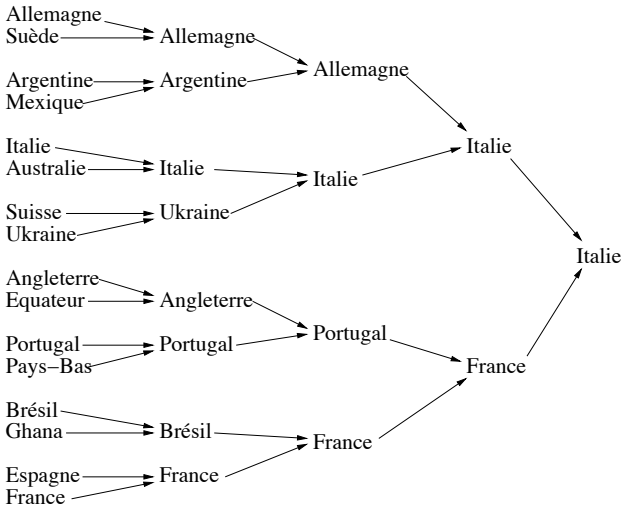
Théorie des graphes

G. Montcouquiol

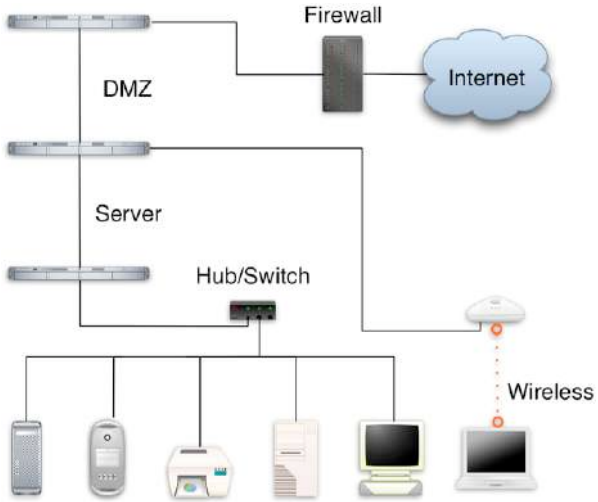
IUT Orsay

2006-2007

Quelques exemples...



Quelques exemples...



Typologie

De manière générale, un graphe, c'est :

- des sommets, et
- des arêtes (ou des arcs) qui relient les sommets.

Différents types de graphes :

oriented : chaque arête (arc) a un sens, habituellement représenté par une petite flèche.

non oriented : les arêtes n'ont pas de sens particuliers.

multigraphe : il peut y avoir plusieurs arcs ayant les mêmes extrémités.

$\Rightarrow$ plusieurs définitions, une pour chaque type de graphe.

Modélisation

Outil mathématique simple, permettant de modéliser des situations très différentes.

Quelques exemples concrets :

1. circulation dans une ville : sommets $\leftrightarrow$ carrefours, arcs $\leftrightarrow$ rues (éventuellement à sens unique)
2. trafic aérien : sommets $\leftrightarrow$ aéroports, arêtes $\leftrightarrow$ vols existants
3. réseau informatique : sommets $\leftrightarrow$ ordinateurs, arêtes $\leftrightarrow$ connexions physiques

Modélisation

Quelques exemples plus abstraits :

4. groupe social : les sommets représentent les membres du groupe, deux personnes sont reliés par une arête si ils se connaissent
5. organisation logistique : les sommets représentent des évènements, deux évènements sont reliés par une arête si ils ne peuvent pas avoir lieu en même temps
6. ordonnancement de projet : les sommets représentent les différentes tâches composant un projet, deux tâches sont reliés par une flèche si la deuxième ne peut pas commencer avant que la première soit terminée

Graphes orientés

Définition

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.*

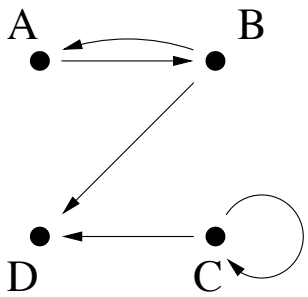
Un arc $(x, y) \in A$ est aussi indifférement noté $x \to y$.

Les boucles sont autorisées, mais pas les arcs parallèles.

Exemple

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

Représentation **sagittale** du graphe :



Terminologie

Si $a = (s_1, s_2) \in A$ est un arc de G, les sommets s_1 et s_2 sont les **extrémités** de a :

s_1 est le **début** (ou **l'origine**) de a et s_2 est la **fin** (ou **l'extrémité finale**) de a.

On dit aussi que s_2 est un **successeur** de s_1 et que s_1 est un **prédécesseur** de s_2.

Si les deux extrémités d'un arc sont égales, l'arc est une **boucle**.

Terminologie

Définition

On note $G(s)$ l'ensemble des successeurs du sommet s :

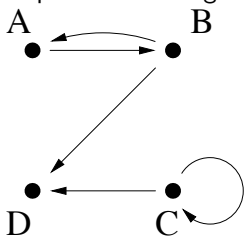
$$G(s) = \{t \in S \mid (s,t) \in A\}$$

On note $G_{-1}(s)$ l'ensemble des prédécesseurs du sommet s :

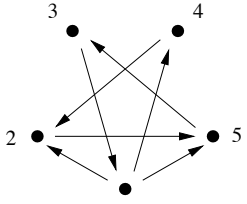
$$G(s) = \{r \in S \mid (r,s) \in A\}$$

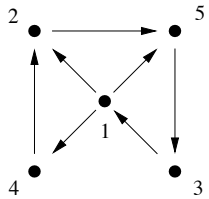
Représentation sagittale

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



Un même graphe peut avoir des représentations sagittales en apparence très différentes :

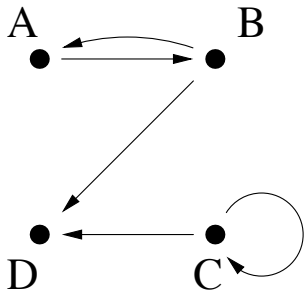




Dictionnaire d'un graphe

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 :



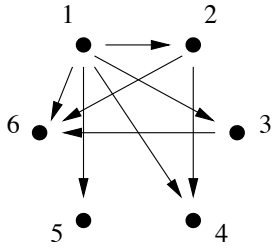
$S = \{A, B, C, D\}$
$G(A) = \{B\}$, $G(B) = \{A, D\}$,
$G(C) = \{C, D\}$, $G(D) = \emptyset$

Propriété caractéristique

Exemple : $G = (S, A)$ avec $S = \{1, 2, 3, 4, 5, 6\}$ et

$$\forall x, y \in S,\ (x, y) \in A \Leftrightarrow x \text{ divise strictement } y$$

Représentation sagittale :



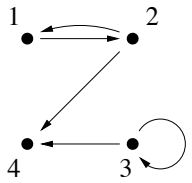
Matrice d'adjacence

Définition

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 $n \times n$, définie par*

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

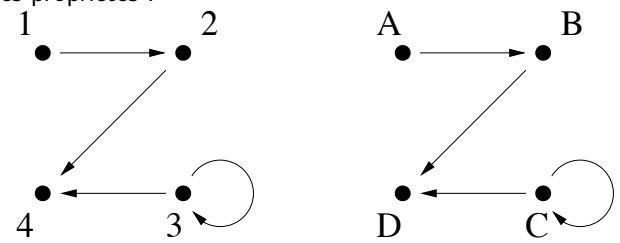
Exemple :



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

Remarques

- On a vu qu'un graphe pouvait avoir des représentations sagittales en apparence très différentes.
- Si on change les noms des sommets d'un graphe, on ne change pas ses propriétés :



sont des graphes en tout point semblables.

