# Chapitre I: Théorie des graphes (modélisation et algorithmes)

## Introduction

La théorie des graphes est un outil privilégié de modélisation et de résolution de problèmes dans un grand nombre de domaines allant de la science fondamentale aux applications technologiques concrètes. Par exemple, les graphes déterministes et aléatoires sont utilisés en chimie (modélisation de structure), en sciences sociales (pour représenter des relations entre groupes d'individus), en mathématiques combinatoires, en informatique (structures de données et algorithmique). Concernant les applications, elles sont très nombreuses : réseaux électriques et transport d'énergie, routage du trafic dans les réseaux de télécommunications et les réseaux d'ordinateurs, routage de véhicules et organisation de tournées, problèmes de localisation (localisation d'entrepôts dans les réseaux de distribution de marchandises, d'antennes ...), problèmes d'ordonnancements de tâches et d'affectation de ressources (problèmes de rotation d'équipages dans les compagnies aériennes)...

## I. Quelques problèmes formalisables par des graphes

### 1. Choix d'un itinéraire

Comment faire pour aller le plus rapidement possible de Bordeaux à Grenoble sachant que :

| | |
|---|---|
| Bordeaux → Nantes | 4h |
| Bordeaux → Marseille | 9h |
| Bordeaux → Lyon | 12h |
| Nantes → Paris-Montparnasse | 2h |
| Nantes → Lyon | 7h |
| Paris-Montparnasse → Paris-Lyon | 1h |
| Paris-Lyon → Grenoble | 4h30 |
| Marseille → Lyon | 2h30 |
| Marseille → Grenoble | 4h30 |
| Lyon → Grenoble | 1h15 |

Ce problème est facile à représenter par un graphe dont les arcs sont valuées par les durées des trajets. Il s'agit alors de déterminer un plus court chemin entre Bordeaux et Grenoble.

### 2. Organisation d'une session d'examen

8 groupes d'étudiants (numérotés de G1 à G8) doivent passer des examens dans différentes disciplines, chaque examen occupant une demi-journée :

| | |
|---|---|
| chimie | G1 et G2 |
| Electronique | G3 et G4 |
| Informatique | G3, G5, G6 et G7 |
| Mathématiques | G1, G5, G6 et G8 |
| Physique | G2, G6, G7 et G8 |

On cherche à organiser la session d'examen la plus courte possible.

On peut représenter chacune des disciplines par un sommet et relier par des arêtes les sommets correspondant aux examens ne pouvant se dérouler simultanément. Le problème est alors de colorier tous les sommets du graphe en utilisant le moins de couleurs possible sachant que deux sommets reliés par une arête doivent être de couleurs distinctes.

## 3. Problème d'ordonnancement de tâches

La mise en exploitation d'un nouveau gisement minier demande l'exécution d'un certain nombre de tâches :

| Codes | Tâches | Durées | Tâches antérieures |
|---|---|---|---|
| 1 | Obtention d'un permis d'exploitation | 120 | - |
| 2 | Etablissement d'une piste de 6 km | 180 | 1 |
| 3 | Transport et installation de 2 sondeuses | 3 | 2 |
| 4 | Création de bâtiments provisoires pour le bureau des plans et le logement des ouvriers | 30 | 2 |
| 5 | Goudronnage de la piste | 60 | 2 |
| 6 | Adduction d'eau | 90 | 4 |
| 7 | Campagne de sondage | 240 | 3, 4 |
| 8 | Forage et équipement de 3 puits | 180 | 5, 6, 7 |
| 9 | Transport et installation du matériel d'exploitation | 30 | 10, 8 |
| 10 | Construction de bureaux et de logements définitifs ouvriers et ingénieurs | 240 | 6, 5, 7 |
| 11 | Traçage et aménagement du fond | 360 | 10, 8 |

Résoudre un problème d'ordonnancement consiste à déterminer le temps minimum de réalisation de l'ensemble, ainsi que l'intervalle de temps dans lequel on doit débuter chacune des tâches pour réaliser l'ensemble dans le temps minimum. Pour cela, on doit rechercher un plus long chemin dans un graphe, appelé graphe potentiels-tâches, dont les sommets représentent les tâches à réaliser et les arcs les contraintes d'antériorité entre les tâches : un arc (i, j) représente la nécessité de réaliser la tâche i avant la tâche j et l'arc (i, j) est valué par la durée d'exécution de i.

### 4. Routage dans les réseaux de télécommunications

Le problème de savoir s'il est possible d'acheminer un ensemble d'informations entre deux centres reliés par un réseau de télécommunications revient à chercher un flot de valeur maximale dans un graphe représentant ce réseau (les sommets représentent les centres et les arêtes les liaisons entre ces centres).

## II. Définition et concepts de base

### 1. typologie

De manière générale, un graphe, c'est des sommets, et des arêtes (ou des arcs) qui relient les sommets. Il existe différents types de graphes ; nous pouvons citer entre autres :

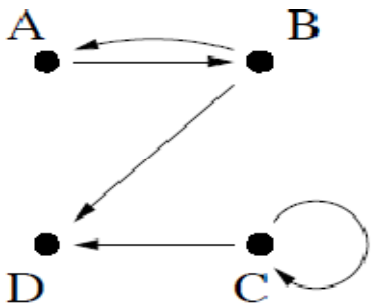
- Les graphes orientés : chaque arête (arc) a un sens, habituellement représenté par une petite flèche.
- Graphes non orienté : les arêtes n'ont pas de sens particuliers.
- multigraphe : il peut y avoir plusieurs arcs ayant les mêmes extrémités.

### 2. Graphe orienté

Un graphe orienté G = (S, A) est la donnée:

- d'un ensemble S dont les éléments sont les sommets du graphe,
- d'un ensemble A dont les éléments, les arcs du graphe, sont des couples d'éléments de S.

