pas deux fois le même arc.
- un *circuit* est un chemin simple *fermé* c'est-à-dire $x_0 = x_k$.
- un chemin est *élémentaire* s'il ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un circuit élémentaire).
- un chemin est *eulérien* s'il est simple et contient toutes les arcs.
- un chemin est *hamiltonien* s'il est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe orienté.

- un chemin est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = (x_{i-1}, x_i)$ pour $i = 1 \cdots k$. On dit que ce chemin est de longueur k et relie x_0 à x_k.
- un chemin est *simple* s'il ne contient pas deux fois le même arc.
- un *circuit* est un chemin simple *fermé* c'est-à-dire $x_0 = x_k$.
- un chemin est *élémentaire* s'il ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un circuit élémentaire).
- un chemin est *eulérien* s'il est simple et contient toutes les arcs.
- un chemin est *hamiltonien* s'il est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe orienté.

- un chemin est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = (x_{i-1}, x_i)$ pour $i = 1 \cdots k$. On dit que ce chemin est de longueur k et relie x_0 à x_k.
- un chemin est *simple* s'il ne contient pas deux fois le même arc.
- un *circuit* est un chemin simple *fermé* c'est-à-dire $x_0 = x_k$.
- un chemin est *élémentaire* s'il ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un circuit élémentaire).
- un chemin est *eulérien* s'il est simple et contient toutes les arcs.
- un chemin est *hamiltonien* s'il est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe orienté.

- un chemin est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = (x_{i-1}, x_i)$ pour $i = 1 \cdots k$. On dit que ce chemin est de longueur k et relie x_0 à x_k.
- un chemin est *simple* s'il ne contient pas deux fois le même arc.
- un *circuit* est un chemin simple *fermé* c'est-à-dire $x_0 = x_k$.
- un chemin est *élémentaire* s'il ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un circuit élémentaire).
- un chemin est *eulérien* s'il est simple et contient toutes les arcs.
- un chemin est *hamiltonien* s'il est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe orienté.

- un chemin est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = (x_{i-1}, x_i)$ pour $i = 1 \cdots k$. On dit que ce chemin est de longueur k et relie x_0 à x_k.
- un chemin est *simple* s'il ne contient pas deux fois le même arc.
- un *circuit* est un chemin simple *fermé* c'est-à-dire $x_0 = x_k$.
- un chemin est *élémentaire* s'il ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un circuit élémentaire).
- un chemin est *eulérien* s'il est simple et contient toutes les arcs.
- un chemin est *hamiltonien* s'il est élémentaire et contient tous les sommets.

Chaînes et chemins

Définition

Dans un graphe orienté.

- un chemin est une suite $x_0 e_1 x_1 e_2 x_2 \cdots e_k x_k$ avec $e_i = (x_{i-1}, x_i)$ pour $i = 1 \cdots k$. On dit que ce chemin est de longueur k et relie x_0 à x_k.
- un chemin est *simple* s'il ne contient pas deux fois le même arc.
- un *circuit* est un chemin simple *fermé* c'est-à-dire $x_0 = x_k$.
- un chemin est *élémentaire* s'il ne contient pas deux fois le même sommet (sauf aux extrêmités si c'est un circuit élémentaire).
- un chemin est *eulérien* s'il est simple et contient toutes les arcs.
- un chemin est *hamiltonien* s'il est élémentaire et contient tous les sommets.

Connexité

Définition

La relation '≪ être reliés par une chaîne ≫ est une relation d'équivalence sur l'ensemble des sommets de G d'un graphe non orienté. Les sous-graphes engendrés par cette relation sont les *composantes connexes* de G (ce sont les sous-graphes *maximaux* par rapport à cette relation).

La *composante connexe* $CC(x)$ *du sommet* x est alors l'ensemble des sommets de G reliés à x par une chaîne sachant que x est relié à lui- même par une chaîne de longueur nulle. On dit que G est un graphe non orienté *connexe* s'il n'a qu'une composante connexe c'est-à-dire si deux sommets quelconques de G sont reliés par une chaîne.

Propriété

Soit G un graphe non orienté d'ordre n à m arêtes. Si G est connexe alors $m \geq n - 1$.

Preuve

Par récurrence sur n, en supposant G simple (sans perte de généralité)

- pour $n = 1$, c'est clair
- soit $n \geq 1$, supposons la propriété vraie pour tous les graphes non orientés d'ordre au plus n et considérons un graphe connexe G d'ordre $n+1$. On considérera deux cas :

1. il existe une arête e tel que $G \setminus e$ n'est plus connexe,
2. pour toute arête e, $G \setminus e$ est connexe.

Preuve

Par récurrence sur n, en supposant G simple (sans perte de généralité)

- pour $n = 1$, c'est clair
- soit $n \geq 1$, supposons la propriété vraie pour tous les graphes non orientés d'ordre au plus n et considérons un graphe connexe G d'ordre $n+1$. On considérera deux cas :

1. il existe une arête e tel que $G \setminus e$ n'est plus connexe,
2. pour toute arête e, $G \setminus e$ est connexe.

Preuve

Par récurrence sur n, en supposant G simple (sans perte de généralité)

- pour $n = 1$, c'est clair
- soit $n \geq 1$, supposons la propriété vraie pour tous les graphes non orientés d'ordre au plus n et considérons un graphe connexe G d'ordre $n+1$. On considérera deux cas :

1. il existe une arête e tel que $G \setminus e$ n'est plus connexe,
2. pour toute arête e, $G \setminus e$ est connexe.

Preuve

Par récurrence sur n, en supposant G simple (sans perte de généralité)

- pour $n = 1$, c'est clair
- soit $n \geq 1$, supposons la propriété vraie pour tous les graphes non orientés d'ordre au plus n et considérons un graphe connexe G d'ordre $n+1$. On considérera deux cas :

1. il existe une arête e tel que $G \setminus e$ n'est plus connexe,
2. pour toute arête e, $G \setminus e$ est connexe.

1er cas

soit e une arête de G telle que $G \setminus e$ n'est plus connexe : alors $G \setminus e$ a exactement 2 composantes connexes.

1er cas

soit e une arête de G telle que $G \setminus e$ n'est plus connexe : alors $G \setminus e$ a exactement 2 composantes connexes.
En effet si $e = \{x, y\}$ et $G \setminus e$ non connexe, alors, sachant que G est connexe, on peut partitionner les sommets de G en deux parties : les sommets reliés à x par une chaîne ne passant pas par e et les sommets reliés à y par une chaîne ne passant pas par e. Ce sont les deux composantes connexes de $G \setminus e$.

1er cas

soit e une arête de G telle que $G \setminus e$ n'est plus connexe : alors $G \setminus e$ a exactement 2 composantes connexes.
En effet si $e = \{x, y\}$ et $G \setminus e$ non connexe, alors, sachant que G est connexe, on peut partitionner les sommets de G en deux parties : les sommets reliés à x par une chaîne ne passant pas par e et les sommets reliés à y par une chaîne ne passant pas par e. Ce sont les deux composantes connexes de $G \setminus e$.
Soient alors C_1, C_2 les deux composantes connexes de $G \setminus e$.
L'ordre n_i de C_i est au plus égal à n pour $i = 1, 2$ et $n_1 + n_2 = n + 1$.
Par hypothèse de récurrence, C_i a donc au moins $n_i - 1$ arêtes.

1er cas

soit e une arête de G telle que $G \setminus e$ n'est plus connexe : alors $G \setminus e$ a exactement 2 composantes connexes.
En effet si $e = \{x, y\}$ et $G \setminus e$ non connexe, alors, sachant que G est connexe, on peut partitionner les sommets de G en deux parties : les sommets reliés à x par une chaîne ne passant pas par e et les sommets reliés à y par une chaîne ne passant pas par e. Ce sont les deux composantes connexes de $G \setminus e$.
Soient alors C_1, C_2 les deux composantes connexes de $G \setminus e$.
L'ordre n_i de C_i est au plus égal à n pour $i = 1, 2$ et $n_1 + n_2 = n + 1$.
Par hypothèse de récurrence, C_i a donc au moins $n_i - 1$ arêtes.
Donc $G \setminus e$ a au moins $(n_1 - 1) + (n_2 - 1) = n + 1 - 2 = n - 1$ arêtes. Donc G a au moins n arêtes.

2ème cas

soit $k > 1$ le plus petit entier tel qu'il existe k arêtes $e_1, \dots e_k$ vérifiant $G \setminus \{e_1, \dots e_{k-1}\}$ est connexe et $G \setminus \{e_1, \dots e_k\}$ n'est pas connexe. Alors en appliquant le même raisonnement que précédemment $G \setminus \{e_1, \dots e_k\}$ a 2 composantes connexes et a au moins $n - 1$ arêtes.
Donc G a au moins $n - 1 + k \geq n + 1 > n$ arêtes. □

Forte connexité

Dans le cadre des graphes orientés, la relation "être reliés par un chemin" n'est pas symétrique. On définit une notion plus forte que la simple connexité :

Définition

un graphe orienté G est *fortement connexe* si pour tous sommets $x \neq y$ de G, il existe un chemein de x à y et un chemin de y à x.

Définition

Une composante fortement connexe d'un graphe orienté est un sous-graphe fortement connexe maximal pour cette propriété.

exemple : un circuit est un graphe fortement connexe.

Représentation des graphes

Il existe de nombreuses façons de représenter les graphes comme une structure de données. On en verra deux :

Représentation des graphes

Il existe de nombreuses façons de représenter les graphes comme une structure de données. On en verra deux :

matrice d'adjacence : le graphe est réprésenté par une matrice carrée $n \times n$ A, à coefficients dans $\mathbb{N}$ indexée par les sommets du graphe et telle que $A_{i,j}$ est égal au nombre d'arêtes (ou arcs) du sommet i au sommet j

Représentation des graphes

Il existe de nombreuses façons de représenter les graphes comme une structure de données. On en verra deux :

matrice d'adjacence : le graphe est réprésenté par une matrice carrée $n \times n$ A, à coefficients dans $\mathbb{N}$ indexée par les sommets du graphe et telle que $A_{i,j}$ est égal au nombre d'arêtes (ou arcs) du sommet i au sommet j

matrice d'incidence : le graphe sans boucle est réprésenté par une matrice $n \times m$ M indexée par les sommets et les arêtes et telle que $M_{i,e_j} = 1$ si i est l'origine de e_j, $M_{i,e_j} = -1$ si i est le but de e_j, $M_{i,e_j} = 0$ sinon

Représentation des graphes

Il existe de nombreuses façons de représenter les graphes comme une structure de données. On en verra deux :

matrice d'adjacence : le graphe est réprésenté par une matrice carrée $n \times n$ A, à coefficients dans $\mathbb{N}$ indexée par les sommets du graphe et telle que $A_{i,j}$ est égal au nombre d'arêtes (ou arcs) du sommet i au sommet j

matrice d'incidence : le graphe sans boucle est réprésenté par une matrice $n \times m$ M indexée par les sommets et les arêtes et telle que $M_{i,e_j} = 1$ si i est l'origine de e_j, $M_{i,e_j} = -1$ si i est le but de e_j, $M_{i,e_j} = 0$ sinon

matrice d'adjacence

pour une matrice d'adjacence, on peut noter :

- la taille d'une matrice d'adjacence est n^2
- si le graphe est non orienté, alors sa matrice d'adjacence est *symétrique*
- le présence et l'absence d'arêtes sont représentées : beaucoup d'espaces mémoire utilisés pour rien
- si le graphe est simple on peut se contenter d'une matrice d'adjacence à coefficients dans $\{0, 1\}$

matrice d'adjacence

pour une matrice d'adjacence, on peut noter :

- la taille d'une matrice d'adjacence est n^2
- si le graphe est non orienté, alors sa matrice d'adjacence est *symétrique*
- le présence et l'absence d'arêtes sont représentées : beaucoup d'espaces mémoire utilisés pour rien
- si le graphe est simple on peut se contenter d'une matrice d'adjacence à coefficients dans $\{0, 1\}$

matrice d'adjacence

pour une matrice d'adjacence, on peut noter :

- la taille d'une matrice d'adjacence est n^2
- si le graphe est non orienté, alors sa matrice d'adjacence est *symétrique*
- le présence et l'absence d'arêtes sont représentées : beaucoup d'espaces mémoire utilisés pour rien
- si le graphe est simple on peut se contenter d'une matrice d'adjacence à coefficients dans $\{0, 1\}$

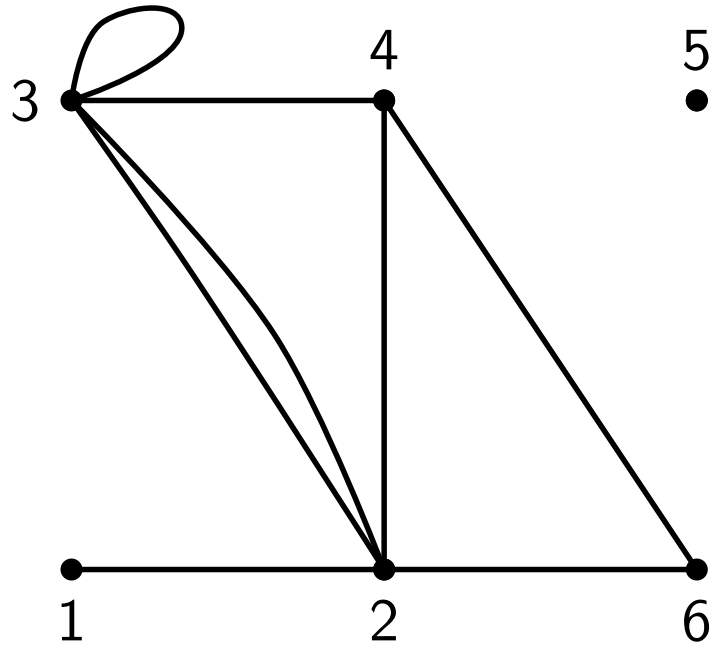
matrice d'adjacence

pour une matrice d'adjacence, on peut noter :

- la taille d'une matrice d'adjacence est n^2
- si le graphe est non orienté, alors sa matrice d'adjacence est *symétrique*
- le présence et l'absence d'arêtes sont représentées : beaucoup d'espaces mémoire utilisés pour rien
- si le graphe est simple on peut se contenter d'une matrice d'adjacence à coefficients dans $\{0, 1\}$

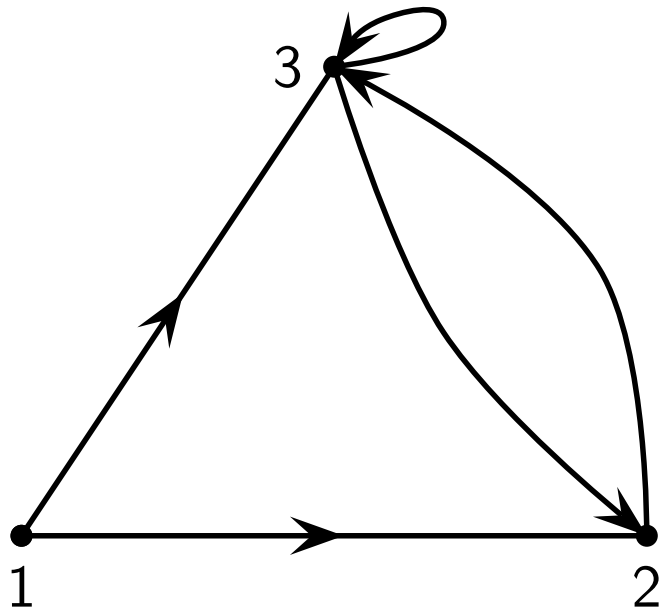
La matrice d'adjacence suivante représente le graphe non orienté ci-dessous

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 & 1 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$



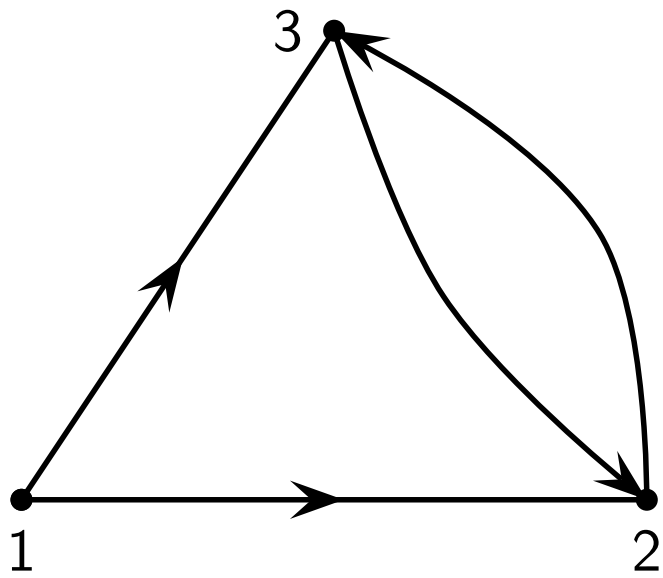
La matrice d'adjacence suivante représente le graphe orienté ci-dessous

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$



La matrice d'incidence suivante représente le graphe orienté ci-dessous

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \end{pmatrix}$$



Distance dans un graphe non orienté

Définition

La *distance* $d(x, y)$ entre deux sommets x, y d'un graphe non orienté G est défini par :

- $d(x, y) = 0$ si $x = y$.
- $d(x, y)$ est la longueur d'une plus courte chaîne reliant x à y s'il existe une telle chaîne
- $d(x, y) = \infty$ sinon

Propriété

Pour tous sommets x, y, z

- $d(x, x) = 0$
- $d(x, y) = d(y, x)$
- $d(x, y) \leq d(x, z) + d(y, z)$

Diamètre d'un graphe non orienté

Définition

Le diamètre d'un graphe non orienté est le maximum des distances entre deux sommets de ce graphe.

Par exemple, le diamètre d'un graphe non connexe est infini. Le diamètre d'un graphe complet est 1.

Les 7 ponts de Königsberg

Cet exemple historique est important puisque sa résolution par L.Euler initia la théorie des graphes.

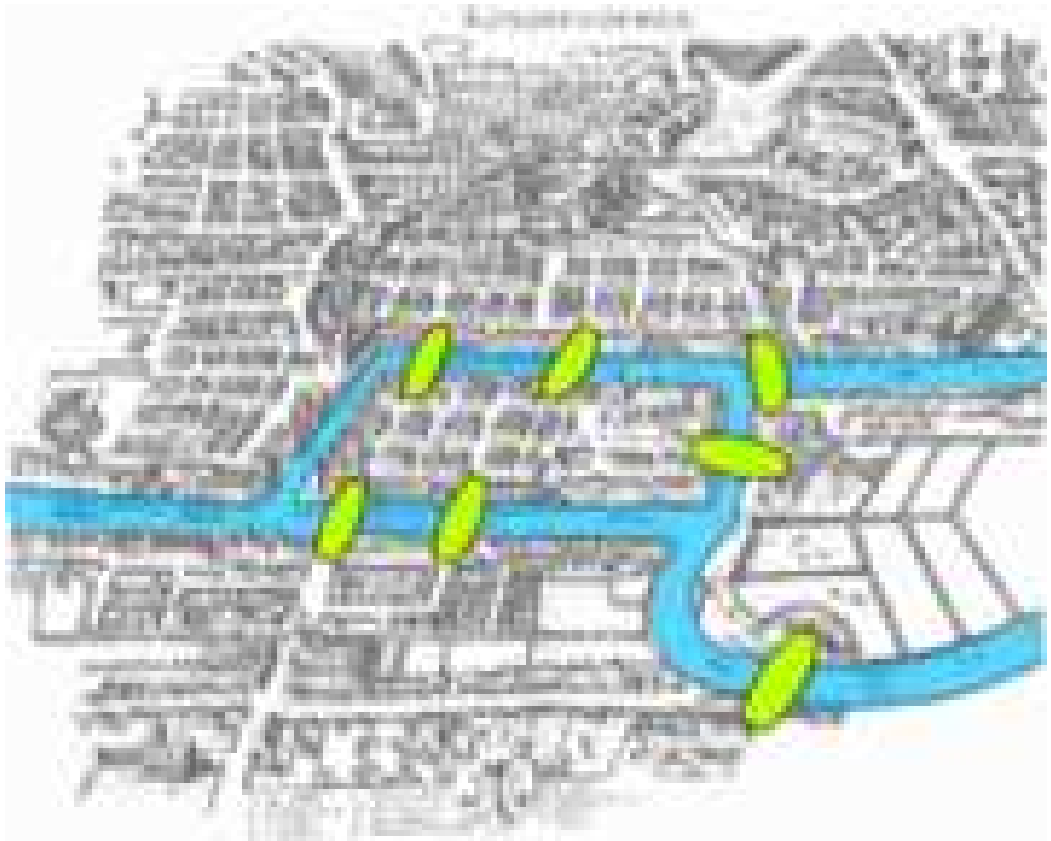


FIGURE: les sept ponts de Königsberg

Les 7 ponts de Königsberg

Le problème posé était le suivant : peut-on visiter toute la ville en emprutant chaque pont une et une seule fois ?

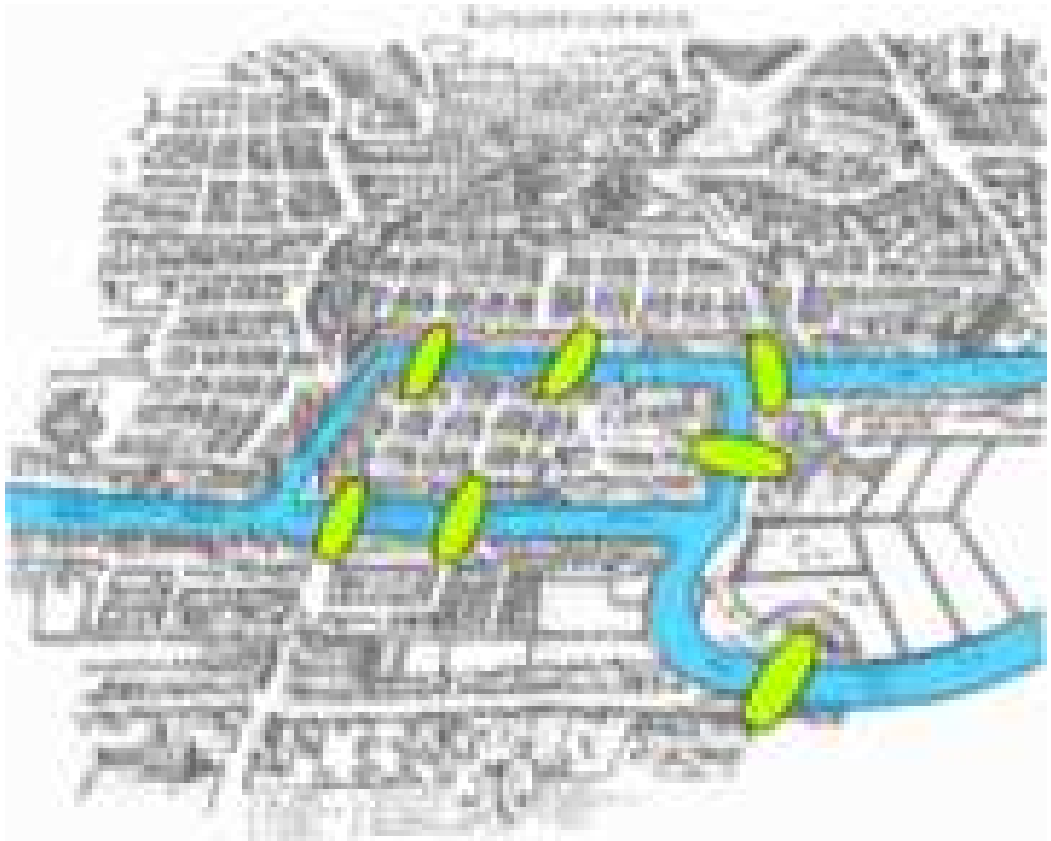


FIGURE: les sept ponts de Königsberg

Les 7 ponts de Königsberg

La réponse est non et L.Euler montra pourquoi en établissant le théorème suivant :

Théorème (Euler)

Soit G un graphe non orienté connexe. G a une chaîne eulérienne *si et seulement si* G a 0 ou 2 sommets de degré impair.

Ce théorème explique pourquoi on peut dessiner d'un seul trait sans repasser la maison avec un toit mais qu'il est impossible de le faire pour la maison sans toit.

Preuve $\Rightarrow$

Supposons que G a une chaîne eulérienne c. On notera $c = \alpha - e_0 - x_1 - e_1 - \ldots e_{k-1} - x_k - e_k - \omega$
Comme G est connexe, chaque sommet a au moins une occurrence dans c. On notera $n(x)$ le nombre d'occurrences d'un sommet x dans c.
On considère deux cas pour c :

1. c est un cycle : chaque occurrence d'un sommet x dans c est entourée de deux arêtes incidentes à x ; puisque toutes les arêtes sont présentes une et une seule fois dans c, le degré de est $d(x) = 2 \times n(x)$. Donc $d(x)$ est pair.
2. c n'est pas un cycle : on distinguera les extrêmités α et ω de c et les autres sommets. Pour les autres sommets on peut raisonner comme dans le cas précédent. Pour α, les arêtes incidentes sont e_0 et e_{i-1}, e_i pour $x_i = \alpha$, donc $d(\alpha)$ est impair. De même pour ω.

Preuve $\Rightarrow$

Supposons que G a une chaîne eulérienne c. On notera $c = \alpha - e_0 - x_1 - e_1 - \ldots e_{k-1} - x_k - e_k - \omega$
Comme G est connexe, chaque sommet a au moins une occurrence dans c. On notera $n(x)$ le nombre d'occurrences d'un sommet x dans c.
On considère deux cas pour c :

1. c est un cycle : chaque occurrence d'un sommet x dans c est entourée de deux arêtes incidentes à x ; puisque toutes les arêtes sont présentes une et une seule fois dans c, le degré de est $d(x) = 2 \times n(x)$. Donc $d(x)$ est pair.
2. c n'est pas un cycle : on distinguera les extrêmités α et ω de c et les autres sommets. Pour les autres sommets on peut raisonner comme dans le cas précédent. Pour α, les arêtes incidentes sont e_0 et e_{i-1}, e_i pour $x_i = \alpha$, donc $d(\alpha)$ est impair. De même pour ω.

Preuve $\Rightarrow$

Supposons que G a une chaîne eulérienne c. On notera $c = \alpha - e_0 - x_1 - e_1 - \ldots e_{k-1} - x_k - e_k - \omega$
Comme G est connexe, chaque sommet a au moins une occurrence dans c. On notera $n(x)$ le nombre d'occurrences d'un sommet x dans c.
On considère deux cas pour c :

1. c est un cycle : chaque occurrence d'un sommet x dans c est entourée de deux arêtes incidentes à x ; puisque toutes les arêtes sont présentes une et une seule fois dans c, le degré de est $d(x) = 2 \times n(x)$. Donc $d(x)$ est pair.
2. c n'est pas un cycle : on distinguera les extrêmités α et ω de c et les autres sommets. Pour les autres sommets on peut raisonner comme dans le cas précédent. Pour α, les arêtes incidentes sont e_0 et e_{i-1}, e_i pour $x_i = \alpha$, donc $d(\alpha)$ est impair. De même pour ω.

Preuve $\Rightarrow$

Donc si G a un cycle eulérien, tous ses sommets sont de degré pair, sinon G aune chaîne eulérienne et il y a exactement 2 sommets de degré impair.

Preuve $\Leftarrow$

Supposons que tous les sommets de G sont pairs. Soit x_0 un sommet de G. On considère une chaîne simple la plus longue possible en partant de x_0 : une telle chaîne est nécessairement un cycle c_0, sinon son extrêmité finale serait de degré impair.
Si c_0 ne contient pas toutes les arêtes on considère le graphe partiel $G - c_0$ puis son sous-graphe H obtenu en supprimant les sommets isolés de $G - c_0$. Les sommets de H sont toujours de degré pair car s'ils ont une occurrence dans c_0, on leur a retiré un nombre pair d'arêtes incidentes ce qui laisse invariant la parité de leur degré.
D'autre part il y a nécessairement un sommet x_1 de H qui appartient à c_0 car G est connexe.
On considère alors dans H une chaîne simple la plus longue possible en partant de x_1 : elle sera, elle aussi, un cycle c'.
On établit un cycle c_1 en raccrochant c' à c_0 au sommet x_1.
Si c_1 ne contient pas toutes les arêtes on réitère le procédé jusqu'à ce que le cycle trouvé soit eulérien.

Chapitre II – Coloration

- Coloration des sommets
- Coloration des arêtes
- Graphe planaire

Chapitre II – Coloration

- Coloration des sommets
- Coloration des arêtes
- Graphe planaire

Chapitre II – Coloration

- Coloration des sommets
- Coloration des arêtes
- Graphe planaire

Coloration des sommets

Définition

Soit G un graphe non orienté sans boucle. Une *coloration* de G est une application $\gamma : X \to C$ de l'ensemble des sommets de G dans un ensemble fini C de couleurs, telle que si x et y sont adjacents alors $\gamma(x) \neq \gamma(y)$ (deux sommets adjacents n'ont pas la même couleur)

Remarque

Une coloration immédiate consiste en l'utilisation de n couleurs distinctes pour un graphe d'ordre n.

nombre chromatique

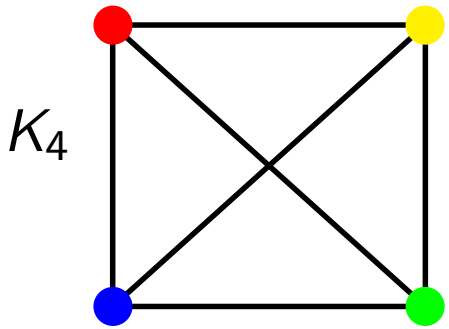
Ce qui nous interesse est l'optimisation de nombre de couleurs utilisées :

Définition

Le *nombre chromatique* de G est le nombre minimal de couleurs permettant de colorier G.

Remarque

Il est clair que le nombre chromatique de K_n est n.



On peut prouver :

Propriétés

- le nombre chromatique d'un graphe d'ordre n est inférieur ou égal à n,
- le nombre chromatique du graphe complet K_n est n,
- si un graphe d'ordre n a un sous-graphe complet d'ordre k alors son nombre chromatique est compris entre k et n.
- si Δ est le degré maximal d'un graphe alors son nombre chromatique est inférieur ou égal à $\Delta + 1$

Exemple

Cinq étudiants A, B, C, D, E doivent passer les examens suivants :

A	Micro Economie, Anglais, Théorie des jeux
B	Probabilités, Analyse financière
C	Macro, Théorie des jeux
D	Anglais, Probabilités, Macro
E	Micro Economie, Analyse financière

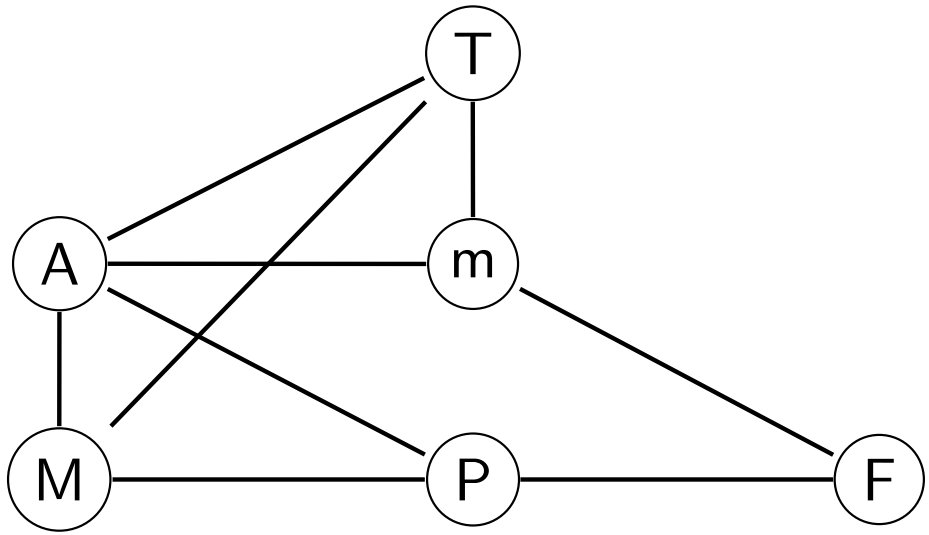
Sachant que chaque étudiant ne peut se présenter qu'à une épreuve par jour, quel est le nombre minimal de jours nécessaire à l'organisation de toutes les épreuves ?

Exemple

Pour résoudre ce problème, on définit le graphe simple non orienté des "incompatibilités" ainsi : les sommets sont les matières et deux sommets sont adjacents si un étudiant doit passer ces deux matières (*i.e.* elles sont incompatibles en ce sens qu'elles ne peuvent pas se dérouler la même journée). Une fois ce graphe établi, on le colorie et chaque couleur représente une journée nécessaire ; deux épreuves ayant la même couleur, donc ne concernant pas un même étudiant, pourront se dérouler la même journée.