Isomorphismes de graphes

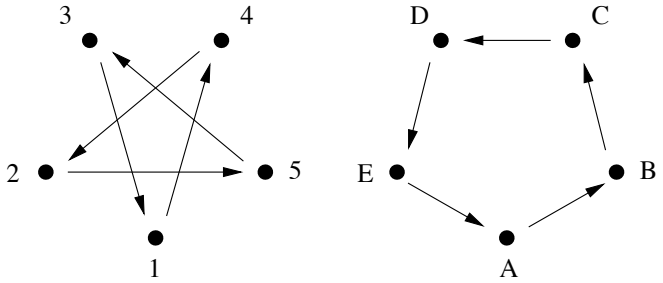
Définition

Deux graphes orientés $G = (S, A)$ et $G' = (S', A')$ sont **isomorphes** *si il existe une application bijective $f : S \to S'$ telle que*

$$\forall x, y \in S,\ (x, y) \in A \Leftrightarrow (f(x), f(y)) \in A'$$

L'application f est alors un **isomorphisme** *de graphes orientés.*

Exemple



L'application $f : \begin{cases} 1 \mapsto A \\ 2 \mapsto C \\ 3 \mapsto E \\ 4 \mapsto B \\ 5 \mapsto D \end{cases}$ est un isomorphisme entre les deux graphes.

Multigraphes orientés

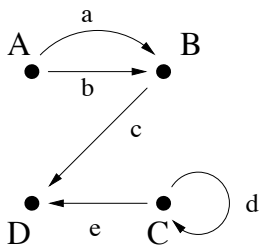
Dans un graphe orienté, on a au plus un arc entre deux sommets donnés. Pour pallier à cette limitation, on utilise les multigraphes, qui autorisent les arcs parallèles.

Définition

Un **multigraphe orienté** $G = (S, A, \alpha, \omega)$ *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 $\alpha : A \to S$ et $\omega : A \to S$ qui à chaque arc $a \in A$ associent son* **origine** *$\alpha(a)$ et son* **extrémité** *$\omega(a)$.*

Exemple



$S = \{A, B, C, D\}$
$A = \{a, b, c, d, e\}$

α	:	$a \mapsto A$	ω	:	$a \mapsto B$
		$b \mapsto A$			$b \mapsto B$
		$c \mapsto B$			$c \mapsto D$
		$d \mapsto C$			$d \mapsto C$
		$e \mapsto C$			$e \mapsto D$

Graphes non orientés

Dans un graphe non orienté, il n'y a pas de distinction entre les deux extrémités d'une arête.

Définition

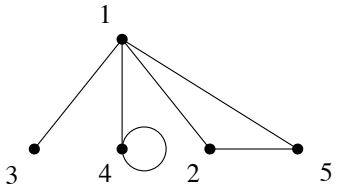
Un **graphe non orienté** $G = (S, A)$ *est la donnée :*

- *d'un ensemble S dont les éléments sont les* **sommets** *du graphe,*
- *d'un ensemble A dont les éléments, les* **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 orienté.

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

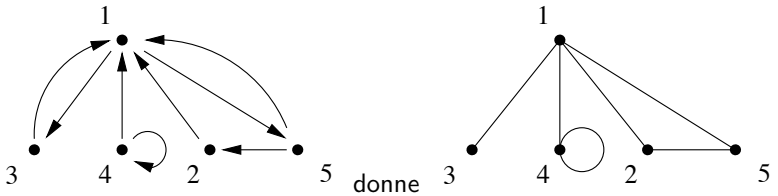
A chaque graphe orienté on peut associer un graphe non orienté, appelé **graphe non orienté associé** ou **sous-jacent**.

Définition

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

$$\{x, y\} \in A' \Leftrightarrow (x, y) \in A \text{ ou } (y, x) \in A$$

Exemples de graphes associés



Les trois graphes suivants :

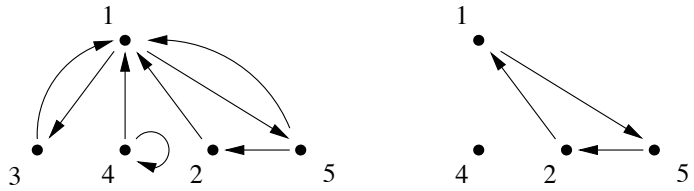
ont le même graphe non orienté associé :

Sous-graphes

Définition

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

Exemple :



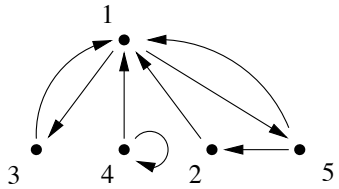
Sous-graphes induits

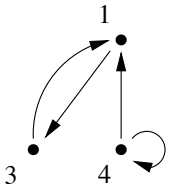
Définition

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' :*

$$\forall x, y \in S',\ (x, y) \in A' \Leftrightarrow (x, y) \in A.$$

Exemple :





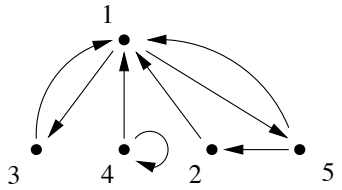
Sous-graphe induit par $S' = \{1, 3, 4\}$

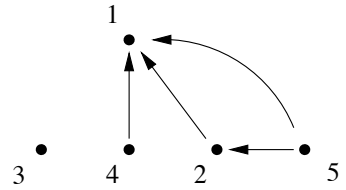
Sous-graphes couvrants

Définition

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 :





Degrés d'un sommet

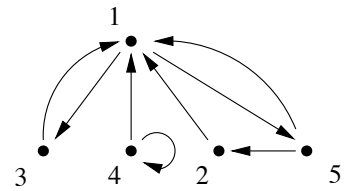
Définition

Soit G un graphe orienté.

- *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 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



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

Théorème des poignées de mains

Théorème

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.

Chemins

Définition

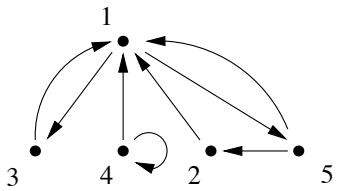
Soit $G = (S, A)$ un graphe orienté. Un **chemin** *C est une suite $(x_0, x_1, \ldots, x_{n-1}, x_n)$ de sommets de G tel que deux sommets consécutifs quelconques x_i et x_{i+1} sont reliés par un arc de G :*

$$\forall i,\ 0 \leq i \leq n-2,\ (x_i, x_{i+1}) \in A$$

Les sommets x_0 et x_n sont respectivement **l'origine** *et* **l'extrémité** *du chemin C.*
Le chemin C est formé 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

Terminologie

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 $\Rightarrow$ simple.

Graphes non orientés

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

orienté		**non orienté**
arc	$\leftrightarrow$	arête
chemin	$\leftrightarrow$	chaîne
circuit	$\leftrightarrow$	cycle

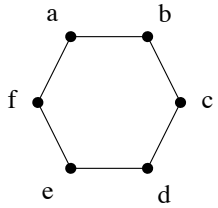
Connexité

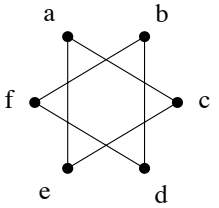
Définition

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.