Un arc (x, y) ∈ A est aussi indifféremment noté x → y.
Les boucles sont autorisées, mais pas les arcs parallèles.
S = {A, B, C,D}  A = {(A, B), (B, A), (B,D), (C, C), (C,D)}




**Terminologie**

Si a = (s1, s2) ∈ A est un arc de G, les sommets s1 et s2 sont les extrémités de a :
s1 est le début (ou l'origine) de a et s2 est la fin (ou l'extrémité finale) de a.
On dit aussi que s2 est un successeur de s1 et que s1 est un prédécesseur de s2.
Si les deux extrémités d'un arc sont égales, l'arc est une boucle.

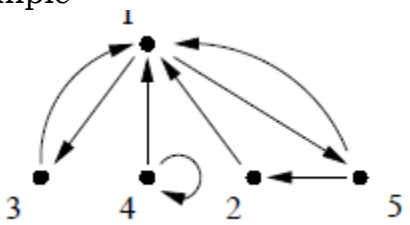
On note G(s) l'ensemble des successeurs du sommet s :
G(s) = {t ∈ S | (s, t) ∈ A}

On note $G_{-1}(s)$ l'ensemble des prédécesseurs du sommet s :
$G_{-1}(s) = \{r \in S \mid (r, s) \in A\}$

**Degré d'un sommet**

On appelle degré entrant d'un sommet x et on note $d_-(x)$ le nombre d'arcs dont l'extrémité est x, c'est à dire le nombre de prédécesseurs de x : $d_-(x) = |G_{-1}(x)|$
On appelle degré sortant d'un sommet x et on note $d_+(x)$ le nombre d'arcs dont l'origine est x, c'est à dire le nombre de successeurs de x :
$d_+(x) = |G(x)|$
On appelle degré total d'un sommet x et on note d(x) le nombre d'arcs dont x est l'origine ou l'extrémité (en comptant deux fois les boucles) :
$d(x) = d_-(x) + d_+(x)$
Si G est un graphe non orienté, on définit de la même façon le degré d(x) d'un sommet x comme le nombre d'arêtes ayant x pour extrémité (en comptant deux fois les boucles).

Exemple




$d_-(1) = 4$ $d_+(1) = 2$ $d(1) = 6$
$d_-(2) = 1$ $d_+(2) = 1$ $d(2) = 2$
$d_-(3) = 1$ $d_+(3) = 1$ $d(3) = 2$
$d_-(4) = 1$ $d_+(4) = 2$ $d(4) = 3$
$d_-(5) = 1$ $d_+(5) = 2$ $d(5) = 3$

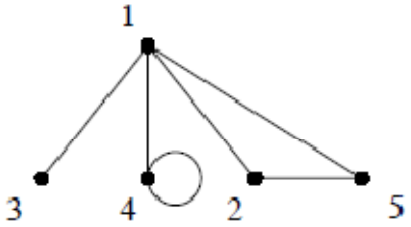
## 3. Graphe non orienté

Dans un graphe non orienté, il n'y a pas de distinction entre les deux extrémités d'une arête.
Un graphe non oriente G = (S, A) est la donnée :

- d'un ensemble S dont les éléments sont les sommets du graphe,
- d'un ensemble A dont les éléments, les arêtes du graphe, sont des parties à un ou deux éléments de S.

Le ou les sommets d'une arête sont appelés extrémités de l'arête.
Les arêtes n'ayant qu'une seule extrémité sont des boucles.
On peut de la même façon définir un multigraphe non oriente.

Exemple :




$$S = \{1, 2, 3, 4, 5\}$$

$$A = \{\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,5\}, \{4\}\}$$

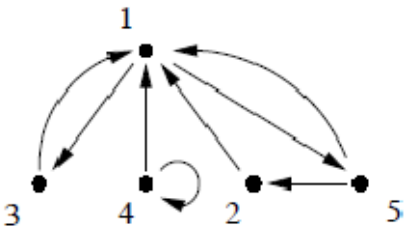
**Graphe orienté et graphe non orienté**

A chaque graphe orienté on peut associer un graphe non orienté, appelé graphe non orienté associé ou sous-jacent.

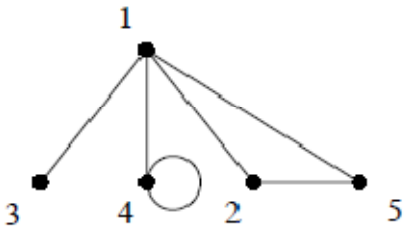
Soit G =(S, A) un graphe orienté. Son graphe non orienté associe est le graphe (non oriente) G' = (S, A') ayant le même ensemble de sommets S et dont l'ensemble d'arêtes A' vérifie

{x, y} ∈ A' ⇔ (x, y) ∈ A ou (y, x) ∈ A

Exemple :




donne




Les trois graphes suivants :

ont le même graphe non orienté associé :  .

Les notions correspondantes existent pour les graphes non orientés. On utilise alors de préférence le vocabulaire suivant :

| orienté | | non orienté |
|---|---|---|
| arc | ↔ | arête |
| chemin | ↔ | chaîne |
| circuit | ↔ | cycle |

**Théorème des poignées de mains**

Soit G = (S, A) un graphe orienté. On alors les égalités suivantes :

$$\sum_{x \in S} d_-(x) = \sum_{x \in S} d_+(x) = |A|$$

Si G = (S, A) est non orienté, on a encore l'égalité suivante :

$$\sum_{x \in S} d(x) = 2|A|$$

**Corollaire**

Dans un graphe G, le nombre de sommets dont le degré est impair est toujours pair.

## 4. Multigraphes orientés

Dans un graphe orienté, on a au plus un arc entre deux sommets donnés. Pour pallier à cette limitation, on utilise les multigraphes, qui autorisent les arcs parallèles.

Un multigraphe oriente G = (S, A, α, ω) est la donnée :

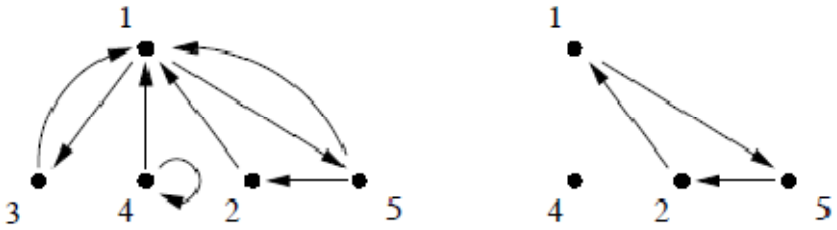
- d'un ensemble S dont les éléments sont les sommets du graphe,
- d'un ensemble A dont les éléments sont les arcs du graphe,
- de deux fonctions α : A → S et ω : A → S qui à chaque arc a ∈ A associent son origine α(a) et son extrémité ω(a).