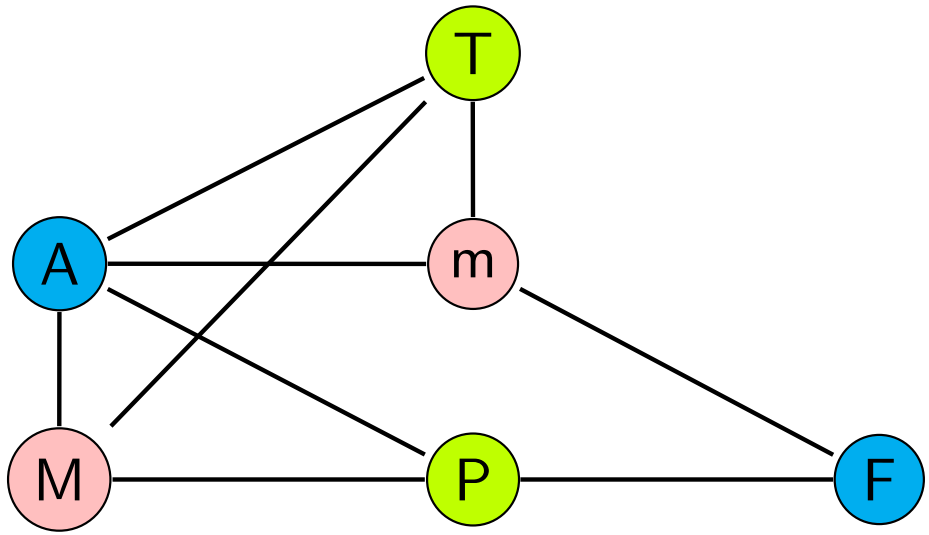
Exemple

A	Micro Economie, Anglais, Théorie des jeux
B	Probabilités, Analyse financière
C	Macro, Théorie des jeux
D	Anglais, Probabilités, Macro
E	Micro Economie, Analyse financière



Exemple

A	Micro Economie, Anglais, Théorie des jeux
B	Probabilités, Analyse financière
C	Macro, Théorie des jeux
D	Anglais, Probabilités, Macro
E	Micro Economie, Analyse financière



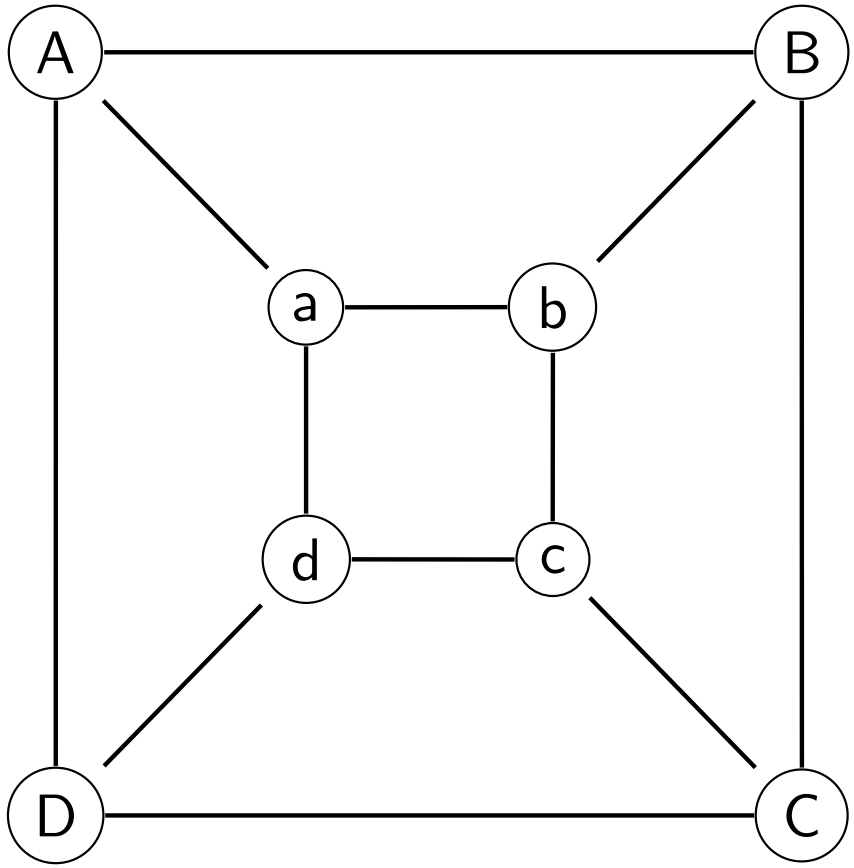
Le cas particulier du nombre chromatique 2

On s'intéresse au cas des graphes non orientés dont le nombre chromatique est 2. Ils peuvent être définis ainsi

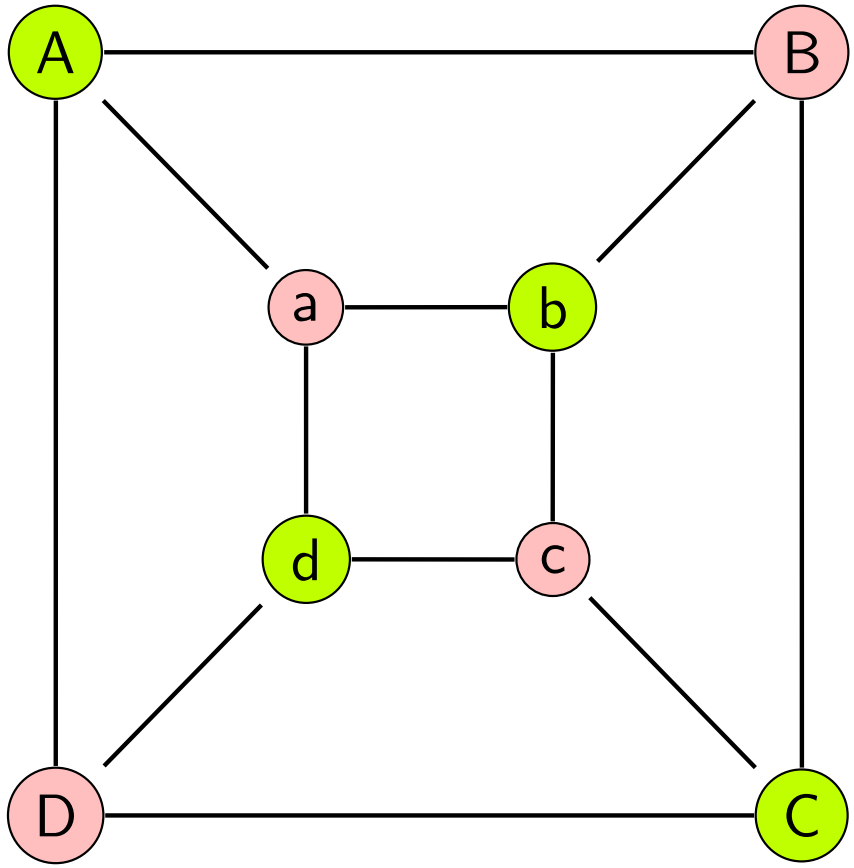
Définition

Un graphe non orienté G est *biparti* s'il existe une partition de l'ensemble de ses sommets en deux classes disjointes telle que toute arête a une extrêmité dans chaque classe.

Exemple



Exemple



Caractéristique

Propriété

un graphe est biparti *si et seulement si* il n'a pas de cycle impair

Preuve

($\Rightarrow$) Supposons que G soit biparti et ait un cycle impair $x_0 e_1 x_2 e_2 x_3 \cdots x_{2k} e_{2k+1} x_{2k+1}$ avec $x_{2k+1} = x_0$.
Soit $Y \cup Z$ la bipartition des sommets de G.
Si $x_0 \in Y$ alors $x_1 \in Z$ et donc $x_2 \in Y$.
On obtient $x_0, x_2, \cdots x_{2k} \in Y$ et $x_1, \cdots x_{2k+1} \in Z$.
Mais $x_{2k+1} = x_0$: contradiction

Preuve

($\Leftarrow$) soit G sans cycle impair. On supposera dans un premier temps que G est connexe et on choisit un sommet x. On définit alors les deux ensembles suivants :
$X_p = \{y \in X \mid$ la longueur d'une plus courte chaîne élémentaire reliant x à y est paire$\}$
$X_i = \{y \in X \mid$ la longueur d'une plus courte chaîne élémentaire reliant x à y est impaire$\}$.
On a bien $X_p \cup X_i = X$ puisque G est connexe.

Preuve

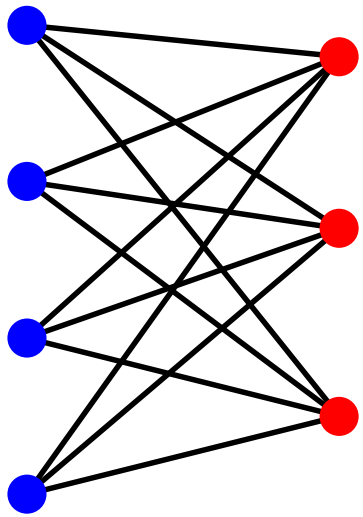
Montrons qu'il est impossible que deux sommets de X_p soient adjacents : supposons qu'il existe une arête e reliant deux sommets y et z de X_p.
On a alors une chaîne élémentaire reliant x à y de longueur paire et une chaîne élémentaire reliant x à z de de longueur paire.
On a donc un cycle $x \cdots yez \cdots x$ de longueur impaire (pair + pair + 1) : contradiction.
On montre de même qu'il n'existe pas d'arête reliant deux sommets de X_i.
Donc G est biparti.
Dans le cas général où le graphe G n'est pas connexe, on applique le raisonnement précédent à ses composantes connexes pour obtenir une bipartition des sommets de G.

graphe biparti complet

Définition

un graphe G est *biparti complet* si G est biparti et si tout sommet d'une partie est adjacent à tout sommet de l'autre partie. On note $K_{p,r}$ le graphe biparti complet dont les deux parties ont respectivement p et r sommets.

$K_{4,3}$



Coloration des arêtes

On peut aussi colorier les arêtes d'un graphe pour que deux arêtes incidentes à un même sommet n'aient pas la même couleur. Bien sûr, cela nécessite que le graphe non orienté n'ait aucune boucle. Plus formellement,

Définition

Soit G un graphe non orienté sans boucle. Une *coloration* des arêtes de G est une application $\gamma : E \to C$ de l'ensemble des arêtes de G dans un ensemble fini C de couleurs, telle que si un sommet x est une extrêmité des arêtes e et e' alors alors $\gamma(e) \neq \gamma(e')$.

Définition

On appelle *indice chromatique* de G le nombre minimal de couleurs permettant de colorier les arêtes de G.

Le problème de coloration des arêtes peut se ramener à celui des sommets pour un graphe en définissant le graphe adjoint G^a ainsi : les sommets de G^a sont les arêtes de G et deux sommets de G^a sont adjacents si, dans G ces deux arêtes ont une extrêmité communes.
exemple : 5 amis font un tournoi d'échecs, dont les parties se déroulent en temps limité d'une heure, chacun devant rencontrer les quatre autres. Combien d'heures nécessite ce tournoi ?
L'idée est le suivante : chaque partie est caractérisée par ses deux joueurs ; donc on peut représenter ce tournoi par un graphe dans lequel les sommets seront les joueurs et les arêtes seront les parties. Ici on obtient le graphe complet K_5.
En coloriant les arêtes de ce graphe on détermine par une couleur une plage horaire durant laquelle les parties de cette couleur pourront se dérouler. L'indice chromatique de K_5 nous donne donc la réponse.

Propriété

Propriété

si Δ est le degré maximal d'un graphe alors son indice chromatique est inférieur ou égal à $\Delta + 1$.

Plus précisement :

Théorème de Vizing

si Δ est le degré maximal d'un graphe **simple** alors son indice chromatique est égal à Δ ou $\Delta + 1$.

Pour l'instant, il n'existe pas d'algorithme en temps polynomial qui permet de décider entre Δ et $\Delta + 1$ l'indice chromatique d'un graphe simple.

Propriété des graphes bipartis

Le résultat précédent devient plus précis pour les graphes bipartis

Théorème de König

si Δ est le degré maximal d'un graphe biparti alors son indice chromatique est égal à Δ.

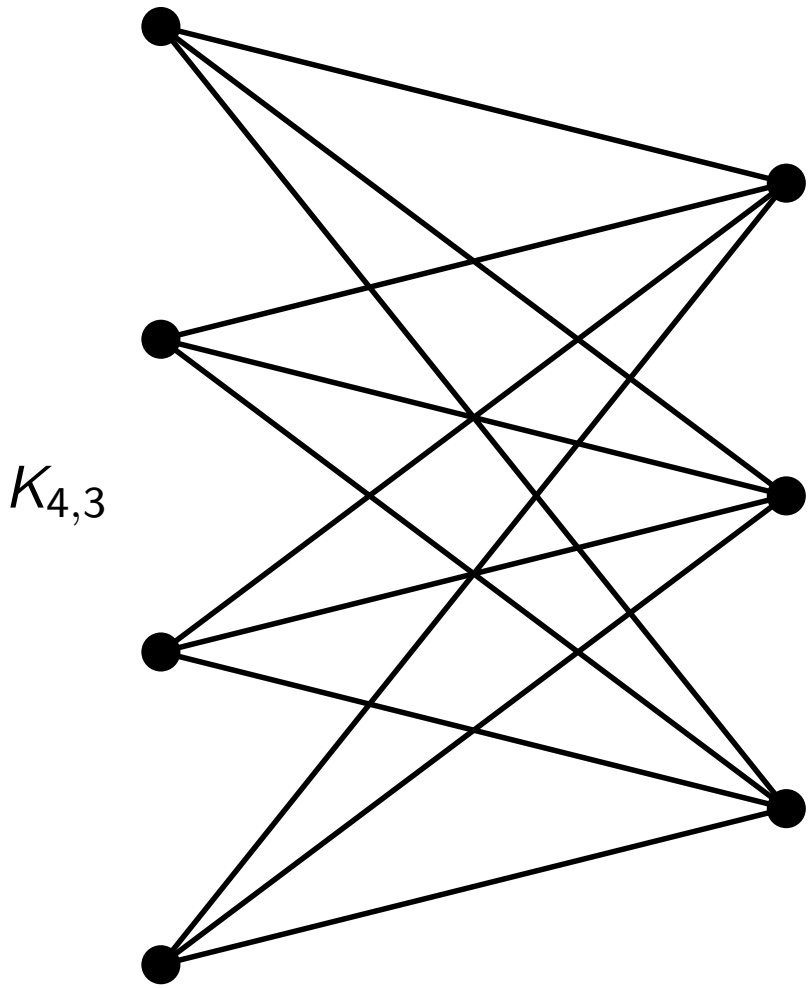
Preuve

Soit un graphe biparti.
Si $\Delta = 1$ alors une seule couleur est clairement suffisante dans un graphe biparti.
Si $\Delta > 1$ alors on définit un ensemble de Δ couleurs que l'on représentera par les entiers $1, 2, \ldots, \Delta$. on affecte à chaque arête une couleur sans chercher à « colorier » les arêtes : si cette affectation est une coloration alors c'est terminé.

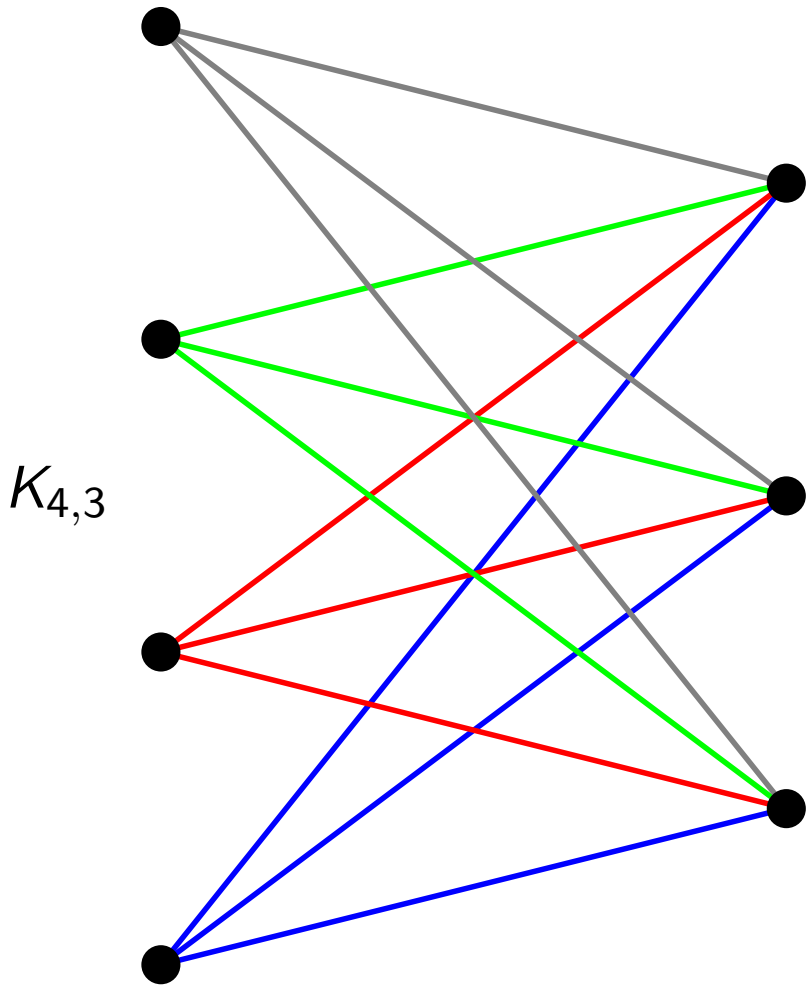
Preuve

Sinon on va améliorer cette affectation : il existe un sommet x qui est l'extrêmité de deux arêtes e, e' de même couleur i.
Comme x a au plus Δ arêtes incidentes, alors il existe nécessairement une couleur j qui ne colore aucune arête incidente à x.
Changeons alors la couleur de $e = \{x, y\}$ en couleur j.
Il se peut que ce changement fasse que y a maintenant deux arêtes incidentes e, e'' de couleur j : dans ce cas on change la couleur j de e'' en couleur i. Puis on s'intéresse à l'autre extrêmité de e'' pour lui appliquer le même traitement qu'à y.
On obtient ainsi une chaîne issue de x de couleurs alternées : $x - j - y - i - \dots$.
Si cette chaîne se refermait en cycle sur x avec une arête de couleur i alors G aurait un cycle de longueur impaire ce qui est impossible dans un graphe biparti : on a donc résolu un problème pour x.
On poursuit ainsi de suite.

