COURS DE THEORIE DES GRAPHES ET APPLICATIONS

Prof Adama COULIBALY [1]

22 avril 2016

1. Université Félix Houphouët-Boigny, UFR de Mathématiques et Informatique ; 22 BP 582 Abidjan 22

Table des matières

Chapitre 1

Généralités sur les graphes

1.1 Graphes orientés

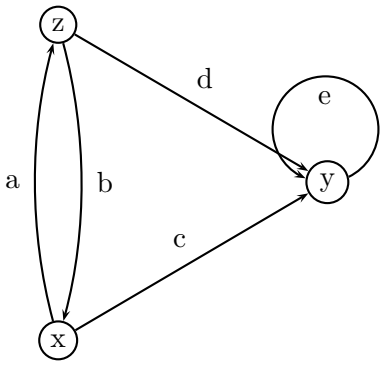
1.1.1 Définitions et propriétés de base

Définition 1.1.1 *Un graphe orienté G est défini par la donnée d'un ensemble fini X dont les éléments sont appelés des sommets ou noeuds et d'un ensemble U dont les éléments sont des couples d'éléments de X appelés arcs. On le note $G = (X, U)$. Le nombre n de sommets de G est appelé l'ordre de G et m celui des arcs de G est appelé la taille de G.*

Graphiquement, les sommets sont représentés par des points ou des cercles et un arc $u = (x, y)$ est représenté par une flèche joignant les sommets x et y, la flèche étant pointée sur y. On parle de représentation sagittale du graphe.

Exemple

Soit $G = (X, U)$ le graphe orienté défini par : $X = \{x, y, z\}$ et $U = \{a, b, c, d, e\}$ où $a = (x, z)$, $b = (z, x)$, $c = (x, y)$, $d = (z, y)$ et $e = (y, y)$.



Définition 1.1.2 *Etant donné un arc $u = (x, y)$ d'un graphe orienté $G = (X, U)$, on dit que le sommet x est l'origine ou l'extrémité initiale de u ; le sommet y est le but ou l'extrémité finale ou terminale de u. On dit aussi que y est un successeur de x et que x est un prédécesseur de y. On dit également que les sommets x et y sont adjacents ou voisins et que l'arc u est incident à x et y. Plus précisement il est dit incident intérieurement à y et incident extérieurement à x.*

Définition 1.1.3 *Un sommet qui n'est adjacent à aucun sommet est dit isolé.*

Un graphe dont les sommets sont isolés est dit discret.

Définition 1.1.4 *Un arc dont les deux extrémités coïncident est appelé une boucle.*

Demi-degré et degré

Deux arcs sont *adjacents* s'ils sont incidents à au moins une extrémité commune.
Le demi-degré extérieur ou *demi-degré sortant* de sommet x, noté $d_G^+(x)$ ou plus simplement $d^+(x)$ est le nombre d'arcs ayant x pour extrémité initiale.
Le demi-degré intérieur ou *demi-degré entrant* de sommet x, noté $d_G^-(x)$ ou plus simplement $d^-(x)$ est le nombre d'arcs ayant x pour extrémité finale.
Le degré du sommet x, noté $d_G(x)$ ou plus simplement $d(x)$ est : $d(x) = d^+(x) + d^-(x)$.

Proposition 1.1.1 *La somme des degrés des sommets de G est égale à deux fois la taille de G.*

Preuve : Il suffit de remarquer que dans la somme des degrés chaque arc est compté deux fois : une fois dans le demi-degré extérieur et une autre fois dans le demi-degré intérieur. □

Corollaire 1.1.1 *Dans un graphe orienté, le nombre de sommets de degrés impairs est pair.*

Etant donné $S \subset X$, on définit :

- $\omega^+(S)$ est l'ensemble des arcs ayant leur extrémité initiale dans S et leur extrémité terminale dans $X - S$.
- $\omega^-(S)$ est l'ensemble des arcs ayant leur extrémité initiale dans $X - S$ et leur extrémité terminale dans S.
- $\omega(S) = \omega^+(S) \cup \omega^-(S)$ est appelé un *cocycle* du graphe.

Propriétés de graphe

Soit $G = (X, U)$ un graphe orienté. On dit que G est :
- *réflexif* si $\forall\, x \in X, (x, x) \in U$. Il y a une boucle sur chaque sommet.
- *anti-réflexif* si $\forall\, x \in X, (x, x) \notin U$. C'est un graphe sans boucle.
- *symétrique* si $\forall\, x \in X,\ \forall\, y \in X : (x, y) \in U \Rightarrow (y, x) \in U$.
L'existence d'un arc implique l'existence de l'arc opposé.
- *anti-symétrique* si $\forall\, x \in X,\ \forall\, y \in X : (x, y) \in U$ et $(y, x) \in U \Rightarrow x = y$.
L'existence d'un arc interdit l'existence de l'arc opposé, sauf dans le cas d'une boucle.
- *transitif* si $\forall\, x, y, z \in X : (x, y) \in U$ et $(y, z) \in U \Rightarrow (x, z) \in U$.
- *strict* si G est sans boucle et deux sommets quelconques sont tous deux incidents à au plus un arc.
- *biparti* si l'ensemble des sommets X peut être partitionné en deux sous ensembles X_1 et X_2 de tel sorte que, tout arc a une extrémité dans X_1 et l'autre dans X_2.
- *complet* si G est strict tel que deux sommets distincts quelconques sont reliés par un seul arc.

Graphes complémentaires

Si $G = (X, U)$ est strict on appelle *graphe complémentaire* de G le graphe $\overline{G} = (X, \bar{U})$ ayant le même ensemble de sommets que G et ayant pour arcs les arcs complémentaires à U. C'est-à-dire :

$$(x, y) \in U \rightarrow (x, y) \notin \bar{U}$$

$$(x, y) \notin U \rightarrow (x, y) \in \bar{U}$$

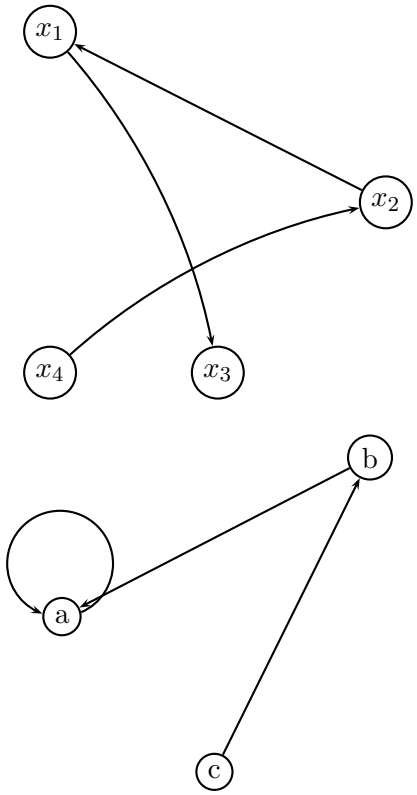
Graphes homomorphes et isomorphes

Définition 1.1.5 *Deux graphes orientés $G = (X, U)$ et $G' = (X', U')$ sont dits homomorphes s'il existe une fonction $f : X \rightarrow X'$ telle que :*

$$(x, y) \in U \Rightarrow (f(x), f(y)) \in U'.$$

Exemple

Les graphes ci-dessous sont homomorphes.



Définition 1.1.6 *Deux graphes orientés $G = (X, U)$ et $G' = (X', U')$ sont dits isomorphes s'il existe deux bijections $B_s : X \to X'$ et $B_a : U \to U'$ telles que deux arcs qui se corespondent dans la bijection B_a aient pour extrémités initiales et terminales respectivement des sommets qui se correspondent dans la bijection B_s. C'est-à-dire si :*

$$u = (x, y) \text{ et } u' = (x', y') \text{ avec } u' = B_a(u), \text{ alors } B_s(x) = x' \text{ et } B_s(y) = y'.$$

Sous-graphe et graphe partiel

Soient $G = (X, U)$ un graphe orienté, Y une partie de X et V une partie de U.

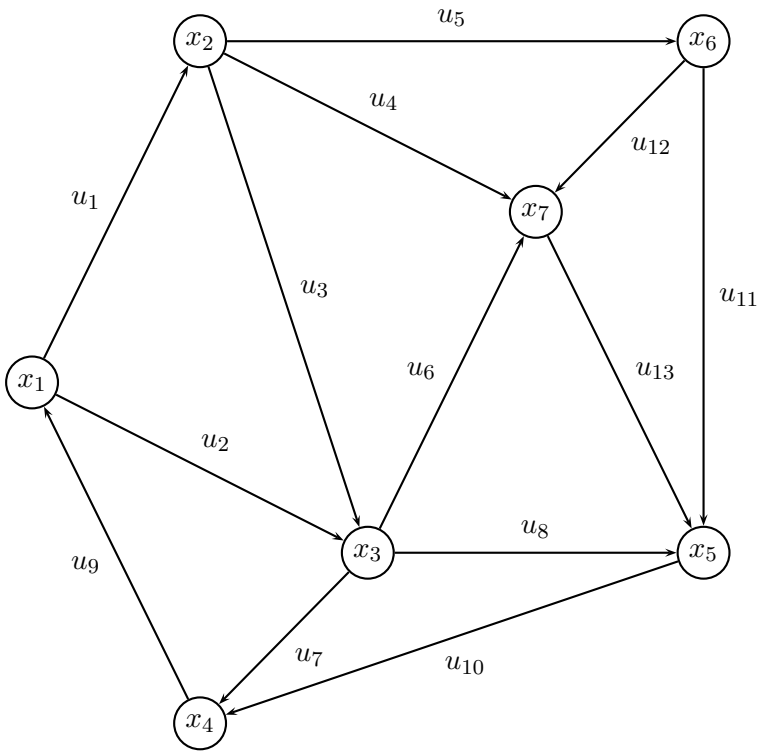
On appelle *sous-graphe* engendré par Y, le graphe noté G_Y dont les sommets sont les éléments de Y et dont les arcs sont les arcs de G ayant leurs deux extrémités dans Y.

On appelle *graphe partiel* engendré par V, le graphe noté $G[V]$ ayant le même ensemble X de sommets que G et dont les arcs sont ceux de V.

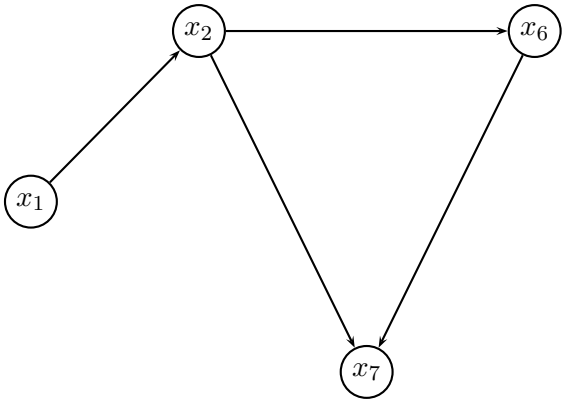
Le *sous-graphe partiel* engendré par Y et V est le graphe partiel de G_Y engendré par V.