## 5. Définitions complémentaires

### a) Sous-graphes

Soit G = (S, A) un graphe (orienté ou non). Un sous-graphe de G est un graphe G' = (S', A') tel que S' ⊂ S et A' ⊂ A.

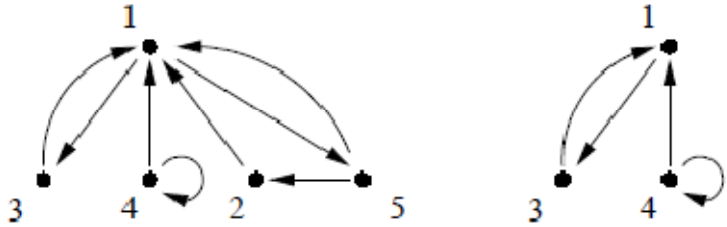
Exemple :




### b) Sous-graphes induits

Un sous-graphe G' = (S', A') d'un graphe G = (S, A) est un sous-graphe induit si A' est formé de tous les arcs (ou arêtes) de G ayant leurs extrémités dans S' : ∀x, y ∈ S', (x, y) ∈ A' ⇔ (x, y) ∈ A.

Exemple :

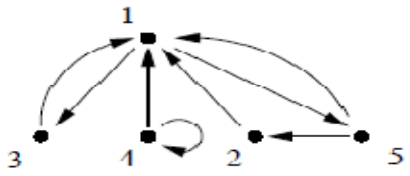


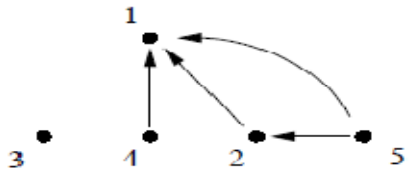

Sous-graphe induit par
S' = {1, 3, 4}

### c) Sous-graphes couvrants

Un sous-graphe G' = (S', A') d'un graphe G = (S, A) est couvrant si il contient tous les sommets de G : S' = S.

Exemple :

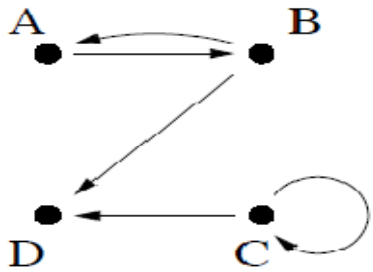



## 6. Représentation des graphes

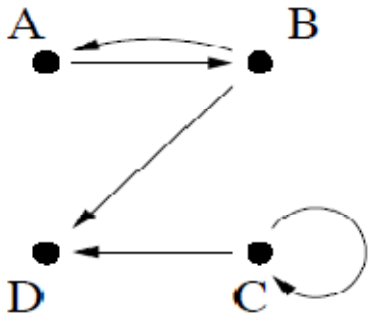
### a) Représentation sagittale

La représentation sagittale est la représentation sous forme d'un dessin :




**Dictionnaire d'un graphe**

Le dictionnaire d'un graphe consiste en la donnée de l'ensemble des sommets du graphe et de l'ensemble des successeurs de chaque sommet.
Exemple :



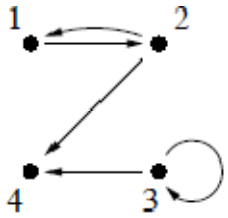

$S = \{A, B, C, D\}$
$G(A) = \{B\},\ G(B) = \{A, D\},$
$G(C) = \{C, D\},\ G(D) = \emptyset$

### b) Matrices d'adjacence

Soit G = (S, A) un graphe dont on a numéroté les sommets de 1 à n. La matrice d'adjacence de G est la matrice carrée M = ($m_{ij}$), de taille n × n, définie par :

$$m_{ij} = \begin{cases} 1 & si\ (i,j) \in A \\ 0 & sinon \end{cases}$$

Exemple :




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

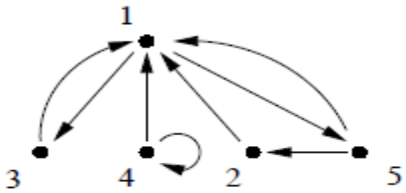
### c) Matrice d'incidence

Soit G =S, A) un graphe orienté sans boucle, la matrice d'incidence sommets-arcs est une matrice A = ($m_{ij}$) (n×m) à coefficients entiers 0, +1, -1 telle que chaque colonne correspond à un arc et chaque ligne à un sommet. Si u = (i, j) est un arc du graphe alors la colonne u a tous ses termes nuls sauf : $m_{iu}$ = +1 ; $m_{ju}$ = −1.
Le nombre de +1 (resp. de -1) dans la ligne i égale $d_+(i)$ (resp. $d_-(i)$).
Soit G = (S, A) un graphe non orienté sans boucle, la matrice d'incidence sommets-arêtes est une matrice A = ($m_{ij}$) (n×m) à coefficients entiers 0 ou 1 telle que chaque colonne correspond à une arête et chaque ligne à un sommet. Si e = ij est une arête du graphe alors la colonne e à tous ses termes nuls sauf :
$m_{iu}$ = 1 ;
$m_{ju}$= 1.

## 7. Connexité

### a) Chemin

Soit G = (S, A) un graphe orienté. Un chemin C est une suite ($x_0$, $x_1$, . . ., $x_{n-1}$, $x_n$) de sommets de G tel que deux sommets consécutifs quelconques xi et xi+1 sont reliés par un arc de G :
$\forall i,\ 0 \le i \le n-2,\ (x_i, x_{i+1}) \in A$ Les sommets $x_0$ et $x_n$ sont respectivement l'origine et l'extrémité du chemin C. Le chemin C est forme de n + 1 sommets et de n arcs mis bout à bout, sa longueur est n. Un chemin peut comporter un seul sommet et être de longueur 0.
Exemple :




(3, 1, 5) est un chemin, (1, 3, 4) n'est pas un chemin, (2) est un chemin, (4, 1, 2, 5) n'est pas un chemin, (2, 1, 5, 2, 1, 3, 1, 3) est un chemin, (4, 4, 4, 4, 4, 4) est un chemin.