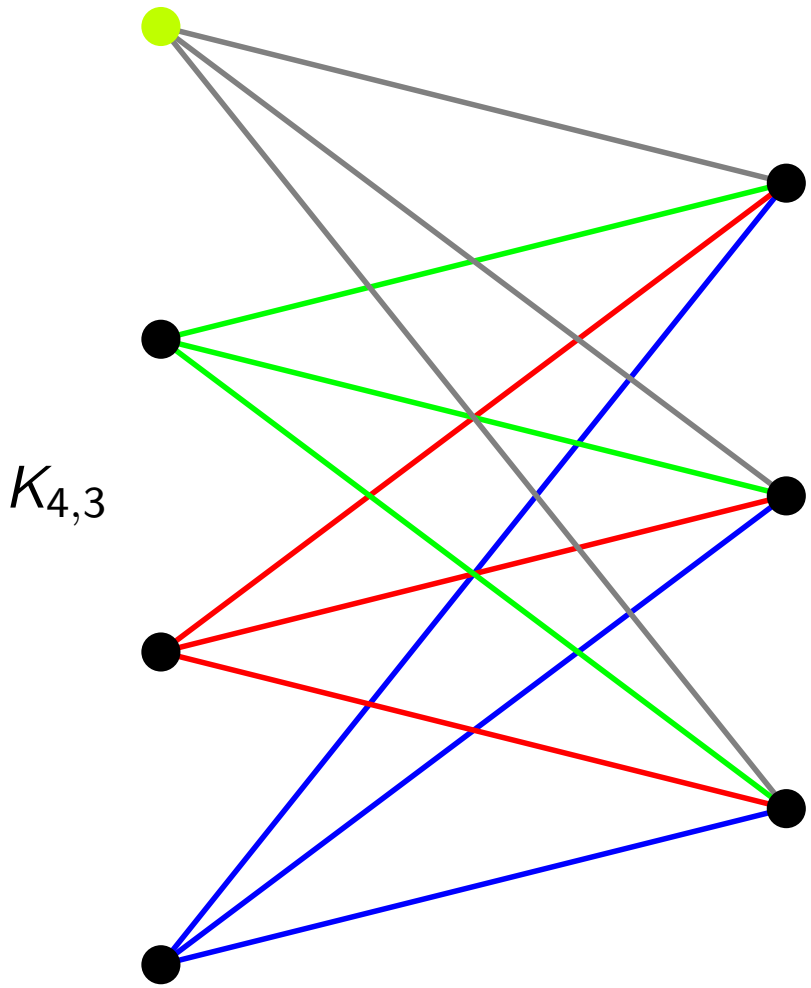
Exemple



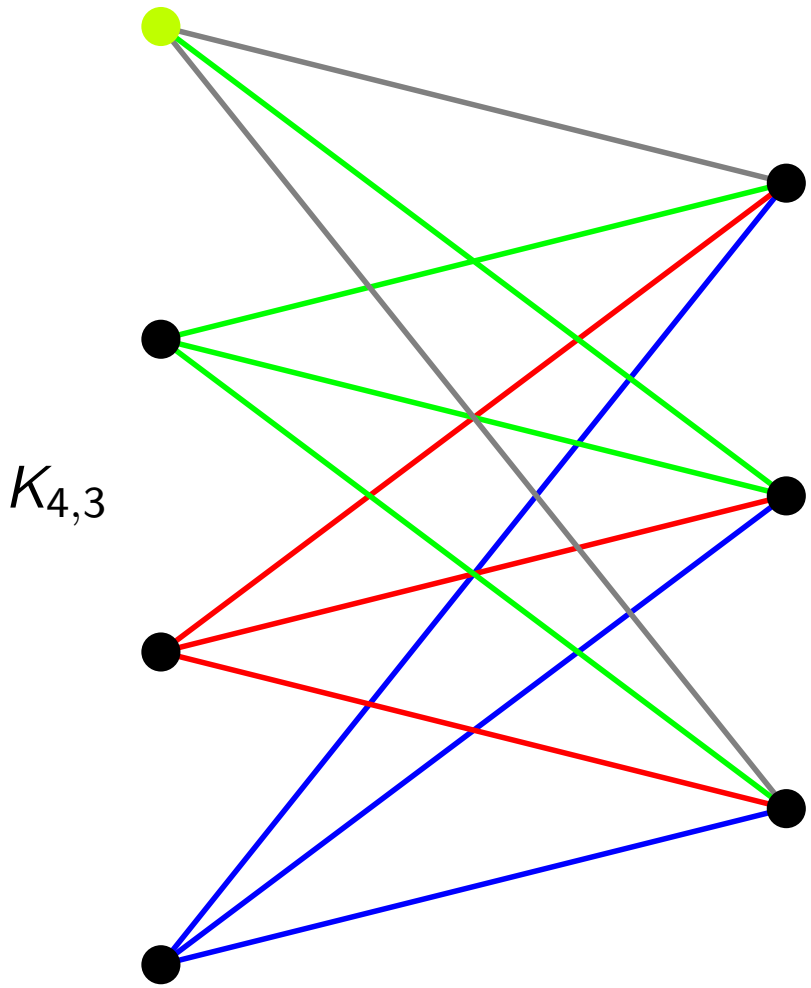
Exemple



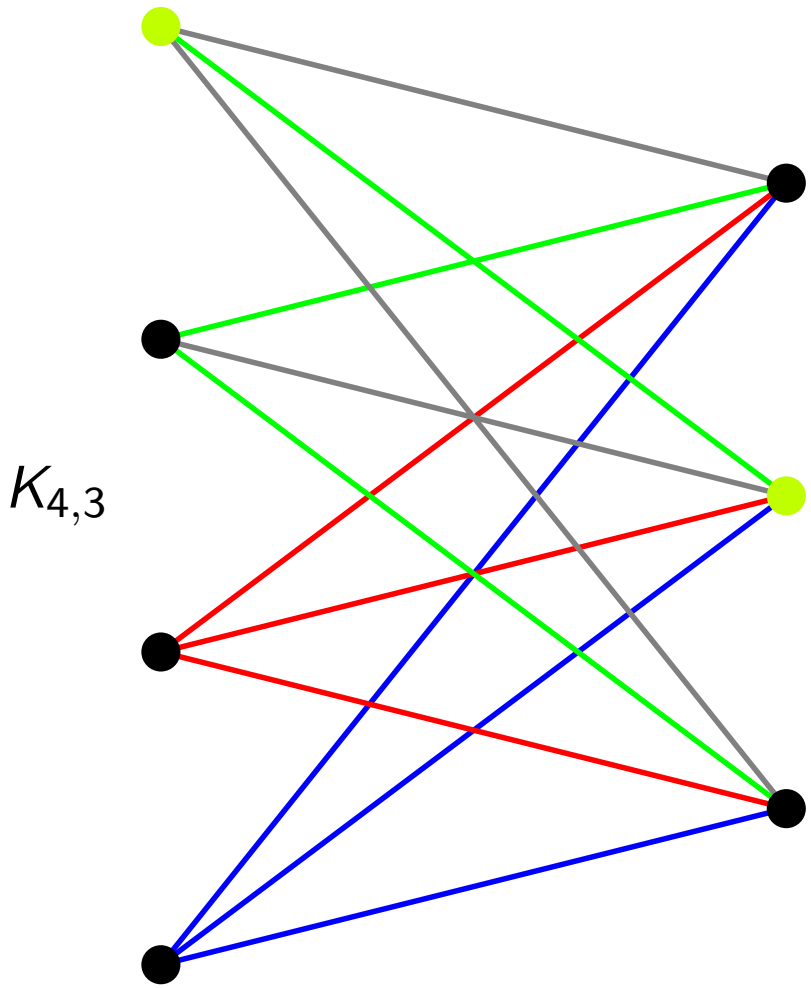
Exemple



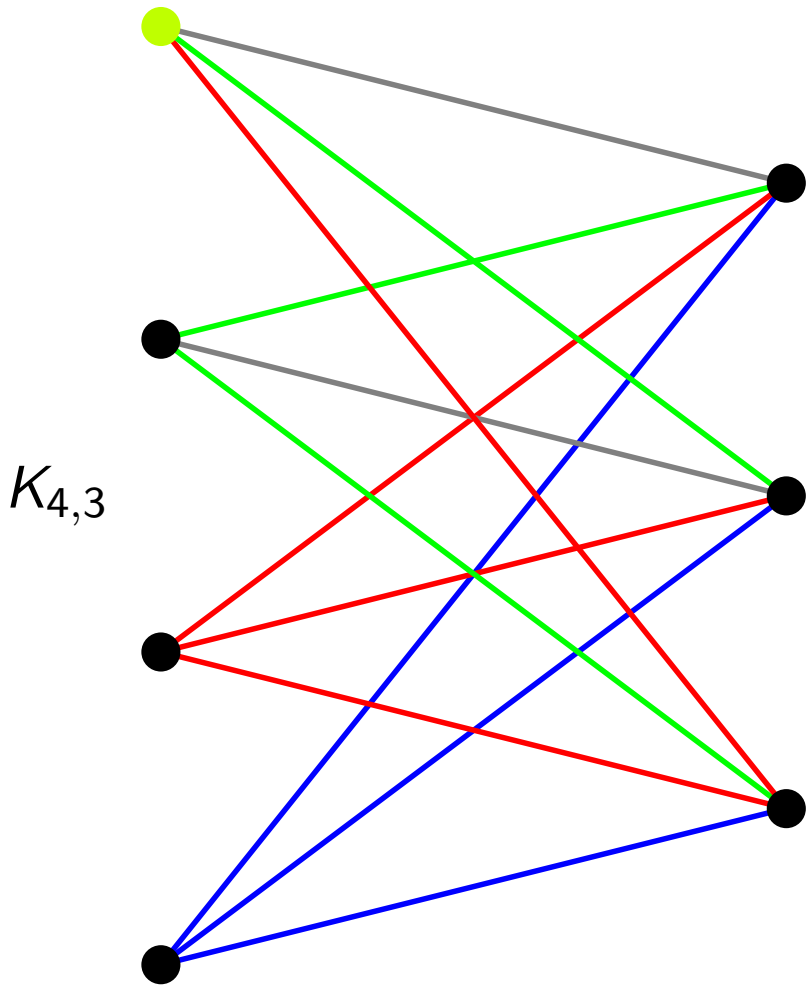
Exemple



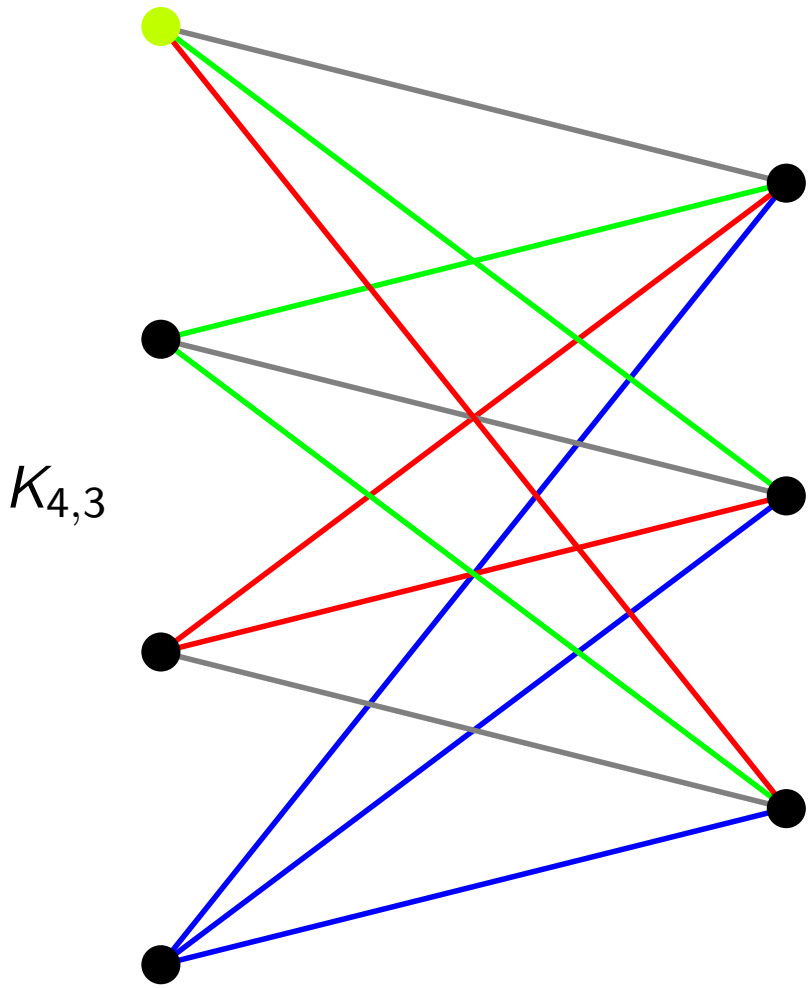
Exemple



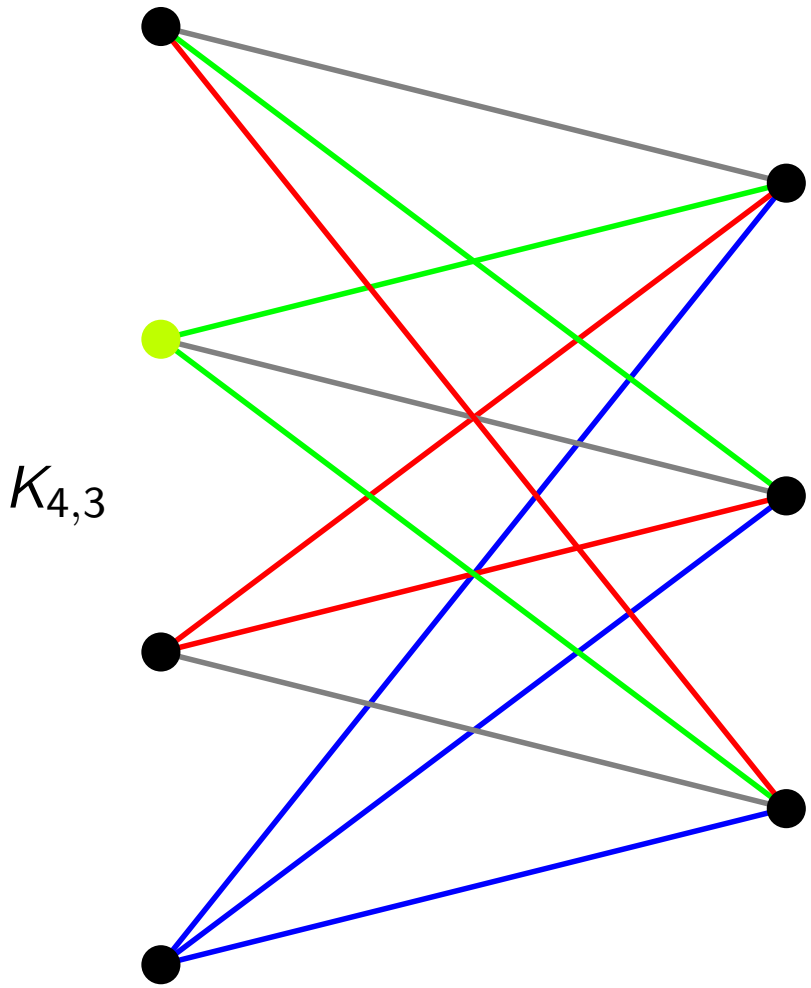
Exemple



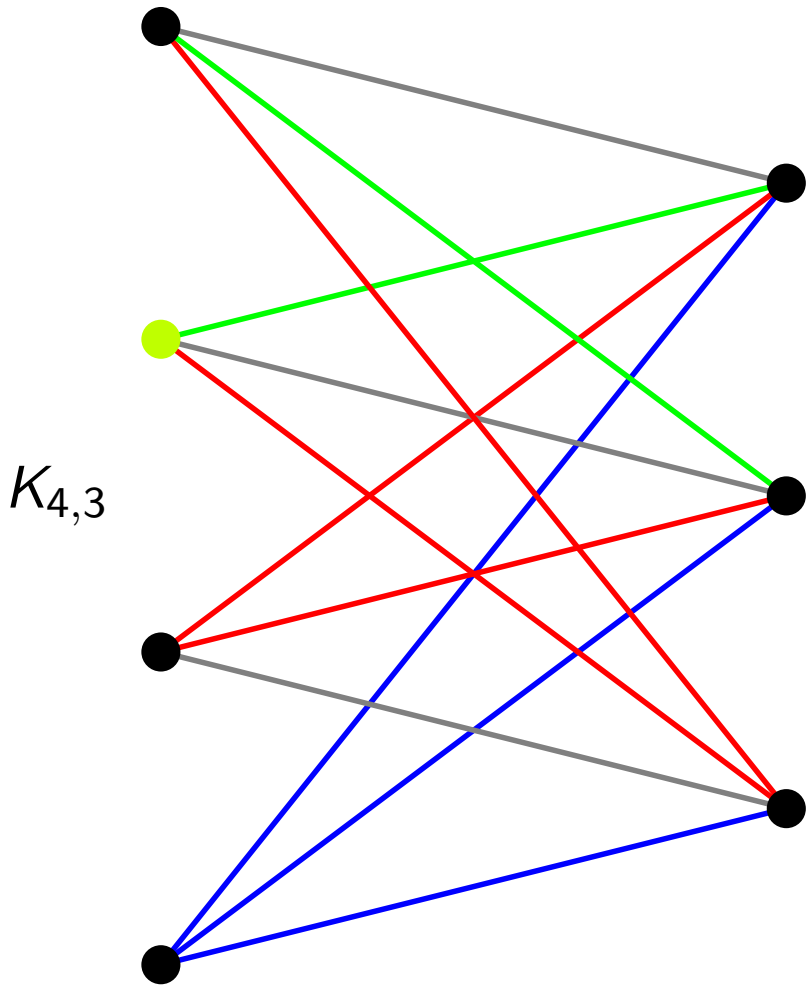
Exemple



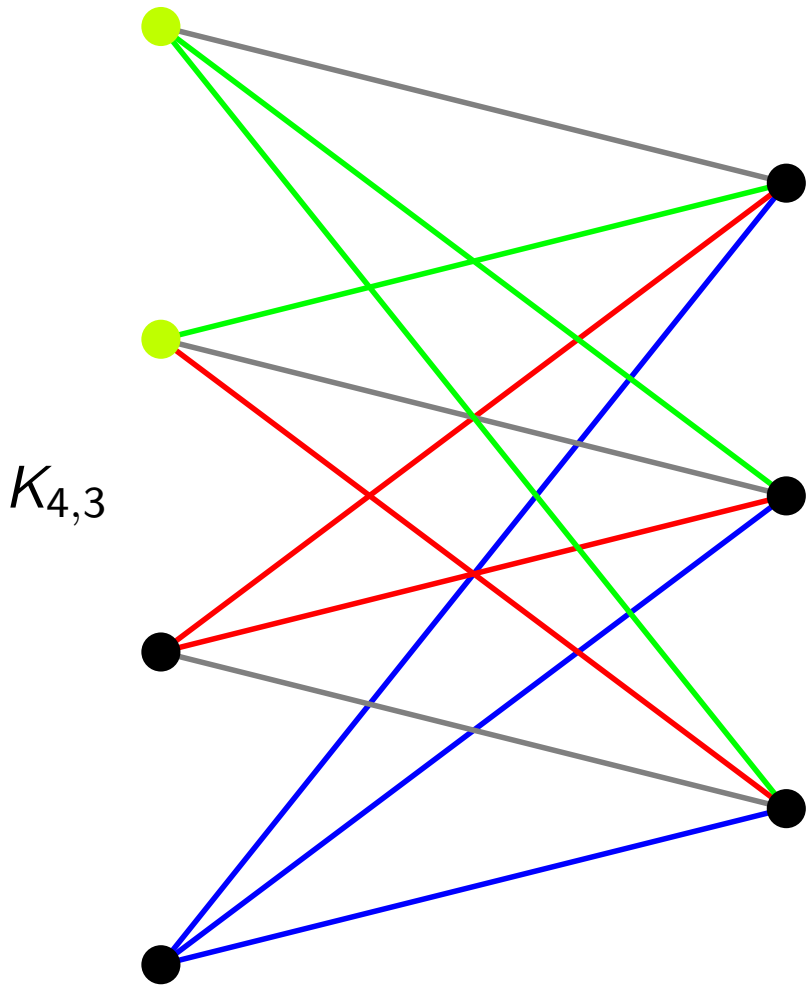
Exemple



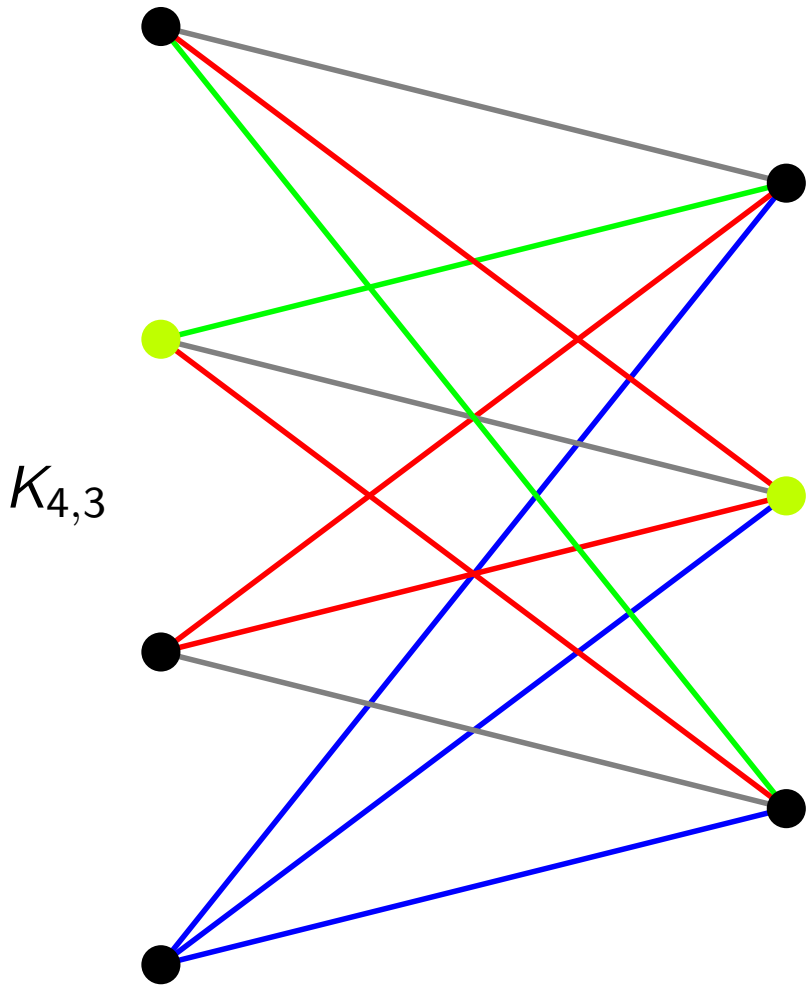
Exemple



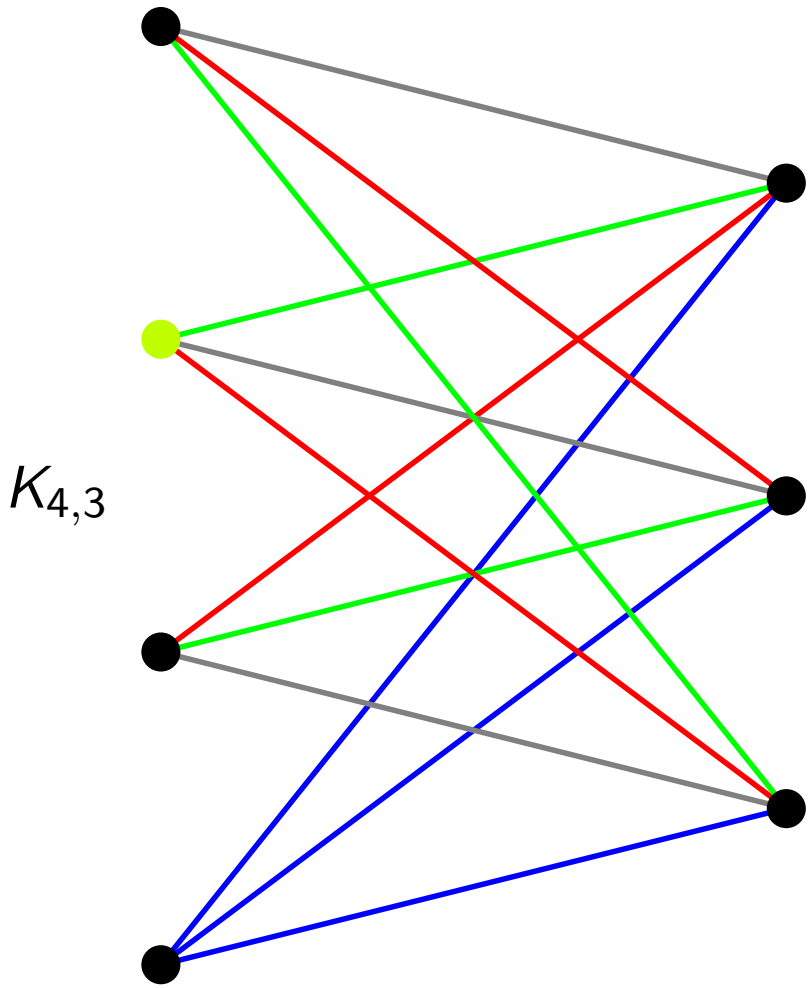
Exemple



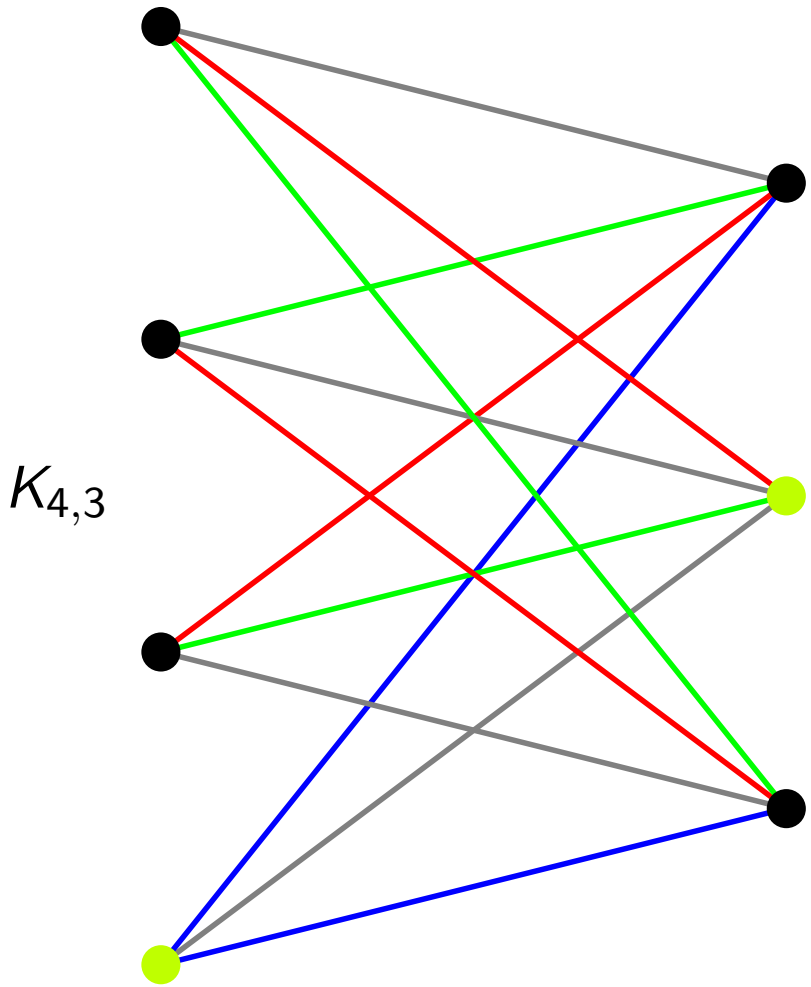
Exemple



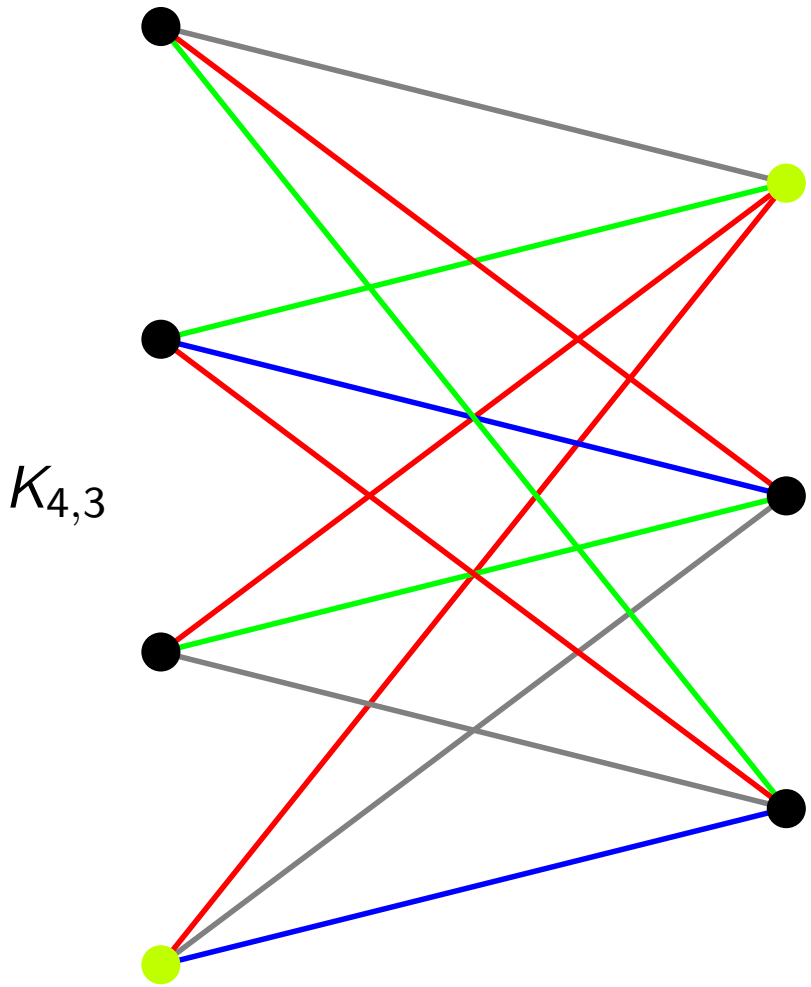
Exemple



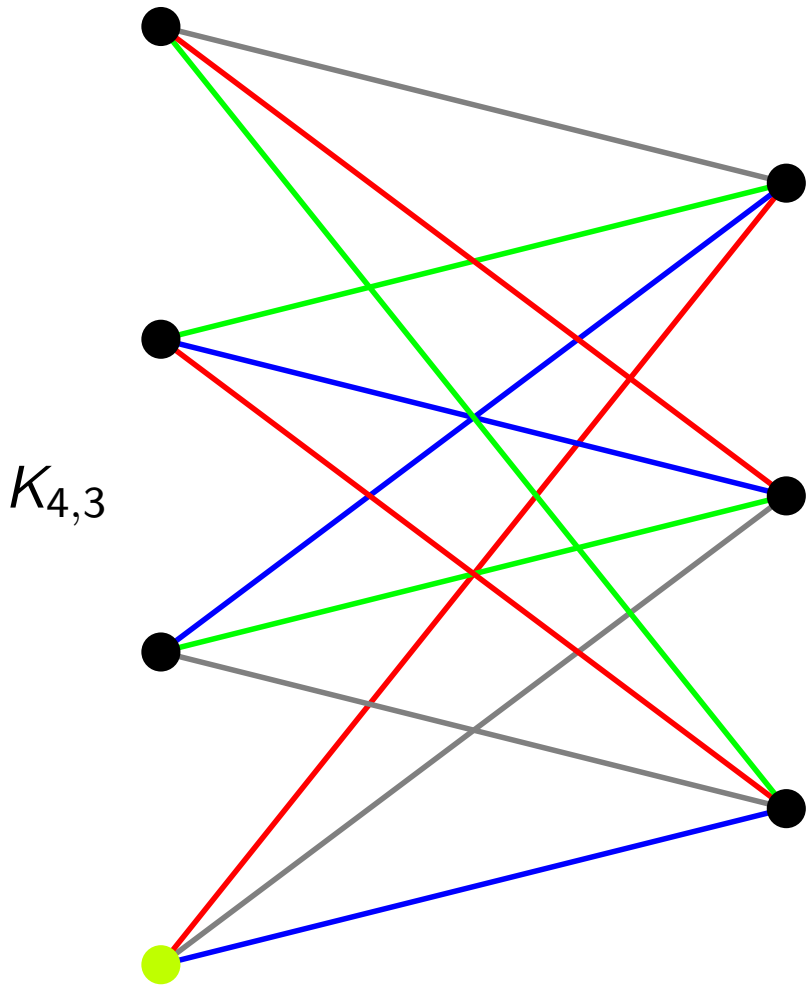
Exemple



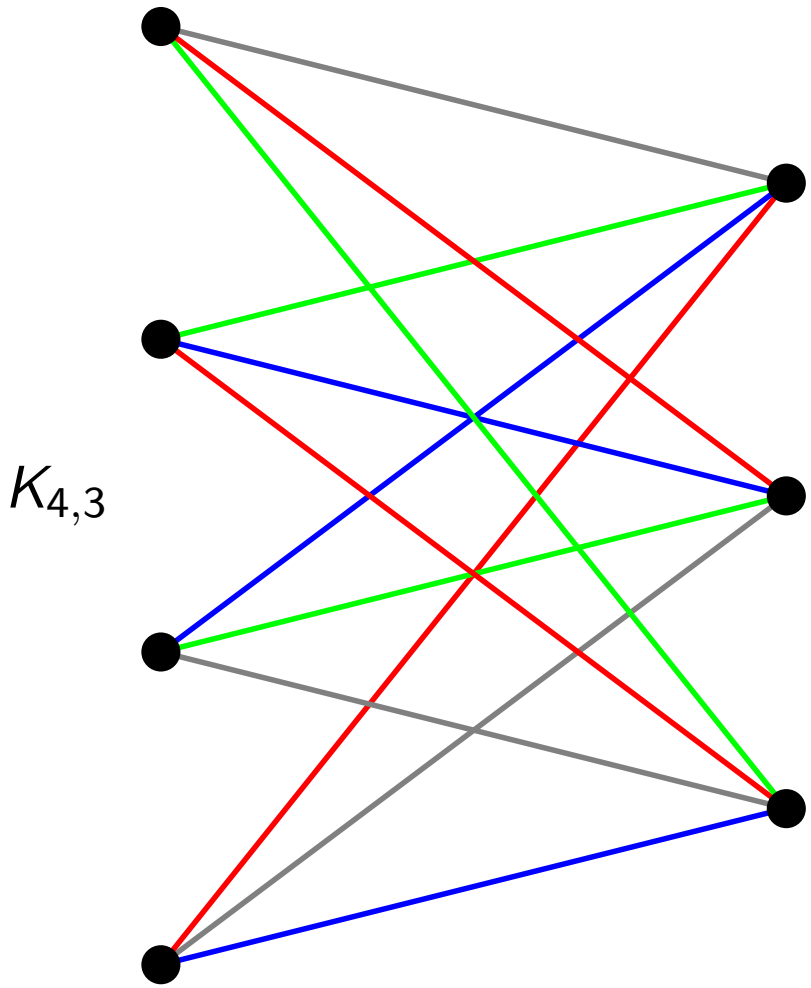
Exemple



Exemple



Exemple



Notion de couplage

En regroupant les arêtes de même couleur nous obtenons une partition de l'ensemble des arêtes ; l'ensemble des arêtes ayant la même couleur forme un *couplage*
Plus formellement :

Définition

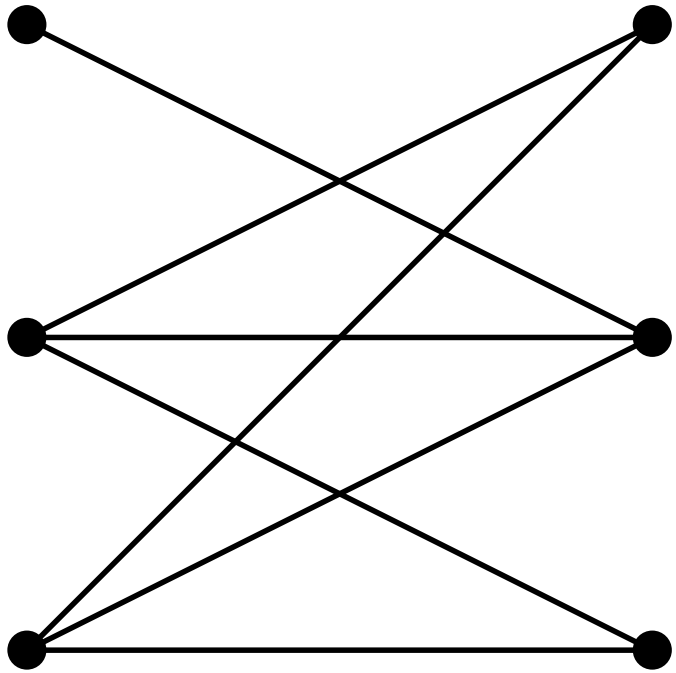
Dans un graphe non orienté, un couplage est un ensemble d'arêtes n'ayant aucune extrêmité commune deux à deux.

Définition

Pour un graphe donné, un couplage est

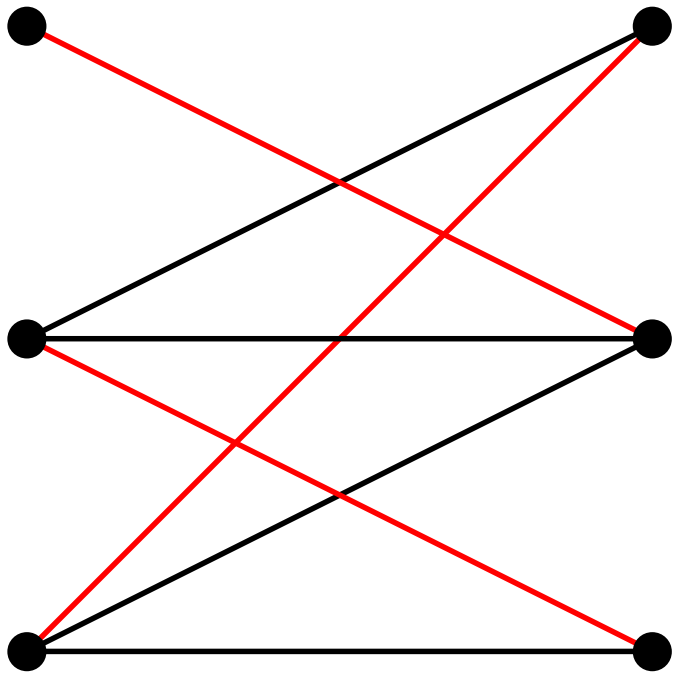
- maximum s'il contient le plus grand nombre d'arêtes possible
- maximal si toute arête du graphe est incident à au moins une arête du couplage
- parfait si tout sommet est incident à une arête du couplage

Exemple



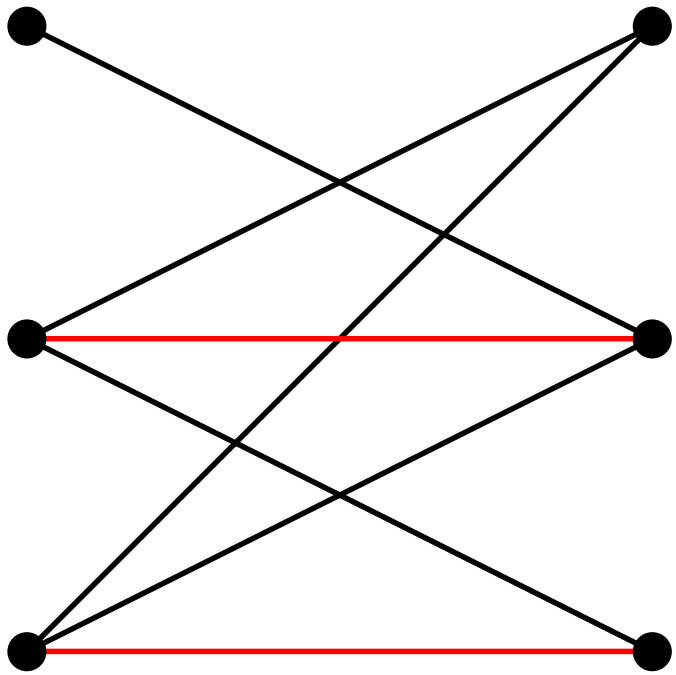
Exemple

un couplage maximum et parfait :



Exemple

un couplage maximal et non maximum :

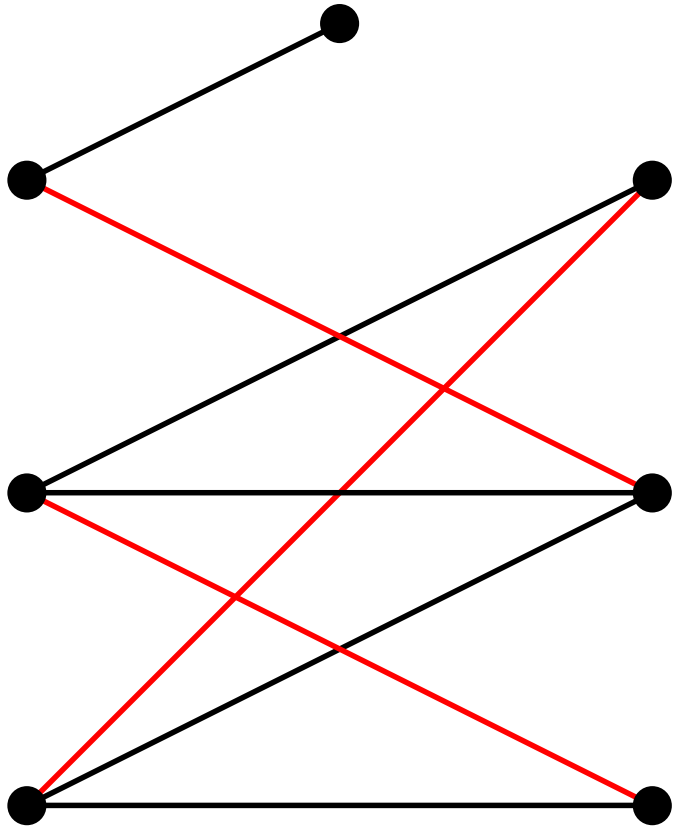


Remarques

- Un couplage parfait est maximum
- Un couplage maximum est maximal
- les deux réciproques sont fausses

Remarques

Voici un couplage maximum et non parfait :



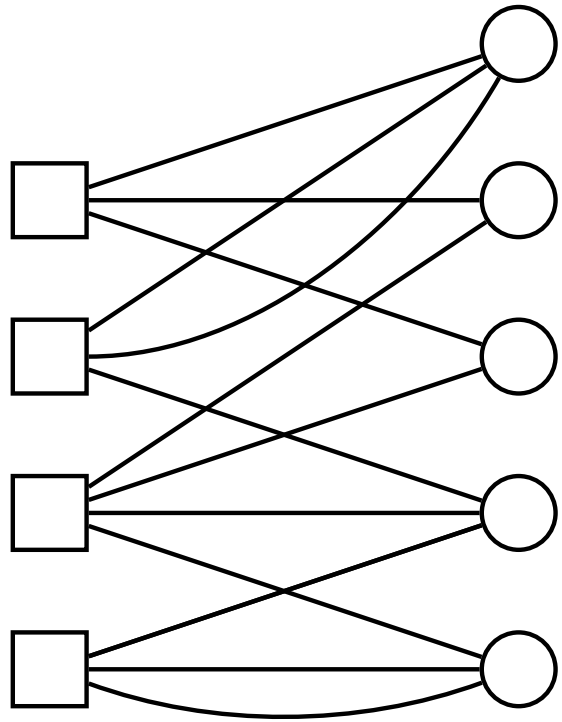
Problème de l'emploi du temps

un ensemble de classes doit recevoir une série de cours d'une heure réalisés par un ensemble de professeurs. Bien sûr, chaque classe reçoit un cours unique par heure et chaque professeur donne un cours unique par heure. Le problème est de minimiser le nombre d'heures totales permettant la réalisation de tous les cours.

	c_1	c_2	c_3	c_4	c_5
p_1	1	1	1	0	0
p_2	2	0	0	1	0
p_3	0	1	1	1	1
p_4	0	0	0	1	2

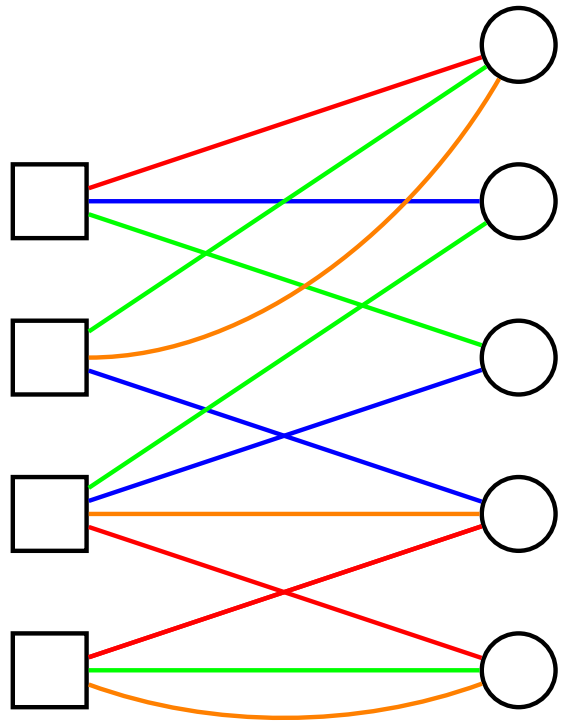
Problème de l'emploi du temps

On modélise le problème par un graphe non orienté biparti dont les sommets sont les professeurs d'une part et les classes d'autre part, et les arêtes sont les cours.



Problème de l'emploi du temps

La coloration des arêtes donnent le nombre de heures minimales : ici 4 couleurs, correspondant au degré maximal selon le théorème de König.



Problème de l'emploi du temps

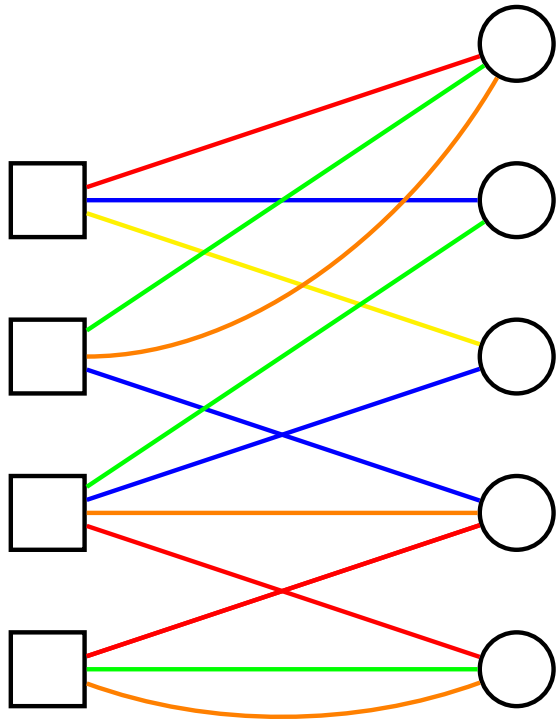
On rajoute maintenant une contrainte supplémentaire ; le nombre de salles est limité par un entier σ.
Donc par tranche horaire il ne peut pas y avoir plus de σ cours.
Donc, si le nombre total de cours est m (nombre total d'arêtes), il faut au moins $\frac{m}{\sigma}$ heures pour réaliser tous les cours.
Le résultat optimal est la coloration des arêtes avec un nombre de couleurs égal à $max\{\Delta, \lceil \frac{m}{\sigma} \rceil\}$.
Prenons $\sigma = 3$. Puisque $\Delta = 4$ et $m = 13$ le nombre optimal de couleurs est $\lceil \frac{m}{\sigma} \rceil = 5$.

Problème de l'emploi du temps

Pour l'instant, avec 4 couleurs, on a les couplages suivants
$C_r = \{(p_1, c_1), (p_3, c_5), (p_4, c_4)\}$,
$C_b = \{(p_1, c_2), (p_2, c_4), (p_3, c_3)\}$,
$C_v = \{(p_1, c_3), (p_2, c_1), (p_3, c_2), (p_4, c_5)\}$,
$C_o = \{(p_2, c_1), (p_3, c_4), (p_4, c_5)\}$.
Mais on voit bien que C_v (couplage maximum non parfait) nécessite 4 salles.

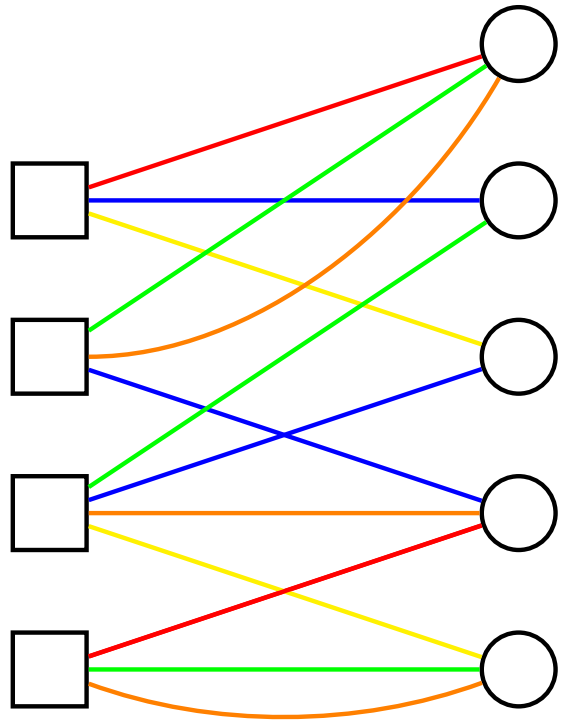
Problème de l'emploi du temps

On va donc rajouter un couplage vide pour une couleur suppémentaire $C_j = \emptyset$ puis on va rééquilibrer C_v et C_j. On affecte (p_1, c_3) à C_j.



Problème de l'emploi du temps

On peut continuer à rééquilibrer en dimunuant par exemple C_r par l'affectation de (p_3, c_5) à C_j.



Graphe planaire

Définition

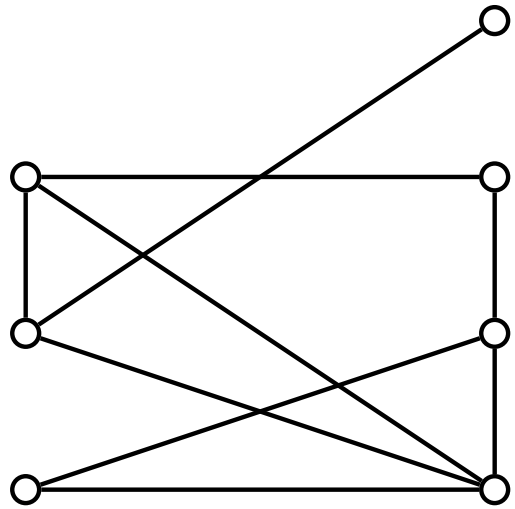
Un graphe est planaire s'il peut être dessiné dans le plan sans qu'aucune paire d'arêtes ou d'arcs ne s'intersecte.