Exemple

On considère le graphe $G = (X, U)$ avec $X = \{x_1, x_2, \cdots, x_7\}$, et $U = \{u_1, u_2, \cdots, u_{12}\}$



Le sous-graphe engendré par $Y = \{x_1, x_2, x_6, x_7\}$ est



1.1.2 Cheminement dans un graphe

Chemin, circuit

Définition 1.1.7 *Etant donné un graphe orienté $G = (X, U)$ et x, y deux sommets de G, on appelle chemin de x à y toute suite ordonnée d'arcs $\mu = (u_1, \cdots, u_q)$ telle que :*

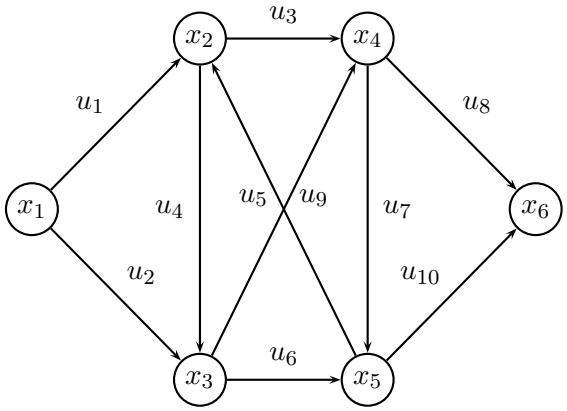
i) l'origine de u_1 est x,

ii) pour tout $i \in \{1, \cdots, q-1\}$, l'origine de u_{i+1} est l'extrémité de u_i,

iii) l'extrémité de u_q est y.

Les sommets x et y sont alors dits extrémités du chemin μ qui est de longueur q au sens des arcs.

Exemple



dans le graphe ci-dessus, $\mu = (u_1, u_3, u_7, u_9, u_4, u_5, u_8)$ est un chemin de x_1 à x_6.

Un *chemin élémentaire* est un chemin tel qu'en le parcourant, on ne rencontre pas deux fois le même sommet. Dans un chemin élémentaire tous les sommets sont de degré 2 au plus.

Un *chemin simple* est chemin dont la sequence d'arcs ne contient pas plusieurs fois un même élément. C'est-à-dire en le parcourant, on ne rencontre pas plusieurs fois un même arc.

Proposition 1.1.2 *Tout chemin élémentaire est simple.*

Proposition 1.1.3 *De tout chemin on peut extraire un chemin élémentaire.*

Définition 1.1.8 *Etant donné un graphe orienté $G = (X, U)$, un sommet y est dit descendant de x, s'il existe un chemin de x à y. dans ce cas , x est un ascendant de y.*

Un *circuit* est un chemin dont les extrémités coïncident.
Un *circuit élémentaire* est un chemin élémentaire dont les deux extrémités coïncident.
Un *circuit simple* est un chemin simple dont les deux extrémités coïncident.
On montre que

Proposition 1.1.4 *Tout circuit simple est union disjointe de circuits élémentaires.*

Définition 1.1.9 *Etant donné un graphe orienté $G = (X, U)$,*

- un chemin est hamiltonien s'il passe une fois et une seule par chaque sommet du graphe. Ou encore un chemin élémentaire passant par tous les sommets du graphe.

- Un circuit hamiltonien est un chemin hamiltonien dont les deux extrémités coïncident.

On parle de chemin pré-hamiltonien s'il passe au moins une fois par chaque sommet du graphe.

Définition 1.1.10 *Etant donné un graphe orienté $G = (X, U)$,*

- un chemin est eulérien s'il passe une fois et une seule par chaque arc du graphe. Ou encore un chemin simple passant par tous les arcs du graphe.

- Un circuit eulérien est un chemin eulérien dont les deux extrémités coïncident.

On parle de chemin pré-eulérien s'il passe au moins une fois par chaque arc du graphe.

Chaîne, cycle

Une *Chaîne* de longueur k est une suite $\mu = a_1, a_2, \dots a_k$ de k arcs telle que deux arcs consécutifs sont adjacents. On notera $\mu = x_0 a_1 x_1 \dots x_{k-1} a_k x_k$, si l'on veut préciser les sommets rencontrés. Les sommets x_0 et x_k sont appelés les *extrémités de la chaîne* μ. On dit que la chaîne μ rélie les sommets x_0 et x_k.

Une chaîne peut être "vue" comme une suite de sommets telle que deux sommets consécutifs soient liés par un arc, certains arcs peuvent être parcourus dans le sens de la chaîne (sens +), les autres arcs dans le sens opposé (sens -).

Une *chaîne élémentaire* est une chaîne telle qu'en la parcourant, on ne rencontre pas deux fois le même sommet.

Une *chaîne simple* est une chaîne dont la sequence d'arcs ne contient pas plusieurs fois un même élément.

Proposition 1.1.5 *Toute chaîne élémentaire est simple.*

Proposition 1.1.6 *De toute chaîne on peut extraire un chaîne élémentaire.*

Un *cycle* est une chaîne dont les extrémités coïncident.

Un *cycle élémentaire* (resp. *simple*) est une chaîne élémentaire (resp. simple) dont les deux extrémités coïncident.

Proposition 1.1.7 *Tout cycle simple est union disjointe de cycles élémentaires.*

Définition 1.1.11 *Etant donné un graphe orienté $G = (X, U)$,*

- une chaîne est hamiltonienne si elle passe une fois et une seule par chaque sommet du graphe. Ou encore une chaîne élémentaire passant par tous les sommets du graphe.

- Un cycle hamiltonien est une chaîne hamiltonienne dont les deux extrémités coïncident.

Le graphe G est dit hamiltonien s'il possède un cycle hamiltonien.

On parle de chaîne pré-hamiltonienne si elle passe au moins une fois par chaque sommet du graphe.

Définition 1.1.12 *Etant donné un graphe orienté $G = (X, U)$,*

- une chaîne est eulérienne si elle passe une fois et une seule par chaque arc du graphe. Ou encore une chaîne simple passant par tous les arcs du graphe.

- Un cycle eulérien est un cycle simple passant par tous les arcs du graphe.

Le graphe G est dit eulérien s'il possède un cycle eulérien

1.1.3 Connexité et forte connexité

Graphe connexe

Soit $G = (X, U)$ un graphe orienté. On dit que G est *connexe* si, pour tout couple de sommets (x, y), il existe une chaîne reliant x et y. On associe à cette notion de connexité une relation d'équivalence $\mathcal{R}$ définie par :

$$x \mathcal{R} y \Leftrightarrow \begin{cases} \text{soit} \quad x = y, \\ \text{soit il existe une chaîne reliant} \quad x \text{ et } y. \end{cases}$$

Les classes d'équivalence induites sur X par cette relation forment une partition de X en : $X_1, \dots, X_p$. Le nombre p de classes distinctes est appelé le *nombre de connexité* du graphe.

Proposition 1.1.8 *G est connexe si et seulement si $p = 1$.*

Les sous graphes $G_1, \ldots, G_p$ engendrés par les sous ensembles $X_1, \ldots, X_p$ sont appelés les *composantes connexes* du graphe G. Chaque composante connexe est un graphe connexe.

Un *point d'articulation* d'un graphe est un sommet dont la suppression augmente le nombre de composantes connexes. Un *isthme* est un arc dont la suppression a le même effet.

Graphe fortement connexe

Soit $G = (X, U)$ un graphe orienté. On dit que G est *fortement connexe* si, pour tout couple de sommets (x, y), il existe un chemin allant de x à y. On associe à cette notion de forte connexité une relation d'équivalence $\mathcal{T}$ définie par :

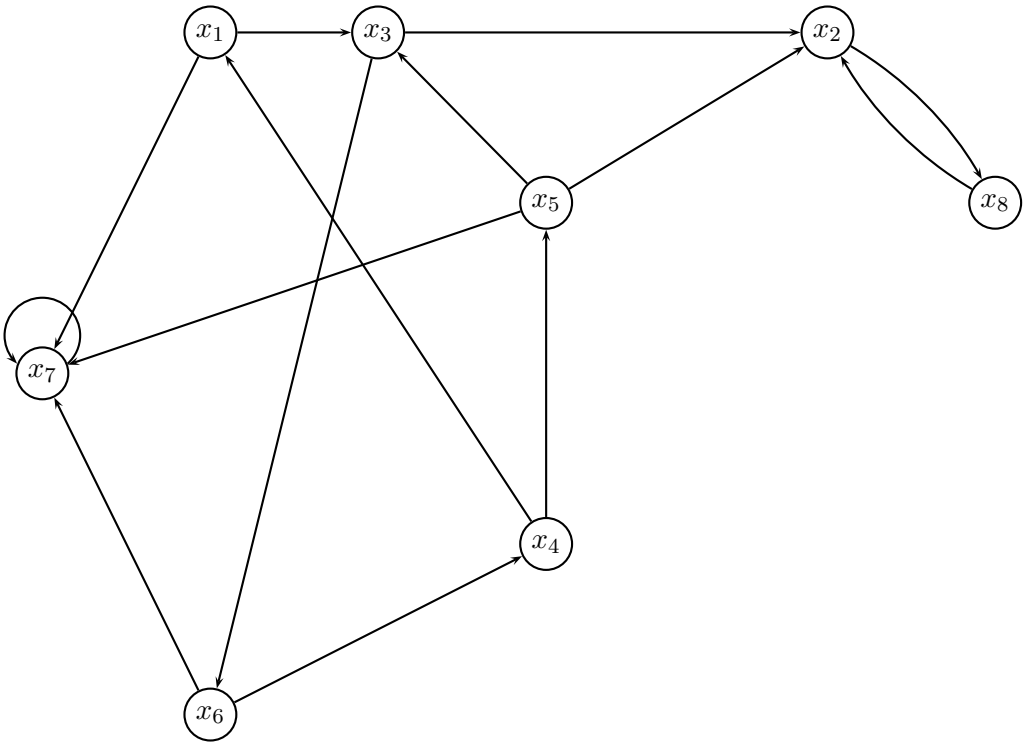
$$x \mathcal{T} y \Leftrightarrow \begin{cases} \text{soit } \ x = y, \\ \text{soit il existe un chemin allant de } \ x \text{ à } y \text{ et un chemin allant de } y \text{ à } x. \end{cases}$$

Les classes d'équivalence induites sur X par cette relation forment une partition de X en : $X_1, \ldots, X_q$. Le nombre q de classes distinctes est appelé le *nombre de forte connexité* du graphe.

Les sous graphes $G_1, \ldots, G_q$ engendrés par les sous ensembles $X_1, \ldots, X_q$ sont appelés les *composantes fortement connexes* du graphe G. Chaque composante fortement connexe est un graphe fortement connexe.