### b) Circuit

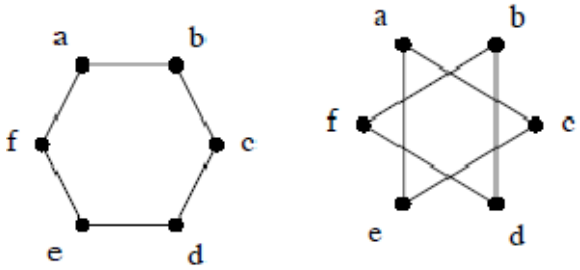
On appelle circuit un chemin de longueur non nulle et dont l'origine et l'extrémité sont identiques. Un chemin est dit :

- simple si il ne passe pas deux fois par le même arc

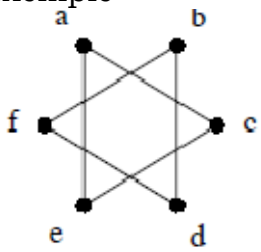
- élémentaire si il ne passe pas deux fois par le même sommet (à l'exception de l'origine et l'extrémité pour un circuit). Remarque : élémentaire ⇒ simple

## 8. graphes connexes et composantes connexes

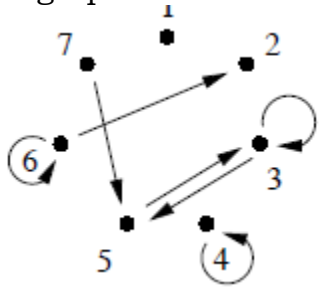
Un graphe non orienté est **connexe** si pour tout couple de sommets x, y il existe une chaîne reliant x à y. Un graphe orienté est connexe si le graphe non orienté associé est connexe.
Exemple




Une **composante connexe** C d'un graphe G = (S, A) est un sous-ensemble maximal de sommets tels que deux quelconques d'entre eux soient reliés par une chaîne : si x ∈ C, alors ∀y ∈ C, il existe une chaîne reliant x à y,
∀z ∈ S \ C, il n'existe pas de chaîne reliant x à z.
Les composantes connexes d'un graphe G = (S, A) forment une partition de S.
Un graphe est connexe si et seulement si il a une seule composante connexe.
Le sous-graphe induit par une composante connexe C est connexe.
La composante connexe C qui contient un sommet x ∈ S est
C = {y ∈ S | il existe une chaîne reliant x à y}
Exemple



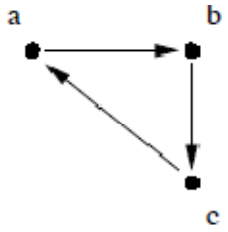

Ce graphe a deux composantes connexes : {a, c, e} et {b, d, f }.

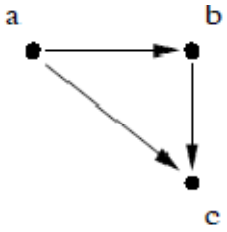



Ce graphe a quatre composantes connexes : {}, {2, 6}, {3, 5, 7}, et {4}.

Un graphe orienté est **fortement connexe** si pour tout couple de sommets x, y il existe un chemin reliant x à y.

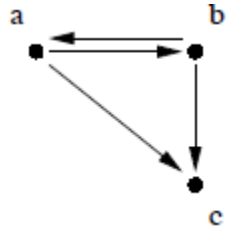
**Remarque** : la définition implique que si x et y sont deux sommets d'un graphe fortement connexe, alors il existe un chemin de x à y et un chemin de y à x.
Exemples :



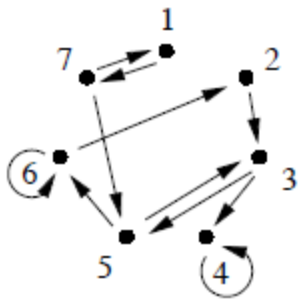




**Théorème**
Un graphe orienté fortement connexe est connexe.
**Théorème**
Un graphe est fortement connexe si et seulement si pour tout couple de sommets x, y il existe un circuit passant par x et y.

**Une composante fortement connexe** C d'un graphe G = (S, A) est un sous-ensemble maximal de sommets tels que deux quelconques d'entre eux soient reliés par un chemin : si $x \in C$, alors $\forall y \in C$, il existe un circuit passant par x et y, $\forall z \in S \setminus C$, il n'existe pas de circuit passant par x et z.
Les composantes fortement connexes d'un graphe G = (S, A) forment une partition de S.
Un graphe est fortement connexe si et seulement si il a une seule composante fortement connexe.
Le sous-graphe induit par une composante fortement connexe C est fortement connexe.
La composante fortement connexe C contenant un sommet x est :
$\{y \in S \mid$ il existe un chemin reliant x à y et un chemin reliant y à x$\}$
Exemple




Ce graphe a deux composantes fortement connexes : {a, b} et {c}.

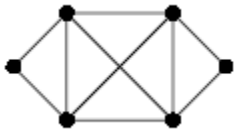



Ce graphe a trois composantes fortement connexes :{1, 7}, {2, 3, 5, 6} et {4}.

## 9. Cycles (resp circuits) eulériens

Au 18e siècle un casse-tête est populaire chez les habitants de Konigsberg : ***est-il possible de se promener dans la ville en ne passant qu'une seule fois par chacun des sept ponts de Konigsberg?*** C'est le célèbre mathématicien Euler qui montre le premier que ce problème n'a pas de solution, en utilisant pour la première fois la notion de graphe.

Soit G un graphe non orienté. Une chaîne (respectivement un cycle) eulérienne est une chaîne (resp. un cycle) qui passe une et une seule fois par toutes les arêtes de G. On définit les mêmes notions pour un graphe orienté G : un chemin ou un circuit eulérien est un chemin ou un circuit passant une et une seule fois par tous les arcs de G.