Exemples :





Composantes connexes

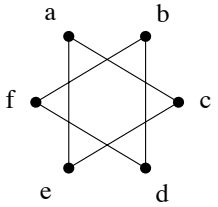
Définition

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 $x \in C$, alors*

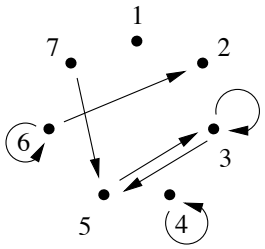
$$\forall y \in C, \text{ il existe une chaîne reliant } x \text{ à } y,$$
$$\forall z \in S \setminus C, \text{ il n'existe pas de chaîne reliant } x \text{ à } z.$$

- 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 $x \in S$ est $C = \{y \in S \mid \text{il existe une chaîne reliant } x \text{ à } y\}$

Exemples



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



Ce graphe a quatre composantes connexes :
$\{1\}$, $\{2, 6\}$, $\{3, 5, 7\}$, et $\{4\}$.

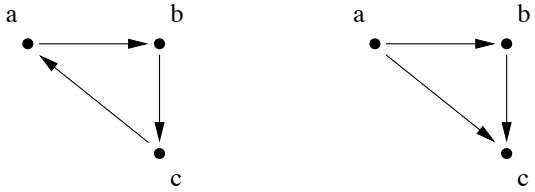
Connexité forte

Définition

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 :



Propriétés

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.

Composantes fortement connexes

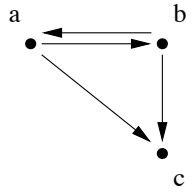
Définition

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*

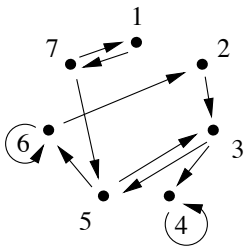
$$\forall y \in C, \text{ il existe un circuit passant par } x \text{ et } y,$$
$$\forall z \in S \setminus C, \text{ il n'existe pas de circuit passant par } x \text{ et } z.$$

- 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 : $\{y \in S \mid \text{il existe un chemin reliant } x \text{ à } y \text{ et un chemin reliant } y \text{ à } x\}$

Exemples



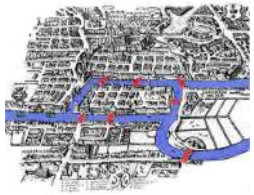
Ce graphe a deux composantes fortement connexes :
$\{a, b\}$ et $\{c\}$.



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

Cycles eulériens - Introduction

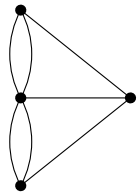
Au 18^{e} siècle un casse-tête est populaire chez les habitants de Königsberg : est-il possible de se promener dans la ville en ne passant qu'une seule fois par chacun des sept ponts de Königsberg ?



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.

Cycles eulériens - Introduction

Le problème se reformule ainsi en terme de graphes :



existe-t-il dans le (multi)graphe ci-dessus un cycle qui passe exactement une fois par toutes les arêtes ?

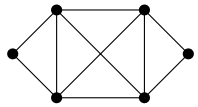
Définitions

Définition

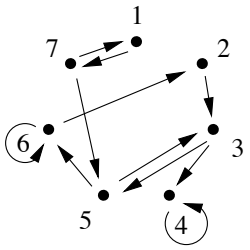
Soit G un graphe non orienté. Une chaîne (respectivement un cycle) **eulérienne** *est une chaîne (resp. un cycle) qui passe une et une seule fois par toutes les arêtes de G.*

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.*

Exemples :



Ce graphe admet un cycle eulérien.

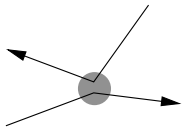


Ce graphe admet un chemin eulérien mais pas de circuits eulériens.

Conditions nécessaires

Une condition évidemment nécessaire pour qu'un graphe admette un cycle ou un circuit eulérien est qu'il soit **connexe**.

On constate aussi qu'un cycle ou un circuit contient autant d'arêtes arrivant à un sommet donné que d'arêtes en partant : on part d'un sommet autant de fois qu'on y arrive.



Une deuxième condition nécessaire pour qu'un graphe admette un cycle ou un circuit euclidien est donc :

$\forall x \in S$, $d(x)$ est pair (cas non orienté)

$\forall x \in S$, $d_+(x) = d_-(x)$ (cas orienté)

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 :*

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

Démonstration (cas orienté)

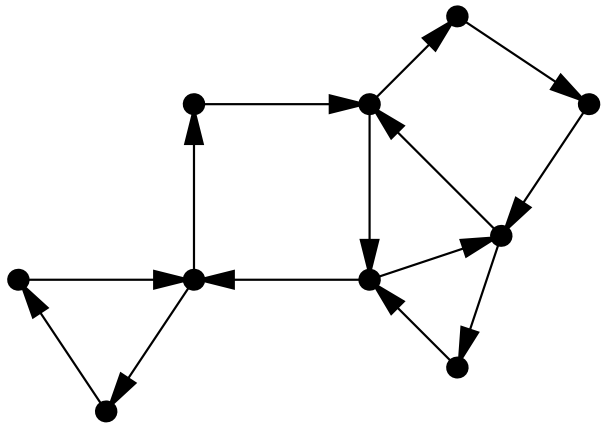
Par récurrence sur le nombre d'arcs n du graphe.

Initialisation : $n = 0$**.** Un graphe connexe sans arcs se réduit à un seul sommet isolé.

Induction : On suppose que le théorème est vrai pour tout graphe ayant un nombre d'arcs strictement inférieur à n (*hypothèse de récurrence forte*).

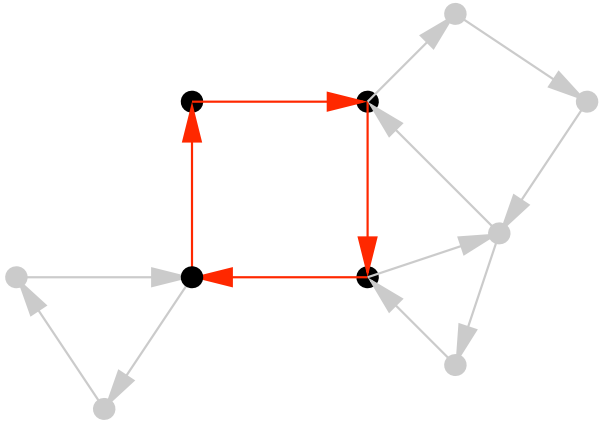
- On construit un premier circuit C.
- On considére le graphe partiel H constitué des arcs en dehors du circuit C. Chaque composante connexe H_i de H vérifie l'hypothèse de récurrence, et admet donc un circuit eulérien C_i.
- Pour reconstruire un circuit eulérien sur G, on fusionne le circuit C avec les différents cycles C_i.