Graphe planaire

Définition

Un graphe est planaire s'il peut être dessiné dans le plan sans qu'aucune paire d'arêtes ou d'arcs ne s'intersecte.

exemple :

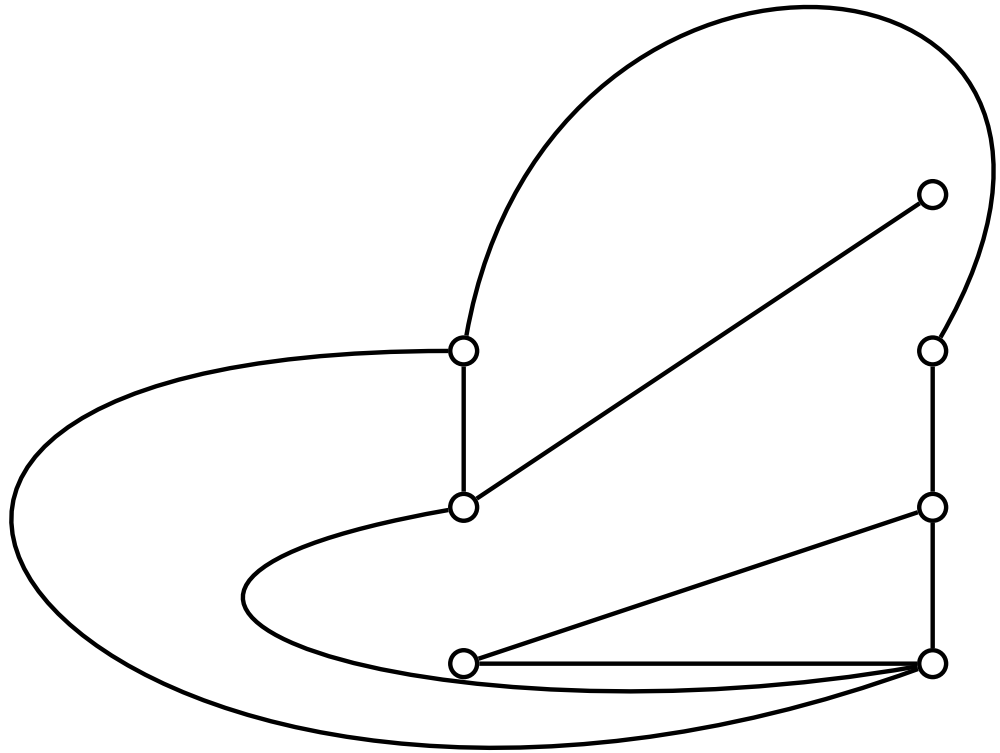


Graphe planaire

Définition

Un graphe est planaire s'il peut être dessiné dans le plan sans qu'aucune paire d'arêtes ou d'arcs ne s'intersecte.

exemple :

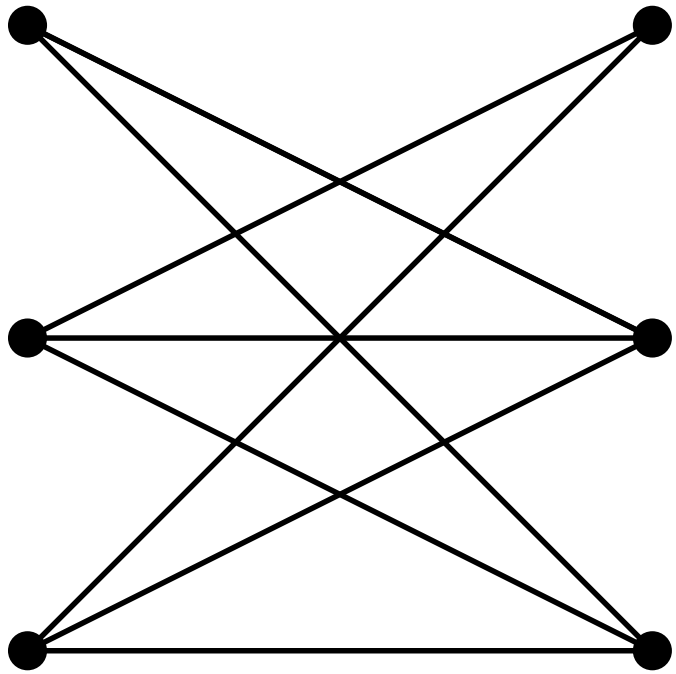


Graphe planaire

Propriété

K_5, $K_{3,3}$ ne sont pas planaires.

$K_{3,3}$



Graphe planaire

On peut se poser le problème de la planarité lors de la construction d'un réseau de voies ferrées afin de minimiser le nombre d'aiguillages.
Cette notion de graphe planaire a un lien célèbre avec le problème des 4 couleurs.
L'assertion est la suivante : toute carte géographique peut être coloriée avec 4 couleurs seulement de telle sorte que deux pays frontaliers n'aient pas la même couleur.
Cette assertion n'a été prouvée que récemment (à l'aide de nombreuses heures de calcul sur ordinateur).
Une carte géographique peut être modélisée par un graphe planaire (les sommets sont les pays et les arêtes représentent une frontière commune).

Théorème

Le nombre chromatique d'un graphe planaire est inférieur ou égal à 4.

Formule d'Euler

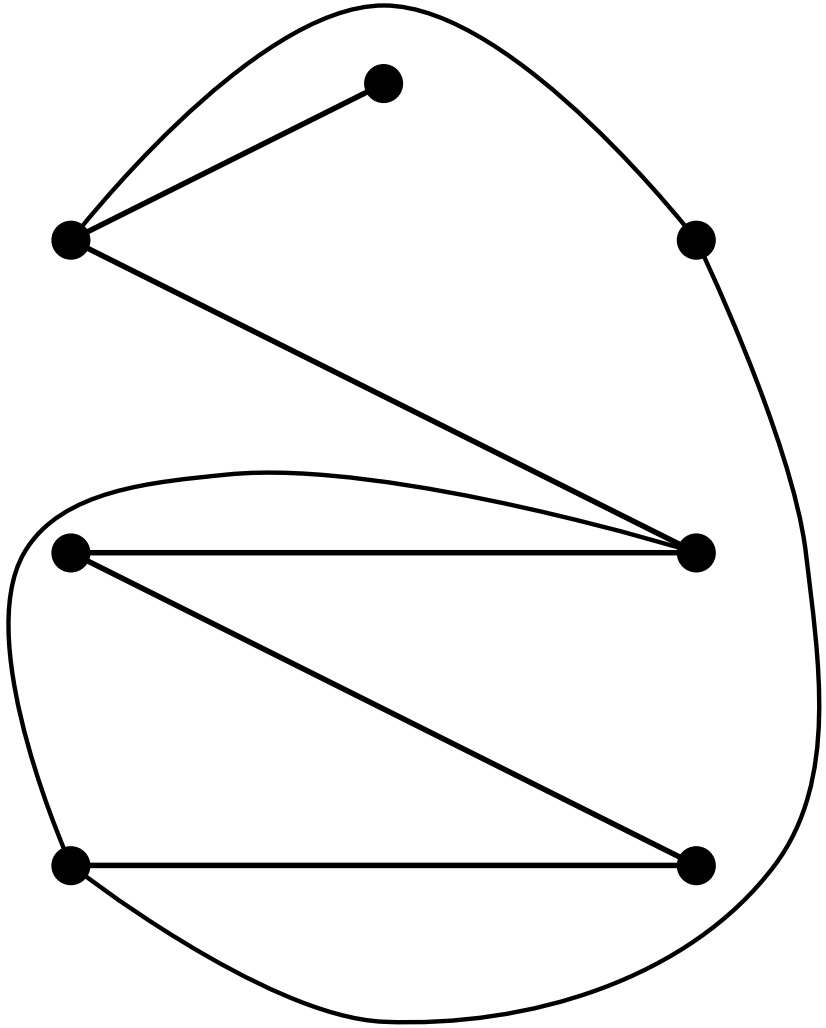
On doit à L. Euler le résultat suivant sur les graphes planaires : leur réprésentation dans le plan délimite un certain nombre de régions.

Théorème

Soit G un graphe plan connexe d'ordre n à m arêtes délimitant r régions.
On a alors la *formule d'Euler* : $n - m + r = 2$.

Formule d'Euler

Pour le graphe ci-dessous, $n = 7, m = 8, r = 3$



Remarque sur r=1

Les graphes connexes planaires ne délimitant qu'une région sont en quelque sorte ≪ ouverts ≫.
Par la formule d'Euler de tels graphes non orientés connexes ont $n-1$ arêtes : ce sont des arbres.

Chapitre III – Arbres

- Définitions des arbres
- Caractérisations des arbres
- Arbre couvrant
- Problème de l'arbre couvrant minimal
- Algorithme de Kruskal
- Algorithme de Prim
- Algorithme de Prim

Chapitre III – Arbres

- Définitions des arbres
- Caractérisations des arbres
- Arbre couvrant
- Problème de l'arbre couvrant minimal
- Algorithme de Kruskal
- Algorithme de Prim
- Algorithme de Prim

Chapitre III – Arbres

- Définitions des arbres
- Caractérisations des arbres
- Arbre couvrant
- Problème de l'arbre couvrant minimal
- Algorithme de Kruskal
- Algorithme de Prim
- Algorithme de Prim

Chapitre III – Arbres

- Définitions des arbres
- Caractérisations des arbres
- Arbre couvrant
- Problème de l'arbre couvrant minimal
- Algorithme de Kruskal
- Algorithme de Prim
- Algorithme de Prim

Chapitre III – Arbres

- Définitions des arbres
- Caractérisations des arbres
- Arbre couvrant
- Problème de l'arbre couvrant minimal
- Algorithme de Kruskal
- Algorithme de Prim
- Algorithme de Prim

Chapitre III – Arbres

- Définitions des arbres
- Caractérisations des arbres
- Arbre couvrant
- Problème de l'arbre couvrant minimal
- Algorithme de Kruskal
- Algorithme de Prim
- Algorithme de Prim

Chapitre III – Arbres

- Définitions des arbres
- Caractérisations des arbres
- Arbre couvrant
- Problème de l'arbre couvrant minimal
- Algorithme de Kruskal
- Algorithme de Prim
- Algorithme de Prim

Arbres

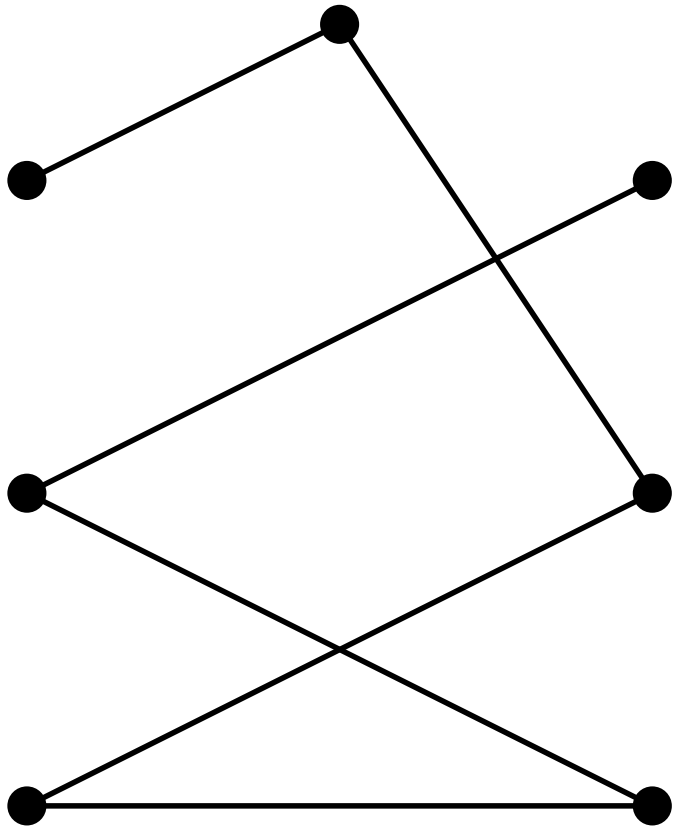
Définition

Un arbre est un graphe non orienté *connexe* et *sans cycle*.

Arbres

Définition

Un arbre est un graphe non orienté *connexe* et *sans cycle*.



Exercice

Trouver les 6 arbres d'ordre 6 (non isomorphes)

Arborescence

En ce qui concerne les graphes orientés, on parle d'arborescence :

Définition

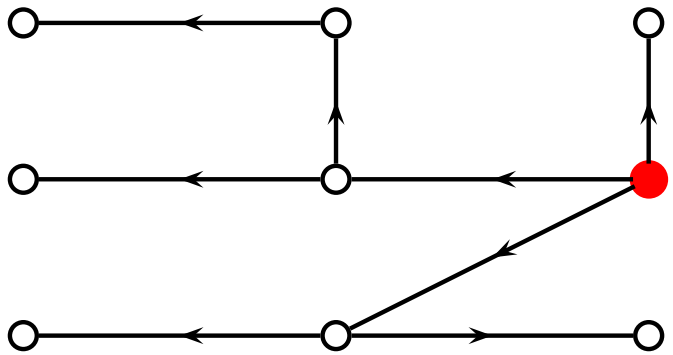
Une arborescence est un graphe orienté possédant un sommet priviligié, la *racine*, telle qu'il existe un et un seul chemin depuis la racine vers tout autre sommet du graphe.

Arborescence

En ce qui concerne les graphes orientés, on parle d'arborescence :

Définition

Une arborescence est un graphe orienté possédant un sommet priviligié, la *racine*, telle qu'il existe un et un seul chemin depuis la racine vers tout autre sommet du graphe.

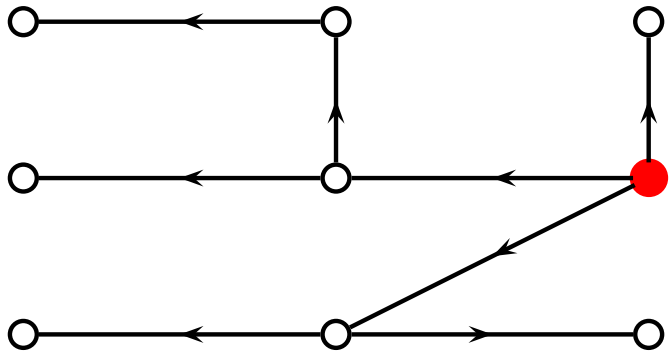


Propriété

On peut orienter les arêtes d'un arbre à partir de n'importe quel sommet de façon à obtenir une arborescence.

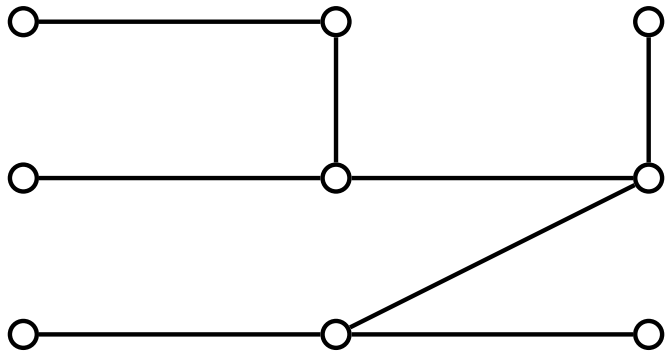
Propriété

On peut orienter les arêtes d'un arbre à partir de n'importe quel sommet de façon à obtenir une arborescence.



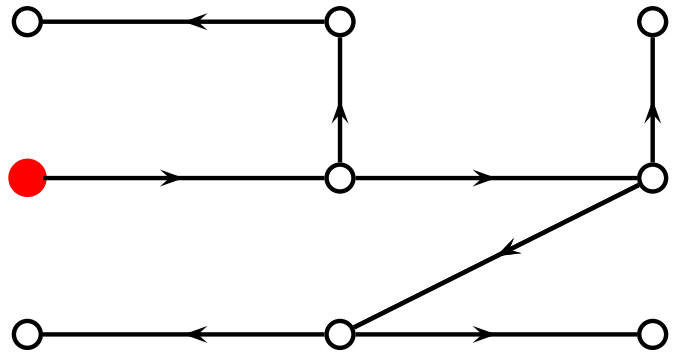
Propriété

On peut orienter les arêtes d'un arbre à partir de n'importe quel sommet de façon à obtenir une arborescence.



Propriété

On peut orienter les arêtes d'un arbre à partir de n'importe quel sommet de façon à obtenir une arborescence.



Caractérisations des arbres

Propriété

Soit G un graphe non orienté d'ordre n. Les propositions suivantes sont équivalentes :

1. G est un arbre
2. G est connexe et a $n-1$ arêtes
3. G est sans cycle et a $n-1$ arêtes
4. G est sans boucle et pour tous sommets x, y, $x \neq y$, il existe une unique chaîne reliant x à y

Caractérisations des arbres

Propriété

Soit G un graphe non orienté d'ordre n. Les propositions suivantes sont équivalentes :

1. G est un arbre
2. G est connexe et a $n-1$ arêtes
3. G est sans cycle et a $n-1$ arêtes
4. G est sans boucle et pour tous sommets x, y, $x \neq y$, il existe une unique chaîne reliant x à y

Caractérisations des arbres

Propriété

Soit G un graphe non orienté d'ordre n. Les propositions suivantes sont équivalentes :

1. G est un arbre
2. G est connexe et a $n-1$ arêtes
3. G est sans cycle et a $n-1$ arêtes
4. G est sans boucle et pour tous sommets x, y, $x \neq y$, il existe une unique chaîne reliant x à y

Caractérisations des arbres

Propriété

Soit G un graphe non orienté d'ordre n. Les propositions suivantes sont équivalentes :

1. G est un arbre
2. G est connexe et a $n-1$ arêtes
3. G est sans cycle et a $n-1$ arêtes
4. G est sans boucle et pour tous sommets x, y, $x \neq y$, il existe une unique chaîne reliant x à y

Preuve

La preuve de ce théorème repose en partie sur les deux lemmes suivants :

Lemme 1

un graphe non orienté connexe d'ordre n a au moins n-1 arêtes.

Lemme 2

un graphe non orienté d'ordre n qui a au moins n arêtes possède un cycle.

Le premier lemme a été prouvé dans le paragraphe connexité.
Il reste le lemme 2 à prouver.

▷ Attention, les réciproques des lemmes 1 et 2 ne sont pas vraies.

Preuve de lemme 2

Soit G un graphe non orienté d'ordre n ayant au moins n arêtes. Montrons par récurrence sur n que G possède un cycle.

Preuve de lemme 2

Pour $n = 1$, c'est clair (c'est une boucle).

Preuve de lemme 2

Soit $n \geq 1$. On suppose la propriété vraie pour tout graphe non orienté d'ordre n.
Soit G graphe non orienté d'ordre $n+1$ ayant au moins $n+1$ arêtes.
Alors G a au moins une composante connexe C_j d'ordre n_j ayant au moins n_j arêtes.
En effet, si toutes les composantes connexes C_i de G d'ordre n_i ont m_i arêtes avec $m_i < n_i$ alors,

puisque $\sum_{i=1}^{i=p} n_i = n+1$,

G aurait $m = \sum_{i=1}^{i=p} m_i < \sum_{i=1}^{i=p} n_i$ arêtes d'où $m < n+1$:

contradiction avec l'hypothèse sur le nombre d'arêtes de G.
Dans la composante connexe C_j d'ordre $n_j \leq n$, par hypothèse de récurrence, il y a un cycle.

Remarques

▷ Attention, les réciproques des lemmes 1 et 2 ne sont pas vraies.

Autre caractérisation

Propriété

Soit G un graphe non orienté d'ordre n. Les propositions suivantes sont équivalentes :

1. G est un arbre
2. G est connexe et si on lui retire une arête il n'est plus connexe
3. G est sans cycle et si on lui rajoute une arête on forme un cycle.

Autre caractérisation

Propriété

Soit G un graphe non orienté d'ordre n. Les propositions suivantes sont équivalentes :

1. G est un arbre
2. G est connexe et si on lui retire une arête il n'est plus connexe
3. G est sans cycle et si on lui rajoute une arête on forme un cycle.

Autre caractérisation

Propriété

Soit G un graphe non orienté d'ordre n. Les propositions suivantes sont équivalentes :

1. G est un arbre
2. G est connexe et si on lui retire une arête il n'est plus connexe
3. G est sans cycle et si on lui rajoute une arête on forme un cycle.

Isthme

Définition

On appelle isthme une arête e telle que le nombre de composantes connexes de $G - e$ est strictement supérieur au nombre de composantes connexes de G.

Toute arête d'un arbre est un isthme et sa suppression engendre exactement deux composantes connexes.

Arbre couvrant

Définition

Soit G un graphe non orienté. Un *arbre couvrant* de G est un graphe partiel de G qui est un arbre.

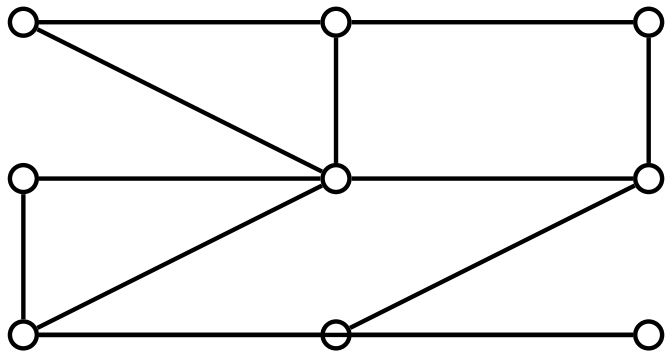
remarque : l'existence d'un arbre couvrant de G implique que G est connexe.
Pour un graphe non connexe, on parle de forêt comme un ensemble d'arbres couvrants de ses composantes connexes.

Arbre couvrant

Définition

Soit G un graphe non orienté. Un *arbre couvrant* de G est un graphe partiel de G qui est un arbre.

remarque : l'existence d'un arbre couvrant de G implique que G est connexe.
Pour un graphe non connexe, on parle de forêt comme un ensemble d'arbres couvrants de ses composantes connexes.

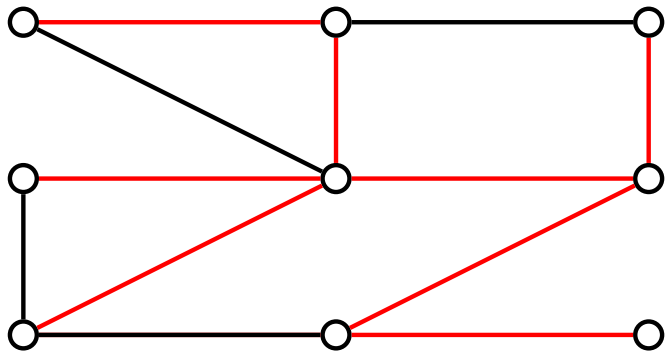


Arbre couvrant

Définition

Soit G un graphe non orienté. Un *arbre couvrant* de G est un graphe partiel de G qui est un arbre.

remarque : l'existence d'un arbre couvrant de G implique que G est connexe.
Pour un graphe non connexe, on parle de forêt comme un ensemble d'arbres couvrants de ses composantes connexes.

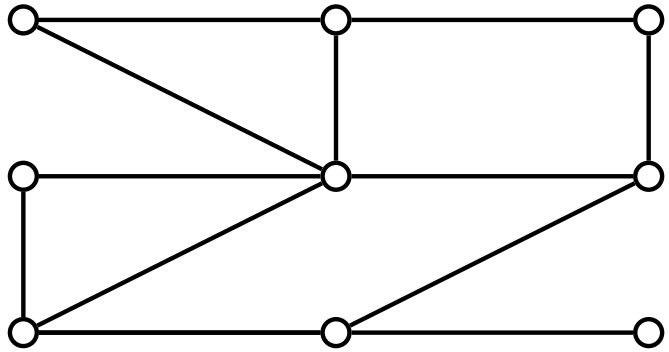


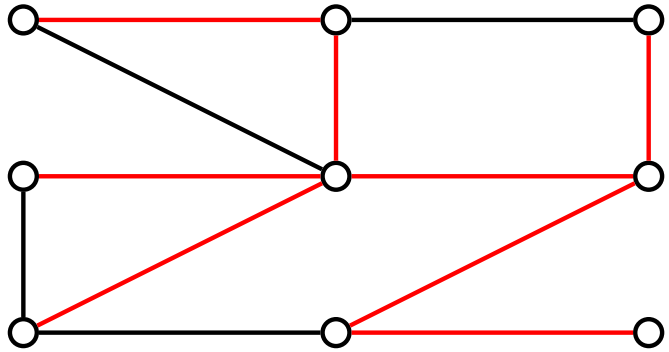
Propriété

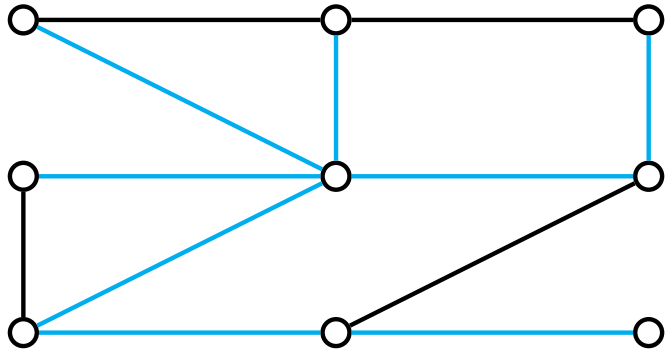
Propriété

tout graphe connexe a un arbre couvrant.

remarque : pour un graphe connexe, il peut y avoir plusieurs arbres couvrants non isomorphes.







Caractérisations

Propriété

Un graphe partiel d'un graphe connexe G est un arbre couvrant de G *si et seulement si* il est connexe et minimal avec cette propriété relativement à la suppression d'arête - *i.e.* s'il perd une arête il n'est plus connexe.

Propriété

Un graphe partiel d'un graphe connexe G est un arbre couvrant de G *si et seulement si* il est acyclique et maximal avec cette propriété relativement à l'ajout d'arête - *i.e.* si on lui ajoute une arête on forme un cycle

Conséquences

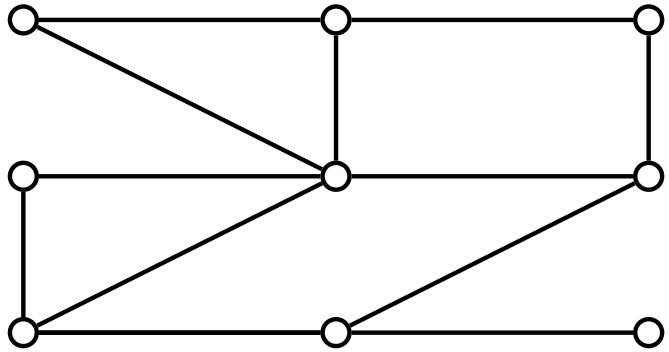
Les deux propriétés suivantes permettent de préciser la conséquence d'un ajout d'arêtes à un arbre couvrant.

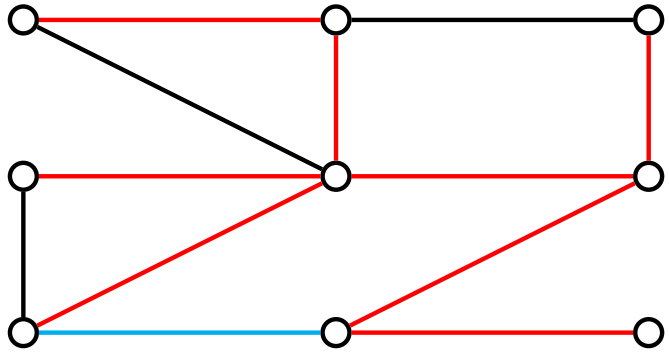
Propriété

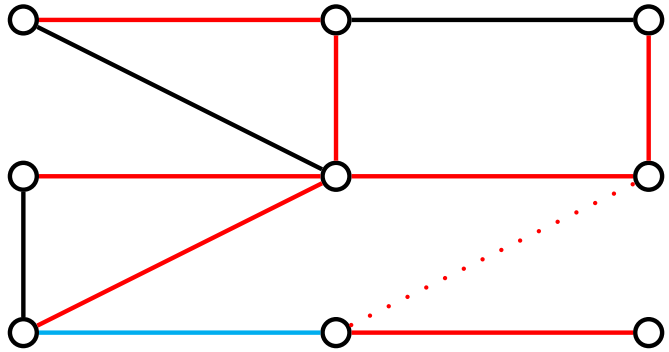
Soit T un arbre couvrant de G et e une arête de G n'appartenant pas à T. Le graphe partiel $T + e$ contient un unique cycle élémentaire.

Propriété

Soit T un arbre couvrant de Get e une arête de G n'appartenant pas à T. Soit e' une arête du cycle de $T + e$. Alors $T + e - e'$ est un arbre couvrant de G non nécessairement isomorphe à T.







Problème de l'arbre couvrant minimal

Le problème abordé ici correspond à la minimisation du coût de construction d'un réseau de communications connaissant le coût de chaque liaison possible entre les noeuds du réseau à construire. Puisqu'on souhaite un réseau à coût minimal,

- on choisira le nombre juste suffisant d'arêtes pour assurer la connexité
- on choisira, dans cette optique, un arbre couvrant ayant un coût total minimal

C'est le problème de l'arbre couvrant minimal

Problème de l'arbre couvrant minimal

Le problème abordé ici correspond à la minimisation du coût de construction d'un réseau de communications connaissant le coût de chaque liaison possible entre les noeuds du réseau à construire. Puisqu'on souhaite un réseau à coût minimal,

- on choisira le nombre juste suffisant d'arêtes pour assurer la connexité
- on choisira, dans cette optique, un arbre couvrant ayant un coût total minimal

C'est le problème de l'arbre couvrant minimal

Problème de l'arbre couvrant minimal

Le problème abordé ici correspond à la minimisation du coût de construction d'un réseau de communications connaissant le coût de chaque liaison possible entre les noeuds du réseau à construire. Puisqu'on souhaite un réseau à coût minimal,

- on choisira le nombre juste suffisant d'arêtes pour assurer la connexité
- on choisira, dans cette optique, un arbre couvrant ayant un coût total minimal

C'est le problème de l'arbre couvrant minimal

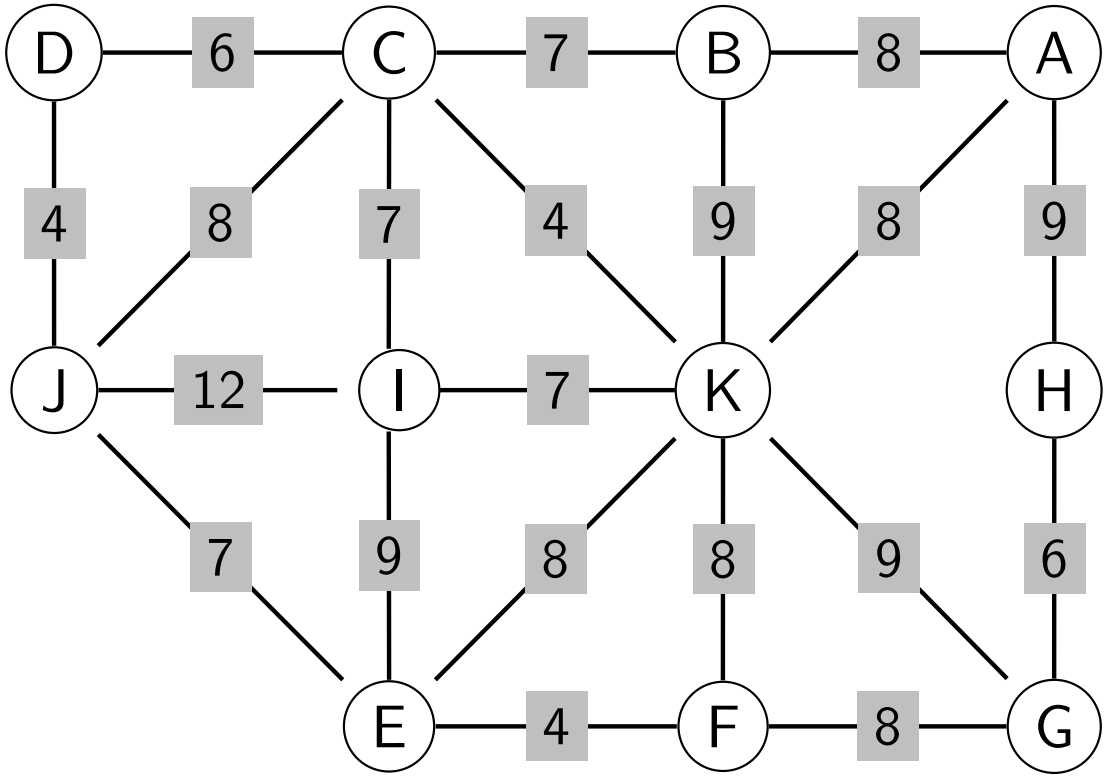
Problème de l'arbre couvrant minimal

Le problème abordé ici correspond à la minimisation du coût de construction d'un réseau de communications connaissant le coût de chaque liaison possible entre les noeuds du réseau à construire. Puisqu'on souhaite un réseau à coût minimal,

- on choisira le nombre juste suffisant d'arêtes pour assurer la connexité
- on choisira, dans cette optique, un arbre couvrant ayant un coût total minimal

C'est le problème de l'arbre couvrant minimal

Exemple



Une solution naïve

Une solution serait de

- de calculer tous les arbres couvrants du graphe
- de garder le moins coûteux

Mais, sachant que le graphe K_n a n^{n-2} arbres couvrants, cette méthode prendrait trop de temps.
Par exemple, pour 20 sommets, si on compte $1\mu s$ pour calculer 1000 arbres couvrants et leur coût, il faudrait $20^{18}\,ns$ soit plus de 20 000 siècles.