Ce graphe admet un cycle eulérien.

**Conditions nécessaires et suffisantes**

Les conditions précédentes sont en fait suffisantes.

**Théorème**

Un graphe G =(S, A) admet un cycle (cas non orienté) ou un circuit (cas orienté) eulérien si et seulement si les deux conditions suivantes sont vérifiées :

- le graphe G est connexe
- $\forall x \in S$, d(x) est pair (cas non orienté), $\forall x \in S$, $d_+(x) = d_-(x)$ (cas orienté)

**Théorème**

Un graphe G =(S, A) admet une chaîne (cas non orienté) ou un chemin (cas orienté) eulérien(ne) si et seulement si les deux conditions suivantes sont vérifiées :

- graphe G est connexe
- Pour tous les sommets x sauf éventuellement deux, d(x) est pair (cas non orienté), pour tous les sommets x sauf éventuellement deux, $d_+(x) = d_-(x)$, les deux derniers vérifiant $d_+(x) = d_-(x) \pm 1$ (cas orienté)

## 10. Cycles hamiltoniens

Soit G un graphe non orienté. Un cycle (respectivement une chaîne) hamiltonien est un cycle (resp. une chaîne) qui passe une et une seule fois par tous les sommets de G. On définit les mêmes notions pour un graphe oriente G : un circuit ou un chemin hamiltonien est un circuit ou un chemin passant une et une seule fois par tous les sommets de G.

**Conditions nécessaires et suffisantes ?**

Question : comment déterminer si un graphe admet des cycles (circuits) hamiltoniens ?

Contrairement au cas des cycles eulériens, il n'existe pas de réponses simples : ce problème est algorithmiquement difficile.

## 11. Graphes valués

Un graphe (orienté ou non) G = (S, A) est valué si il est muni d'une application

$v : A \rightarrow R$

$(x, y) \rightarrow v(x, y)$ qui sera appelée valuation. On peut étendre la valuation en une fonction $S \times S \rightarrow R \cup \{+\infty\}$ en posant $v(x, y) = +\infty$ si (x, y) n'appartient pas A.

## 12. Distance et plus court chemin

Soit G = (S, A, v) un graphe valué et soient x, y deux sommets de G.

- On appelle distance de x à y et on note d(x, y) le minimum des valuations des chemins / chaînes allant de x à y.
- On appelle plus court chemin / plus courte chaîne de x à y tout chemin / chaîne dont la valuation est égale à d(x, y).

## 13. Recherche de plus courts chemins

De nombreux problèmes concrets peuvent se modéliser comme des recherches de plus courts chemins dans des graphes valués. Par exemple :

- recherche de l'itinéraire le plus rapide en voiture entre deux villes, ou en métro entre deux stations.
- routage dans des réseaux de télécommunications.

Certains problèmes d'ordonnancement font aussi appel à des recherches de plus longs chemins.

On étudiera principalement des algorithmes qui résolvent le problème suivant : étant donné un sommet x, déterminer pour chaque sommet y la distance et un plus court chemin de x à y.

**Remarques**

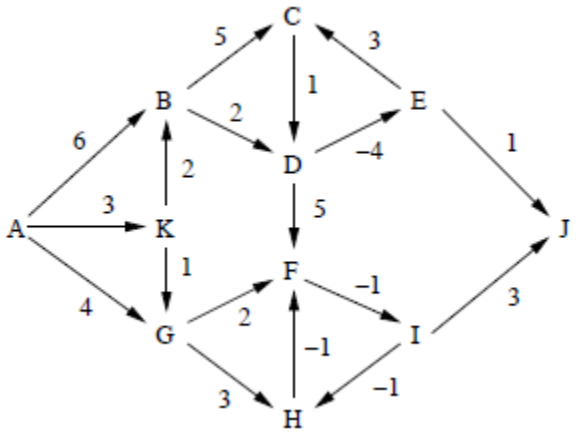
Dans un graphe non orienté, on a toujours d(x, y) = d(y, x), et toute plus courte chaîne de x à y parcourue à l'envers est une plus courte chaîne de y à x.

Etant donnés deux sommets x et y, plusieurs cas se présentent :

- il n'y a pas de chemins / chaînes de x à y

- il existe un ou plusieurs plus courts chemins / chaînes de x à y
- il existe des chemins / chaînes de x à y mais pas de plus court.

Exemple




De A à B : il existe un unique plus court chemin (A,K, B).
De A à G : il existe deux plus courts chemins (A,K, G) et (A, G).
De E à A : il n'existe pas de chemins, donc pas de plus courts chemins.

**Circuit absorbant**
Un circuit absorbant est un circuit de valuation négative. Si un graphe possède un circuit absorbant, alors il n'existe pas de plus courts chemins entre certains de ces sommets.

**Théorème**
Soit G un graphe orienté valué n'ayant pas de circuits absorbants, et x et y deux sommets de G. Si il existe un chemin allant de x à y, alors la distance d(x, y) est bien définie et il existe au moins un plus court chemin de x à y.
Dans la suite, les graphes seront donc sans circuits absorbants. On définit de la même manière un cycle absorbant dans un graphe non orienté. Le théorème reste vrai en remplaçant chemin par chaîne.

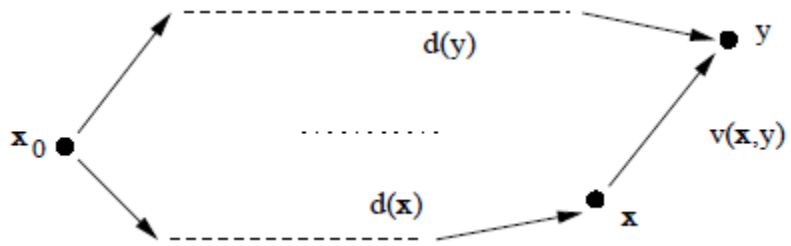
**Propriétés des plus courts chemins**

1. Tout sous-chemin d'un plus court chemin est un plus court chemin.
2. Si il existe un plus court chemin entre deux sommets x et y, alors il existe un plus court chemin élémentaire entre x et y.