Définition 1.1.13 *G est fortement connexe s'il n'a qu'une seule composante fortement connexe.*

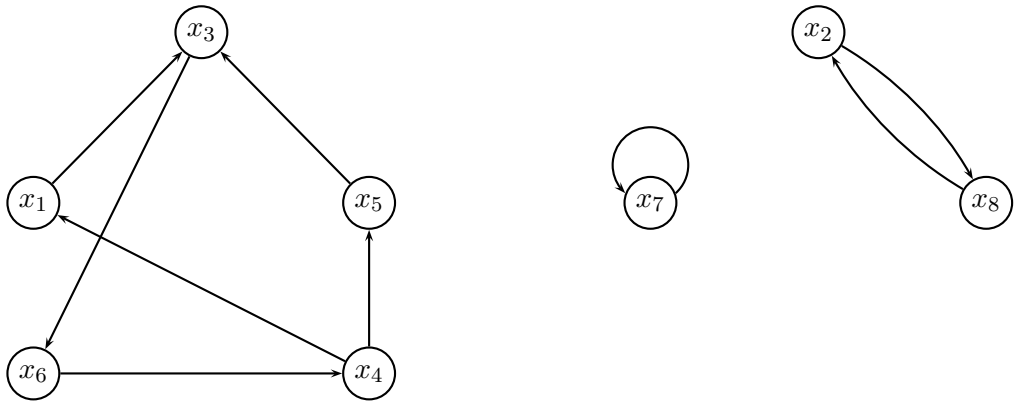
Considérons ci-dessous



Les classes fortement connexes sont

$$\{x_1, x_3, x_4, x_5, x_6\}, \quad \{x_2, x_8\}, \quad \{x_7\}.$$

Les composantes fortement connexes sont :



Définition 1.1.14 *Un graphe orienté $G = (X, U)$ est quasi-fortement connexe si $\forall\, x, y \in X$ il existe un sommet $z(x, y) \in X$ d'où partent à la fois un chemin allant à x et un chemin allant à y.*

Il est immédiat que

Proposition 1.1.9 *Tout graphe fortement connexe est quasi-fortement connexe.*

Tout graphe quasi-fortement connexe est connexe.

Définition 1.1.15 *Soit $G = (X, U)$ un graphe orienté. On appelle graphe réduit de G, le graphe quotient $G_r = G/\mathcal{T}$ du graphe G par la relation d'équivalence $\mathcal{T}$. Les sommets de G_r sont les composantes fortement connexes et il existe un arc de G_i à G_j si et seulement s'il existe au moins un arc allant d'un sommet de G_i à un sommet de G_j dans G.*

1.1.4 Graphe sans circuit

Soient $G = (X, U)$ un graphe orienté, s et p deux sommets de G. On dit que s est un *sommet source* ou simplement *source* de G si $d^-(s) = 0$. Le sommet p est appelé *sommet puits* ou simplement *puits* de G si $d^+(p) = 0$.

Proposition 1.1.10 *Si G est sans circuit alors il admet un sommet source et un sommet puits.*

Preuve : Exercice (Ind. faire une preuve par construction). □

Proposition 1.1.11 *Si G est strict alors G est sans circuit si et seulement si*

$$\begin{cases} \bullet \text{ il existe un sommet } s \text{ tel que } \ d^-(s) = 0, \\ \bullet \text{ pour tout } s \in X \text{ tel que } \ d^-(s) = 0, \text{ le sous graphe } \ G - \{s\} \text{ est sans circuit.} \end{cases}$$

Preuve : Soit $G = (X, U)$ une graphe orienté strict.

Si G est sans circuit alors d'après la proposition 1.1.10, il existe $s \in X$ tel que $d^-(s) = 0$ et comme tout sous graphe d'un graphe sans circuit est un graphe sans circuit on deduit que pour toute source s le graphe $G - \{s\}$ est sans circuit.

Réciproquement supposons que s est une source de G et donc le sous graphe $G - \{s\}$ est sans circuit. Si G contenait un circuit alors ce circuit contiendrait s et $d^-(s) > 0$; ce qui contredit le fait que s est une source de G. □

Proposition 1.1.12 *Le graphe G_r est sans circuit.*

Proposition 1.1.13 *Si $G = (X, U)$ est orienté strict tel que pour tout $x \in X$ on a : $d^-(x) = d^+(x)$ alors G est fortement connexe si et seulement si G est connexe.*

Preuve : Soit $G = (X, U)$ un graphe orienté strict tel que pour tout $x \in X$ on a : $d^-(x) = d^+(x)$.

Si G est fortement connexe il est clair que G est connexe.

Reciproquement supposons que G est connexe mais n'est pas fortement connexe. Donc G admet $q > 1$ composantes fortement connexes, notés $G_1, \ldots, G_q$. D'après la proposition 1.1.12, le graphe réduit G_r est sans circuit. Il existe, d'après la proposition 1.1.10, un sommet G_p (avec $p \in \{1, \ldots, q\}$) dans G_r tel que $d^+(G_p) = 0$. Tout arc de G dont l'extrémité initiale est dans G_p a aussi son extrémité finale dans G_p. D'après l'hypothèse, dans le graphe $G_p = (X_p, U_p)$ on a :

$$\sum_{x \in X_p} d^-(x) = \sum_{x \in X_p} d^+(x),$$

mais comme G est connexe il existe une composante fortement connexe $G_i = (X_i, U_i)$, un sommet $y \in X_i$ et un sommet $x \in X_p$ tels que $(y, x) \in U$. Ce qui donne dans G :

$$\sum_{x \in X_p} d^-(x) > \sum_{x \in X_p} d^+(x),$$

ce qui contredit l'hypothèse sur les degrés. □

1.2 Graphe non orienté

Dans l'étude de certaines propriétés des graphes, il arrive que l'orientation des arcs, c'est-à-dire la distinction entre extrémité initiale et extrémité terminale ne joue aucun rôle. On s'intéresse simplement à l'existence ou non d'un arc entre deux sommets sans en préciser l'ordre. On parle alors de graphe non orienté ou simplement de *graphe*.

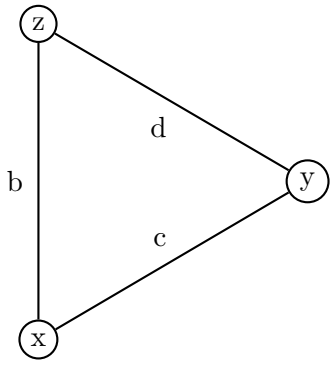
1.2.1 Définitions

Définition 1.2.1 *Un graphe non orienté G est défini par la donnée d'un ensemble fini X dont les éléments sont appelés des sommets ou noeuds et d'un ensemble E dont les éléments sont des sous-ensembles à 1 ou 2 éléments de X appelés arêtes. On le note $G = (X, E)$. Le nombre n de sommets de G est appelé l'ordre de G et le nombre m des arêtes de G est appelée la taille de G.*

Les arêtes de la forme $u = \{x\} \in E$ sont appelées boucles.

Exemple

Soit $G = (X, E)$ le graphe défini par : $S = \{x, y, z\}$ et $E = \{a, b, c, d\}$ où $b = \{z, x\}$, $c = \{x, y\}$ et $d = \{z, y\}$.



A un graphe orienté, on associe un graphe non orienté en "laissant tomber l'orientation". De même à un graphe non orienté, on associe un graphe orienté, soit en introduisant pour toute arête les deux arcs qui lui correspondent, soit en choisissant un seul des deux arcs.

Définition 1.2.2 *Etant donnée une arête $u = \{x, y\}$ d'un graphe non orienté $G = (X, E)$, on dit que les sommets x et y sont les extrémités de u. On dit également que les sommets x et y sont adjacents et que l'arête u est incidente à x et y. On appelle voisinage de x, l'ensemble des sommets adjacents à x. On le note $\mathcal{V}(x)$.*

Un sommet qui n'est adjacent à aucun autre sommet est dit isolé.

Définition 1.2.3 *Un graphe non orienté G est dit simple s'il est sans boucle.*

Définition 1.2.4 *Un graphe non orienté G est dit discret si sa taille est nulle. C'est-à-dire tous ses sommets sont isolés.*

Définition 1.2.5 *Un graphe non orienté G est dit complet si deux sommets quelconques sont adjacents.*

Définition 1.2.6 *Deux arêtes sont adjacentes si elles sont incidentes à au moins une extrémité commune.*
Le degré du sommet x, noté $d_G(x)$ ou plus simplement $d(x)$ est le nombre d'arêtes ayant x pour extréminté.

Définition 1.2.7 *Etant donné $Y \subset X$, on appelle cocycle de G l'ensemble $w(Y)$ des arêtes ayant une extrémité dans Y et l'autre extrémité dans $X - Y$.*

Les résultats de la proposition 1.1.1 et du corollaire 1.1.1 restent valables.

Proposition 1.2.1 *Soit $G = (X, E)$ un graphe non orienté simple d'ordre n et de taille m. Alors on a $m \leq \frac{n(n-1)}{2}$.*

Proposition 1.2.2 *Soit $G = (X, E)$ un graphe non orienté simple d'ordre n et de taille m. Alors G est complet si et seulement si $m = \frac{n(n-1)}{2}$.*

Proposition 1.2.3 *Soit $G = (X, E)$ un graphe non orienté simple d'ordre n et de taille m. Alors il existe au moins deux sommets ayant même degré.*

1.2.2 Chaîne, cycle

Une *Chaîne* de longueur k est une suite $\mu = e_1, e_2, \ldots, e_k$ de k arêtes telle que deux arêtes consécutives sont adjacentes. On notera $\mu = x_1 e_1 x_2 \ldots x_k e_k x_{k+1}$, si l'on veut préciser les sommets rencontrés. Les sommets x_1 et x_k sont appelés les *extrémités de la chaîne* μ. On dit que la chaîne μ rélie les sommets x_1 et x_k.

Une *chaîne élémentaire* est une chaîne telle qu'en la parcourant, on ne rencontre pas deux fois le même sommet.
Une chaîne est *simple* si la séquence d'arêtes qui la constitue ne comporte pas plusieurs fois le même élément.
La proposition 1.1.6 est valable pour les chaînes.

Un *cycle* est une chaîne dont les extrémités coïncident.

Un *cycle élémentaire* est un cycle minimal pour l'inclusion i.e ne contenant strictement aucun autre cycle.
Les notions de demi-degré, chemin, connexité forte et circuit sont essentiellement orientées. Elles n'ont donc pas de correspondant dans l'univers des graphes non orientés. Par contre celle d'isomorphie, de connexité de sous-graphe et graphes partiel s'étendent de manière directe au cas des graphes non orientés.