Algorithme de Kruskal

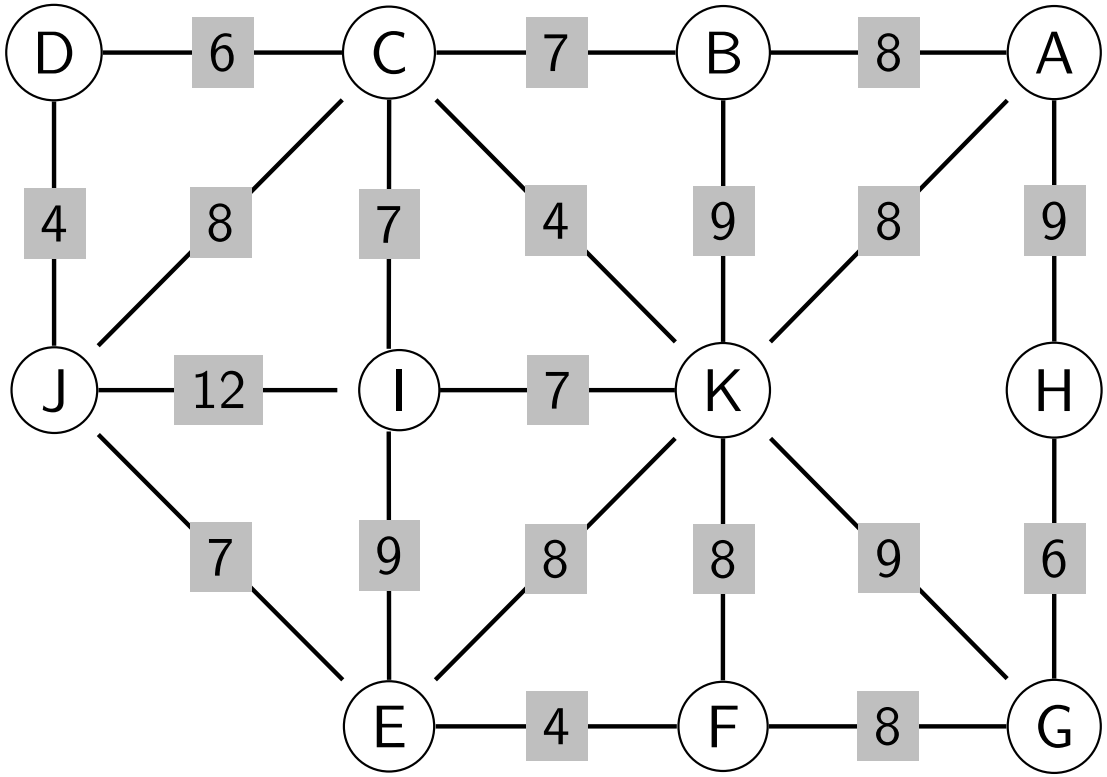
L'algorithme de Kruskal est un algorithme glouton qui repose sur le fait qu'un arbre couvrant d'un graphe d'ordre n a $n-1$ arêtes et est acyclique.
Le choix des arêtes se fera donc sur deux critères : la conservation de l'acyclicité et un coût minimal et s'arrêtera lorsque le bon nombre aura été choisi.

Algorithme de Kruskal

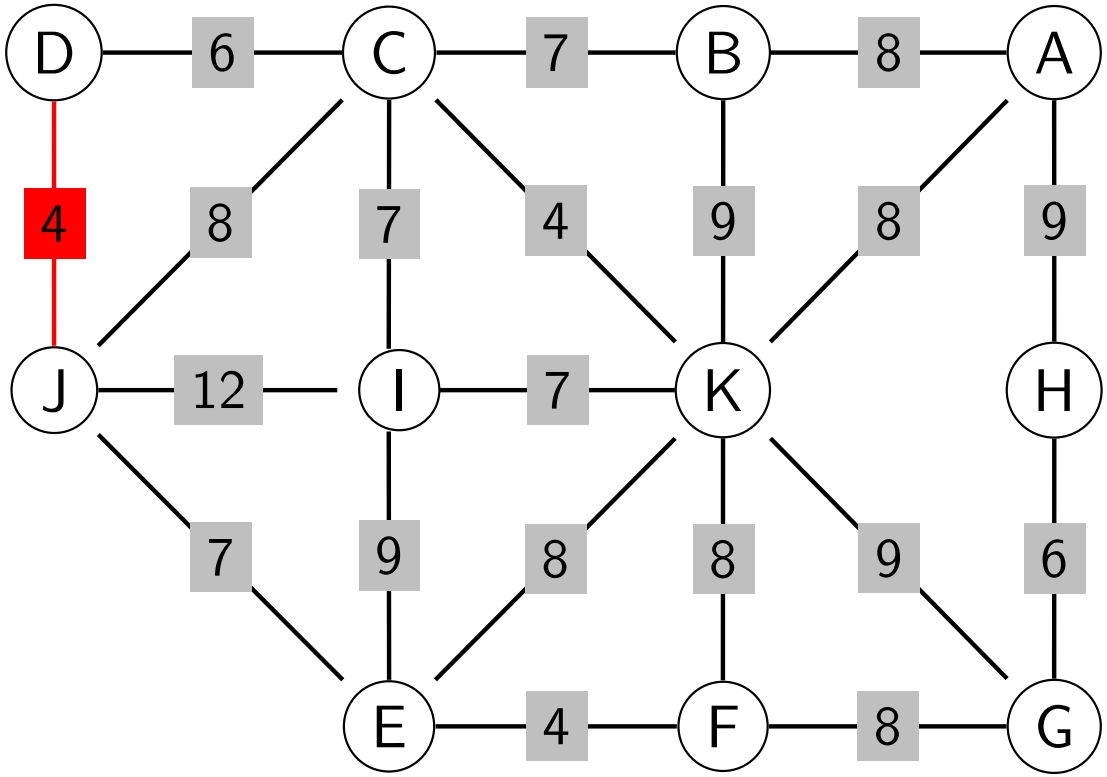
Soit E l'ensemble des sommets de G. On utilisera un ensemble d'arêtes T qui sera en sortie l'arbre couvrant minimal et un ensemble d'arêtes F qui représentera les arêtes qui peuvent être choisies.
Pour simplifier, on supposera que l'ensemble F est trié par ordre de coût croissant.

ALGORITHME DE KRUSKAL(G : graphe).
  $T \leftarrow \emptyset$
  $F \leftarrow E$
  TANT QUE $Card(T) < n - 1$ FAIRE
    soit $e \in F$ la première arêtede F
    $F \leftarrow F - e$
    SI $T + e$ est sans cycle ALORS $T \leftarrow T \cup e$ FINSI
  FINTANT QUE
FIN