**Principe des algorithmes dans le cas général**
Etant donnés un graphe valué G = (S, A, v) et un sommet $x_0$, on veut déterminer pour chaque sommet s la distance et un plus court chemin de $x_0$ à s.
Les algorithmes de recherche de distance et de plus court chemin dans un graphe valué fonctionnent de la façon suivante.
• On calcule les distances d($x_0$, s) par approximations successives. A un stade donné de l'algorithme on dispose d'estimations d(s) (éventuellement égales à +∞) pour ces distances, et de la donnée d'un prédécesseur P(s) pour les plus courts chemins.

- A chaque étape, on considère un sommet x et un successeur y de x.

On compare la valeur d(y) à celle que l'on obtiendrait en passant par x, c'est-à-dire d(x) + v(x, y).




- Si cette deuxième valeur est plus petite que d(y), on remplace l'estimation d(y) par d(x) + v(x, y) et le père P(y) par x.

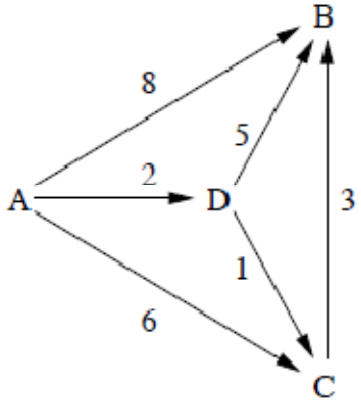
## Algorithme de Bellman-Ford

On applique le principe précédent en explorant systématiquement tous les sommets et tous leurs successeurs. On s'arrête quand les valeurs des distances sont stabilisées.

**Initialisation :**
$d(x_0) = 0$, $P(x_0) = nul$
**pour tout** $s \in S$, $s \neq x_0$ **répéter**
    $d(s) = +\infty$, $P(s) = nul$
$fin = FAUX$

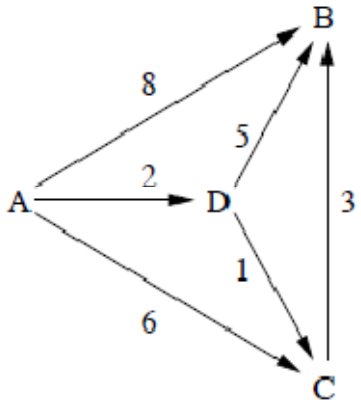
**tant que** $fin = FAUX$ **répéter**
    $fin \leftarrow VRAI$
    **pour tout** $x \in S$ **répéter**
        **pour tout** $y \in G(x)$ **répéter**
            **si** $d(x) + v(x,y) < d(y)$ **alors**
                $d(y) \leftarrow d(x) + v(x,y)$ (màj distance)
                $P(y) \leftarrow x$ (màj père)
                $fin \leftarrow FAUX$ (pas encore stabilisé)
**fin tant que**

Exemple




| Itération | $d(A)$ | $d(B)$ | $d(C)$ | $d(D)$ | $P(A)$ | $P(B)$ | $P(C)$ | $P(D$ |
|---|---|---|---|---|---|---|---|---|
| Initialisation | 0 | $+\infty$ | $+\infty$ | $+\infty$ | nul | nul | nul | nul |
| 1 | 0 | 7 | 3 | 2 | nul | D | D | A |
| 2 | 0 | 6 | 3 | 2 | nul | C | D | A |
| 3 | 0 | 6 | 3 | 2 | nul | C | D | A |

Pour trouver un plus court chemin entre $x_0$ et s, on se sert du tableau des pères.
On construit le chemin à partir de la fin : on part de s, le tableau des pères donne son prédécesseur p le long d'un plus court chemin, et on recommence avec p.
Exemple :




| $P(A)$ | $P(B)$ | $P(C)$ | $P(D)$ |
|---|---|---|---|
| nul | C | D | A |

Un plus court chemin de $A$ à $B$ est
A→D→C→B

**Remarques**

- Le nom de cet algorithme et de ses variantes similaires est assez variable selon les sources : Bellman, Ford, Bellman-Ford, Bellman-Kalaba,. . .
- L'algorithme permet de détecter la présence de circuits absorbants : si les valeurs d(s) ne sont pas stabilisées après n passages de boucles, alors le graphe contient au moins un circuit absorbant.
- On peut améliorer légèrement l'algorithme en ne regardant que les successeurs des sommets ayant été modifiés récemment.

## Bellman-Ford amélioré

C'est la version de l'algorithme que l'on utilisera dans les exercices.

**Initialisation :**
$d(x_0) = 0$, $P(x_0) = nul$
**pour tout** $s \in S$, $s \neq x_0$ **répéter**
    $d(s) = +\infty$, $P(s) = nul$
$L = \{x_0\}$ (liste des sommets à explorer)

**tant que** $L \neq \emptyset$ **répéter**
    **choisir** $x \in L$
    $L \leftarrow L \setminus \{x\}$
    **pour tout** $y \in G(x)$ **répéter**
        **si** $d(x) + v(x,y) < d(y)$ **alors**
            $d(y) \leftarrow d(x) + v(x,y)$ (màj distance)
            $P(y) \leftarrow x$ (màj père)
            $L \leftarrow L \cup \{y\}$ (màj sommets à explorer)
**fin tant que**

## Algorithme de Dijkstra

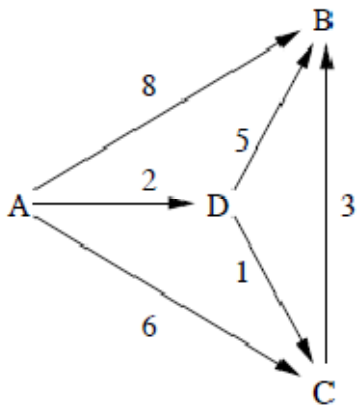
L'algorithme de Dijkstra est un autre algorithme de recherche de distance et de plus court chemin.
Il est plus efficace que Bellman-Ford, mais ne fonctionne que dans le cas où toutes les valuations des arcs sont positives.
Principe :

- On construit petit à petit, à partir de $\{x_0\}$, un ensemble M de sommets marqués. Pour tout sommet marqué s, l'estimation d(s) est égale à la distance $d(x_0, s)$.
- A chaque étape, on sélectionne le (un) sommet non marqué x dont la distance estimé d(x) est la plus petite parmi tous les sommets non marqué.
- On marque alors x (on rajoute x à M), puis on met à jour à partir de x les distances estimées des successeurs non marqués de x.
- On recommence, jusqu'à épuisement des sommets non marqués.