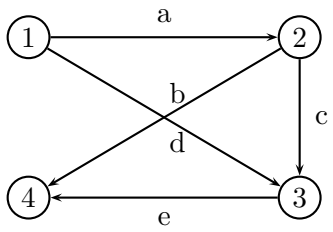
1.3 Autres représentations des graphes

1.3.1 Cas des graphes orientés

Définition 1.3.1 (Matrice d'incidence (sommets-arcs)) *Soit $G = (X, U)$ un graphe orienté tel que $X = \{x_1, \cdots, x_n\}$ et $U = \{u_1, \cdots, u_m\}$. La matrice d'incidence sommets-arcs de G est la $n \times m$ matrice $A = (a_{ij})$ à coefficients entiers : 0, -1 et 1 telle que*

$$a_{ij} = \begin{cases} 1 & \text{si } x_i \text{ est origine de l'arc } u_j \\ -1 & \text{si } x_i \text{ est extrémité de l'arc } u_j \\ 0 & \text{dans les autre cas} \end{cases}$$

Exemple



On obtient la matrice d'incidence :

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

Définition 1.3.2 (Matrice d'adjacence (sommets-sommets)) *Soit $G = (X, U)$ un graphe orienté tel que $X = \{x_1, \cdots, x_n\}$. La matrice d'adjacence sommets-sommets ou matrice booléenne de G est la matrice carrée d'ordre n $A = (a_{ij})$ à coefficients entiers : 0 et 1 telle que pour tout x_i et x_j dans X on a :*

$$a_{ij} = 1 \text{ si et seulement si } (x_i, x_j) \in U.$$

La matrice d'adjacence de l'exemple précédent est

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

PROPRIETE

```
Soit M la matrice
d'adjacence de G
M^n=(Aij)
Aij répresente le nombre
de chaine de longueur n
entre les sommets i et j
```

Tableaux α et β

On considère ici que les sommets sont rangés et désignés par leur rang. Ainsi on pose $X = \{1, 2, \ldots, n\}$. Les tableaux $\alpha()$ et $\beta()$ sont respectivement des matrices unilignes de dimension $n+1$ et m. Pour chaque sommet i, la liste des sommets successeurs de i est contenue dans le tableau $\beta()$ à partir de la case numéro $\alpha(i)$. Et donc l'ensemble des successeurs de i est contenu entre les cases $\alpha(i)$ et $\alpha(i+1) - 1$ du tableau $\beta()$.

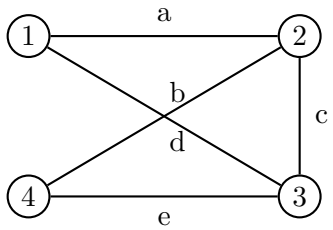
$$\begin{cases} \alpha(1) & = & 1 \\ \alpha(i+1) & = & d^+(i) \quad + \quad \alpha(i) \end{cases}$$

1.3.2 Cas des graphes non orientés

Définition 1.3.3 (Matrice d'incidence (sommets-arêtes)) *Soit $G = (X, U)$ un graphe non orienté tel que $X = \{x_1, \cdots, x_n\}$ et $E = \{e_1, \cdots, e_m\}$. La matrice d'incidence sommets-arêtes de G est la $n \times m$ matrice $A = (a_{ij})$ à coefficients entiers : 0 et 1 telle que*

$$a_{ij} = \begin{cases} 1 & \text{si } x_i \text{ est une extrémité l'arête } u_j \\ 0 & \text{dans les autres cas} \end{cases}$$

Exemple



On obtient la matrice d'incidence :

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

Définition 1.3.4 (Matrice d'adjacence (sommets-sommets)) *La matrice d'adjacence sommets-somm ou matrice booléenne d'un graphe non orienté $G = (X, E)$ avec $X = \{x_1, \cdots, x_n\}$ est la matrice $A = (a_{ij})$ à coefficients entiers : 0 et 1 telle que pour tout x_i et x_j dans X on a :*

$$a_{ij} = 1 \text{ si et seulement si } \{x_i, x_j\} \in E.$$

La matrice d'adjacence de l'exemple précédent est

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

Tableaux α et β

On considère ici que les sommets rangés et sont désignés par leur rang. Ainsi on pose $S = \{1, 2, \ldots, n\}$. Les tableaux $\alpha()$ et $\beta()$ sont respectivement des matrices unilignes de dimension $n+1$ et $2m$. Pour chaque sommet i, la liste des sommets successeurs de i est contenue dans le tableau $\beta()$ à partir de la case numéro $\alpha(i)$. Et donc l'ensemble des successeurs de i est contenu entre les cases $\alpha(i)$ et $\alpha(i+1) - 1$ du tableau $\beta()$.

$$\begin{cases} \alpha(1) & = & 1 & & \\ \alpha(i+1) & = & d(i) & + & \alpha(i) \end{cases}$$

Chapitre 2

Arbres et arborescences

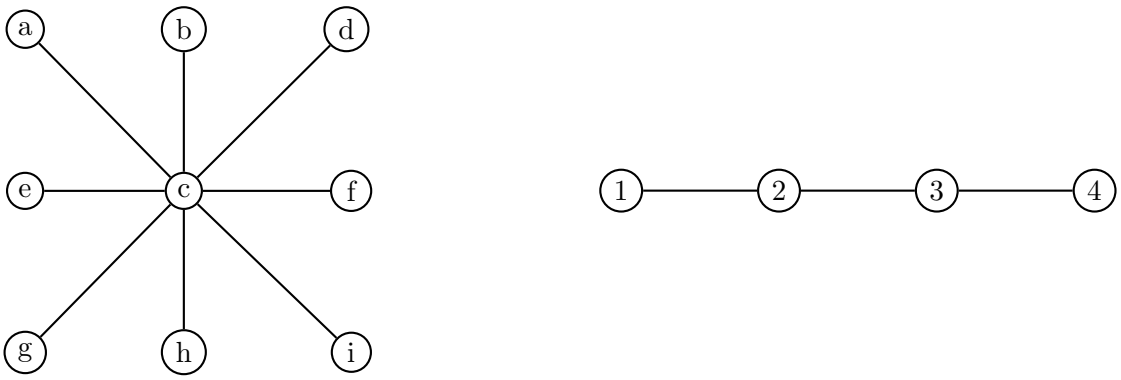
La notion d'arbre est fondamentale en recherche opérationnelle. Elle est utilisée dans de treès nombreux domaines notamment l'informatique, les sciences sociales, l'optimisation combinatoire et la théorie des réseaux électriques.

2.1 Arbres

2.1.1 Définitions et propriétés des arbres

Un *arbre* est un graphe simple connexe sans cycles. Un graphe simple sans cycles qui n'est pas nécessairement connexe est appelé une *forêt* (chaque composante connexe est un arbre). Une chaîne élémentaire est un arbre.

Exemples d'arbre



Proposition 2.1.1 *Un arbre tel que $n \geq 2$ possède au moins deux sommets pendants (ie de degré 1).*

Preuve : Il suffit de considérer une chaîne $\mu = \{s_0, e_1, s_1, \ldots, s_{k-1}, e_k, s_k\}$ de longueur la plus grande possible (maximale) où $k \geq 1$ car $n \geq 2$. On montre par l'absurde que s_0 et s_k sont des sommets pendants. Par exemple supposons que s_0 n'est pas pendant alors il existerait une arête $f \neq e_1$ incidente à s_0 et soit y l'autre sommet incident à f. Si $y \in \{s_0, s_1, \ldots, s_k\}$ alors l'arbre contiendrait un cycle ce qui est absurde ; sinon la chaîne $\sigma = \{y, f, s_0, e_1, s_1, \ldots, s_{k-1}, e_k, s_k\}$ contiendrait μ ce qui est également absurde. □

Proposition 2.1.2 *Si G est un arbre alors $m = n - 1$.*

Preuve : On fait une preuve par récurrence sur n. Il est facile d'établir le résultat pour $n = 1$ car un arbre ne peut contenir une boucle. On suppose que $n > 1$. D'après la proposition 2.1.1, il existe un

sommet pendant s. Il est facile de voir que le graphe $G' = G - \{s\}$ est un arbre et donc par hypothèse de récurrence on a : $m_{G'} = n_{G'} - 1$. Comme $m_{G'} = m_G - 1$ et $n_{G'} = n_G - 1$ on obtient $m_G = n_G - 1$. □

Proposition 2.1.3 *Dans un arbre, deux sommets quelconques sont reliés par une chaîne élémentaire unique.*

Preuve : Soient x et y deux sommets d'un arbre. Il existe une chaîne élémentaire qui relie x et y car le graphe est connexe et supposons qu'il existe une autre chaîne distincte de la précédente qui les relie. En concaténant les deux chaînes on obtient un cycle ce qui contredit le fait que le graphe est acyclique.□

Proposition 2.1.4 *Une arête $e \in E$ est un isthme de $G = (S, E)$ si et seulement si e n'appartient à aucun cycle de G*

Preuve : Soit $G = (S, E)$ un graphe connexe (cela suffit). Si $e = \{s, t\} \in E$ n'est pas un isthme alors il existe dans $G - \{e\}$ une chaîne μ reliant s et t les extrémités de e et donc en concaténant μ et e on obtient un cycle contenant e. Donc si e n'appartient à aucun cycle de G alors e est un isthme. Inversement supposons que e appartient à un cycle $c = (s_0, e, s_1, e_2, s_2, \ldots, s_{k-1}, e_k, s_0)$ et x et y deux sommets de $G - \{e\}$. Comme G est connexe il existe une chaîne $\sigma = \{x = t_0, f_1, t_1, f_2, \ldots, t_{k-1}, f_k, t_k = y\}$ qui relie x et y. Si σ est $G - \{e\}$ alors $G - \{e\}$ est connexe et e n'est pas un isthme. Sinon $e = f_i$ avec $i = 1, \ldots, k$ et posons $s_0 = t_{i-1}$ et $s_1 = t_i$. Il suffit de considérer la chaîne obtenue par la concaténation des trois chaînes suivantes : $\{x = t_0, f_1, \ldots, t_{i-1}\}$, $\{s_1, e_2, \ldots, e_k, s_0\}$ et $\{t_i, f_{i+1}, \ldots, f_k, t_k = y\}$. Elle est dans $G - \{e\}$ et donc $G - \{e\}$ est connexe. Donc si e est un isthme alors e n'appartient à aucun cycle de G. □

Corollaire 2.1.1 *Toute arête d'un arbre est un isthme.*

Théorème 2.1.1 *Soit G un graphe. Les assertions suivantes sont équivalentes :*

1. *G est un arbre.*
2. *G est acyclique et $m = n - 1$.*
3. *G est connexe et $m = n - 1$.*
4. *G est acyclique et si on ajoute une arête, on crée un cycle et un seul.*
5. *G est connexe et toute arête est un isthme.*
6. *Deux sommets quelconques de G sont reliés par une et une seule chaîne élémentaire.*

Preuve : (En exercice)

Proposition 2.1.5 *Une forêt d'ordre n et ayant p arbres possède $n - p$ arêtes.*

2.1.2 Problème de l'arbre couvrant

A) Position du problème

Soit $G = (S, E)$ un graphe simple. Un *arbre de G* ou encore *arbre couvrant de G* est un graphe partiel connexe et sans cycle de G. Une *forêt de G* ou encore *forêt couvrante de G* est un graphe partiel sans cycle de G (non nécessairement connexe).