Illustration



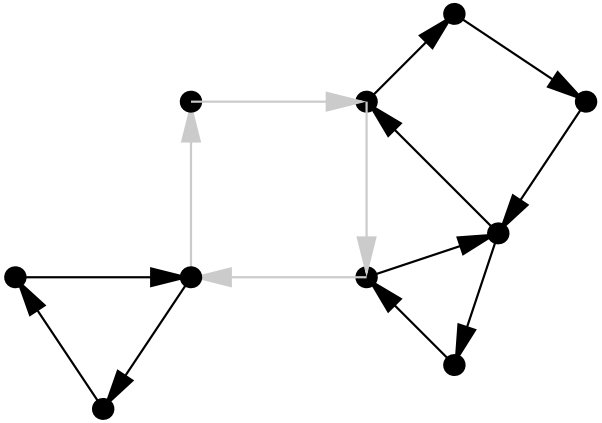
Graphe de départ

Illustration



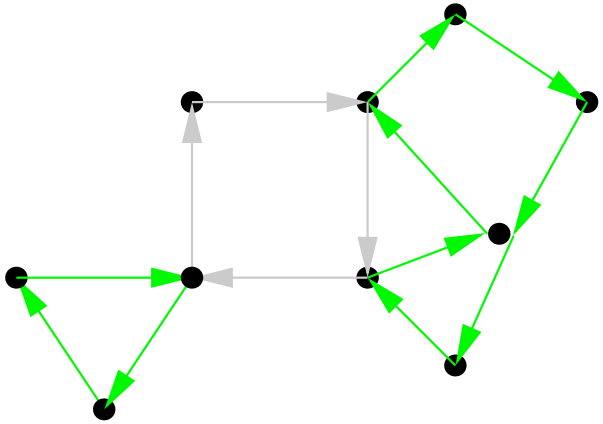
On construit un premier circuit.

Illustration



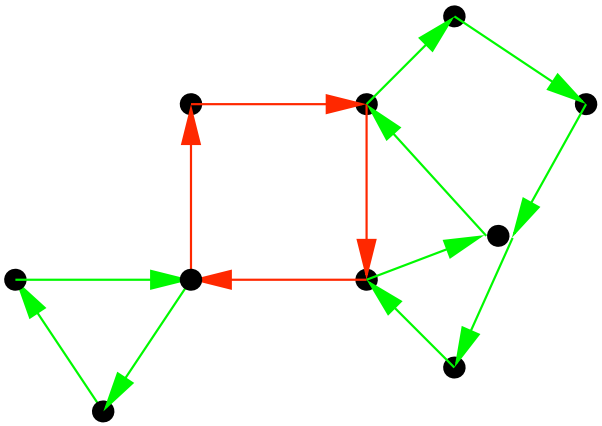
On considère le graphe privé des arcs du circuit.

Illustration



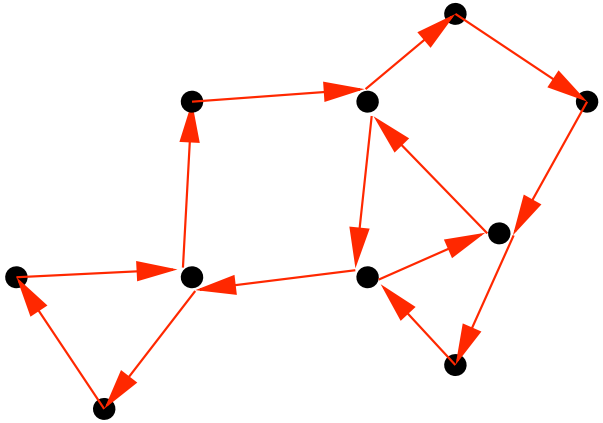
Par récurrence, chacune des composantes connexes du nouveau graphe possède un circuit eulérien.

Illustration



Il ne reste qu'à fusionner ces circuits avec le circuit initial.

Illustration



Et voilà.

Chemins et chaînes eulérien(ne)s

Soit G un graphe admettant une chaîne eulérienne.
Si cette chaîne n'est pas déjà un cycle, le graphe obtenu en ajoutant une arête entre les deux extrémités de la chaîne admet alors un cycle eulérien.

Réciproquement, si on supprime une arête à un graphe G admettant un cycle eulérien, le nouveau graphe admet encore une chaîne eulérienne.

Chemins et chaînes eulérien(ne)s

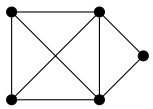
Les graphes qui admettent une chaîne eulérienne se déduisent donc des graphes qui admettent un cycle eulérien en supprimant éventuellement une arête.
Même chose pour les graphes orientés.

Théorème

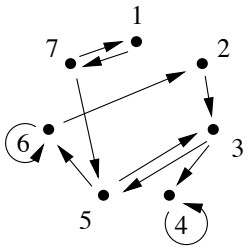
Un graphe $G = (S, A)$ admet une chaîne (cas non orienté) ou un chemin (cas orienté) **eulérien(ne)** *si et seulement si les deux conditions suivantes sont vérifiées :*

1. *le graphe G est* **connexe**
2. *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)

Exemples :



Ce graphe admet une chaîne eulérienne mais pas de cycles eulériens.



Ce graphe admet un chemin eulérien mais pas de circuits eulériens.

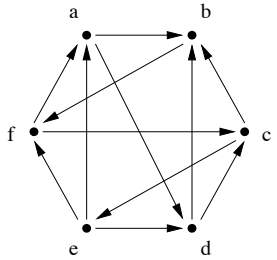
Cycles hamiltoniens

Définition

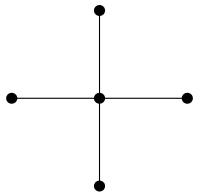
Soit G un graphe non orienté. Un cycle (respectivement une chaîne) **hamiltonien** *est un cycle (resp. une chaîne) qui passe une et une seule fois par tous les sommets de G.*

On définit les mêmes notions pour un graphe orienté 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.*

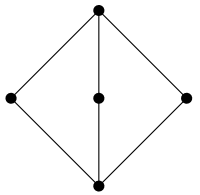
Exemples :



Ce graphe admet un circuit hamiltonien.



Ce graphe n'admet ni chaînes ni cycles hamiltoniens.

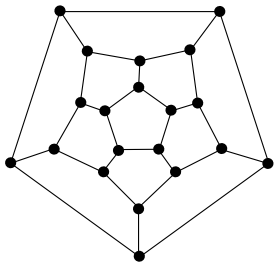


Ce graphe admet une chaîne hamiltonienne mais pas de cycles hamiltoniens

Conditions nécessaires et suffisantes ?

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

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



Graphes valués

Définition

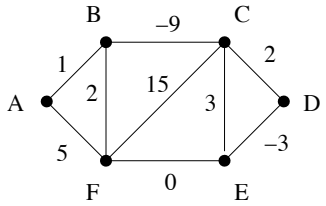
Un graphe (orienté ou non) $G = (S, A)$ est **valué** *si il est muni d'une application*

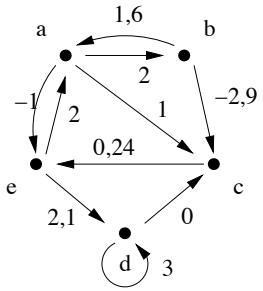
$$\begin{array}{rcl} v : A & \to & \mathbb{R} \\ (x, y) & \mapsto & v(x, y) \end{array}$$

qui sera appelée **valuation**.

On peut étendre la valuation en une fonction $S \times S \to \mathbb{R} \cup \{+\infty\}$ en posant $v(x, y) = +\infty$ si $(x, y) \notin A$.

Exemples :





Représentation matricielle

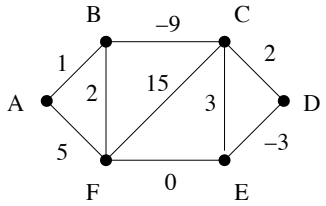
Par analogie avec la matrice d'adjacence, on peut représenter un graphe valué par une matrice carrée, dont les coefficients correspondent à la valuation des arcs.