**Initialisation**
$d(x_0) = 0$, $P(x_0) = nul$
aucun sommet n'est marqué
$min\_dist\_M = 0$ (minimum des distances estimées des sommets non marqués)
**pour tout** $s \in S$, $s \neq x_0$ **répéter**
    $d(s) = +\infty$, $P(s) = nul$

**répéter**
    chercher $x$ non marqué tel que $d(x) = min\_dist\_M$
    marquer $x$
    **pour tout** $y \in G(x)$, $y$ non marqué **répéter**
        **si** $d(x) + v(x, y) < d(y)$ **alors**
            $d(y) \leftarrow d(x) + v(x, y)$ (màj distance)
            $P(y) \leftarrow x$ (màj père)
    $min\_dist\_M = \min\{d(s),\ s \notin M\}$
**jusqu'à ce que** $min\_dist\_M = +\infty$

Exemple




| $M$ | $d(A)$ | $d(B)$ | $d(C)$ | $d(D)$ | $P(A)$ | $P(B)$ | $P(C)$ | $P(D$ |
|---|---|---|---|---|---|---|---|---|
| $\emptyset$ | 0 | $+\infty$ | $+\infty$ | $+\infty$ | nul | nul | nul | nul |
| $\{A\}$ | | 8 | 6 | 2 | | A | A | A |
| $\{A, D\}$ | | 7 | 3 | | | D | D | |
| $\{A, D, C\}$ | | 6 | | | | C | | |
| $\{A, D, C, B\}$ | 0 | 6 | 3 | 2 | nul | C | D | A |

**Commentaires**

- L'algorithme de Dijkstra est plus performant que Bellman-Ford : il est donc à privilégier systématiquement
- Par contre on ne peut pas l'appliquer dès qu'on a des valuations négatives (ou assimilées) : problème des plus longs chemins en particulier.
- Ces algorithmes sont pratiques si on connaît tout le graphe : routage dans un réseau local par exemple.

Les algorithmes de routage dans des réseaux importants sont en général distribués : chaque serveur (↔ sommet) connaît juste son voisinage immédiat et le communique à ses voisins.

**Détermination des plus courts chemins**

Quand on connaît les distances de $x_0$ à s pour tout sommet s, on peut déterminer (tous) les plus courts chemins de $x_0$ à s pour tout sommet s. Si on dispose du tableau des pères, il est facile de trouver un plus court chemin; si on n'a pas ce tableau ou si on recherche tous les plus courts chemins, on procède de la manière étudiée pour les plus courts chemins en nombre d'arcs.

Pour trouver un plus court chemin de $x_0$ à s, on part de la fin :

• on cherche un prédécesseur p de s tel que $d(x_0, s) = d(x_0, p) + v(p, s)$ (ou tous les prédécesseurs vérifiant cette condition si on cherche tous les plus courts chemins)

• on recommence en partant de p

Remarque : Si la distance de $x_0$ à s est bien définie, on a toujours $d(x_0, s) \leq d(x_0, p) + v(p, s)$ pour tout prédécesseur p de s, et il existe au moins un prédécesseur pour lequel il y a égalité.

## 14. Arbres

Un arbre est un graphe non orienté, connexe, sans cycle.

Une forêt est un graphe non orienté sans cycle (chacune de ses composantes connexes est un arbre).

Remarque : Dans cette définition, on ne privilégie aucun sommet en particulier. Les arbres classiquement étudiés en algorithmique sont des arbres enracinés, que l'on verra plus loin.

### a) Caractérisation des arbres

Le théorème fondamental suivant donne six caractérisations alternatives des arbres :

**Théorème**

Soit G un graphe non orienté à n sommets. Les propositions suivantes sont équivalentes :

- le graphe G est connexe et sans cycle
- le graphe G est connexe et a n − 1 arêtes
- le graphe G est connexe et la suppression de n'importe quelle arête le déconnecte
- le graphe G est sans cycle et a n − 1 arêtes
- le graphe G est sans cycle et l'ajout de n'importe quelle arête crée un cycle
- entre toute paire de sommets de G il existe une unique chaîne élémentaire

On voit en particulier qu'un arbre possédant n sommets a toujours exactement n − 1 arêtes.

Cela découle de la propriété plus générale suivante :

**Propriété**

• Un graphe connexe à n sommets a toujours au moins n − 1 arêtes.
• Un graphe sans cycle à n sommets a toujours au plus n − 1 arêtes.