Proposition 2.1.6 *Tout graphe connexe a un arbre couvrant.*

Preuve : Il suffit de supprimer toutes les arêtes qui ne sont pas des isthmes. □

Corollaire 2.1.2 *Si G est connexe alors $m \geq n - 1$ et on a l'égalité si et seulement si G est un arbre.*

Preuve : G admet un arbre T couvrant, donc $m \geq m_T = n_T - 1 = n - 1$. comme T est un graphe partiel de G le cas de l'égalité n'est possible que lorsque $G = T$. □

Proposition 2.1.7 *Un graphe partiel d'un graphe connexe G est un arbre couvrant de G si et seulement s'il est connexe et minimal par rapport à la suppression d'arêtes.*

Preuve : Soit H un graphe partiel d'un graphe connexe G.
Si H est un arbre couvrant de G alors H est connexe et toute arête e de H y est un isthme donc $H - \{e\}$ n'est pas connexe. Réciproquement supposons que H est connexe et minimal par rapport à la suppression d'arêtes. Donc la suppression d'une arête quelconque e de H augmente le nombre de composante connexe de H et l'arête e est un isthme. Donc e n'appartient à aucun cycle et par conséquence H est acyclique et donc H est un arbre couvrant. □

Proposition 2.1.8 *Un graphe partiel d'un graphe connexe G est un arbre couvrant de G si et seulement s'il est acyclique et maximal par rapport à l'ajout d'arêtes.*

Preuve : Soit H un graphe partiel d'un graphe connexe G.
Si H est un arbre couvrant alors H est acyclique et soit e une arête quelconque de G n'appartenant pas à H. Il existe une chaîne $\mu = \{x, e_1, x_1, \ldots, x_{k-1}, e_k, y\}$ reliant les extrémités x et y de e. En concaténant μ et e on obtient le cycle $C = (x, e_1, x_1, \ldots, x_{k-1}, e_k, y, e, x)$; ce qui établit que la condition est nécessaire. Pour montrer qu'elle est suffisante supposons que H est acyclique et maximal par rapport à l'ajout d'arêtes. soient x et y deux sommets quelconques de H, il existe dans G une chaîne unique $\mu = \{x_0 = x, e_1, x_1, \ldots, x_{k-1}, e_k, x_k = y\}$. Si toutes les arêtes e_i avec $i = 1, 2, \ldots, k$ sont dans H alors H est connexe. Sinon on construit à partir de μ une chaîne dans H réliant x et y de la façon suivante : Pour toute arête e_i qui n'est pas dans H, il existe dans H une chaîne unique $\sigma_i = \{x_{i-1}, e_{i_1}, x_{i_1}, \ldots, e_{i_k}, x_i\}$ qui rélie x_{i-1} et x_i. On remplace x_{i-1}, e_i, x_i dans μ par σ_i. □

Définition 2.1.1 (Graphe valué) *Soit $G = (S, E)$ un graphe (orienté ou non). On associe à chaque arc (ou arête) $e \in E$ un réel $l(e)$ appelé poids ou longueur de e. L'application l ainsi définie :*

$$\begin{array}{rccc} l: & E & \to & \mathbb{R} \\ & e & \mapsto & l(e) \end{array}$$

est appelée une valuation du graphe G, on note $G = (S, E, l)$ et on dit que le graphe est valué. On l'appelle aussi réseau.

Définition 2.1.2 *Soit $G = (S; E)$ un graphe simple valué et $E' \subset E$. On appelle poids du graphe partiel engendré par E', le nombre : $c(E') = \sum_{e \in E'} c(e)$.*

Soit $G = (S, E, l)$ un graphe valué. Le problème de l'arbre couvrant minimum (resp. maximum) de G est la détermination d'un arbre couvrant H de G de poids le plus petit (resp. grand) possible. On présente ici deux algorithmes : celui de Kruskal et celui de Prim.

B) Méthode de Kruskal

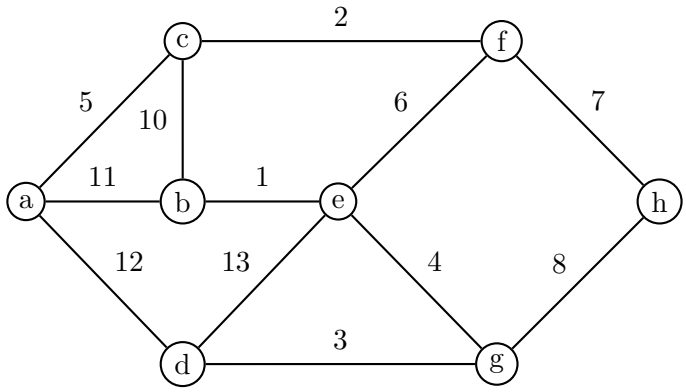
La méthode de Kruskal construit un arbre couvrant de poids minimal en ajoutant à chaque itération une arête de plus petit poids n'ajoutant pas de cycle.

Algorithme de Kruskal

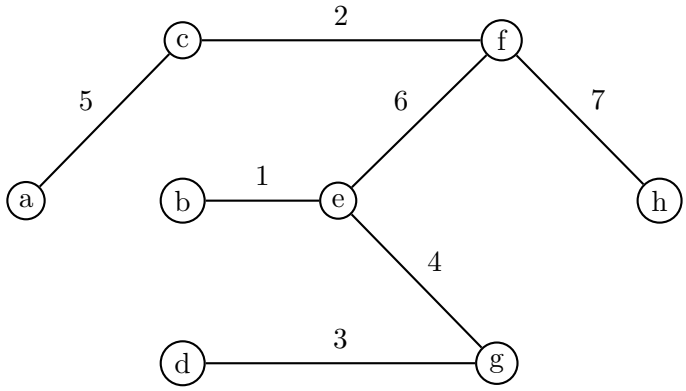
Initialisation $\mathcal{F} = E$ et $E_H = \emptyset$
Tant que $|E_H| < n - 1$ Faire
  Sélectionner $e \in \mathcal{F}$ tel que $l(e)$ soit minimum.
    $\mathcal{F} = \mathcal{F} - \{e\}$
  Si $G[E_H \cup \{e\}]$ est acyclique alors $E_H = E_H \cup \{e\}$
Fin Tant que

Exemple

On considère le graphe connexe suivant :



En appliquant Kruskal on obtient l'arbre couvrant minimum suivant :



Le poids de l'arbre est $1+2+3+4+5+6+7=28$.
Une variante de l'algorithme de Kruskal est la suivante.

Variante de l'Algorithme de Kruskal

Initialisation $\mathcal{F}=E$ et $E_H=E$

Tant que $|E_H|>n-1$ Faire

Sélectionner $e\in\mathcal{F}$ tel que $l(e)$ soit maximum.
$\mathcal{F}=\mathcal{F}-\{e\}$

Si $G[E_H-\{e\}]$ est connexe alors $E_H=E_H-\{e\}$

Fin Tant que

C) Méthode de Prim

Soit $G=(X,E,l)$ un graphe connexe valué. La méthode consiste à construire de proche en proche un arbre couvrant minimum $H=(X,U')$. Pour ce faire on fixe un sommet quelconque s_0. Dans cet algorithme T est l'ensemble des sommets visités et $\omega(T)$ est le cocycle de T.

Algorithme

Initialisation $T=\{s_0\}$, $\theta=\omega(s_0)$ et $U'=\emptyset$;

Tant que $S-T\neq\emptyset$ Faire

Selectionner $e\in\theta$ tel que $l(e)$ soit minimum.
Soit s_i l'extrémité de e qui n'est pas dans T ;
$U'=U'\cup\{e\}$, $T=T\cup\{s_i\}$ et $\theta=\omega(T)=(\theta\cup\omega(s_i))-(\theta\cap\omega(s_i))$

Fin Tant que

Une solution du problème de l'arbre couvrant de poids maximal peut s'obtenir en utilisant les mêmes algorithmes après avoir au préalable multiplié les poids par -1.

On peut aussi modifier très légèrement ces algorithmes pour obtenir un arbre couvrant de poids maximal. Il suffit de remplacer "minimum" par "maximum" dans les algorithmes ci-dessus. Mais dans la variante de Kruskal, il remplacer "maximum" par "minimum".

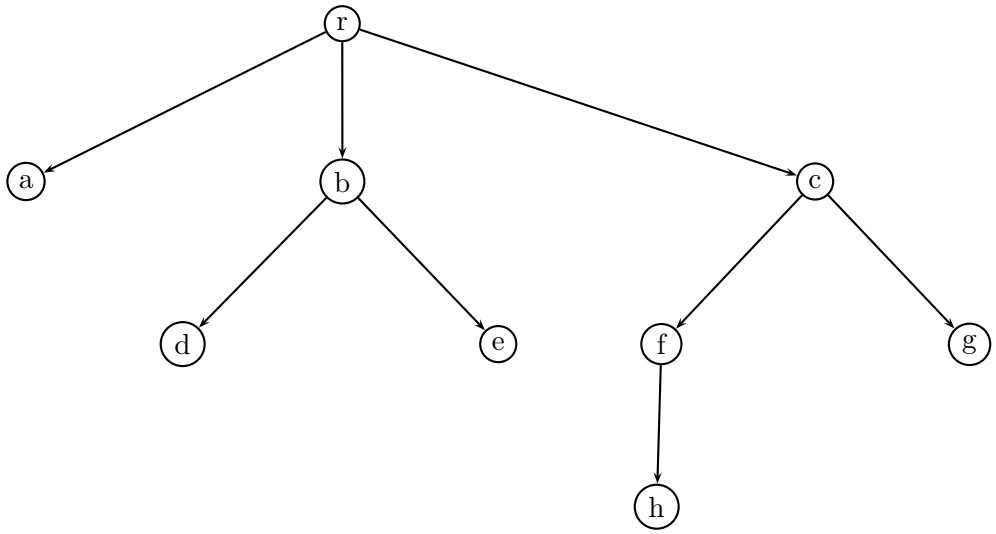
2.2 Arborescences

2.2.1 Définitions

Soient $G = (X, U)$ un graphe orienté et $r \in X$. On dit que r est une *racine* de G si tout sommet de G peut être atteint par un chemin d'origne r. Une *arborescence* est un graphe orienté a une seule racine et sans cycle.