Définition

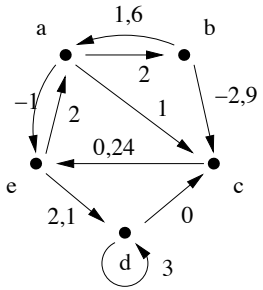
Soit $G = (S, A, v)$ un graphe valué dont on a numéroté les sommets de 1 à n. La **matrice de valuation** *de G est la matrice carrée $M = (m_{ij})$, de taille $n \times n$, définie par*

$$m_{ij} = \begin{cases} v(i,j) & \text{si } (i,j) \in A \\ +\infty & \text{sinon} \end{cases}$$

Exemples :



$$\begin{pmatrix} \infty & 1 & \infty & \infty & \infty & 5 \\ 1 & \infty & -9 & \infty & \infty & 2 \\ \infty & -9 & \infty & 2 & 3 & 15 \\ \infty & \infty & 2 & \infty & -3 & \infty \\ \infty & \infty & 3 & -3 & \infty & 0 \\ 5 & 2 & 15 & \infty & 0 & \infty \end{pmatrix}$$



$$\begin{pmatrix} \infty & 2 & 1 & \infty & -1 \\ 1,6 & \infty & -2,9 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0,24 \\ \infty & \infty & 0 & 3 & \infty \\ 2 & \infty & \infty & 2,1 & \infty \end{pmatrix}$$

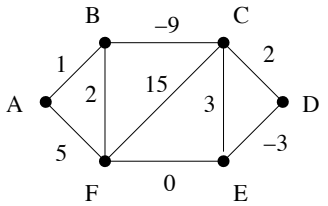
Valuation d'un chemin

Définition

Soit $G = (S, A, v)$ un graphe valué. La **valuation** *ou* **longueur** *d'un chemin (ou d'une chaîne) est la somme des valuations de chacun des arcs qui le composent.*

La valuation d'un chemin ne comportant pas d'arcs, de la forme (s), est égale à 0.

Exemple :



La valuation de la chaîne (A, F, C, E, D) est $5 + 15 + 3 - 3 = 20$

Distance et plus court chemin

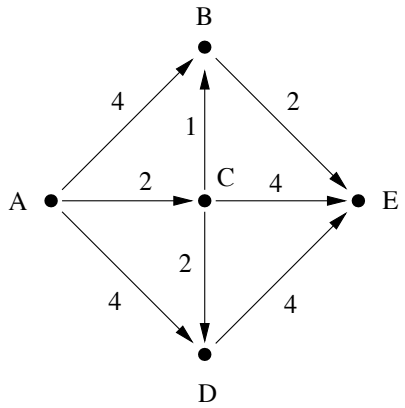
Définition

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 $d(x, y)$.*

Distance et plus court chemin

Exemple :



Distance de A à E : $d(A, E) = 5$
Plus court(s) chemin(s) :
(A, C, B, E)

Distance de A à D : $d(A, D) = 4$
Plus court(s) chemin(s) : (A, D) et (A, C, D)

Distance de E à A : non défini
Plus court(s) chemin(s) : inexistant

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

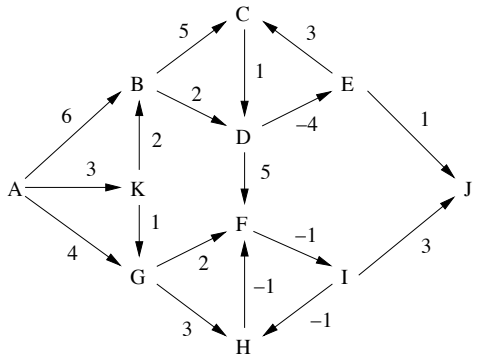
Certains problèmes d'ordonnancement font aussi appel à des recherches de plus longs chemins.

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

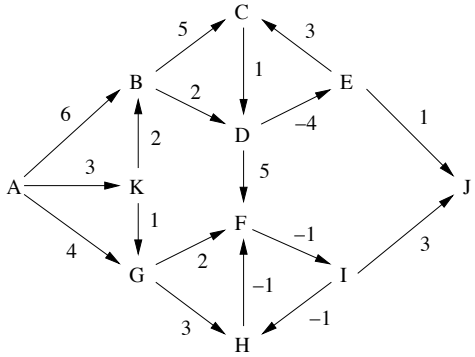
- Dans un graphe non orienté, on a toujours $d(x, y) = d(y, x)$, et toute plus courte chaîne de x à y parcourue à l'envers est une plus courte chaîne de y à x.
- Etant donnés deux sommets x et y, plusieurs cas se présentent :
  1. il n'y a pas de chemins / chaînes de x à y
  2. il existe un ou plusieurs plus courts chemins / chaînes de x à y
  3. 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.

Exemple



De A à E : il existe une infinité de plus courts chemins : (A, K, B, D, E), (A, K, B, D, E, C, D, E), $(A, K, B, D, E, C, D, E, \ldots, C, D, E), \ldots$
De A à J : il existe des chemins mais pas de plus court : les chemins $(A, G, H, F, I, H, F, I, \ldots, H, F, I, J)$ sont arbitrairement courts.

Circuit absorbant

Définition

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

Propriété

Tout sous-chemin d'un plus court chemin est un plus court chemin.

Démonstration.

- Soit $C_0 = (x_0, x_1, x_2, \ldots, x_n)$ un plus court chemin entre x_0 et x_n.
- Soit $C = (x_p, x_{p+1}, \ldots, x_{q-1}, x_q)$ un sous-chemin de C_0, avec $0 \leq p \leq q \leq n$.
- Si C n'est pas un plus court chemin entre x_p et x_q, alors il existe un autre chemin $C' = (x_p, x'_1, x'_2, \ldots, x'_{r-1}, x'_r, x_q)$ entre x_p et x_q, et dont la longueur est strictement plus petite que celle de C.
- Le chemin $C_1 = (x_0, x_1, \ldots, x_{p-1}, x_p, x'_1, x'_2, \ldots, x'_r, x_q, x_{q+1}, \ldots, x_n)$, obtenu en remplaçant C par C' dans C_0, est alors strictement plus court que C_0, ce qui est absurde.

□

Propriétés des plus courts chemins

Propriété

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.

Démonstration

- Soit $C_0 = (x_0, x_1, x_2, \ldots, x_{n-1}, x_n)$ un plus court chemin entre $x = x_0$ et $y = x_n$. Si C_0 n'est pas élémentaire, il existe deux indices p et q, $0 \leq p < q \leq n$, tels que $x_p = x_q$.
- Le sous-chemin $C_1 = (x_p, x_{p+1}, \ldots, x_{q-1}, x_q)$ est alors un circuit, et c'est aussi un plus court chemin d'après la propriété précedente. Il est donc au moins aussi court que le chemin trivial (x_p), de valuation 0.

Propriétés des plus courts chemins

Démonstration - suite

- Si la valuation de C_1 est strictement négative, alors C_1 est un circuit absorbant, et il n'existe pas de plus court chemin entre $x = x_0$ et $y = x_n$, ce qui est absurde.
  $C_0' = (x_0, x_1, \ldots, x_{p-1}, x_p, x_{q+1}, x_{q+2}, \ldots, x_{n-1}, x_n)$
- Si la valuation de C_1 est nulle, le chemin
  $C_0' = (x_0, x_1, \ldots, x_{p-1}, x_p, x_{q+1}, x_{q+2}, \ldots, x_{n-1}, x_n)$ a la même longueur que C_0, c'est donc encore un plus court chemin. On construit ainsi un plus court chemin élémentaire entre $x = x_0$ et $y = x_n$.

Distance en nombre d'arcs

C'est un cas particulier de calcul de distance, dans le cas où *tous les arcs sont de valuation* 1.
Etant donné un sommet initial x, on cherche à déterminer $d(x, y)$ pour tout sommet y.