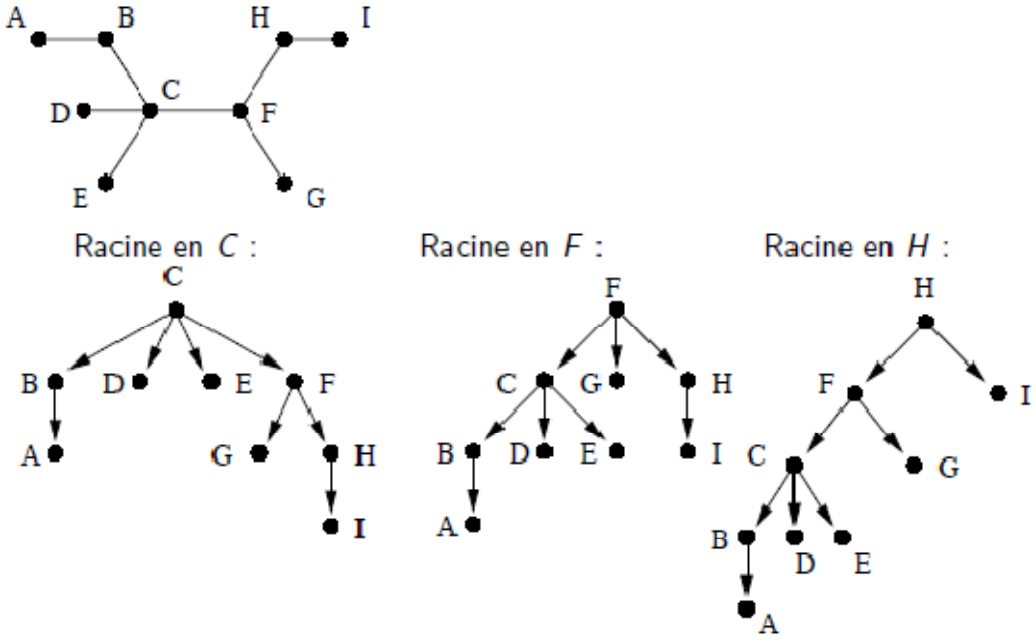
### b) Arbres enracinés

Les arbres utilisés en algorithmique ont le plus souvent une orientation et un sommet qui joue un rôle particulier, la racine : c'est ce type d'arbre que l'on va voir maintenant

Un graphe non orienté est un arbre enraciné s'il est connexe sans cycle et si un sommet particulier, la racine, a été distingué.

Un arbre enraciné est souvent muni d'une orientation naturelle : on oriente chaque arête de telle sorte qu'il existe un chemin de la racine à tout autre sommet. Le graphe orienté résultant est aussi appelé arbre enraciné ou plus simplement arbre.

Exemple




**Caractérisation**

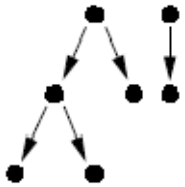
Un graphe orienté est un arbre enraciné si et seulement si

- il est connexe,
- il a un unique sommet sans prédécesseur (la racine),
- et tous ses autres sommets ont exactement un prédécesseur.

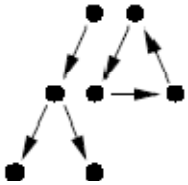
Exemple

N'est pas un arbre : un sommet a deux prédécesseurs

N'est pas un arbre : aucun sommet sans prédécesseur (pas de racine)



N'est pas un arbre : deux sommets sans prédécesseur



N'est pas un arbre : non connexe

Est un arbre

### c) Arbres couvrant de poids minimum

Soit G un graphe valué non orienté connexe.
Un arbre couvrant est un sous-graphe de G couvrant (i.e. contenant tous les sommets), connexe et sans cycle. Son poids est la somme des valuations de ses arêtes.
Un arbre couvrant de poids minimum (en anglais minimum spanning tree) est un arbre couvrant dont le poids est le plus petit possible parmi les arbres couvrants de G. Si toutes les arêtes ont des valuations positives, son poids est le plus petit possible parmi tous les sous-graphes connexes couvrants de G.

### d) Algorithmes

On va étudier deux algorithmes classiques permettant de trouver un arbre couvrant de poids minimum dans un graphe valué G donné.
On supposera toujours que le graphe G est non orienté et que les valuations des arêtes sont toutes positives.
Dans les deux cas, on va construire l'arbre couvrant petit à petit, en s'assurant à chaque étape

- que l'on reste couvrant sans cycle (algorithme de Kruskal)
- ou que l'on reste connexe sans cycle (algorithme de Prim)

Ce sont deux exemples d'algorithmes gloutons : à chaque étape on fait le meilleur choix instantané, dans le but d'obtenir la meilleure solution globale.

#### ➢ Algorithme de Kruskal

Principe : On construit un sous-graphe en ajoutant des arêtes une par une. A chaque étape, on cherche l'arête de plus petite valuation parmi celles que l'on n'a pas déjà explorées. Si elle ne crée pas un cycle, on l'ajoute au sous-graphe, sinon on la laisse de côté. On termine dès que l'on a sélectionnée n – 1 arêtes, ou qu'il ne reste plus d'arêtes ne créant pas de cycles.

Initialisation :
$A$ = ensemble des arêtes du graphe $G$
$F = \emptyset$ (ensemble des arêtes de l'arbre couvrant)

Trier l'ensemble $A$ des arêtes de $G$ par valuations croissantes

**pour tout** $e \in A$ (parcouru dans l'ordre) **répéter**
    **si** $F \cup \{e\}$ est acyclique **alors**
        $F \leftarrow F \cup \{e\}$
**Fin pour**

**Commentaies**

- A chaque étape de l'algorithme, on obtient une forêt couvrante (i.e. un sous-graphe couvrant sans cycle), qui grossit jusqu'à devenir un arbre.
- Si le graphe G n'est pas connexe, l'algorithme donne un arbre couvrant de poids minimum sur chaque composante connexe de G.
- La partie la plus coûteuse de l'algorithme de Kruskal est en fait le tri initial des arêtes. . .

➢ **Algorithme de Prim**

Principe :

On construit un sous-graphe en ajoutant arêtes et sommets les un après les autres. A chaque étape, on cherche l'arête sortante de plus petite valuation. Une arête est sortante si elle joint un sommet du sous-graphe à un sommet qui n'est pas dans le sous-graphe. On ajoute alors l'arête et le sommet qu'elle joint au sous-graphe. On termine dès que l'on a sélectionné n − 1 arêtes.

Initialisation :
$A$ = ensemble des arêtes du graphe $G$
$F = \emptyset$ (ensemble des arêtes de l'arbre couvrant)
$M = \{x_0\}$ (On **marque** un sommet quelconque de $G$)

**tant que** il y a des arêtes sortantes de $M$ **répéter**
    **chercher** l'arête sortante $e = (x, y)$ de **plus petite valuation** ($e$ sortante : $x \in M$ et $y \notin M$)
    $M \leftarrow M \cup \{y\}$
    $F \leftarrow F \cup \{e\}$
**Fin tant que**

**Commentaires**

- A chaque étape de l'algorithme, on obtient un sous-graphe partiel qui est un arbre, et qui grossit jusqu'à devenir couvrant.
- Si le graphe G est bien connexe, le choix du sommet initial n'est pas important : tous les sommets finissent par être visites par l'algorithme.
- Si le graphe G n'est pas connexe, l'algorithme donne un arbre couvrant de poids minimum sur la composante connexe de G contenant le sommet initial.