On dit que G possède la *condition des demi-degrés intérieurs* lorsque $d^-(x) = 1$ pour tout sommet x de G sauf pour un sommet qu'on note r, pour lequel $d^-(r) = 0$.

Exemple



Les sommets puits a, d, e, h, g sont appelés *feuilles*. d et e sont *frères* et *fils* de b. les *descendants* de c sont f, g, h.

2.2.2 Caractérisation des arborescences

Proposition 2.2.1 *Soit $G = (X, U)$ un graphe orienté. Les conditions suivantes sont équivalentes*

1. *G est une arborescence.*
2. *G a une racine et est acyclique.*
3. *G a une racine et $m = n - 1$.*
4. *Il existe un sommet $r \in X$ tel que pour tout sommet $x \in X$, il existe un chemin unique allant de r à x.*
5. *G est connexe et possède la condition des demi-degrés intérieurs.*
6. *G est acyclique et possède la condition des demi-degrés intérieurs.*
7. *G est sans circuits et possède la condition des demi-degrés intérieurs.*

Preuve : Exercice. □

Chapitre 3

Chemins optimaux

3.1 Position du problème de cheminement

3.1.1 Plus court chemin et distance

Définition 3.1.1 *Soient $G = (S, A, l)$ un graphe valué et $\mu(s_0, s_k)$ un chemin allant de s_0 à s_k. On appelle longueur du chemin ou bien poids du chemin $\mu(s_0, s_k)$ et on note $l(\mu(s_0, s_k))$ la somme des poids des arcs de $\mu(s_0, s_k)$.*

Le problème du plus court chemin entre deux sommets x et y de $G = (S, A, l)$ est de déterminer un chemin μ entre x et y qui soit de longueur minimale.

De façon analogue le problème du plus long chemin entre x et y de $G = (S, A, l)$ est de déterminer un chemin μ entre x et y qui soit de longueur maximale.

Remarquer que si la valuation l est constante égale à 1 alors la notion définie ici coïncide avec la notion de longueur d'un chemin comme nombre de ses arcs. On parle de longueur au sens des arcs.

La recherche d'un plus long chemin est analogue à la recherche d'un plus court chemin. En effet, rechercher un plus long chemin entre deux sommets x et y dans $G = (S, A, l)$ est équivalent à la recherche du plus court chemin entre x et y dans $G = (S, A, -l)$. On peut donc sans perdre de généralités se restreindre au problème du plus court chemin.

Le problème de plus court chemin a de nombreuses applications pratiques car la longueur $l(u)$ d'un arc u peut s'interpréter comme un coût de transport sur l'arc, comme les dépenses de construction de l'arc, ou comme le temps nécessaire pour parcourir l'arc u.

Les algorithmes de résolutions du problème de plus court chemin sont différents selon les propriétés du graphe et le problème considéré.

Parmi les propriétés du graphe, on distingue les cas suivants :

1. on a $l(u) \geq 0$ pour tout $u \in A$,
2. on a $l(u) = 1$ pour tout $u \in A$,
3. $l(u)$ est quelconque,
4. le graphe G est sans circuit.

Les problèmes considérés sont :

1. Recherche d'un plus court chemin d'un sommet à un autre (*problème de 2 sommets*).
2. Recherche d'un plus court chemin d'un sommet à tous les autres (*problème avec un sommet origine unique*).
3. Recherche d'un plus court chemin entre tous les couples de sommets (*problème de tous les couples de sommets*).

On remarque que les problèmes 1) et 2) sont identiques. Dans la suite on ne s'intéressera qu'aux problèmes 2) et 3).

3.1.2 Condition d'optimalité

Définition 3.1.2 *Un circuit de longueur strictement négative est appelé circuit absorbant.*

Proposition 3.1.1 *Soient s et t deux sommets de G. Pour qu'il existe un plus court chemin allant de s à t il faut et il suffit que tout chemin allant de s à t ne contienne pas de circuit absorbant.*

Preuve : Soit $\mathcal{C}$ l'ensemble des chemins de s à t. Si $\mu \in \mathcal{C}$ contient un circuit ω alors on note μ' le chemin de s à t associé à μ mais n'empruntant pas le circuit ω et on a : $l(\mu) = l(\mu') + l(\omega)$.

- S'il existe un chemin $\mu \in \mathcal{C}$ contenant un circuit absorbant alors il n'existe pas de plus court chemin de s à t. Il suffit en effet de considérer le chemin obtenu à partir de μ en passant un assez grand nombre de fois par le circuit absorbant.
- Si tout chemin $\mu \in \mathcal{C}$ ne contient pas de circuit absorbant alors $l(\omega) \geq 0$, $l(\mu') \leq l(\mu)$ et on peut se restreindre aux chemins élémentaires. Comme ils forment un ensemble fini, il existera un plus court chemin. □

Soit $G = (S, A, l)$ un graphe orienté, valué et sans circuits absorbants. On appelle *distance* de s à t, deux sommets de G et on note $d(s,t)$ la plus petite longueur des chemins de s à t dans le graphe G.
Un *plus court chemin* de s à t est un chemin μ tel que $l(\mu) = d(s,t)$.

3.2 Quelques algorithmes

3.2.1 Cas où le graphe est non valué

On considère un graphe $G = (S, A)$ orienté non valué. On peut poser $G = (S, A, l)$ où l est constante égale à 1. On note r le sommet origine donc $d(r,r) = 0$. La distance est bien définie.

Calcul des distances

Le calcul des distances repose sur un algorithme de parcours en largeur du graphe G à partir de r. Soit F une file. On rappelle qu'une file fonctionne selon le principe du *premier entré, premier sorti*. Les opérations élémentaires sont.

- *enfiler(F,s)* : met s dans la file à la queue.
- *tête-file(F)* : retourne le sommet qui est en tête de F sans l'enlever.
- *défiler(F,s)* : enlève le sommet qui est en tête sans le retourner.
- *file-vide(F)* : retourne vrai ou faux selon que F est vide ou non.

Algorithme

a) Initialisation

Pour tout sommet $s \in S$
Atteint(s) :=Faux
Fin Pour

b) Visite de r (sommet origine unique)

Atteint(r) :=Vrai
Enfiler(F,r)
$d(r,r) = 0$
$s := r$

c)

Tant que file-vide(F)=Faux

```
void unweighted( Vertex s )
  {
    Queue<Vertex> q = new Queue<Vertex>( );

    for each Vertex v                  // aucun des nœuds n'est atteint
      v.dist = INFINITY;

    s.dist = 0;                        // distance à s, premier nœud, =0
    q.enqueue( s );                    // s est enfilé

    while( !q.isEmpty( ) )
    {
      Vertex v = q.dequeue( );         // le nœud à traiter est défilé

      for each Vertex w adjacent to v
        if( w.dist == INFINITY )       // on ne met dans la file que les nœuds
        {                              // jamais atteints
          w.dist = v.dist + 1;         // distance mise à jour
          w.path = v;                  // précédent mis à jour
          q.enqueue( w ); // le nœud traité est enfilé
        }
    }
  }
```

Pour tout successeur t de s qui n'est pas encore atteint
$d(r,t) = d(r,s) + 1$
Atteint(t) :=Vrai
Enfiler(F,t)

Fin Pour
Défiler(F,s) s :=tête-file(F)

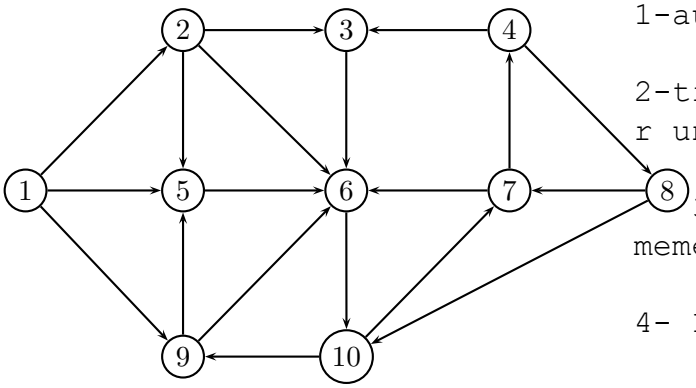
Fin Tant Que

Noter que Atteint( ) est un tableau indexé sur les sommets du graphe G.

Arborescence de plus courts chemins

L'ensemble de plus courts chemins déterminés définit dans le graphe une arborescence de plus courts chemins.

Exemple

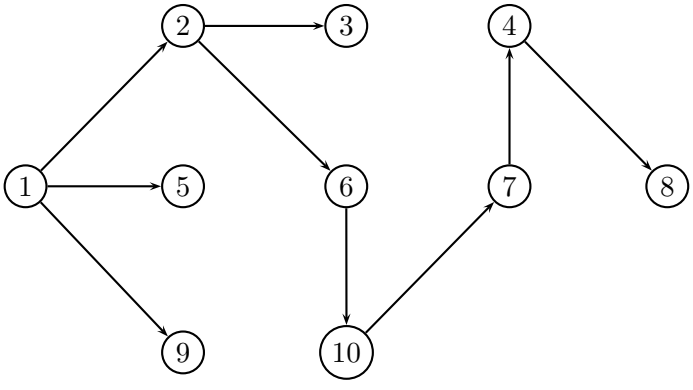



```
le principe est:
1-aucun sommet n'est atteint

2-trouver la racine r (cela fait de
r un sommet atteint et s<-r

3-Mettre s en fin de file(elle
meme qui ne contient rien au départ
      TANTQUE FILE NON VIDE
4- Pour des successeurs t de s:
       d(r,t)<-d(r,s)+1
       => t atteint
       t fin de file
    Enlever le sommet s de la file
```

L'algorithme permet de calculer les distances suivantes :

s	1	2	3	4	5	6	7	8	9	10
$d(r,s)$	0	1	2	5	1	2	4	6	1	3

Il existe plusieurs arborescences, en voici une.



3.2.2 Cas où les longueurs sont positives

Soit $G = (S, A, l)$ un graphe orienté strict où $S = \{s_1, s_2, \ldots, s_n\}$.