Principe :
Un sommet y est à distance n de x si :

1. il existe un chemin de longueur n de x à y
2. il n'existe pas de chemin de longueur strictement inférieure à n de x à y.

Ces deux conditions peuvent se réécrire :

1. y est le successeur d'un sommet à distance $n-1$ de x.
2. la distance de x à y n'est pas plus petite que n.

Distance en nombre d'arcs : algorithme

L'algorithme est similaire à l'exploration en largeur d'un graphe.

On construit deux familles d'ensemble de sommets :
S_i : ensemble des sommets à distance i de x
R_i : ensemble des sommets à distance plus grande que i de x.

Distance en nombre d'arcs : algorithme

Initialisation :

$S_0 = \{x\}$
$R_0 = S \setminus \{x\}$
$i = 0$

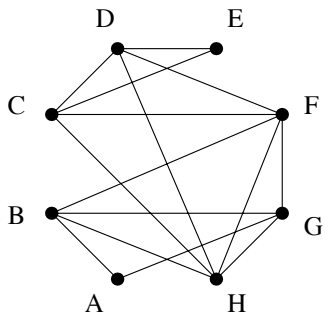
répéter

$S_{i+1} = G(S_i) \cap R_i$
$R_{i+1} = R_i \setminus S_{i+1}$
$i \leftarrow i + 1$

jusqu'à ce que $S_{i+1} = \emptyset$ ou $R_{i+1} = \emptyset$

On utilise la notation
$G(S_i) = \bigcup_{s \in S_i} G(s) = \{\text{successeurs des éléments de } S_i\}$

Exemple



$S_0 = \{A\}$,
$R_0 = \{B, C, D, E, F, G, H\}$

$S_1 = \{B, G\}$, $R_1 = \{C, D, E, F, H\}$

$S_2 = \{F, H\}$, $R_2 = \{C, D, E\}$

$S_3 = \{C, D\}$, $R_3 = \{E\}$

$S_4 = \{E\}$, $R_4 = \emptyset$

$d(A, B) = 1$, $d(A, C) = 3$, $d(A, D) = 3$, $d(A, E) = 4$, $d(A, F) = 2$,
$d(A, G) = 1$, $d(A, H) = 2$

Plus court chemin en nombre d'arcs

Une fois connues les distance de x à s pour tout sommet s, on peut déterminer les plus courts chemins.

Pour trouver un plus court chemin de x à y, on part de la fin :

- on cherche un prédécesseur p de y tel que

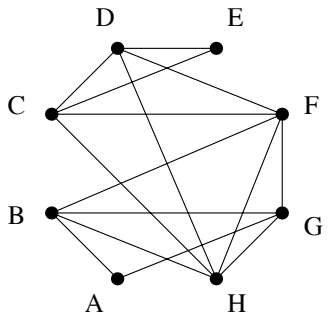
$$d(x, y) = d(x, p) + 1$$

(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 à y est bien définie, on a toujours $d(x, y) \leq d(x, p) + 1$ pour tout prédécesseur p de y, et il existe au moins un prédécesseur pour lequel il y a égalité.

Exemple



$d(A, B) = 1$, $d(A, C) = 3$,
$d(A, D) = 3$, $d(A, E) = 4$,
$d(A, F) = 2$, $d(A, G) = 1$,
$d(A, H) = 2$

Un plus court chemin de A à E est : $A \rightarrow B \rightarrow F \rightarrow C \rightarrow E$

Principe des algorithmes dans le cas général

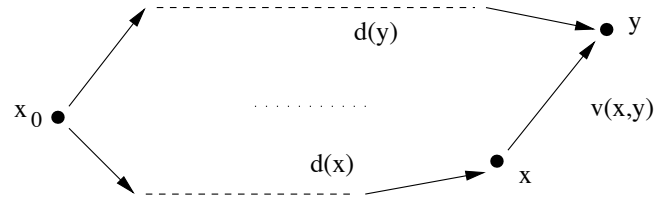
Etant donnés un graphe valué $G = (S, A, v)$ et un sommet x_0, on veut déterminer pour chaque sommet s la distance et un plus court chemin de x_0 à s.

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 $+\infty$) pour ces distances, et de la donnée d'un prédécesseur $P(s)$ pour les plus courts chemins.

Principe des algorithmes dans le cas général

- 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

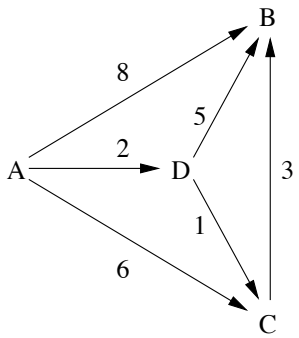
On applique le principe précédent en explorant systématiquement tous les sommets et tous leurs successeurs. On s'arrête quand les valeurs des distances sont stabilisées.

Algorithme de Bellman-Ford

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

Preuve de l'algorithme

On va montrer que (dans le cas où il n'y a pas de circuits absorbants) :

1. l'algorithme se termine (les valeurs se stabilisent) après au plus n passages dans la boucle principale, n étant le nombre de sommets du graphe
2. les valeurs $d(s)$ obtenues à la fin de l'algorithme sont bien les distances de x_0 aux sommets s.

Plus précisément, on va montrer par *récurrence* la propriété suivante :

Si un plus court chemin élémentaire de x_0 à un sommet s comporte k arcs, alors après k passages dans la boucle on a $d(s) = d(x_0, s)$.

Comme un chemin élémentaire a au plus $n-1$ arcs, cela démontrera à la fois les points 1 et 2.

Preuve de l'algorithme (2)

On commence par démontrer les remarques suivantes :

Lemme

1. Les valeurs $d(s)$ ne peuvent que diminuer pendant le déroulement de l'algorithme.
2. A chaque étape de l'algorithme, pour tout sommet s, la valeur $d(s)$ est soit $+\infty$, soit égale à la longueur d'un chemin de x_0 à s.

Le point 1 est évident. Le point 2 se montre facilement par récurrence :

Initialisation : La propriété est manifestement vraie à l'initialisation de l'algorithme.

Hérédité : Supposons que la propriété soit vraie à une étape de l'algorithme. A l'étape suivante on remplace (éventuellement) $d(y)$ par $d(x) + v(x, y)$. Par hypothèse de récurrence $d(x)$ est la longueur d'un chemin C de x_0 à x, et $d(x) + v(x, y)$ est donc la longueur d'un chemin de x_0 à y obtenu en rajoutant l'arc (x, y) à C.

Preuve de l'algorithme (3)

Lemme

1. Les valeurs $d(s)$ ne peuvent que diminuer pendant le déroulement de l'algorithme.
2. A chaque étape de l'algorithme, pour tout sommet s, la valeur $d(s)$ est soit $+\infty$, soit égale à la longueur d'un chemin de x_0 à s.

On déduit de ce lemme que :

- à toute étape de l'algorithme, $d(s) \geq d(x_0, s)$

- quand la valeur $d(s)$ atteint $d(x_0, s)$, elle ne varie plus dans la suite de l'algorithme.

Preuve de l'algorithme (4)

Propriété

Si un plus court chemin élémentaire de x_0 à un sommet s comporte k arcs, alors après k passages dans la boucle on a $d(s) = d(x_0, s)$.

On fait une récurrence sur le nombre d'arcs k d'un plus court chemin élémentaire.