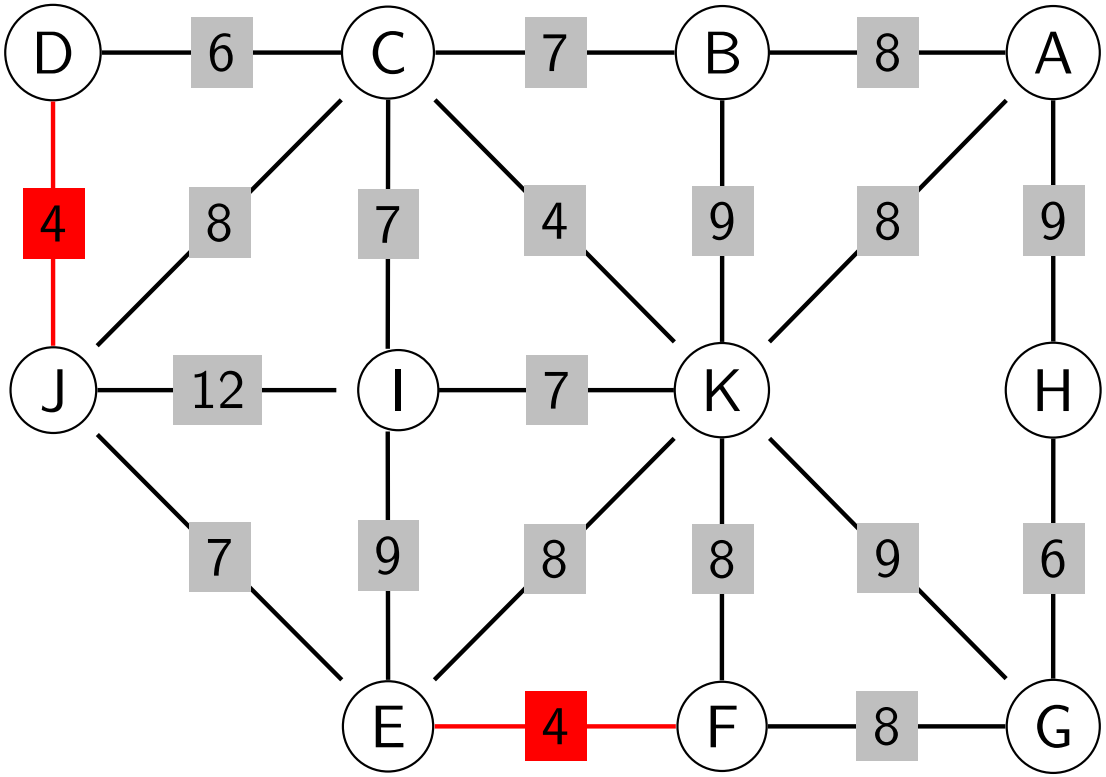
Exemple



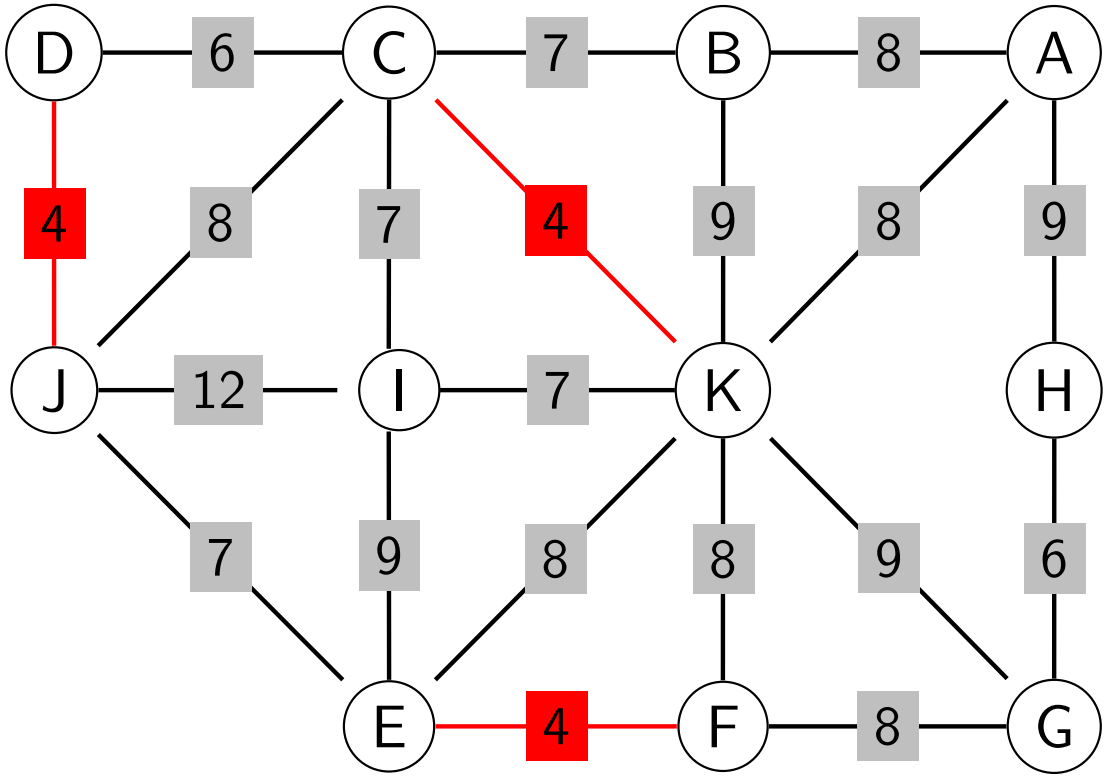
Exemple



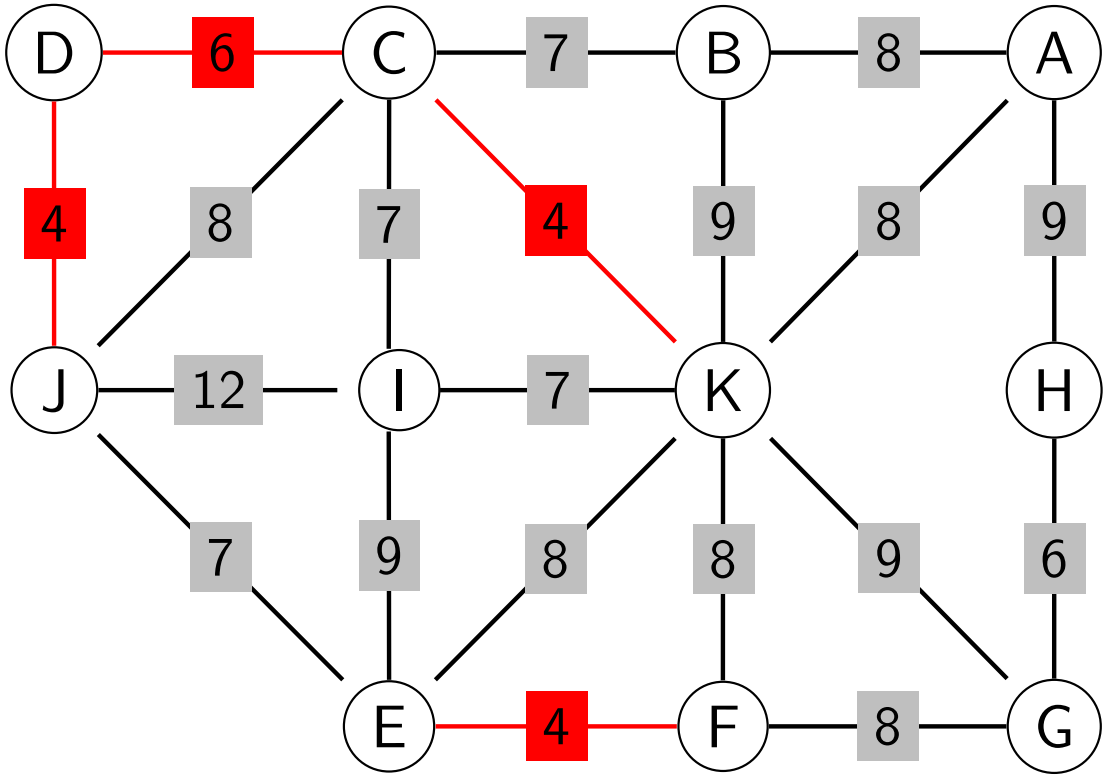
Exemple



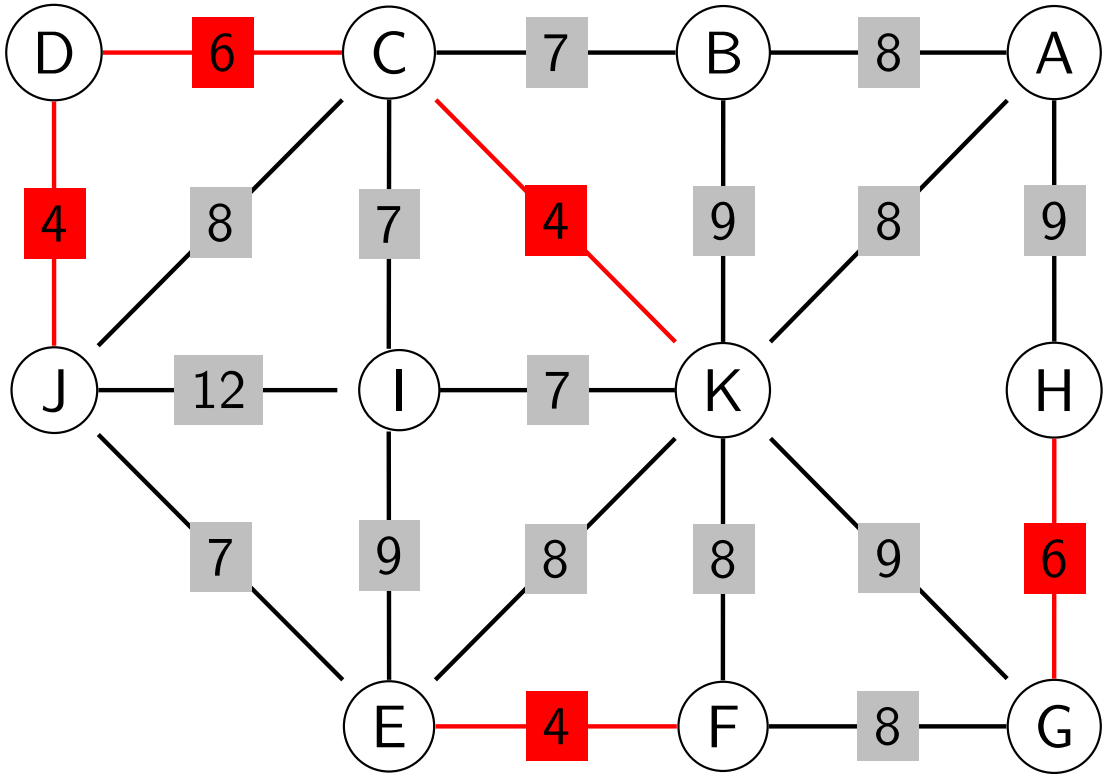
Exemple



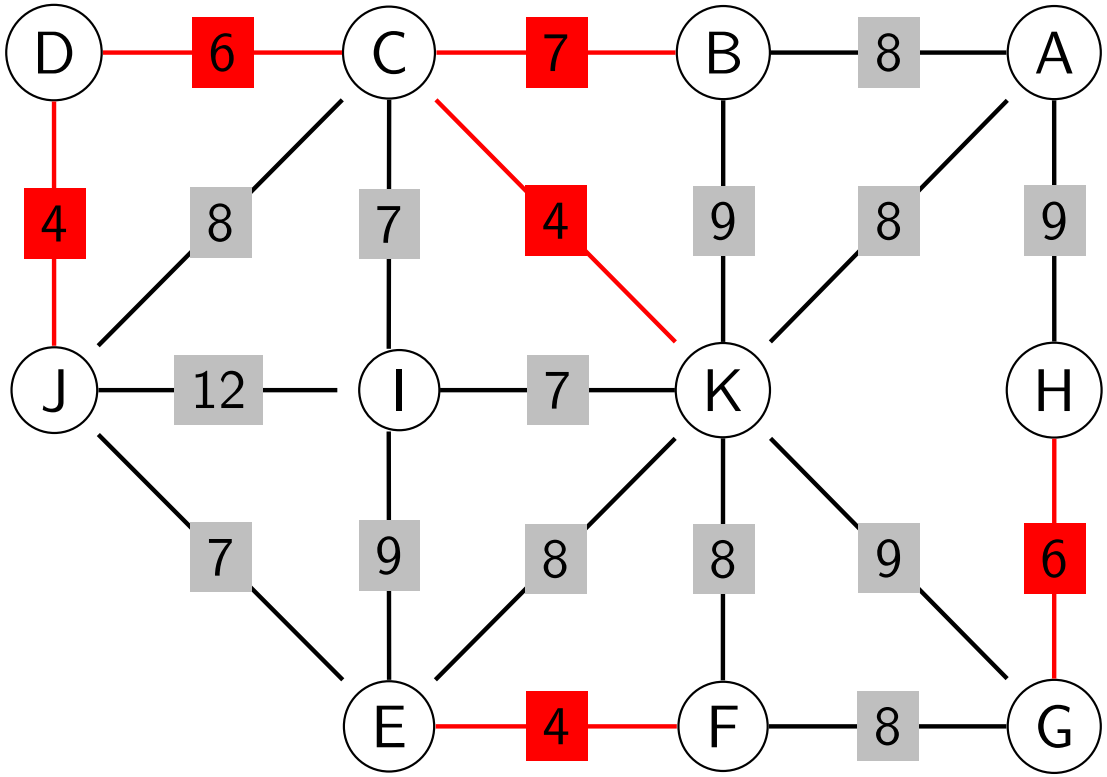
Exemple



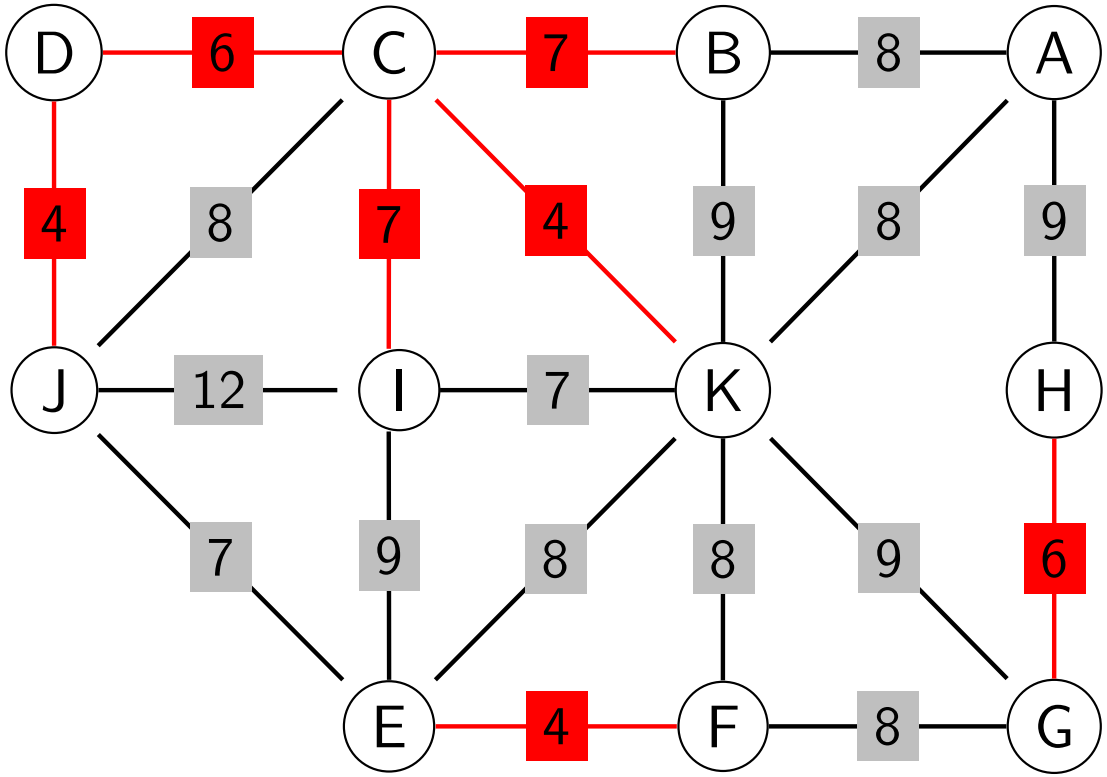
Exemple



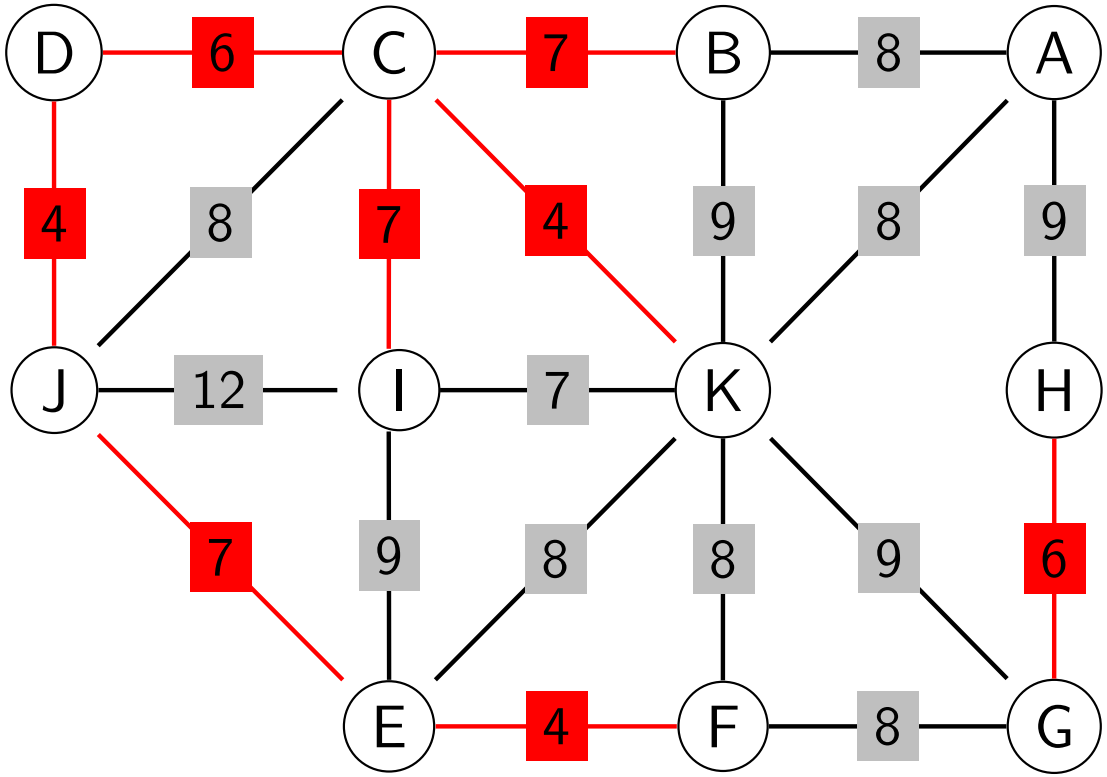
Exemple



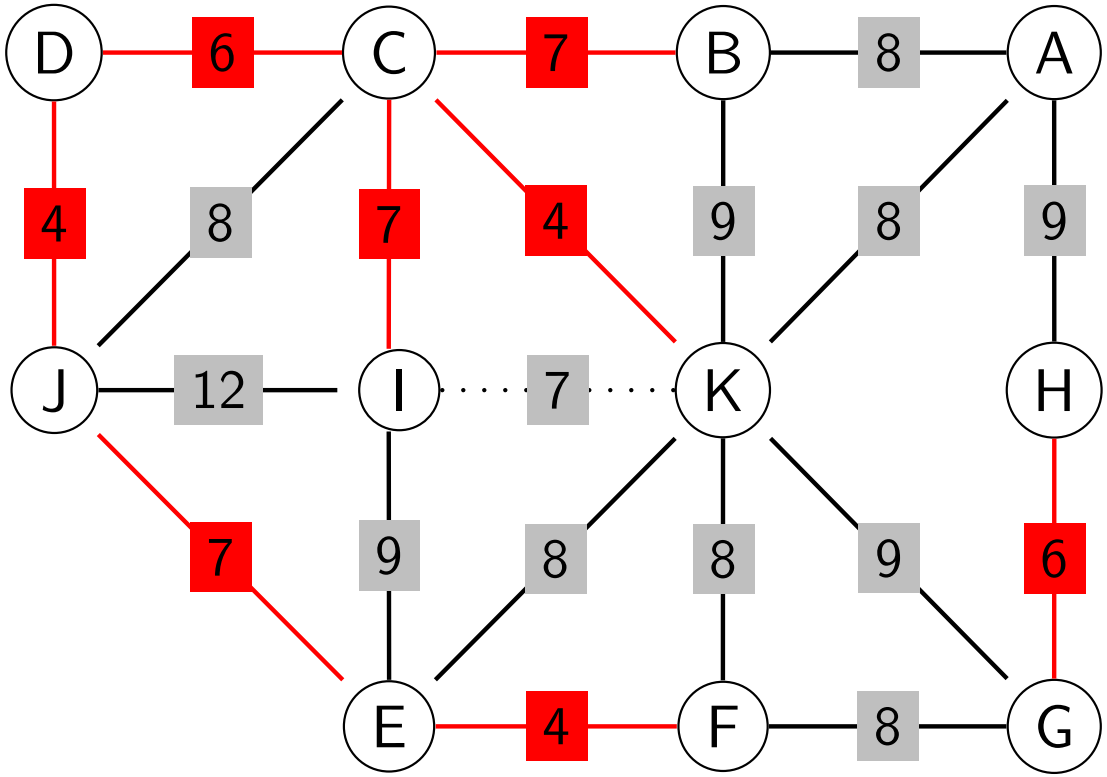
Exemple



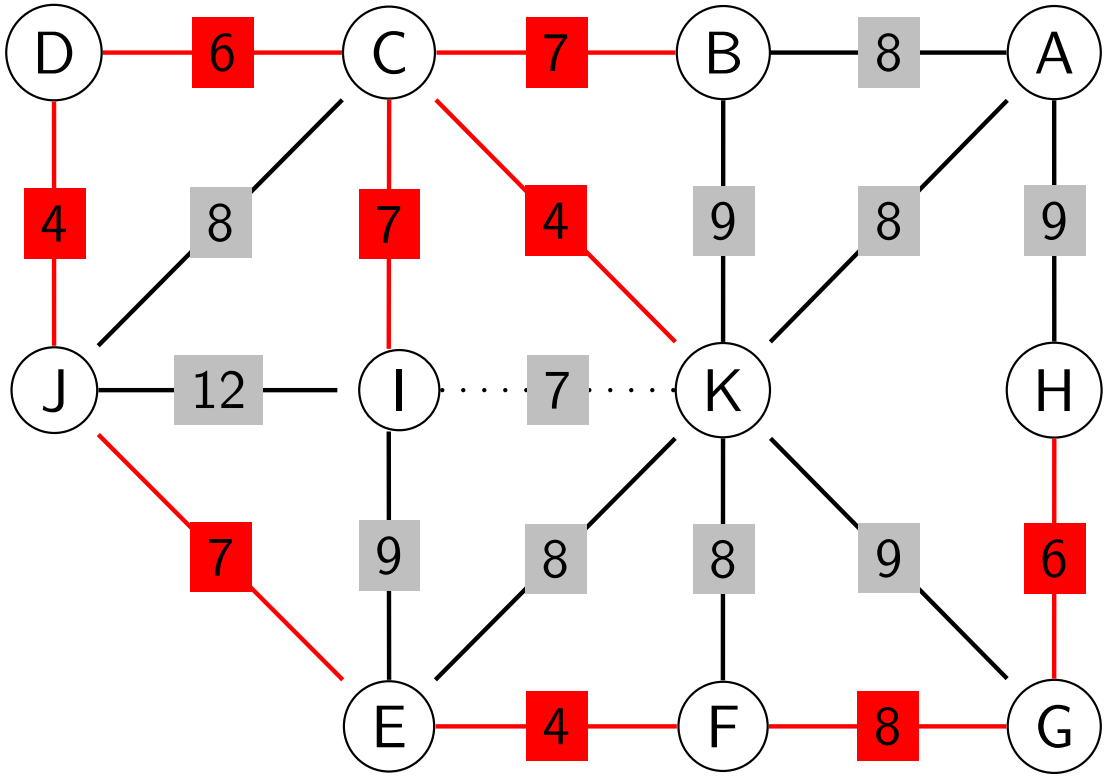
Exemple



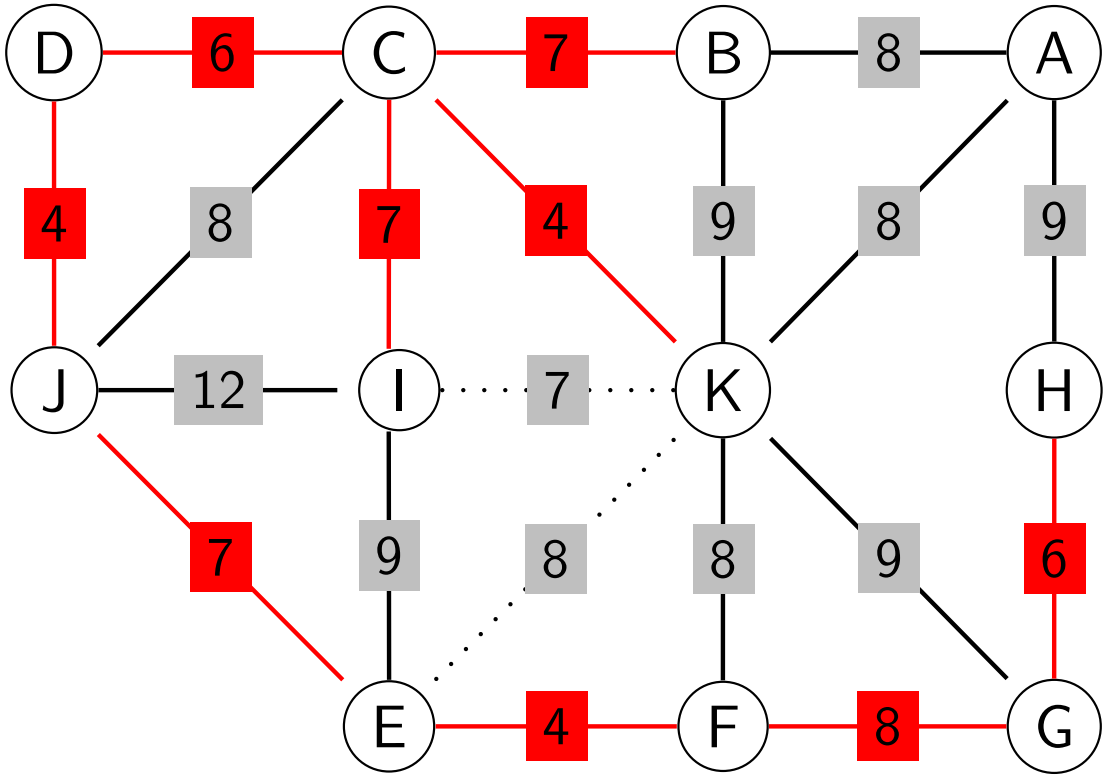
Exemple



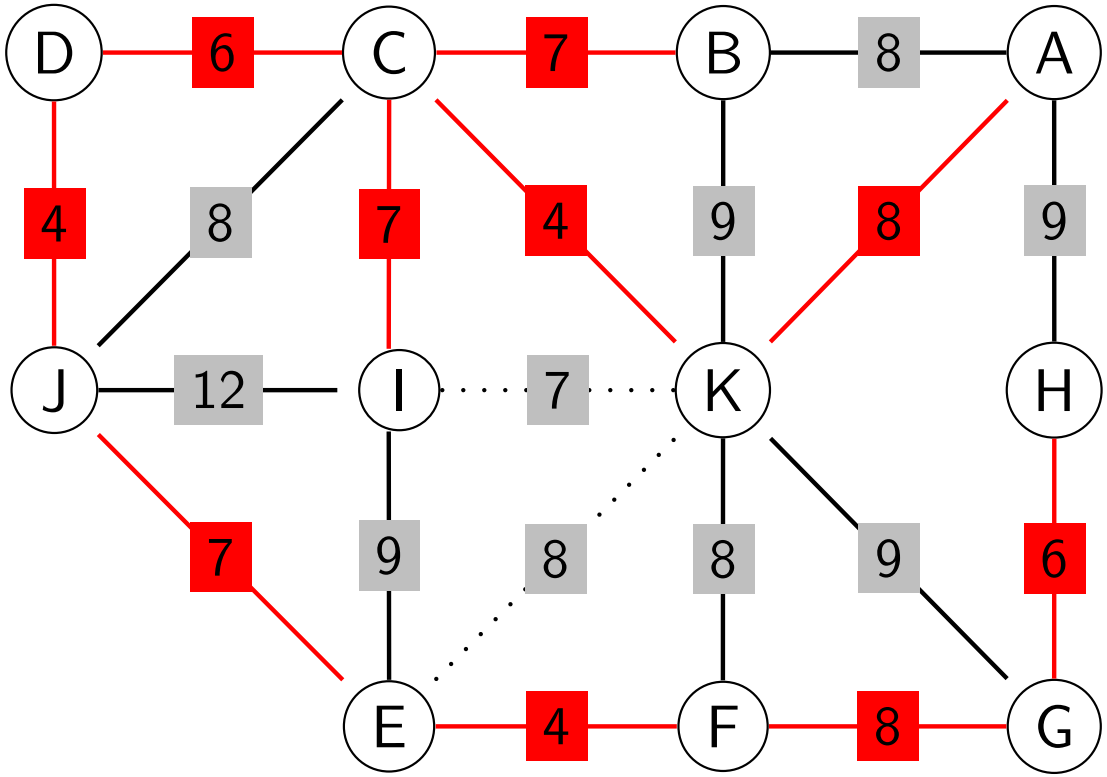
Exemple



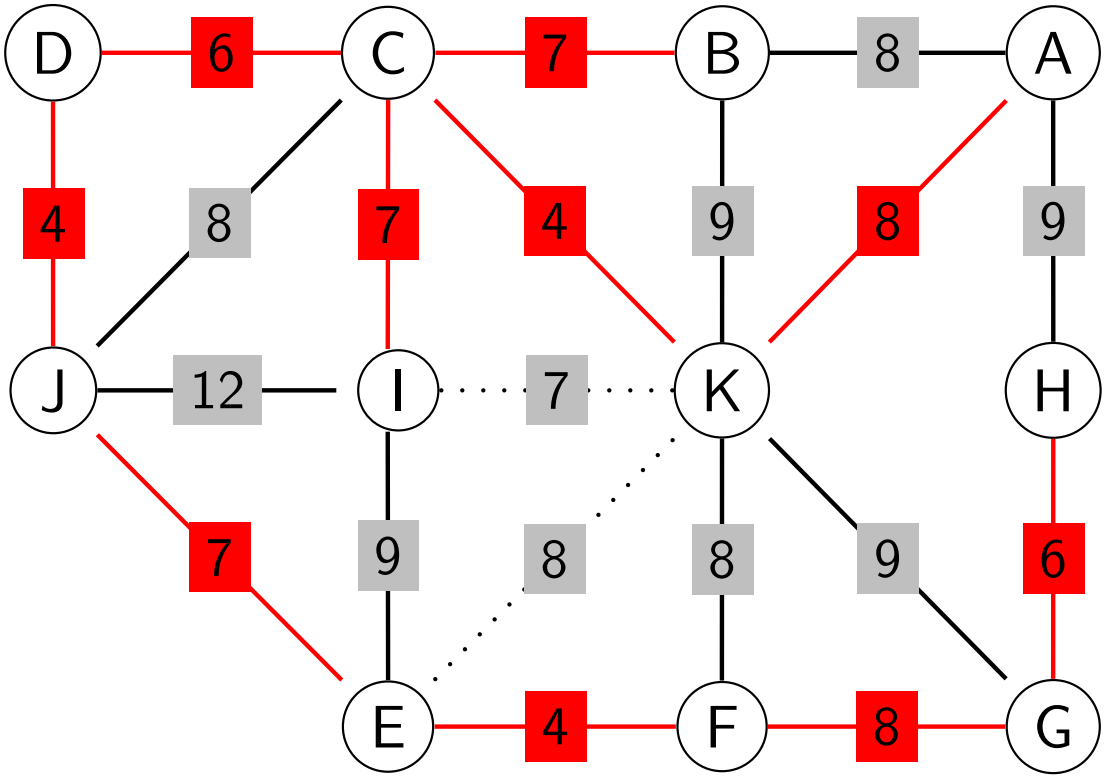
Exemple



Exemple



Exemple



Coût total = 61

Complexité

La répétition TANT QUE se fait au plus m fois et chaque fois le test « sans cycle » peut se faire grâce à un marquage d'arêtes en m instructions ; donc on dit que la complexité de l'algorithme est au maximum de l'odre de m^2.

Preuve

Soit n est l'ordre de G.
L'algorithme produit bien un arbre couvrant de G puisqu'il termine lorsque $n-1$ arêtes sont choisies et T est acyclique.
Supposons que T ne soit pas minimal.

Preuve

Si $e_1, \cdots, e_{n-1}$ sont les arêtes de T dans l'ordre de choix de l'algorithme, alors $p(e_1) \leq \cdots \leq p(e_{n-1})$ par construction.

Soit alors A un arbre couvrant minimal tel que l'indice de la première arête de T qui ne soit pas une arête de A soit maximum.
Soit k cet indice, on a alors $e_1 \cdots e_k \in T$, $e_1 \cdots e_{k-1} \in A$ et $e_k \notin A$.
Alors $A + e_k$ contient un unique cycle C. C ne peut pas être constitué uniquement d'arête de T (hormis e_k) car sinon T contiendrait un cycle.

Preuve

Soit alors e' une arête de C qui appartient à A mais qui n'appartient pas à T.
$A + e_k - e'$ est donc un arbre couvrant de G et $p(A + e_k - e') = p(A) + p(e_k) - p(e')$.
Dans l'exécution de l'algorihtme de Kruskal, e_k a été choisie de poids minimal tel que $G[e_1 \cdots e_k]$ soit acyclique.
Mais $G[e_1 \cdots e_{k-1}, e']$ est aussi acyclique puisque sous-graphe de A.
Donc $p(e') \geq p(e_k)$ et $p(A + e_k - e') \leq p(A)$. On en déduit que $A + e_k - e'$ est optimal et diffère de T par un arête d'indice strictement supérieur à k : contradiction.

Algorithme de Prim

L'algorithme de Kruskal veille à maintenir la propriété d'acyclicité d'un arbre alors que l'algorithme de Prim se base sur la connexité d'un arbre.
L'algorithme de Prim fait pousser un arbre couvrant minimal en ajoutant au sous-arbre T déjà construit une nouvelle branche parmi les arêtes de poids minimal joignant un sommet de T à un sommet n'appartenant pas à T. L'algorithme s'arrête lorsque tous les sommets du graphe appartiennent à T.

Algorithme de Prim

Soit X l'ensemble des sommets de G. On utilisera un ensemble d'arêtes T qui sera en sortie l'arbre couvrant minimal et un ensemble de sommets S qui contiendra les sommets de T.

LGAORITHME DE PRIM (G : graphe)

choisir un sommet x de G.