Algorithme de Dijkstra

1. **Initialisation**
   $S' = \{s_2, s_3, \ldots, s_n\}$

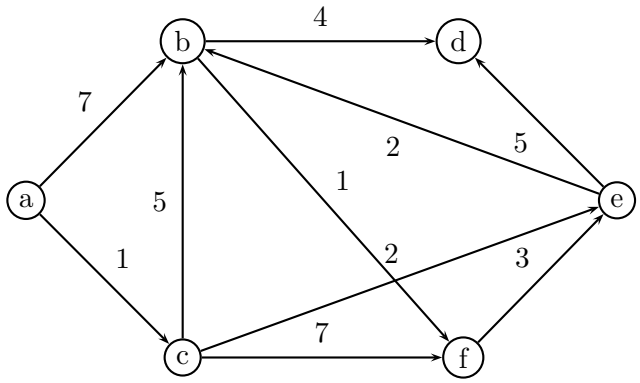
$\lambda(s_1) = 0$
$\lambda(s_i) = l(s_1, s_i)$ si s_i est un successeur de s_1 sinon $\lambda(s_i) = \infty$

2. **Sélection d'un sommet**
   Sélectionner $s_j \in S'$ tel que $\lambda(s_j) = \text{Min}_{s_i \in S'}(\lambda(s_i))$
   Faire $S' \leftarrow S' - \{s_j\}$
   Si S' est vide FIN
   Sinon aller en 3
3. **Calcul des valeurs de λ**
   Faire pour tout s_i à la fois dans $\Gamma(s_j)$ et dans S'
   $\lambda(s_i) \leftarrow \text{Min}(\lambda(s_i), \lambda(s_j) + l(s_j, s_i))$
   et retourner en 2

L'algorithme considère les sommets dans un ordre qui dépend des valeurs $\lambda(s_i)$ appelées labels. A la fin on a $\lambda(s_i) = d(s_1, s_i)$ pour tout $s_i \in S$.

Exemple d'application

On considère le graphe suivant avec a comme sommet source :

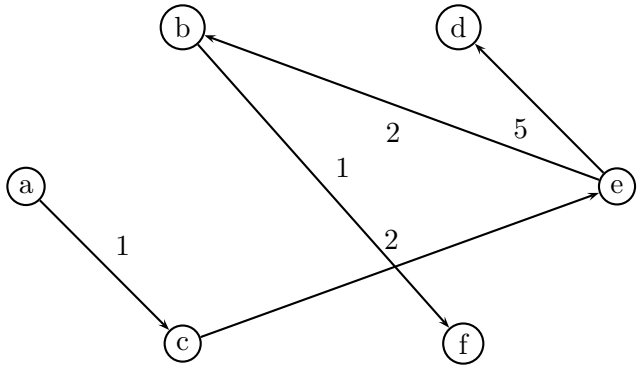


Le principe est:
Après l'initialisation
λ(si)=distance trouvé lors de la precedente ittération

On obtient alors :

Sommets	a	b	c	d	e	f
Initialisation	0	7	1*	∞	∞	∞
Itération 1	⋮	6	⋮	∞	3*	8
Itération 2	⋮	5*	⋮	8	⋮	8
Itération 3	⋮	⋮	⋮	8	⋮	6*
Itération 4	⋮	⋮	⋮	8*	⋮	⋮

On obtient une arborescence de parcours



3.2.3 Cas où les longueurs sont quelconques

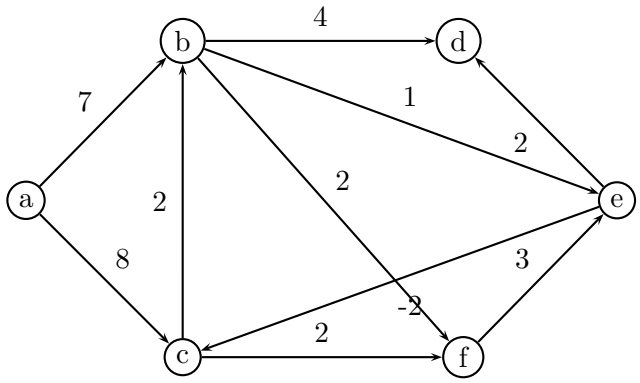
Soit $G = (S, A, l)$ un graphe orienté strict où $S = \{s_1, s_2, \ldots, s_n\}$.

Algorithme de Bellman

1. **Initialisation**
   $\lambda^0(s_1) = 0$
   $\lambda^0(s_i) = \infty$ pour tout $s_i \neq s_1$
   $k = 1$
2. **A l'itération** k
   Faire $\lambda^k(s_1) = 0$
   et $\lambda^k(s_i) = \text{Min}(\lambda^{k-1}(s_i), \text{Min}_{s_j \in \Gamma^{-1}(s_i)}(\lambda^{k-1}(s_j) + l(s_j, s_i)))$ pour tout $s_i \neq s_1$
3. Si $\lambda^k(s_i) = \lambda^{k-1}(s_i)$ pour tout i alors FIN
   Si $k < n - 1$ aller en 2 avec $k \leftarrow k + 1$
   Si $k = n$ alors il existe un circuit absorbant

Exemple d'application

On considère le graphe suivant avec a comme sommet source :



On obtient alors :

Sommets	a	b	c	d	e	f
Initialisation	0	∞	∞	∞	∞	∞
Itération 1	0	7	8	∞	∞	∞
Itération 2	0	7	8	11	8	9
Itération 3	0	7	6	10	8	9
Itération 4	0	7	6	10	8	8
Itération 5	0	7	6	10	8	8

On obtient une arborescence de parcours

3.2.4 Cas d'un graphe sans circuit

A) Détection d'un circuit

Définition 3.2.1 *Soit $G = (X; U)$ un graphe orienté. On appelle ferméture transitive de G, le "plus petit" (en nombre d'arcs) graphe transitif contenant G. C'est-à-dire un graphe $\widehat{G} = (X, \widehat{U})$ ayant G comme graphe partiel et possédant le plus petit nombre d'arcs.*

Schématiquement, on construit $\widehat{U}$ en ajoutant à U un arc (x, z) qui n'appartient pas àU si (x, y) et (yz) appartiennent à U pour au moins un sommet y de G.

On appelle somme booléenne l'opération notée $\oplus$ telle que

1. $0 \oplus 0 = 0$
2. $1 \oplus 0 = 1$
3. $0 \oplus 1 = 1$
4. $1 \oplus 1 = 1$

et le produit booléen est l'opération notée $\otimes$ telle que

1. $0 \otimes 0 = 0$
2. $1 \otimes 0 = 0$
3. $0 \otimes 1 = 0$
4. $1 \otimes 1 = 1$.

La somme et le produit booléens de deux matrices s'obtiennent en substituant $+$ et $\times$ aux opérateurs $\oplus$ et $\otimes$ dans la somme et le produit matriciels usuels.

On montre que

Proposition 3.2.1 *Soit $G = (X; U)$ un graphe orienté d'ordre n avec $X = \{x_1, x_2, \ldots, x_n\}$ et M sa matrice d'adjacence.*

Soit $\widetilde{M} = M \otimes M \otimes \cdots \otimes M$ la $p^{ième}$ puissance booléenne (p un entier naturel non nul) de la matrice d'adjacence M avec $\widetilde{M} = (\widetilde{m}_{ij})$. On a :

$$\widetilde{m}_{ij} = 1 \text{ si et seulement si il existe dans } G \text{ un chemin de cardinalité } p \text{ entre } x_i \text{ et } x_j.$$

B) Ordonnacement par niveaux

Définition 3.2.2 *Soit G un graphe orienté. On appelle dictionnaire des précédents (resp. suivants), le tableau à simple entrée qui à tout sommet x énumère les précédents $P(x)$ (resp. les suivants $S(x)$) de x.*

Définition 3.2.3 *Soit G un graphe orienté sans circuit. On appelle niveau d'un sommet x, la longueur (au sens des arcs) du plus long chemin ayant pour extrémité terminale x.*

Soit G un graphe orienté sans circuit. Pour déterminer les niveaux des sommets de G on peut utiliser soit le dictionnaire des précédents soit la matrice booléenne.

Algorithme utilisant le dictionnaire des précédents

1. On note N_0 l'ensemble des sommets de G qui n'ont pas de précédents, c'est-à-dire de niveau 0.
2. Barrer dans le dictionnaire des précédents les éléments de N_0 partout où ils se trouvent. Les sommets n'ayant plus de précédents sont de niveau 1. Soit N_1 cet ensemble.
3. On réitère cette procédure en augmentant de 1, la valeur des niveaux jusqu'à ce que tous les sommets soient barrés.

Algorithme utilisant la matrice booléenne

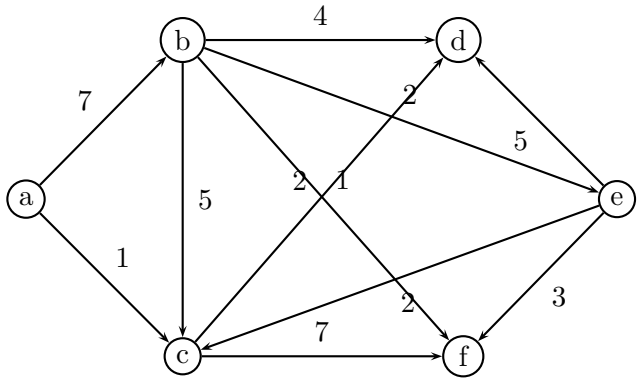
1. L'ensemble N_0 des sommets de niveau 0 est l'ensemble des sommets dont les colonnes sont nulles.
2. Barrer dans la matrice booléenne les colonnes et les lignes des éléments de N_0. Les sommets dont les colonnes sont nulles dans la matrice ainsi obtenue sont de niveau 1. Soit N_1 cet ensemble.

3. On réitère cette procédure en augmentant de 1, la valeur des niveaux jusqu'à ce que toutes les colonnes soient barrées.

Le graphe ordonnancé par niveau du graphe est une représentation graphique plus simple (avec moins d'intersections possibles des arcs). Pour ce faire, on place les sommets d'un même niveau sur une verticale. Les niveaux sont placés de gauche à droite par ordre croissant.

Exemple d'application 1

On considère le graphe suivant :



On a les niveaux suivants :

$$N_0 = \{a\},\ N_1 = \{b\},\ N_2 = \{e\},\ N_3 = \{c\},\ N_4 = \{d, f\}.$$

Dresser le graphe ordonnancé par niveaux.

Exemple d'application 2

On considère le graphe suivant donné par son dictionnaire des précédents : Déterminer les niveaux des sommets et dresser le graphe ordonnancé par niveaux.

Sommets	Précédents
a	-
b	a
c	b
d	a
e	a
f	d
g	f
h	e, c, g
i	h, k
j	a
k	j
m	k
n	m

C) Algorithme de Ford

Soit $G = (S, A, l)$ un graphe orienté strict où $S = \{s_1, s_2, \ldots, s_n\}$. On suppose que $P(s_1) = \emptyset$.

1. **Initialisation** Poser $\lambda(s_1) = 0$ et $S' = \{s_1\}$
2. Rechercher un sommet $s_j \notin S'$ tel que $P(s_j) \subset S'$

3. Poser $\lambda(s_j) = \min[\lambda(s_i) + l(s_i, s_j) : s_i \in P(s_j)], S' := S' \cup \{s_j\}$
4. Si $|S'| = n$, FIN
5. Aller à 2

Exemple d'application

Appliquer l'algorithme de Ford aux graphes ci-dessus.