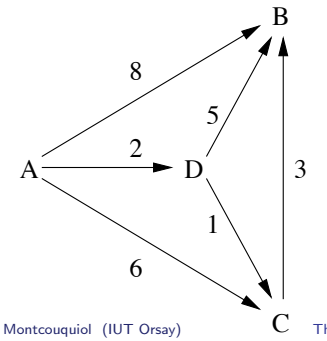
Initialisation : La propriété est vraie pour $k = 0$: c'est la phase d'initialisation de l'algorithme.

Hérédité : On suppose la propriété vraie au rang $k-1$, montrons-la au rang k.
Soit p le prédécesseur de s le long du plus court chemin élémentaire comportant k arcs de x_0 à s. Il existe un plus court chemin élémentaire comportant $k-1$ arcs de x_0 à p, donc par hypothèse de récurrence, après $k-1$ passages dans la boucle on a $d(p) = d(x_0, p)$. Après le k-ième passage dans la boucle, on a alors
$d(s) = d(p) + v(p, s) = d(x_0, p) + v(p, s) = d(x_0, s)$.

Plus court chemin

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 :

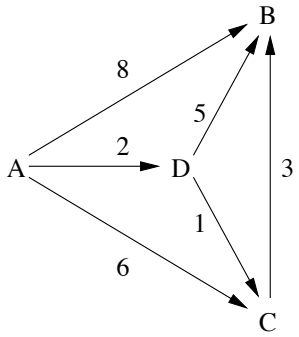
- On construit petit à petit, à partir de $\{x_0\}$, un ensemble M de sommets **marqués**. Pour tout sommet marqué s, l'estimation $d(s)$ est égale à la distance $d(x_0, s)$.
- A chaque étape, on sélectionne le (un) sommet non marqué x dont la distance estimé $d(x)$ est la plus petite parmi tous les 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.

Algorithme de Dijkstra

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

Preuve de l'algorithme

On commence par classer les sommets selon leur distance à x_0 :
$S = \{x_0, x_1, x_2, \ldots, x_n\}$
$0 = d(x_0, x_0) \leq d(x_0, x_1) \leq d(x_0, x_2) \leq \ldots \leq d(x_0, x_n)$
Comme les arcs ont tous des valuations positives, toutes ces distances sont bien positives.
On va montrer la propriété suivante :

Propriété

1. A toutes les étapes de l'algorithme, chaque sommet marqué s vérifie $d(s) = d(x_0, s)$.
2. Après i itérations de la boucle principale de l'algorithme, l'ensemble des sommets marqués est $M = \{x_0, x_1, \ldots, x_{i-1}\}$

En particulier, cette propriété implique qu'à la fin de l'algorithme on a bien calculé les distances à x_0.

Preuve de l'algorithme (2)

Propriété

1. A toutes les étapes de l'algorithme, chaque sommet marqué s vérifie $d(s) = d(x_0, s)$.
2. Après i itérations de la boucle principale de l'algorithme, l'ensemble des sommets marqués est $M = \{x_0, x_1, \ldots, x_{i-1}\}$

Démonstration

On fait une démonstration par récurrence.

Initialisation : Les propriétés sont vraies pour $i = 0$: c'est la phase d'initialisation de l'algorithme.

Hérédité : On suppose les propriétés vraies pour tout rang plus petit que i, montrons-la au rang i.
Soit x_k un prédécesseur de x_i le long d'un plus court chemin de x_0 à x_i. La distance de x_0 à x_k est $d(x_0, x_i) - v(x_k, x_i)$, qui est plus petite que $d(x_0, x_i)$ car tous les arcs sont de valuation positive. Par conséquent x_k est classé avant x_i : $k < i$.

Preuve de l'algorithme (3)

Démonstration - suite

Par hypothèse de récurrence x_k a été marqué pendant la k-ième itération, à la sortie de laquelle on a $d(x_k) = d(x_0, x_k)$ et $d(x_i) = d(x_k) + v(x_k, x_i) = d(x_0, x_i)$.

A la i-ème itération on a alors pour tout sommet non marqué x_j, $j > i$: $d(x_i) = d(x_0, x_i) < d(x_0, s) \leq d(s)$. Le sommet x_i est donc celui qui est marqué à la i-ème itération et il vérifie bien $d(x_i) = d(x_0, x_i)$.

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 ($\leftrightarrow$ sommet) connaît juste son voisinage immédiat et le communique à ses voisins.

Détermination des plus courts chemins

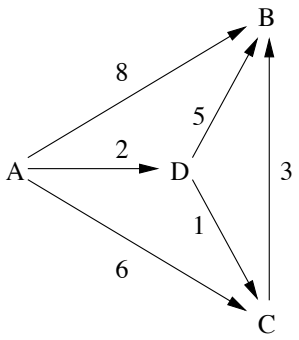
Quand on connaît les distance de x_0 à s pour tout sommet s, on peut déterminer (tous) les plus courts chemins de x_0 à s pour tout sommet s. Si on dispose du tableau des pères, il est facile de trouver un plus court chemin ; si on n'a pas ce tableau ou si on recherche tous les plus courts chemins, on 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.

Exemple



s	A	B	C	D
$d(A,s)$	0	6	3	2

Un plus court chemin de A à B est :
$A \rightarrow D \rightarrow C \rightarrow$B

$$d(A,A) + v(A,B) = 8 > d(A,B)$$
$$d(A,C) + v(C,B) = 6 = d(A,B)$$
$$d(A,D) + v(D,B) = 7 > d(A,B)$$
$$d(A,A) + v(A,C) = 6 > d(A,C)$$
$$d(A,D) + v(D,C) = 3 = d(A,C)$$
$$d(A,A) + v(A,D) = 2 = d(A,D)$$

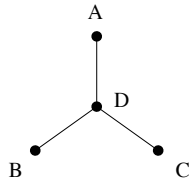
Graphes sans cycles

Définition

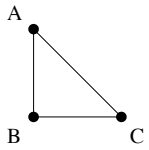
Un graphe non orienté G est **sans cycle** *ou* **acyclique** *s'il ne possède pas de cycles non simples.*

Rappel : un chemin est *simple* s'il ne passe qu'une seule fois par chacune de ses arêtes.

Exemples :



Ce graphe admet des cycles, par exemple (A, D, B, D, A) ou (C, D, C), mais pas de cycles simples : ce graphe est acyclique.



Ce graphe admet un cycle simple (A, B, C, A) : il n'est pas acyclique.

Arbres

Définition

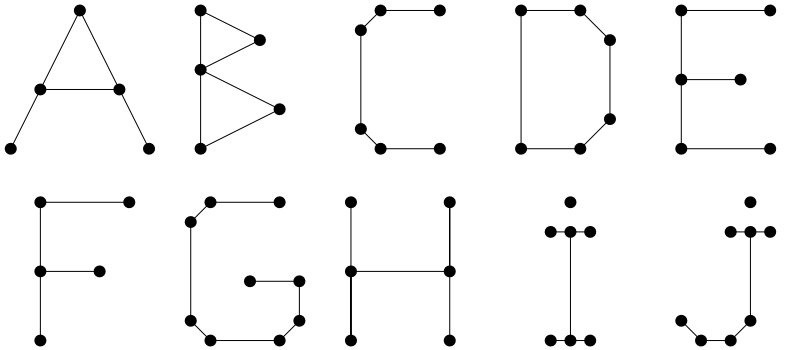
Un **arbre** *est un graphe non orienté, connexe, sans cycle.*

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.

Exemples

Parmi les graphes suivants, lesquels sont des arbres ?