$S \leftarrow S \cup x$

$T \leftarrow \emptyset$

TANT QUE $S \neq X$ FAIRE

trouver une arête $\{ys\}$ de poids minimal telle que $y \in X - S$ et $s \in S$

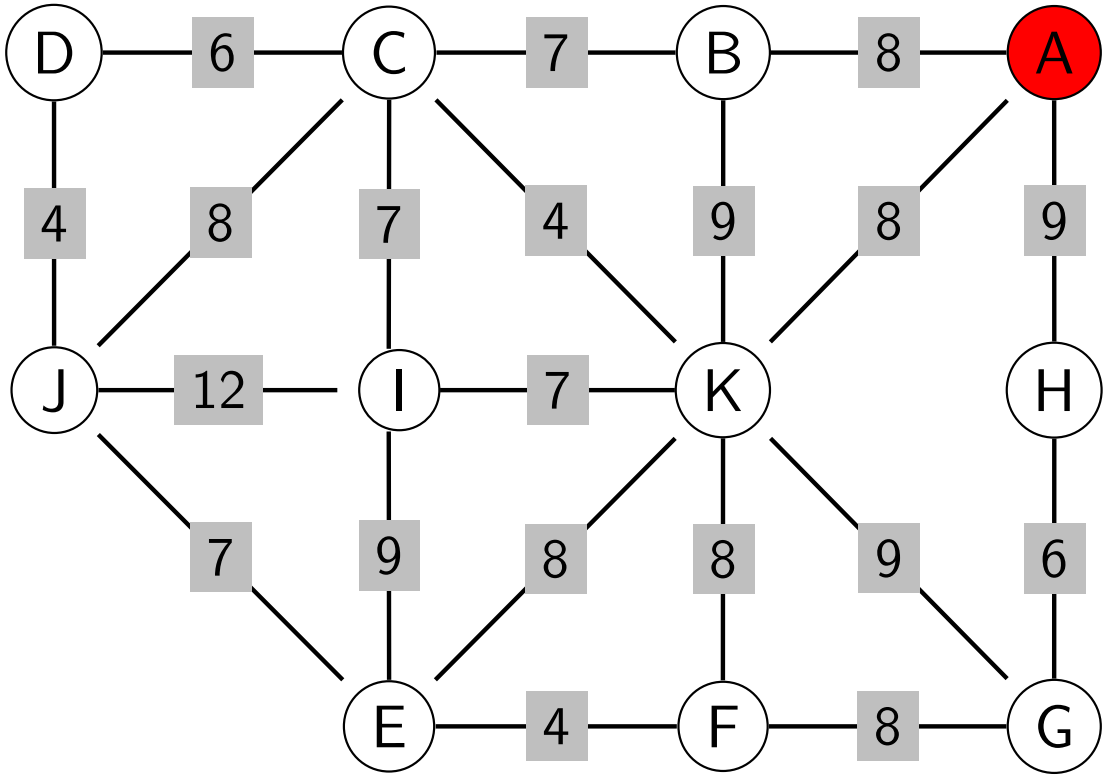
$T \leftarrow T \cup \{ys\}$

$S \leftarrow S \cup y$

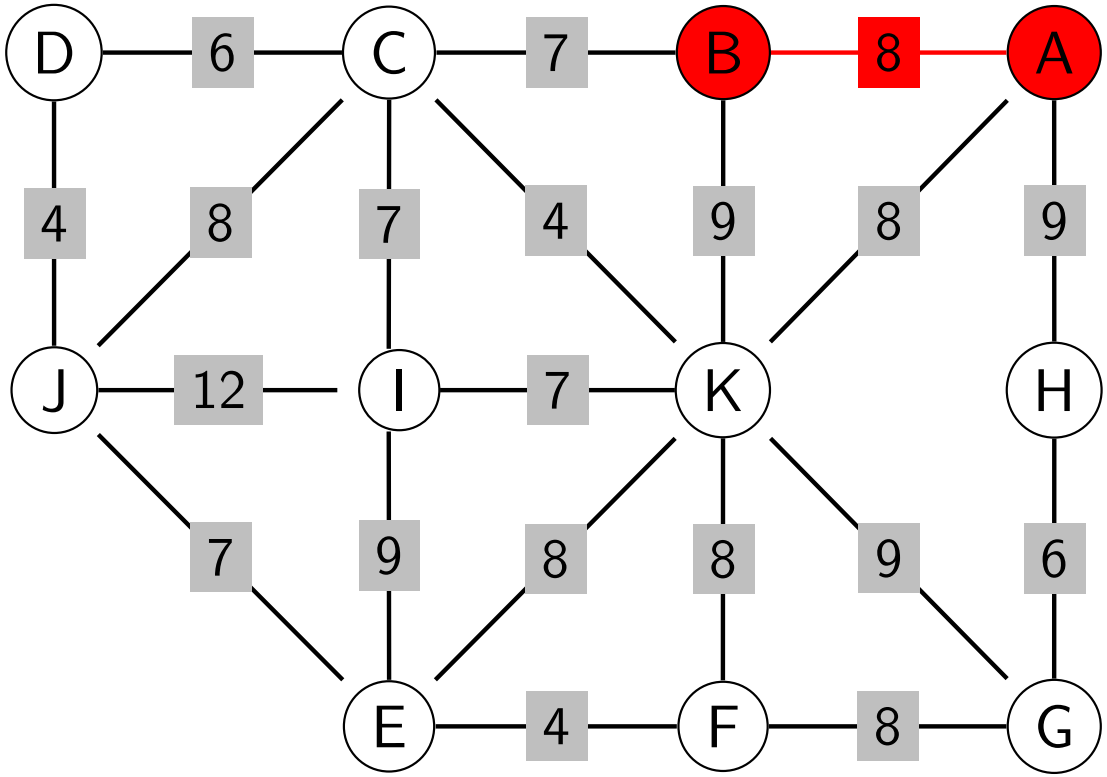
FINTANT QUE

FIN

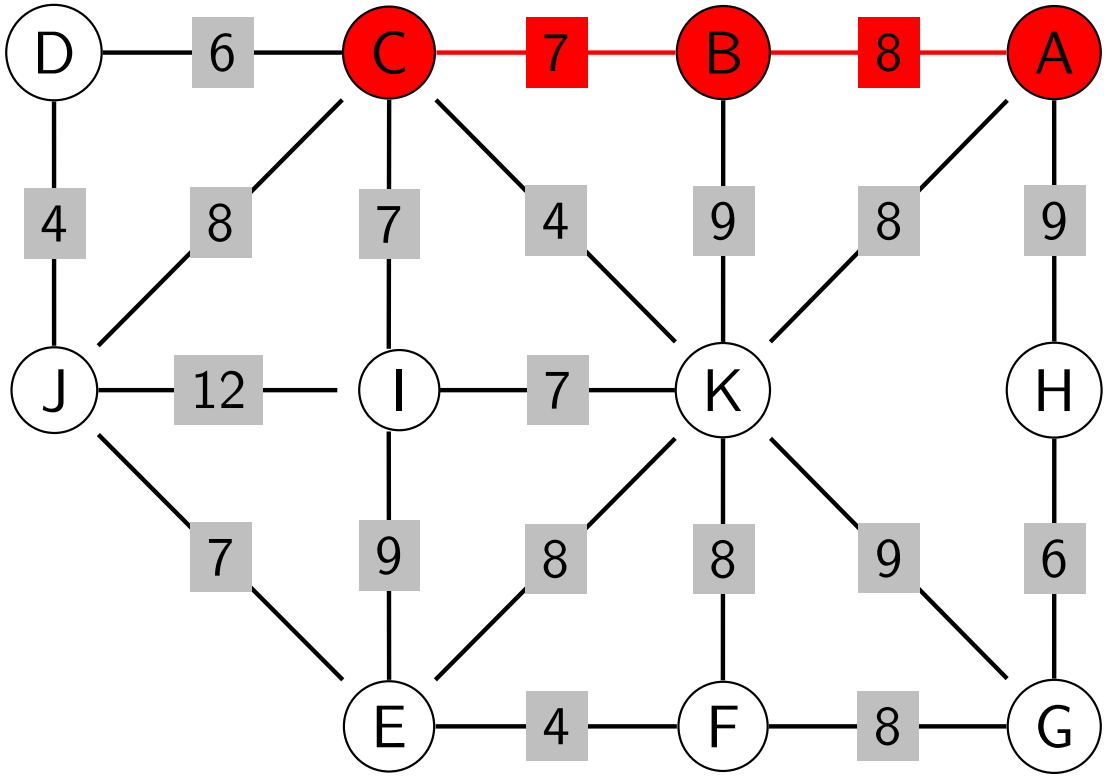
Exemple



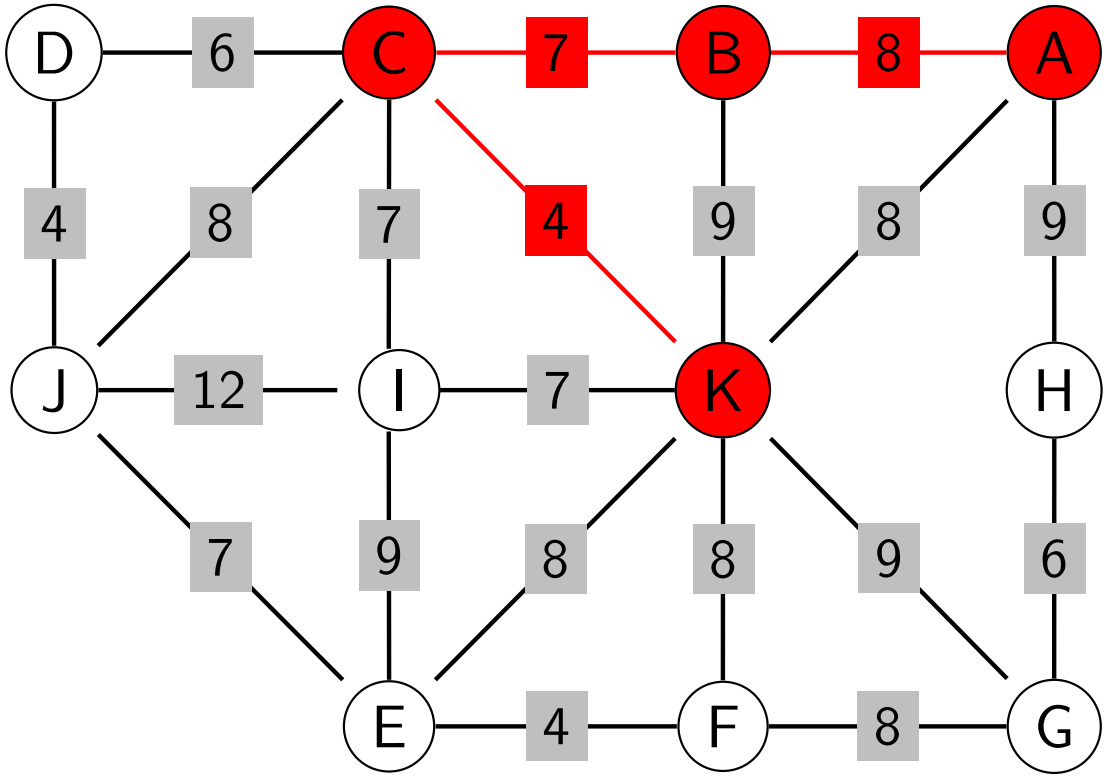
Exemple



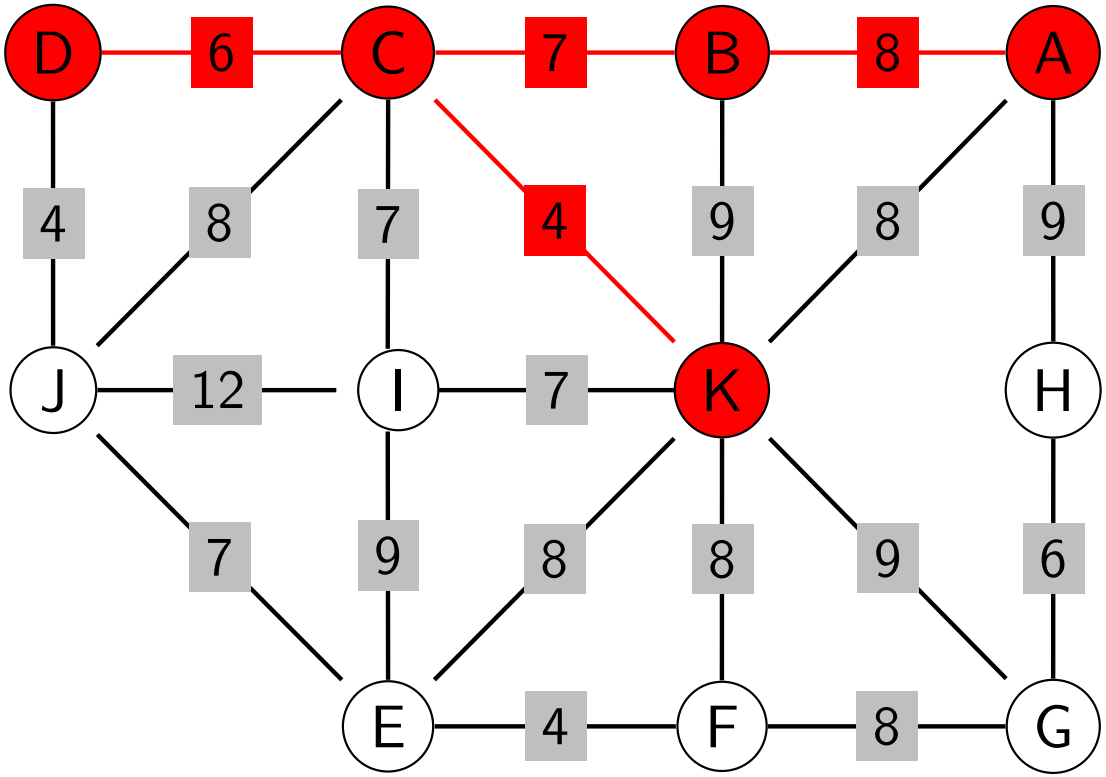
Exemple



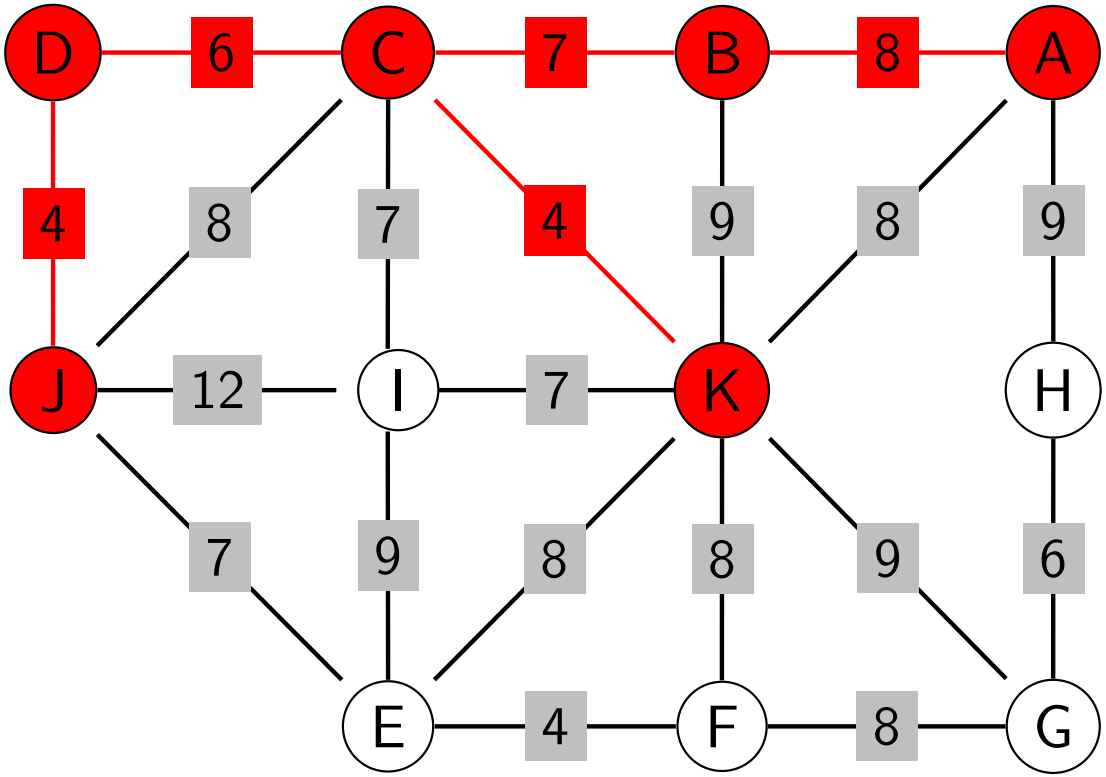
Exemple



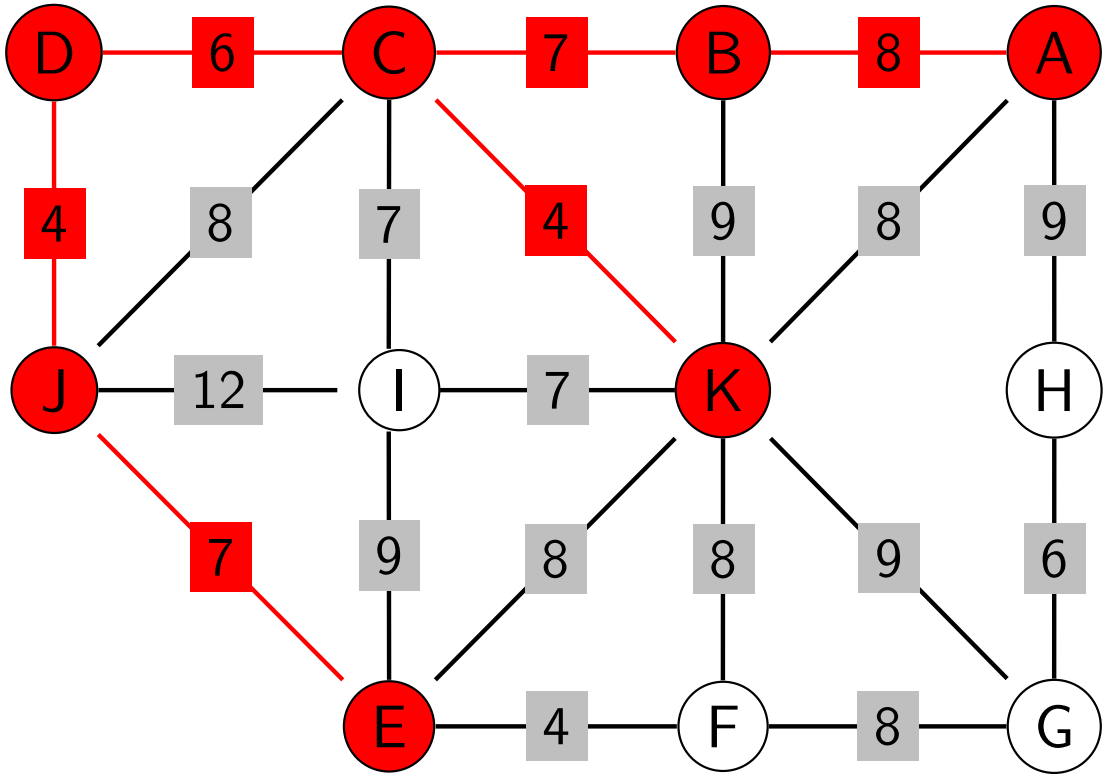
Exemple



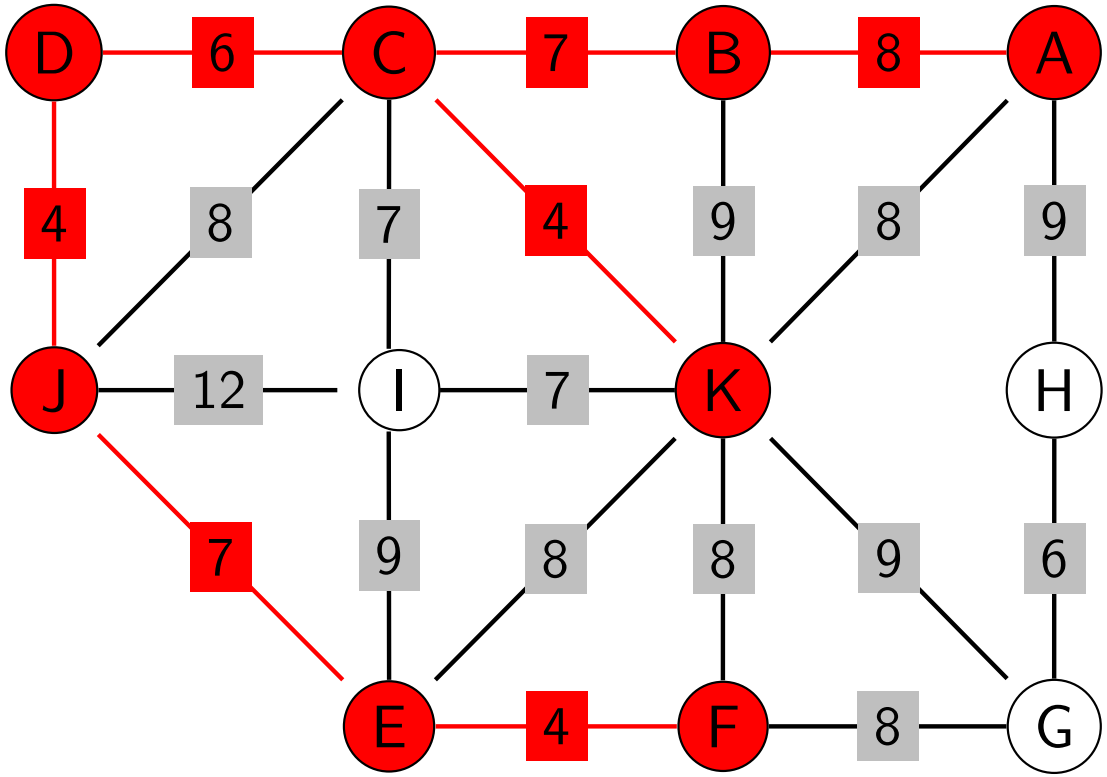
Exemple



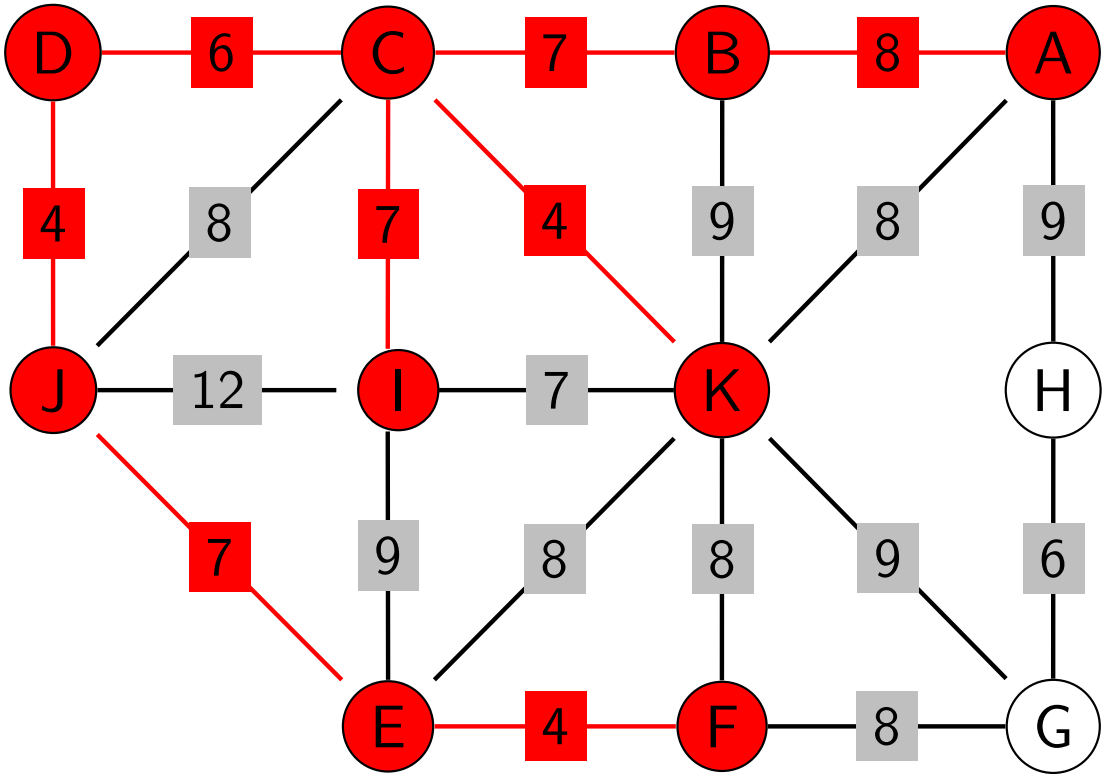
Exemple



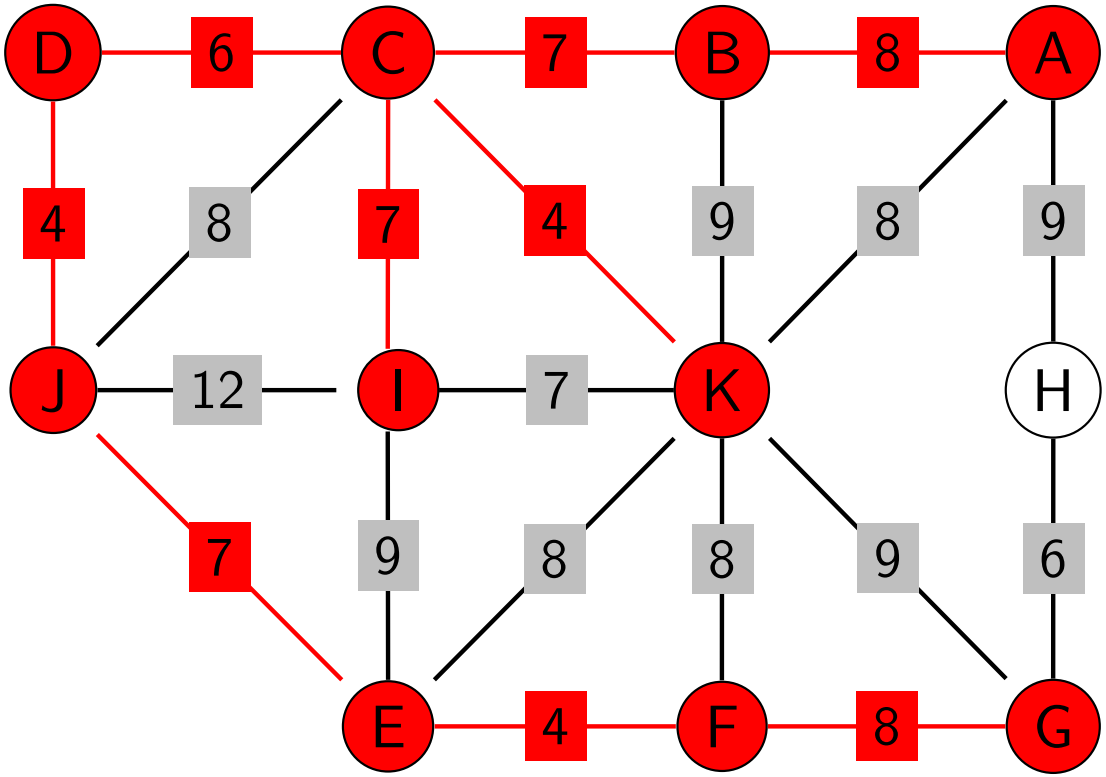
Exemple



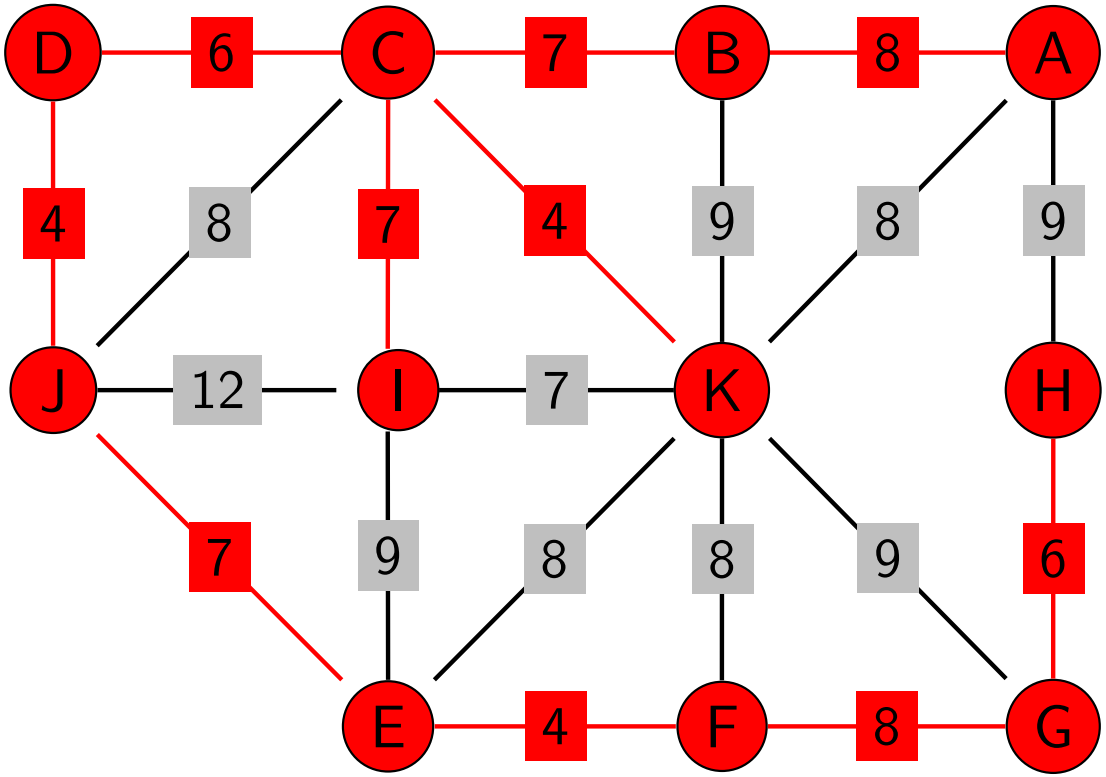
Exemple



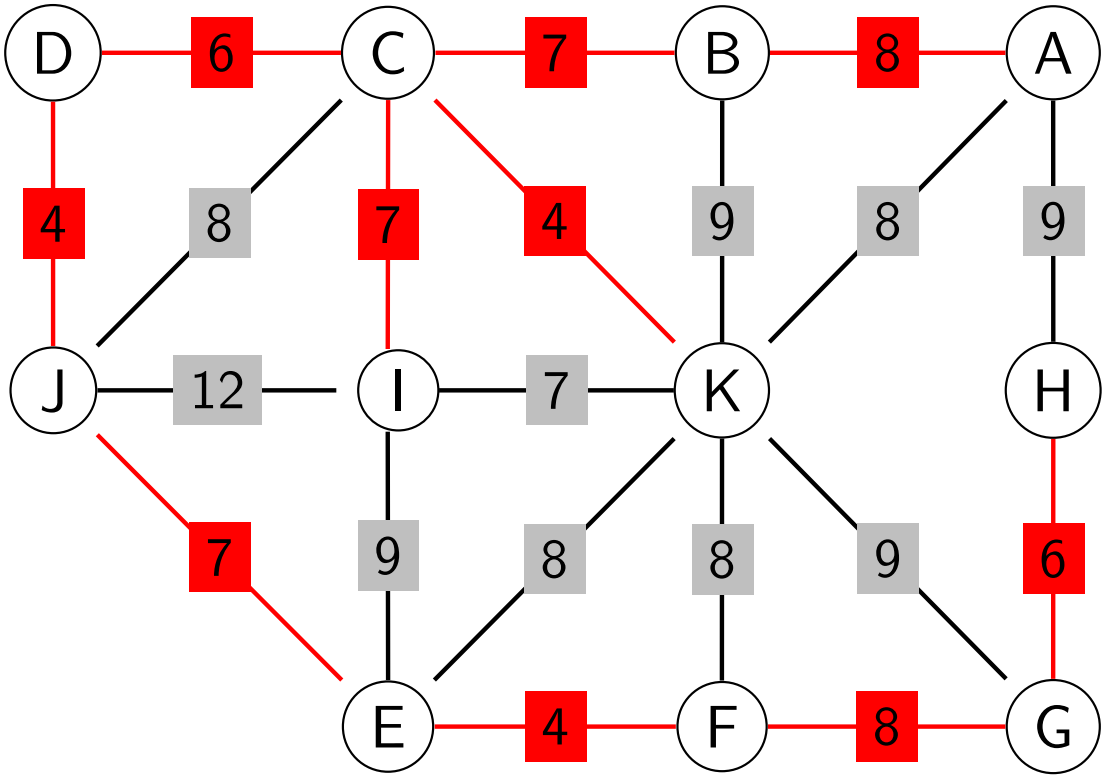
Exemple



Exemple



Exemple



Coût total = 61

Complexité

Soit n l'ordre de G.
Si S a k sommets, pour choisir une arête de poids minimal relient un sommet de S à un sommet de $X - S$, on doit trouver le poids minimum parmi au plus $k(n-k)$ arêtes puisque chaque sommet de S est adjacent à au plus $(n-k)$ sommets de $X - S$.
Or k varie de 1 à n, donc $k(n-k) < (n-1)^2$.
Ce choix est effectué n fois dans la boucle TANT QUE.
Donc la complexité est de l'ordre $O(n^3)$.

Chapitre IV – Problème du plus court chemin

- Algorithme de Dijkstra
- Algorithme de Floyd
- Algorithme de Bellman

Chapitre IV – Problème du plus court chemin

- Algorithme de Dijkstra
- Algorithme de Floyd
- Algorithme de Bellman

Chapitre IV – Problème du plus court chemin

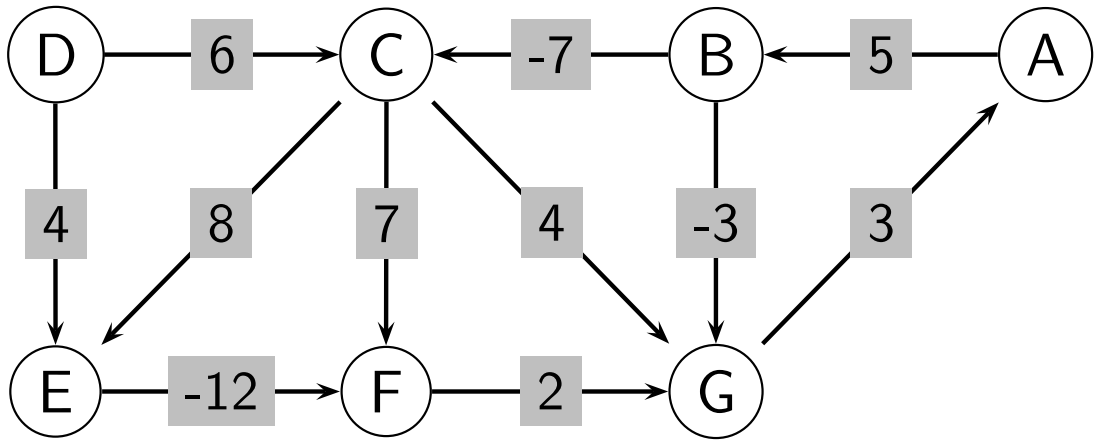
- Algorithme de Dijkstra
- Algorithme de Floyd
- Algorithme de Bellman

Problème

Le problème posé dans ce chapitre est de déterminer dans un graphe *valué* (orienté ou non orienté) le plus court chemin entre deux sommets.
Les arêtes (arcs) du graphe seront donc affectées d'une valeur par une fonction v à valeurs dans $\mathbb{R}$ et on calculera la longueur d'un chemin (chaîne) par la somme des valeurs de ses arêtes (arcs).

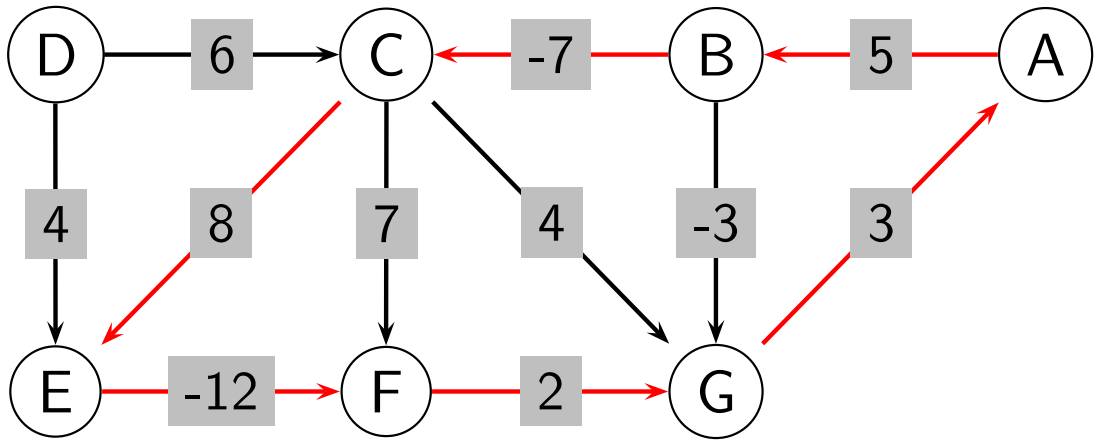
Problème

Selon la fonction v, on peut être amené à des solutions différentes. Plus précisement, il est aisé de remarquer qu'il n'y a pas de solution lorsque le graphe contient un circuit (ou un cycle) absorbant *i.e.* un circuit pour lequel la somme des valeurs de ses arcs est négative.



Problème

Selon la fonction v, on peut être amené à des solutions différentes. Plus précisement, il est aisé de remarquer qu'il n'y a pas de solution lorsque le graphe contient un circuit (ou un cycle) absorbant *i.e.* un circuit pour lequel la somme des valeurs de ses arcs est négative.



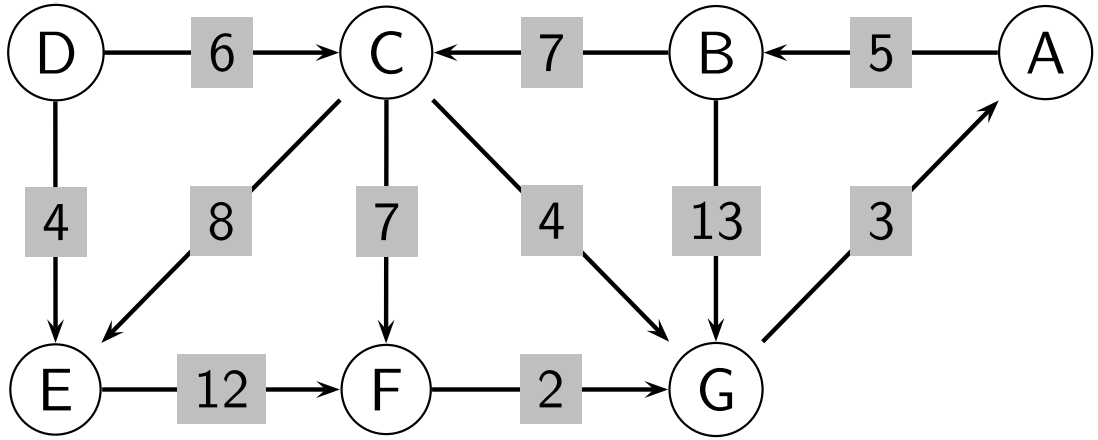
Algorithme de Dijkstra

Cet algorithme est conçu pour des **valuations positives**.
Un sommet de départ x est fixé.
C'est un algorithme glouton qui construit progressivement un ensemble de sommets pour lesquels on connaît un plus court chemin depuis x.
A chaque étape on choisit un sommet dont la distance à x est minimale parmi ceux qui n'ont pas encore été choisis.

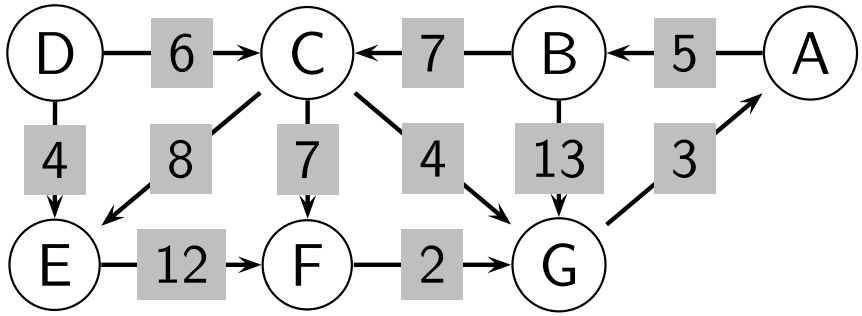
Algorithme de Dijkstra

PROCÉDURE Dijkstra(G : graphe, x : sommet)
  $C \leftarrow \{x\}$; $D[x] \leftarrow 0$
  POUR tout sommet $s \neq x$ FAIRE
    $D[s] \leftarrow v(x,s)$ ou $+\infty$ si s n'est pas adjacent à x
    $P[s] \leftarrow x$
  FINPOUR
  TANT QUE $|C| < n$ FAIRE
    choisir y tel que $D[y] = min\{D[z]; z \notin C\}$
    SI $D[y] = +\infty$ ALORS EXIT FINSI
    $C \leftarrow C \cup \{y\}$
    POUR chaque sommet $z \notin C$ FAIRE
      SI $D[z] > D[y] + v(yz)$ ALORS $D[z] \leftarrow D[y] + v(yz)$;
                                    $P[z] \leftarrow y$ FINSI
    FINPOUR
  FINTANT QUE
FINPROCÉDURE

Exemple



Exemple



A	B	C	D	E	F	G	$\mathcal{C}$
0	5(A)	∞	∞	∞	∞	∞	A
–	5(A)	12(B)	–	–	–	18(B)	+ B
–	–	12(B)	–	20(C)	19(C)	16(C)	+ C
–	–	–	–	–	–	16(C)	+ G
–	–	–	–	–	19(C)	–	+ F
–	–	–	–	20(C)	–	–	+ E
–	–	–	∞	–	–	–	+ D

Complexité

Dans l'algorithme on utilise implicitement l'ensemble $X - C$ que l'on notera CC. C'est l'ensemble des sommets parmi lesquels on choisira le prochain y.
La boucle TANT QUE a au plus $(n-1)$ itérations car on choisit un sommet à chaque itération.
Le choix du sommet y se fera en temps $\Theta(|CC|)$ et puisque $|CC|$ diminue d'une unité à chaque itération, on aura alors $\sum_{i=1}^{i=n}(i-1)$ comparaisons au total soit $\Theta(n^2)$.
Supprimer y de CC prend un temps au maximum $\Theta(n)$ opérations.
La boucle POUR chaque sommet z de CC FAIRE la comparaison $D[z] > D[y] + v(yz)$ se fera au plus $d(y)$ fois donc au plus $\Theta(m)$ fois au total.
On peut améliorer le choix et la suppression du sommet y en représentant CC par un tas ordonné selon les valeurs en cours du tableau D.

Preuve

L'algorithme termine puisqu'on augmente le nombre de sommets choisis à chaque itération.
Avant la boucle TANT QUE, C ne contient que x et $D[x] = 0$ est la longueur d'un plus court chemin depuis x.

Preuve

Au bout de k itérations, supposons que pour chaque sommet z de C, $D[z]$ est la longueur d'un plus court chemin depuis x.
Soit y le sommet choisi dans $X - C$ à la $k+1$ itération. Soit $x = x_0, x_1, x_2, \cdots, x_{l-1}, x_l = y$ un chemin σ quelconque de x à y s'il existe. Soit i le plus grand entier tel que $x, x_1, x_2, \cdots, x_i$ soient des sommets de C. En particulier $x_{i+1} \notin C$.
Par hypothèse de récurrence, $D[x_i] \leq \sum_{j=0}^{j=i-1} v(x_j, x_{j+1})$.
Donc, puisque la fonction p est à valeurs positives, la longueur $l(\sigma)$ du chemin σ est supérieure à $D[x_i] + \sum_{j=i+1}^{j=l-1} v(x_j, x_{j+1})$.
Le sommet x_i a été choisi au cours des k itérations précédentes donc on a nécessairement $D[x_i] + v(x_i, x_{i+1}) \geq D[x_{i+1}]$.
Mais par choix de y, $D[y] \leq D[x_{i+1}]$, puisque x_{i+1} n'a pas été choisi.
Il reste donc $D[y] \leq D[x_i] + v(x_i, x_{i+1}) \leq l(\sigma)$ *i.e.* $D[y]$ est inférieur à la longueur d'un chemin quelconque depuis x.

Algorithme de Floyd

Cet algorithme utilise la matrice d'adjacence adaptée pour contenir les valeurs associées à chaque arête (arc).

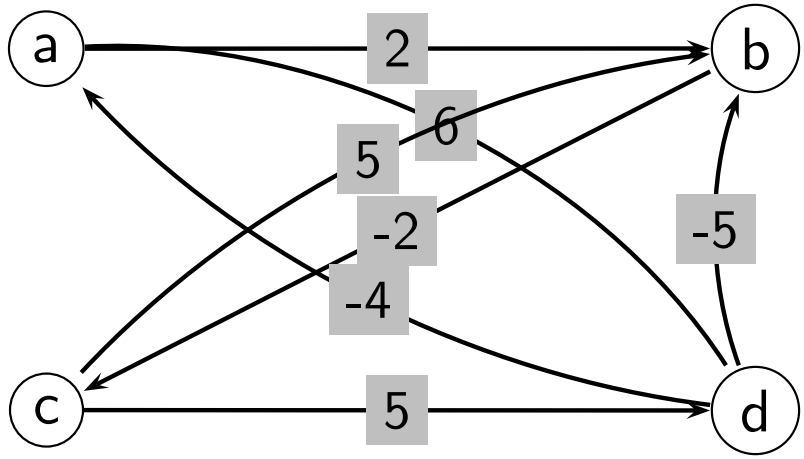
Algorithme

On utilise

- une matrice carrée M d'ordre n initialisée par les valeurs $v(i,j)$
- une matrice carrée P d'ordre n dans laquelle $P_{i,j}$ sera le père de j dans un plus court chemin depuis i.

PROCÉDURE Floyd(M, P : matrice)
  POUR i de 1 à n FAIRE
    POUR j de 1 à n FAIRE $M_{i,j} \leftarrow v(i,j)$; $P_{i,j} \leftarrow i$
    FINPOUR FINPOUR
  POUR k de 1 à n FAIRE
    POUR i de 1 à n FAIRE
      POUR j de 1 à n FAIRE
      SI $M_{i,k} + M_{k,j} < M_{i,j}$
      ALORS $M_{i,j} \leftarrow M_{i,k} + M_{k,j}$; $P_{i,j} \leftarrow P_{k,j}$ FINSI
  FINPOUR FINPOUR FINPOUR
FINPROCÉDURE

Exemple



Exemple

$$M = \begin{pmatrix} 0 & 2 & +\infty & 6 \\ +\infty & 0 & -2 & +\infty \\ +\infty & 5 & 0 & 5 \\ -4 & -1 & +\infty & 0 \end{pmatrix} \quad et \quad P = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{pmatrix}$$

Exemple

pour $k = 1$

$$M = \begin{pmatrix} 0 & 2 & +\infty & 6 \\ +\infty & 0 & -2 & +\infty \\ +\infty & 5 & 0 & 5 \\ -4 & -2 & +\infty & 0 \end{pmatrix} \quad et \quad P = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 1 & 4 & 4 \end{pmatrix}$$

Exemple

pour $k = 2$

$$M = \begin{pmatrix} 0 & 2 & 0 & 6 \\ +\infty & 0 & -2 & +\infty \\ +\infty & 5 & 0 & 5 \\ -4 & -2 & -4 & 0 \end{pmatrix} \text{ et } P = \begin{pmatrix} 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 1 & 2 & 4 \end{pmatrix}$$

Exemple

pour $k = 3$

$$M = \begin{pmatrix} 0 & 2 & 0 & 5 \\ +\infty & 0 & -2 & 3 \\ +\infty & 5 & 0 & 5 \\ -4 & -2 & -4 & 0 \end{pmatrix} \quad et \quad P = \begin{pmatrix} 1 & 1 & 2 & 3 \\ 2 & 2 & 2 & 3 \\ 3 & 3 & 3 & 3 \\ 4 & 1 & 2 & 4 \end{pmatrix}$$

Exemple

pour $k = 4$

$$M = \begin{pmatrix} 0 & 2 & 0 & 5 \\ -1 & 0 & -2 & 3 \\ 1 & 3 & 0 & 5 \\ -4 & -2 & -4 & 0 \end{pmatrix} \quad et \quad P = \begin{pmatrix} 1 & 1 & 2 & 3 \\ 4 & 2 & 2 & 3 \\ 4 & 1 & 3 & 3 \\ 4 & 1 & 2 & 4 \end{pmatrix}$$

Amélioration

On peut modifier l'algorithme de Floyd afin qu'il détecte la présence d'un circuit de longueur négative de la façon suivante :

PROCÉDURE Floyd 2 (M, P : matrice)
  POUR i de 1 à n FAIRE
    POUR j de 1 à n FAIRE $M_{i,j} \leftarrow v(i,j)$; $P_{i,j} \leftarrow i$ FINPOUR
  FINPOUR
  POUR k de 1 à n FAIRE
    POUR i de 1 à n FAIRE
      SI $M_{i,k} \neq +\infty$ ALORS
        SI $M_{i,k} + M_{k,i} < 0$ ALORS circuit absorbant ; EXIT
        SINON
          POUR j de 1 à n FAIRE
      $M_{i,j} \leftarrow min\{M_{i,j}; M_{i,k} + M_{k,j}\}$
      SI $M_{i,j} = M_{i,k} + M_{k,j}$ ALORS $P_{i,j} \leftarrow P_{k,j}$ FINSI
          FINPOUR
  FINSI FINSI FINPOUR FINPOUR FINPROCÉDURE

Complexité

Elle est clairement en $\Theta(n^3)$.

Preuve

La matrice M initiale contient les plus courts chemins de i à j ne passant par aucun autre sommet.
On peut montrer par récurrence qu'à l'étape k, $M_{i,j}$ est égal à un plus court chemin de i à j passant uniquement par les sommets $1, \cdots, k$. Cela constitue l'invariant de la boucle POUR.

Algorithme de Bellman

Cet algorithme recherche un plus court chemin depuis un sommet fixé x dans un graphe *quelconque* ou bien détecte la présence d'un circuit absorbant.

Algorithme

On utilise un tableau de distance D indexé par les sommets du graphe et la liste des voisins (pour un graphe non orienté) ou bien la liste des prédécesseurs (dans la cas d'un graphe orienté).

PROCÉDURE Bellman (G : graphe ; x : sommet)

{initialisation} $D^0[x] \leftarrow 0$,

POUR tout sommet $y \neq x$ FAIRE $D^0[y] \leftarrow +\infty$ FINPOUR

$k \leftarrow 0$

TANT QUE $k < n$ FAIRE

$D^k[x] \leftarrow 0$

POUR tout sommet y FAIRE

$D^{k+1}[y] \leftarrow min(D^k[y], min\{D^k[z] + w(zy)) \ ; zy \in E\}$

FINPOUR

SI $D^{k+1} = D^k$ ALORS FIN FINSI

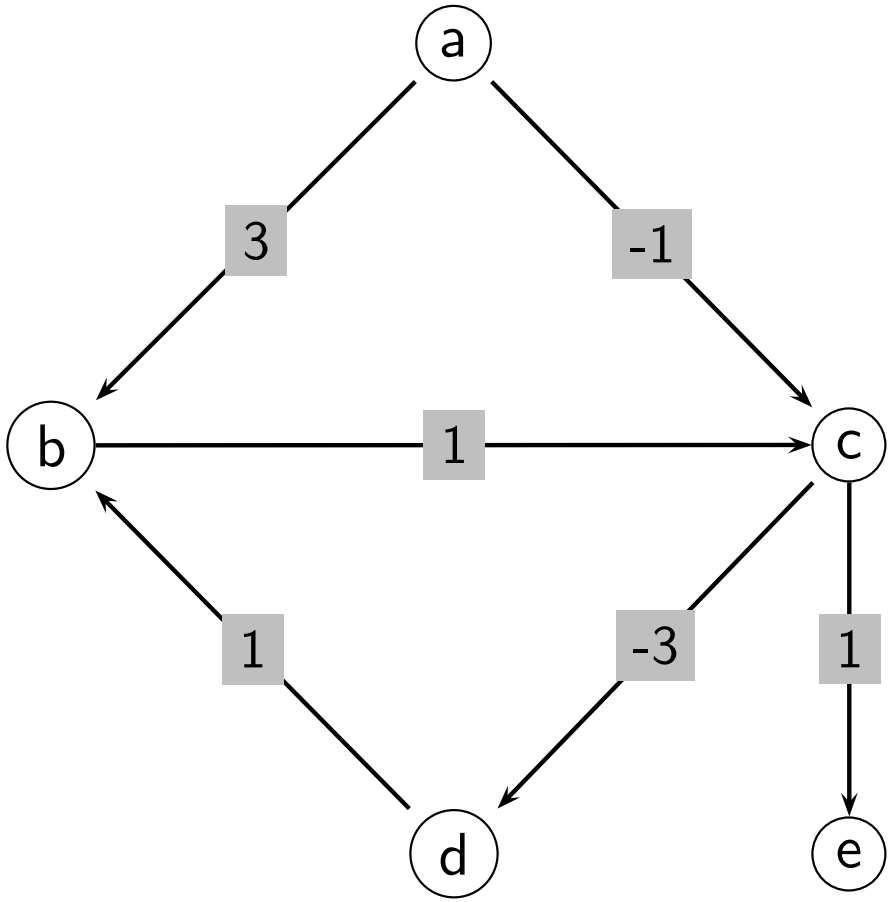
$k \leftarrow k + 1$

FINTANT QUE

SI $k = n$ ALORS il existe un circuit de longueur négative.

FINPROCÉDURE

Exemple



Exemple

si on choisit $x = a$, le tableau D évolue ainsi

sommet	a	b	c	d	e
D^0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$
D^1	0	3^a	-1^a	$+\infty$	$+\infty$
D^2	0	3^a	-1^a	-4^c	0^c
D^3	0	-3^d	-1^a	-4^c	0^c
D^4	0	-3^d	-2^b	-4^c	0^c
D^5	0	-3^d	-2^b	-5^c	-1^c

L'algorithme s'arrête avec $k = 5$ et le circuit absorbant a été détecté.

Complexité

La boucle TANT QUE demande chaque fois m additions et m comparaisons donc la complexité est en $O(m \times n)$.

Preuve

$D^0[y]$ représente la valeur d'un chemin de x à y de longueur minimum et contenant 0 arête (arc).
On peut montrer par récurrence sur k que, à la k-ième itération de la boucle TANT QUE, $D^k[y]$ représente la valeur d'un chemin de x à y de longueur minimum et ne contenant pas plus que k arêtes (arcs).
S'il n'existe pas de circuit absorbant, il existe un plus court chemin (élémentaire) de x à y de moins de $n-1$ arêtes (arcs) et le tableau D se stabilise en moins de $n-1$ itérations à la valeur d'un plus court chemin de x à y, pour chaque sommet y.
S'il n'y a pas de stabilisation alors il existe un circuit absorbant.

Chapitre V – Ordonnancement

Chapitre V – Ordonnancement

- Un premier exemple
- méthode MPM
- Date au plus tôt
- Date au plus tard
- Marge totale et marge libre

Chapitre V – Ordonnancement

- Un premier exemple
- méthode MPM
- Date au plus tôt
- Date au plus tard
- Marge totale et marge libre

Chapitre V – Ordonnancement

- Un premier exemple
- méthode MPM
- Date au plus tôt
- Date au plus tard
- Marge totale et marge libre

Chapitre V – Ordonnancement

- Un premier exemple
- méthode MPM
- Date au plus tôt
- Date au plus tard
- Marge totale et marge libre

Problème de coordination de travaux

Dans ce chapitre, on étudiera le problème de coordination de travaux : un ensemble de tâches doit être accompli en respectant certaines contraintes. Ces contraintes peuvent être de plusieurs natures, par exemple :

- contraintes de précédence : une tâche ne peut être exécutée que si une ou plusieurs autres sont achevées
- contraintes absolues : une tâche doit être achevée avant une date précise ...
- contraintes d'incompatibilité : deux tâches ne peuvent s'exécuter simultanément, deux tâches utilisent une même ressource ...

On ne verra que les contraintes de précédence simples pour lesquelles les tâches ont une durée fixée.

Exemple

Considérons la réfection d'une salle de bain. Les tâches sont les suivantes

- dépot des sanitaires et du carrelages existants (DS, DC)
- pose du nouveau carrelage (PC)
- peinture des murs non carrelés (P)
- mise aux normes électriques (E)
- pose des sanitaires (PS)
- pose du mobilier et de l'éclairage (PM, PE)

On va établir un tableau résumant les contraintes de précédences.

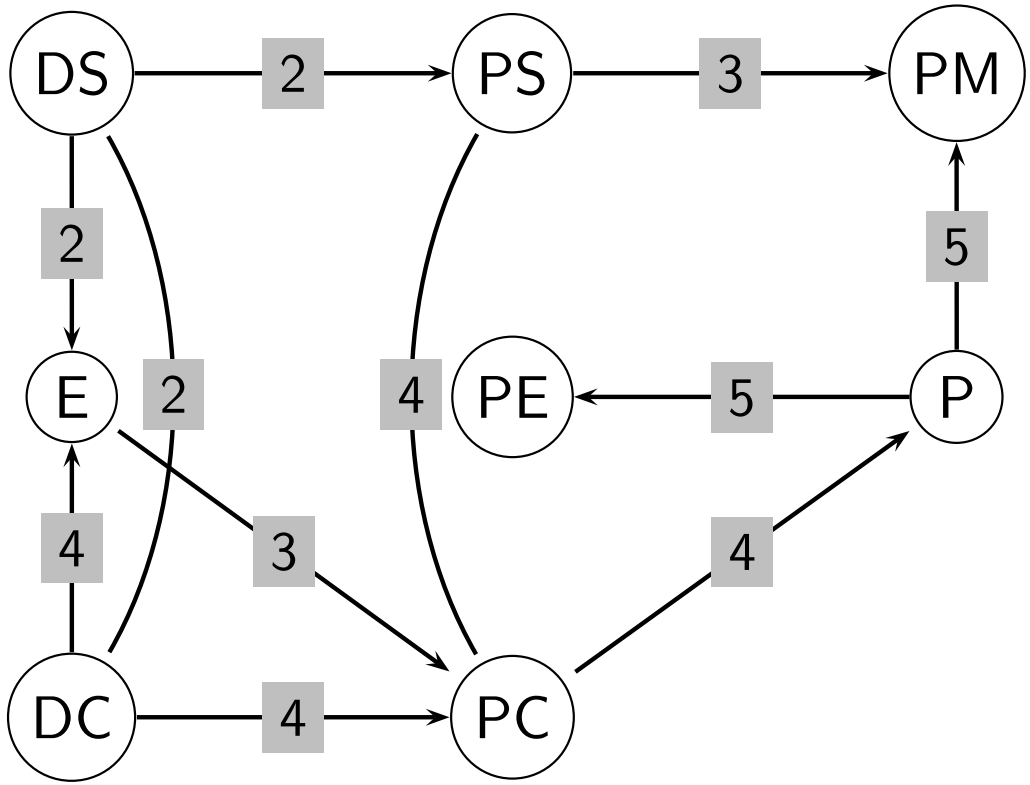
Exemple

tâche	durée	tâche précedente
DS	2	-
DC	4	DS
PC	4	DC, E
P	5	PC
E	3	DC,DS
PS	3	DS,PC
PE	1	P
PM	2	P, PS

Exemple

On peut représenter cela sous forme d'un graphe orienté valué de la façon suivante :

- les sommets sont les tâches
- un arc (x, y) existe si la tâche x est avant la tâche y et cet arc est valué par la durée de x.



Exemple

C'est illisible !
Le but étant d'optimiser les travaux il faut classer ces travaux mais comment ?
par rang.

Exemple

C'est illisible !
Le but étant d'optimiser les travaux il faut classer ces travaux mais comment ?
par rang.

Définition du rang dans un graphe orienté sans circuit

Le calcul du rang correspond à la définition suivante :

Définition

Dans un graphe orienté sans circuit, pour un sommet x, $rang(x)$ est la longueur maximale d'un chemin de but x.

Remarque : la valeur du rang est unique.