Chapitre 4

Problèmes d'ordonnancement

L'objet d'un problème d'ordonnancement est de faciliter la mise en œuvre et de guider l'exécution d'un ensemble complexe de tâches (programme de recherche ou de production, lancement d'un produit, construction d'un édifice ...). La technique d'analyse la plus connue, appelée méthode PERT (Program Evaluation and Review Technique) a été intriduite aux Etats-Unis en 1958 pour la conduite du programme de rechefche et de construction des fusées Polarix. Cette méthode tient une place dominante par sa simplicité, son efficacité et la variété d'extensions qui ont pu être développées.

En toute généralité, les problèmes d'ordonnancement se posent sous la forme suivnate. Etant donné un objectif qu'on se propose d'atteindre et dont la réalisation suppose l'exécution préalable de multiples tâcjes, soumises à de nombreuses contraintes, déterminer l'ordre et le calendrier d'exécution des diverses tâches.

Le critère d'optimalité peut porter sur le minimisation de la durée et/ou du coût de la réalisation du projet. Nous supposerons que le projet est décomposable en un nombre fini de tâches, caractérisées par leur durée d'exécution (généralement fixe, parfois aléatoire), éventuellement aussi par leur coût d'exécution, et soumises à des contraintes de postériorité strcite (une tâche ne peut commencer que si certaines autres tâches sont complètement terminées). Ce type de problème se nomme problème central de l'ordonnancement. Des cas plus complexes peuvent également être envisagés comme par exemple le cas des contraintes disjonctives, c'est-à-dire le cas où, en plus des contraintes de succession, on impose la réalisation non simultanée de certaines paires de tâches.

En pratique, le travail préliminaire à accomplir sera donc de dresser une liste des différentes opérations à mener que l'on décomposera plus ou moins finement selon la précision souhaitée. Généralement, les tâches sont définies pour que leurs durées d'exécution soient du même ordre de grandeur ; de plus, les contraintes de postériorité entre les tâches doivent pouvoir être établies avec précision. Ce travail important est souvent long et nécessite une collaboration etroite entre les différents acteurs du projet.

La représentation par un graphe d'un problème d'ordonnancement permet une bonne appréhension globale du problème. L'étude de ce graphe conduit à l'identification des tâches prioritaires et la détection des retards ou des dépassements de moyens afin de prendre les mesures correctives nécessaires.

La schématisation d'un problème d'ordonnancement sous forme d'un graphe peut se faire selon deux modes de repprésentation :

- la méthode MPM (Méthode des potentiels Metra) développée en France par la SEMA
- la méthode PERT

4.1 Graphe potentiel-tâches ou graphe MPM

Le graphe potentiel-tâches, aussi appelé graphe MPM est une modélisation d'un projet qui permet une planification dans le temps des différentes tâches : c'est un *ordonnancement*.
Soit un projet décomposé en n tâches élémentaires $1, 2, \ldots, n$. Pour chaque tâche i, il est donné sa durée d_i et les tâches antérieures c'est-à-dire les tâches qui doivent être achévées pour que la tâche i puisse

commencer.

Tâches	Durées	Tâches antérieures
⋮	⋮	⋮
i	d_i	j, k
⋮	⋮	⋮

4.1.1 Définition

Le graphe potentiel-tâches associé au projet est le graphe $G = (S, A, l)$ avec $S = \{\alpha, 1, 2, \ldots, n, \omega\}$ où α désigne le début et ω la fin du projet, considérés comme des tâches fictives de durée nulle. $(i, j) \in A$ si la tâche i est antérieure à la tâche j. La longueur ou le poids de l'arc (i, j) est $l(i, j) = d_i$. Il est clair que G est orienté strict et sans circuits.

4.1.2 Notions de dates

La date au plus tôt t_i d'une tâche i est la date la plus rapprochée (optimiste) à laquelle i peut commencer. On a : $t_\alpha = 0$ et t_ω est la durée minimum du projet.

$$\begin{cases} t_\alpha = 0 \\ t_i = \text{Max}_{j \in \Gamma^{-1}(i)}(t_j + d_j) \quad \text{pour } i \in \{1, \ldots, n, \omega\} \end{cases}$$

La date au plus tard T_i d'une tâche i est la date la plus tardive (pessimiste) à laquelle i doit commencer pour que la durée minimale de réalisation du projet soit respectée c'est-à-dire $T_\omega = t_\omega$. Les dates au plus tard se calculent par un compte à rebours à partir de T_ω et

$$T_i = \text{Min}_{j \in \Gamma(i)}(T_j - d_i) \quad \text{pour } i = n, n-1, \ldots, 2, 1.$$

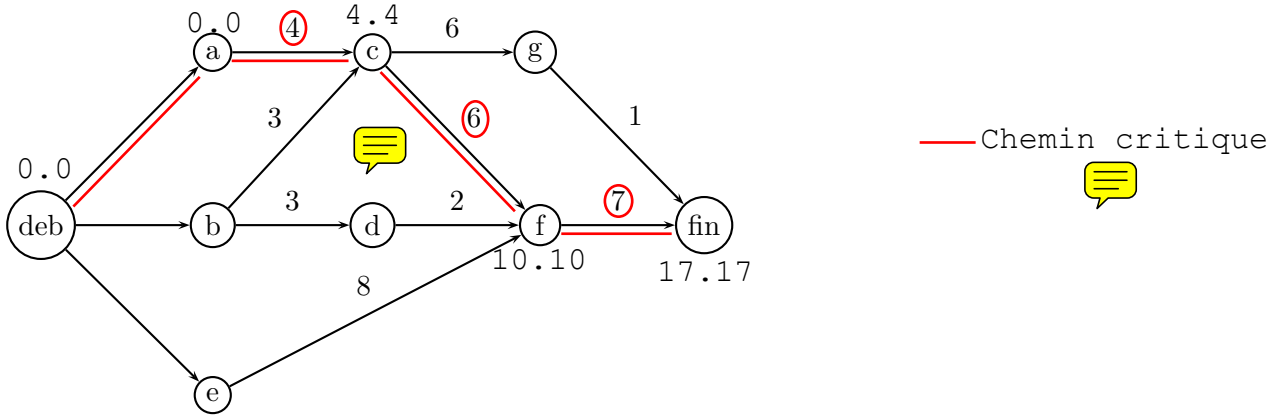
La date de fin au plus tard T'_i d'une tâche i est la date la plus pessimiste de fin d'une tâche et se calcul par la formule suivant :

$$T'_i = \text{Min}_{j \in \Gamma(i)}(t_j - d_i) \quad \text{pour } i = n, n-1, \ldots, 2, 1$$

Exemple : On considère un projet qui comporte 7 tâches : a, b, c, d, e, f, g.

Tâches	a	b	c	d	e	f	g
Durées	4	3	6	2	8	7	1
Tâches antérieures			a,b	b		c,d,e	c

On obtient le graphe potentiel-tâches $G = (S, A, l)$ ci-dessous, où $S = \{deb, a, b, c, d, e, f, g, fin\}$



Tâches	deb	a	b	c	d	e	f	g	fin
Durées	0	4	3	6	2	8	7	1	0
t_i	0	0	0	4	3	0	10	10	17
T_i	0	0	1	4	8	2	10	16	17

4.1.3 Les marges

La marge totale est le délai $MT(i) = T_i - t_i$ pouvant être accordé à une tâche sans repousser la durée minimale du projet. On dit qu'une tâche i est *critique* si $MT(i) = 0$. La succession de taches qui imposent la durée minimale du projet est appelée *chemin critique*.
La marge libre d'une tâche i est le délai $ML(i)$ pouvant être accordé au commencement de la tâche sans modifier la marge totale des tâches qui suivent.

$$ML(i) = \text{Min}_{j \in \Gamma(i)}(t_j - d_i) - t_i$$

ou encore

$$ML(i) = T'_i - t_i$$

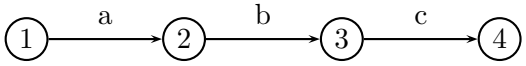
. Contrairement à la marge totale, la marge libre peut être consommée sans aucune conséquence sur les tâches qui suivent dans le projet. La marge libre d'une tâche est toujours inférieure ou égale à sa marge totale.

4.2 Graphe potentiel-étapes ou graphe PERT

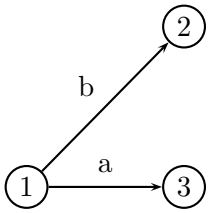
Le graphe potentiel-étapes ou graphe PERT est un graphe orienté valué strict et sans circuits dont les sommets représentent la fin d'une tâche et le début d'une autre et dont les arcs représentent les tâches.

4.2.1 Définitions

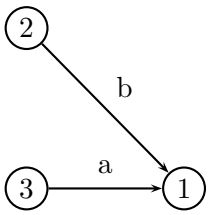
Les successives se suivent dans le temps et sont représentées par un chemin. Par exemple les tâches a, b, c sont successives :



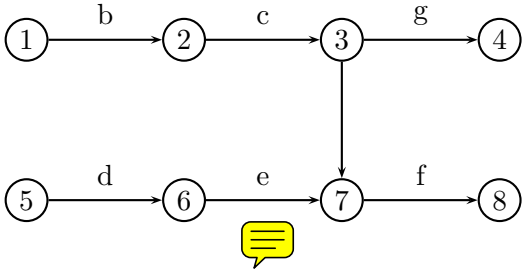
Les tâches simultanées ont le même début d'exécution. Par exemple les tâches a, b sont simultanées :



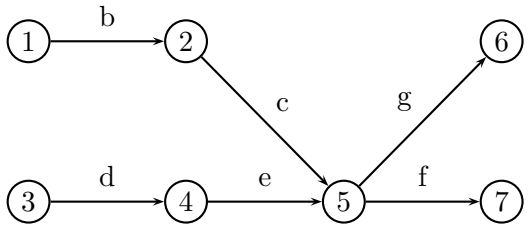
Les tâches convergentes aboutissent à la même étape. Par exemple a, b sont convergentes :



Les contraintes de liaison apparaissent quand une tâche appartenant à une suite de tâches successives ne peut être exécutée avant qu'une autre tâche, n'appartenant pas à cette suite, ne soit achevée.



f ne peut pas commencer avant que c ne soit achevée même si e est terminée. On ne peut pas avoir le graphe suivant :



car cela signifirait que e doit être achevée pour que g commence, ce qui n'est pas nécessaire.

les tâches fictives sont introduites pour éviter que deux tâches soient à la fois simultanées et convergentes.

4.2.2 Construction d'un graphe PERT

Il est clair que le graphe PERT à construire est sans circuits. On peut donc classer les sommets par niveaux. On trace un graphe des niveaux en reliant les tâches successives entre deux niveaux consécutifs. Puis on trace le graphe PERT à partir de ce graphe de mannière à ce que toutes les tâches aboutissent à une étape et que l'antériorité des tâches soit respectée. Noter que toutes les tâches du niveau 1 sont simultanées et celle du dernier niveau sont convergentes.