Calcul du rang

Pour calculer cette fonction *rang*, il suffit donc de résoudre le problème du plus long chemin par opposition au problème du plus court chemin.
Une solution consiste à utiliser les degrés entrants des sommets.

Algorithme

On utilise :

- un entier k qui permet le calcul du rang
- un tableau qui permet d'y écrire la valeur du rang et d'effacer virtuellement les arcs depuis les sommets de rang k.

On initialise

- k à 0
- les sommets de rang 0 sont tous les sommets source

A chaque itération, on construit l'ensemble des sommets de rang $k+1$ en supprimant virtuellement les arcs depuis les sommets de rang k : ce sont les sommets qui se retrouvent sans degré entrant.

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	
PC	4	DC, E	
P	5	PC	
E	3	DC,DS	
PS	3	DS,PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	
PC	4	DC, E	
P	5	PC	
E	3	DC,DS	
PS	3	DS,PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	
P	5	PC	
E	3	DC,DS	
PS	3	DS,PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	
P	5	PC	
E	3	DC,DS	
PS	3	DS,PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	
P	5	PC	
E	3	DC,DS	2
PS	3	DS,PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	
P	5	PC	
E	3	DC,DS	2
PS	3	DS,PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	3
P	5	PC	
E	3	DC,DS	2
PS	3	DS,PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	3
P	5	PC	
E	3	DC,DS	2
PS	3	DS, PC	
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	3
P	5	PC	4
E	3	DC,DS	2
PS	3	DS, PC	4
PE	1	P	
PM	2	P, PS	

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	3
P	5	PC	4
E	3	DC,DS	2
PS	3	DS, PC	4
PE	1	P	
PM	2	P, PS	

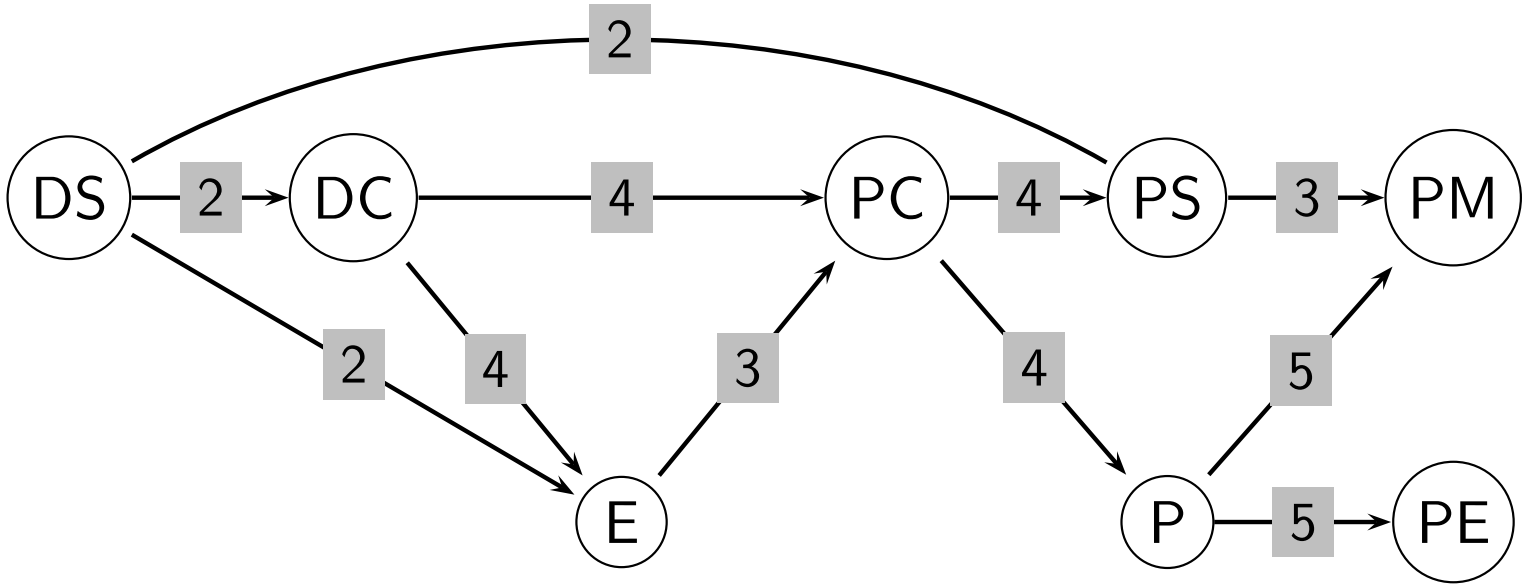
Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	3
P	5	PC	4
E	3	DC,DS	2
PS	3	DS, PC	4
PE	1	P	5
PM	2	P, PS	5

Exemple

tâche	durée	tâche précedente	rang
DS	2	-	0
DC	4	DS	1
PC	4	DC, E	3
P	5	PC	4
E	3	DC,DS	2
PS	3	DS,PC	4
PE	1	P	5
PM	2	P, PS	5

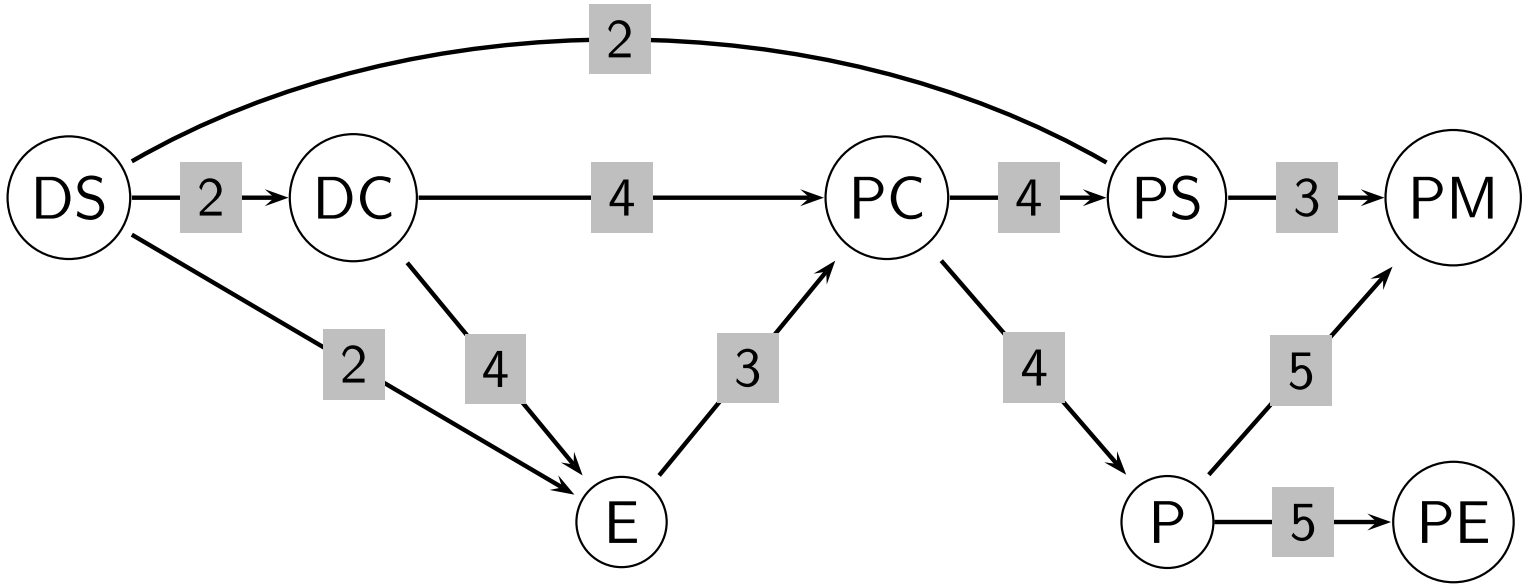
Exemple



Certains arcs sont inutiles pour ordonner les travaux : (DS,E), (DS,PS), (DC,PC)

Il manque les durées des « dernières » tâches PM, PE.

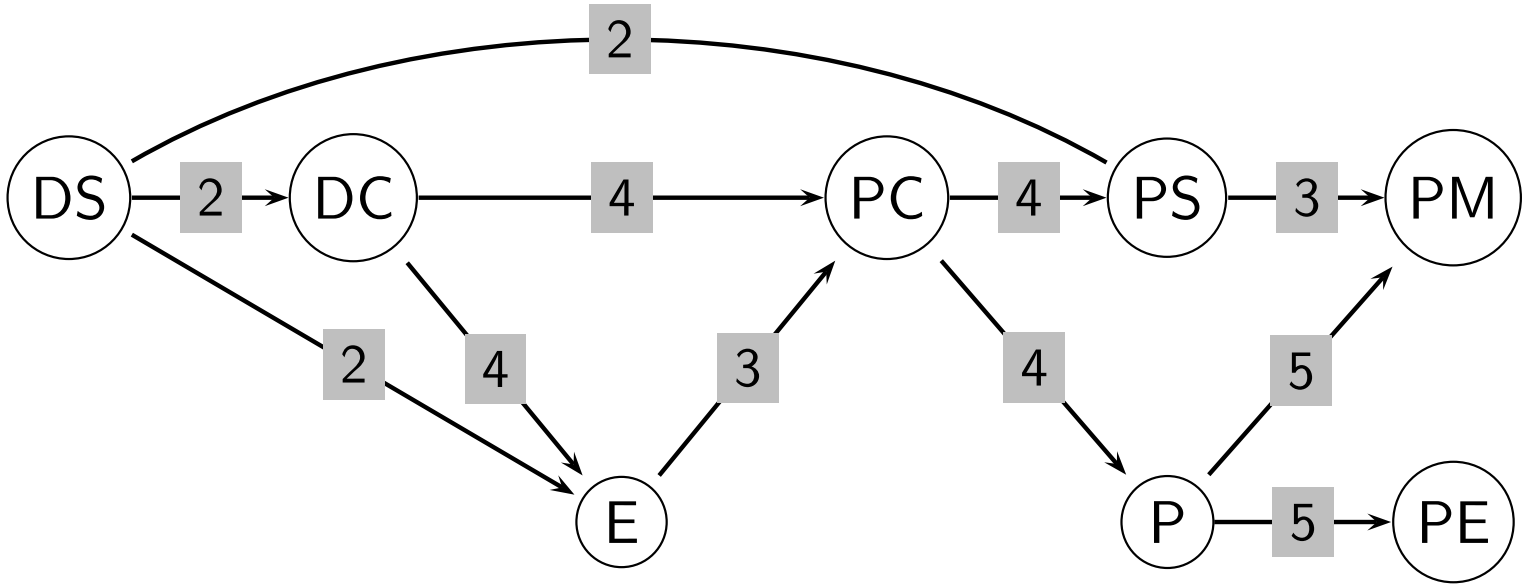
Exemple



Certains arcs sont inutiles pour ordonner les travaux : (DS,E), (DS,PS), (DC,PC)

Il manque les durées des « dernières » tâches PM, PE.

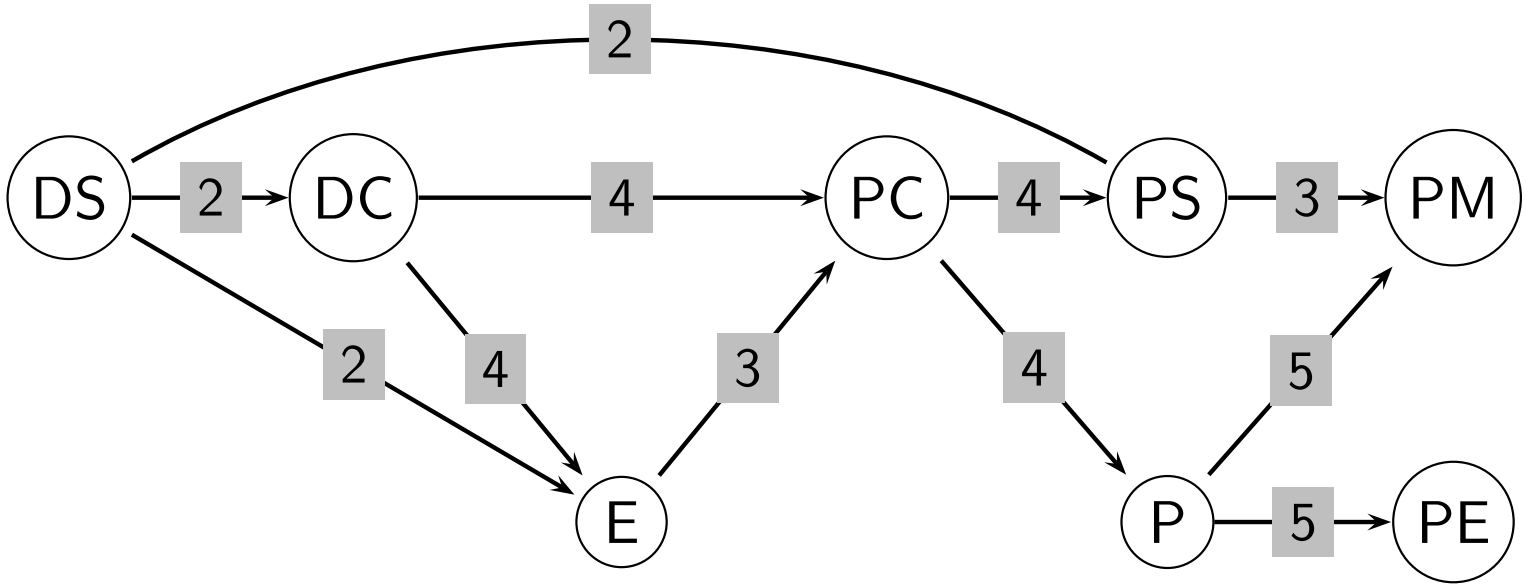
Exemple



Certains arcs sont inutiles pour ordonner les travaux : (DS,E), (DS,PS), (DC,PC)

Il manque les durées des « dernières » tâches PM, PE.

Exemple



Certains arcs sont inutiles pour ordonner les travaux : (DS,E), (DS,PS), (DC,PC)

Il manque les durées des « dernières » tâches PM, PE.

méthode MPM

Dans la méthode potentiels–tâches (MPM pour *Method of Project Management*), la modélisation se fait par un graphe orienté dont

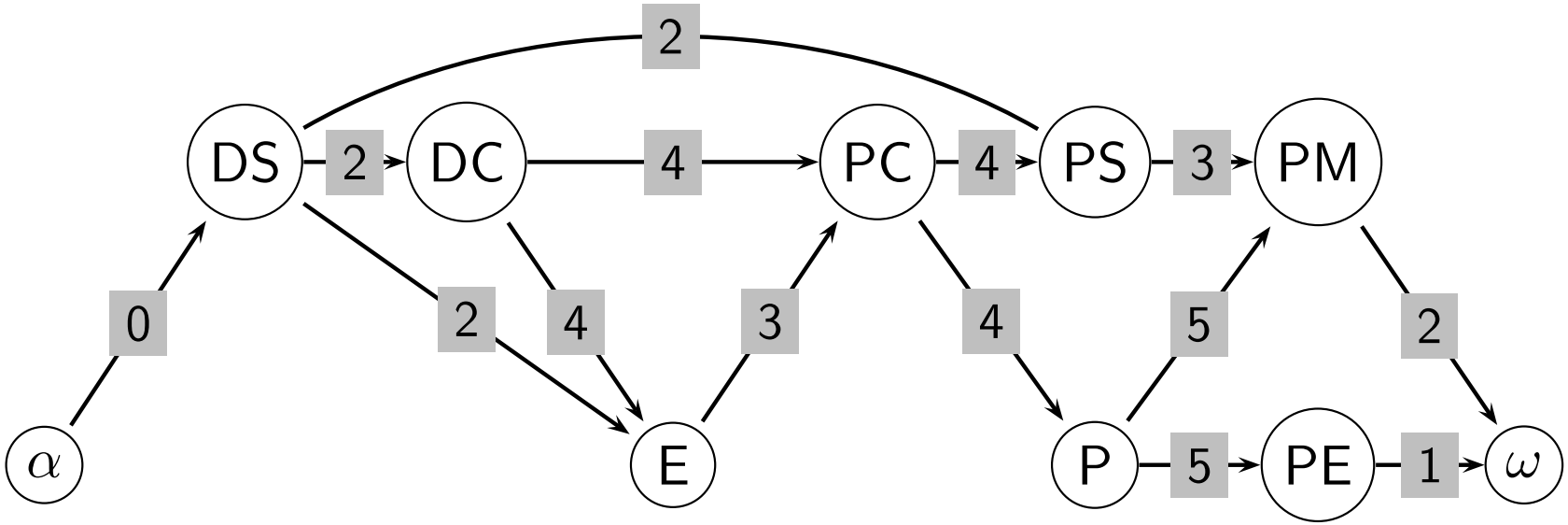
- les sommets sont les tâches,
- un arc indique que la tâche origine doit précéder la tâche but,
- chaque arc est valué par la durée de son origine.

On complète alors le graphe avec deux sommets fictifs α et ω :

- α représente la tâche fictive de début des travaux et précède chaque sommet sans prédécesseur par un arc de valeur nulle,
- ω représente la tâche fictive de fin des travaux et succède à chaque sommet sans successeur par un arc valué par la durée de la tâche origine.

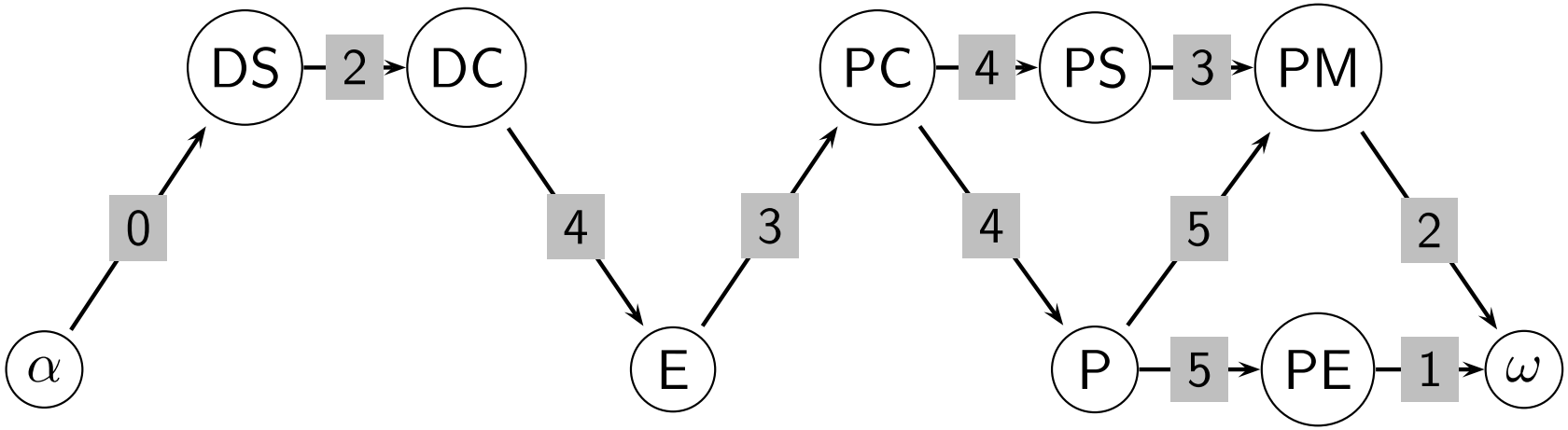
méthode MPM

Le problème est d'établir un calendrier des travaux respectant les contraintes et minimisant la durée totale.



méthode MPM

Le problème est d'établir un calendrier des travaux respectant les contraintes et minimisant la durée totale *en supprimant les arcs inutiles*.



Date au plus tôt

Pour qu'une tâche puisse commencer, il est nécessaire que toutes les tâches qui la précèdent soient terminées : donc, au plus tôt elle ne peut démarrer qu'à la fin de celle qui termine le plus tard parmi toutes la tâches qui la précèdent.
On va donc calculer la **longueur d'un plus long chemin depuis le sommet** α jusqu'à la tâche pour déterminer sa *date au plus tôt*.

Date au plus tôt

Pour qu'une tâche puisse commencer, il est nécessaire que toutes les tâches qui la précèdent soient terminées : donc, au plus tôt elle ne peut démarrer qu'à la fin de celle qui termine le plus tard parmi toutes la tâches qui la précèdent.

On va donc calculer la **longueur d'un plus long chemin depuis le sommet** α jusqu'à la tâche pour déterminer sa *date au plus tôt*.

Date au plus tôt

Pour qu'une tâche puisse commencer, il est nécessaire que toutes les tâches qui la précèdent soient terminées : donc, au plus tôt elle ne peut démarrer qu'à la fin de celle qui termine le plus tard parmi toutes la tâches qui la précèdent.
On va donc calculer la **longueur d'un plus long chemin depuis le sommet** α jusqu'à la tâche pour déterminer sa *date au plus tôt*.

Date au plus tôt

Définition

Pour une tâche x, la date au plus tôt de x est

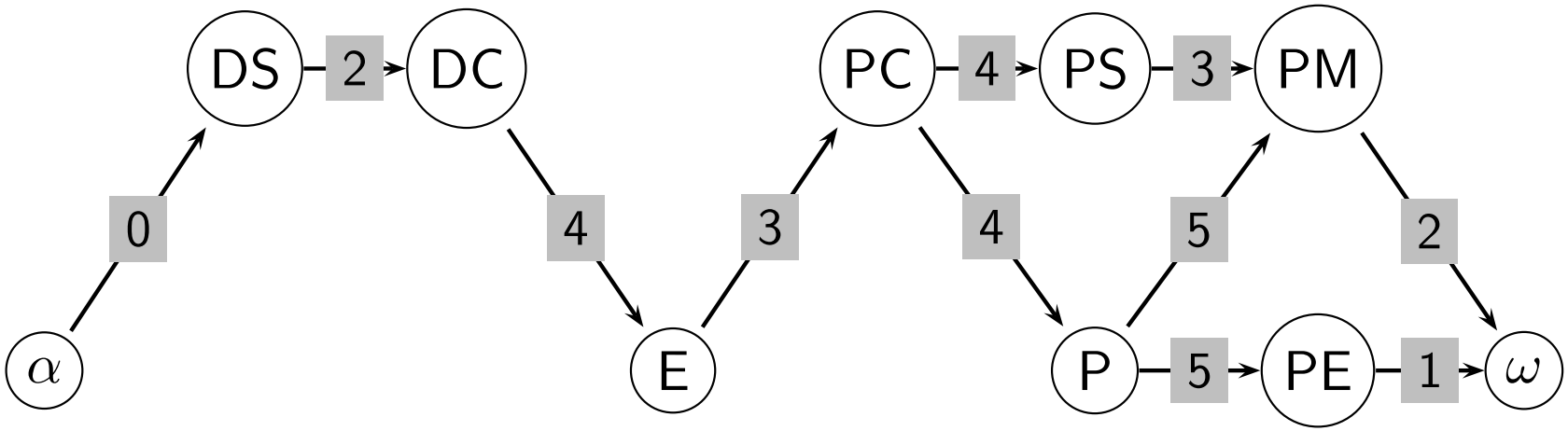
$$d_{tot}[x] = max_{z \ precede \ x} \ (d_{tot}[z] + duree(z))$$

La durée minimale des travaux est donc la date au plus tôt de la tâche ω soit la longueur d'un plus long chemin depuis α jusqu'à ω.

Le calcul pour toutes les tâches se fait par la fonction suivante

FONCTION date au plus tôt(G : graphe) : entier naturel
  $d_{tot}[\alpha] \leftarrow 0$;
  POUR chaque sommet x par ordre de rang croissant FAIRE
    $d_{tot}[x] \leftarrow max_{z \ precede \ x} \ (d_{tot}[z] + duree(z))$;
  FINPOUR
FINFONCTION

Exemple



tâche	DS	DC	PC	P	E	PS	PE	PM	ω
d_{tot}	0	2	9	13	6	13	18	18	20

Date au plus tard

Une fois déterminée la durée minimale du projet, on peut remarquer ques le retard de certaines tâches entraîne un retard global des travaux : ce sont des tâches *critiques*. Il existe au moins un chemin dit critique dont tous les sommets sont des tâches critiques : c'est un plus long chemin de α à ω.
Au contraire, certaines autres tâches peuvent être différées, dans une certaine limite appelée *marge totale*, sans pour autant retarder la fin des travaux.
Pour calculer cette marge, on va déterminer tout d'abord la *date au plus tard* d_{tard} pour chaque tâche en posant $d_{tard}[\omega] \leftarrow d_{tot}[\omega]$.
On cherche donc le ≪ dernier moment ≫ auquel exécuter une tâche sans provoquer de retard global.

Date au plus tard

Pour une tâche x, on détermine la première tâche dans le temps qui doit lui succéder et on tient compte alors de la durée de x :

$$d_{tard}[x] = min_{(xy)\in E}\ (d_{tard}[y]) - d(x)$$

Cela correspond à $d_{tard}[x] = d_{tot}[\omega] - l(x, \omega)$ où $l(x, \omega)$ est la longueur d'un plus long chemin de x à ω.
Le graphe est donc parcouru depuis le sommet ω pour lequel on fixe $d_{tard}[\omega] = d_{tot}[\omega]$. Pour chaque sommet x, on tient compte des sommets qui lui succèdent et on cherche la contrainte la plus forte : x doit être fini avant toutes les tâches qui lui succédent ne démarrent.

Date au plus tard

Le calcul se fait par la fonction suivante

FONCTION date au plus tard(G : graphe) : entier naturel

  $d_{tard}[\omega] \leftarrow d_{tot}[\omega]$;

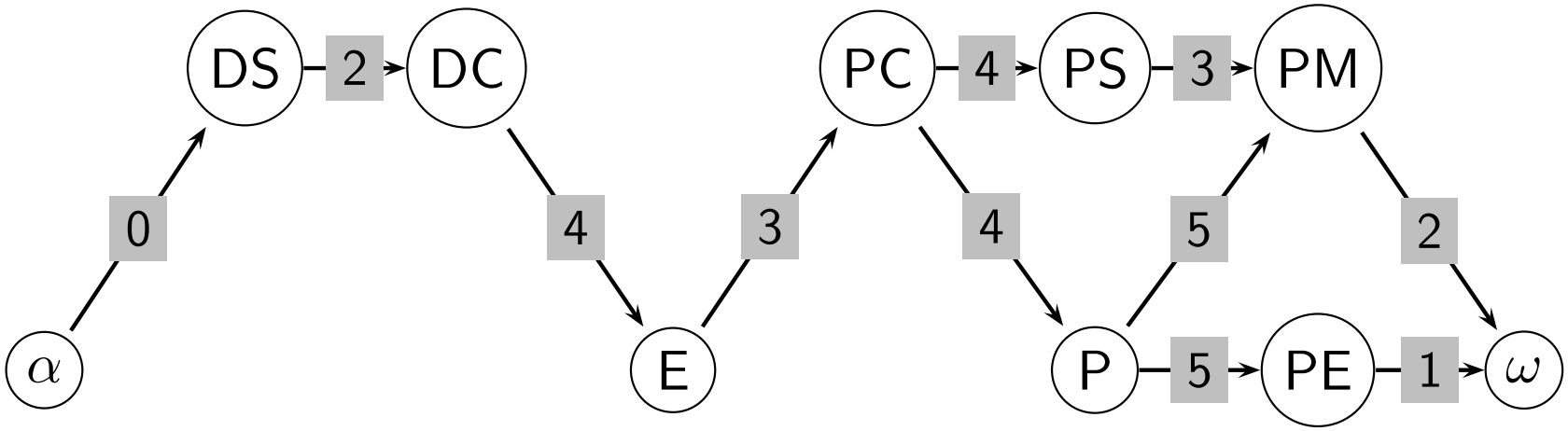
  POUR chaque sommet x par ordre de rang décroissant FAIRE

    $d_{tard}[x] \leftarrow min_{x \ precede \ y} (d_{tard}[y]) - d(x)$;

  FINPOUR

FINFONCTION

Exemple

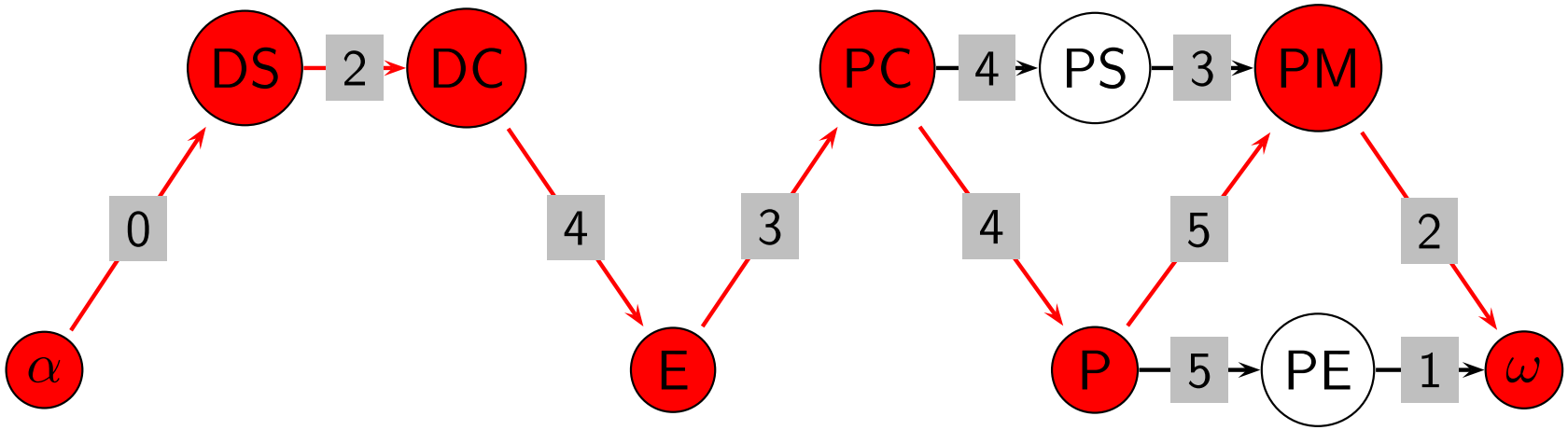


tâche	DS	DC	PC	P	E	PS	PE	PM	ω
d_{tot}	0	2	9	13	6	13	18	18	20
d_{tard}	0	2	9	13	6	15	19	18	20

Marge totale

Les sommets pour lesquels les dates au plus tôt et au plus tard sont égales correspondent aux tâches dites *critiques*.
Pour les autres sommets, leur *marge totale* est obtenue par la différence entre leurs date au plus tard et date au plus tôt.

Exemple



tâche	DS	DC	PC	P	E	PS	PE	PM	ω
d_{tot}	0	2	9	13	6	13	18	18	20
d_{tard}	0	2	9	13	6	15	19	18	20
marge	0	0	0	0	0	2	1	0	0

Marge libre

On a vu la notion de marge totale. Cette marge, si elle est utilisée peut obliger les tâches suivantes à elles-même utiliser leur marge totale.
Une notion plus fine de *marge libre* permet de calculer pour chaque tâche une marge qui n'aura aucune influence sur le comportement des tâches suivantes.

Marge libre

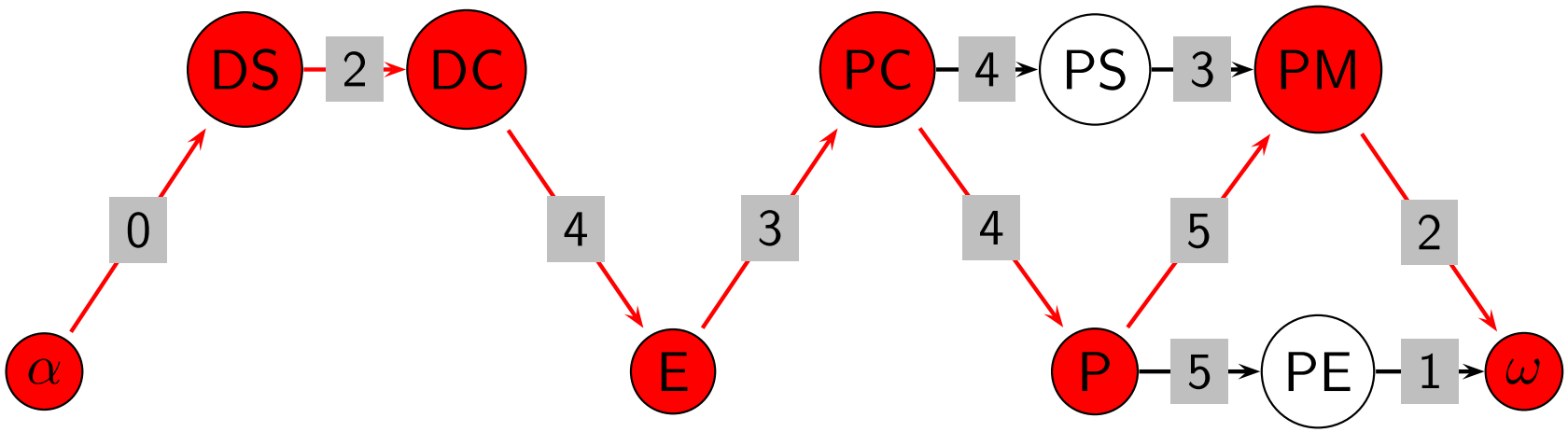
Par exemple, si une tâche A de durée 10 a pour date au plus tôt 12, date au plus tard 17 et si elle précède deux tâches B et C dont les dates au plus tôt sont respectivement 24 et 27 (à cause d'autres prédécesseurs) alors A pourra commencer au plus tard à la date 14 (et se terminer à la date 24) sans que B et C voient leur date au plus tôt perturbée. Sa marge libre est alors de 2.

Marge libre

Le calcul de la marge libre se fait par la fonction suivante :

FONCTION marge libre(G : graphe) : entier naturel
  POUR chaque sommet x FAIRE
    $marge\_libre[x] \leftarrow min_{x \ precede \ y} (d_{tot}[y]) - d_{tot}[x] - d(x)$;
  FINPOUR
FINFONCTION

Exemple



tâche	DS	DC	PC	P	E	PS	PE	PM	ω
d_{tot}	0	2	9	13	6	13	18	18	20
d_{tard}	0	2	9	13	6	15	19	18	20
marge	0	0	0	0	0	2	1	0	0
marge libre	0	0	0	0	0	2	1	0	0