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 :

1. *le graphe G est connexe et sans cycle*
2. *le graphe G est connexe et a $n-1$ arêtes*
3. *le graphe G est connexe et la suppression de n'importe quelle arête le déconnecte*
4. *le graphe G est sans cycle et a $n-1$ arêtes*
5. *le graphe G est sans cycle et l'ajout de n'importe quelle arête crée un cycle*
6. *entre toute paire de sommets de G il existe une unique chaîne élémentaire*

Caractérisation des arbres

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.*

Arbres enracinés

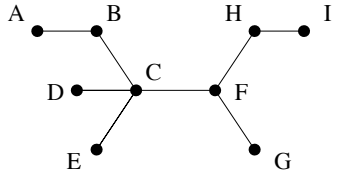
Les arbres utilisés en algorithmique ont le plus souvent une orientation et un sommet qui joue un rôle particulier, la racine : c'est ce type d'arbre que l'on va voir maintenant

Définition

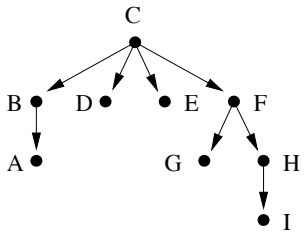
Un graphe non orienté est un **arbre enraciné** *s'il est connexe sans cycle et si un sommet particulier, la* **racine**, *a été distingué.*

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**.

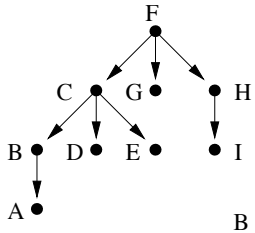
Exemples



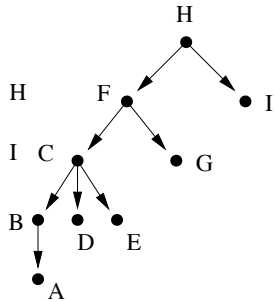
Racine en C :



Racine en F :



Racine en H :



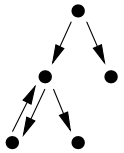
Caractérisation

Propriété

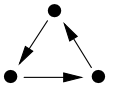
Un graphe orienté est un arbre enraciné si et seulement si

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

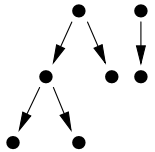
Exemples



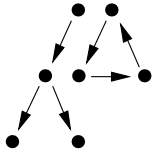
N'est pas un arbre : un sommet a deux prédécesseurs



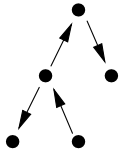
N'est pas un arbre : aucun sommet sans prédécesseur (pas de racine)



N'est pas un arbre : deux sommets sans prédécesseur



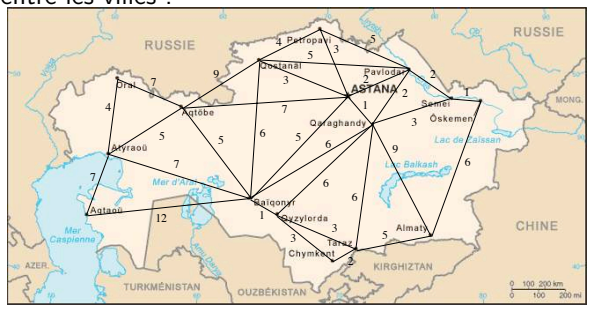
N'est pas un arbre : non connexe



Est un arbre

Un problème de connexion

Vous êtes chargé par le gouvernement kazakh d'équiper le pays en accès internet à haut débit. Pour cela, vous devez relier les 16 plus grandes villes du Kazakhstan avec des câbles de fibres optiques. Après une étude préliminaire, vous estimez les coûts suivants (en millions de KZT) de connexion entre les villes :



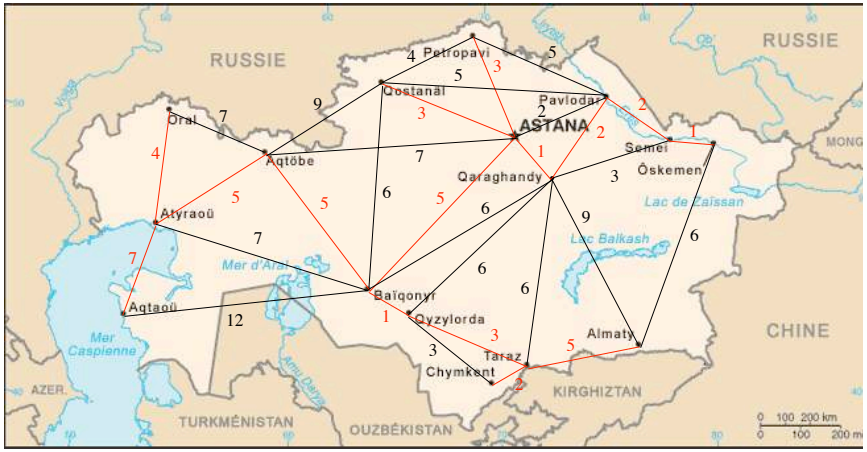
Quelles sont les liaisons à réaliser pour connecter toutes les villes au moindre coût ?

Arbres couvrant de poids minimum

Définition

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.*

Exemple



total : 49 000 000 KZT

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 sélectionné $n-1$ arêtes, ou qu'il ne reste plus d'arêtes ne créant pas de cycles.

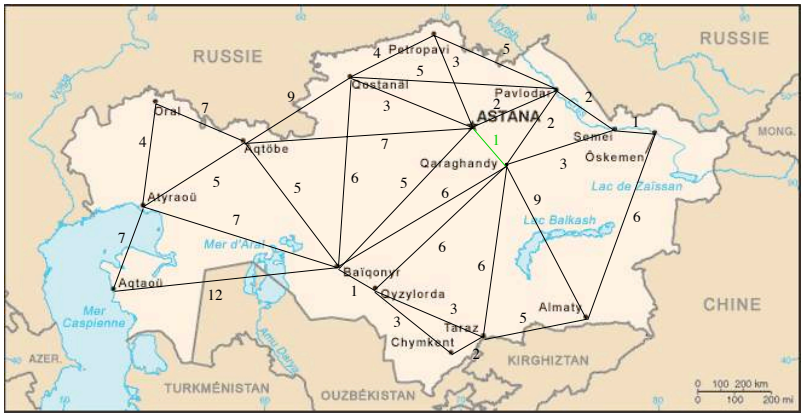
Algorithme de Kruskal

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

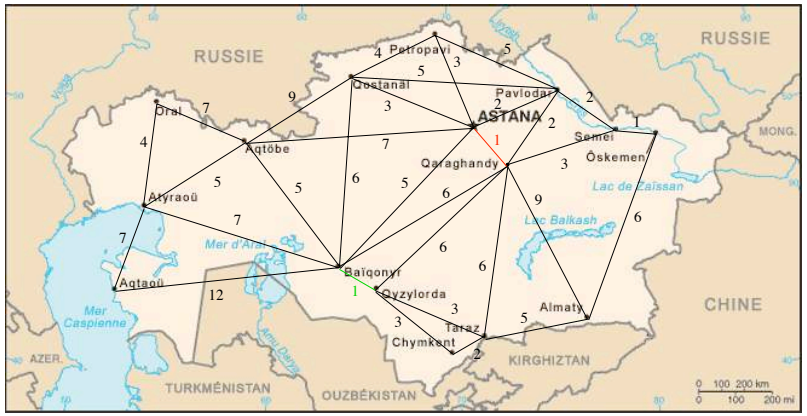
Trier l'ensemble A des arêtes de G par valuations croissantes

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

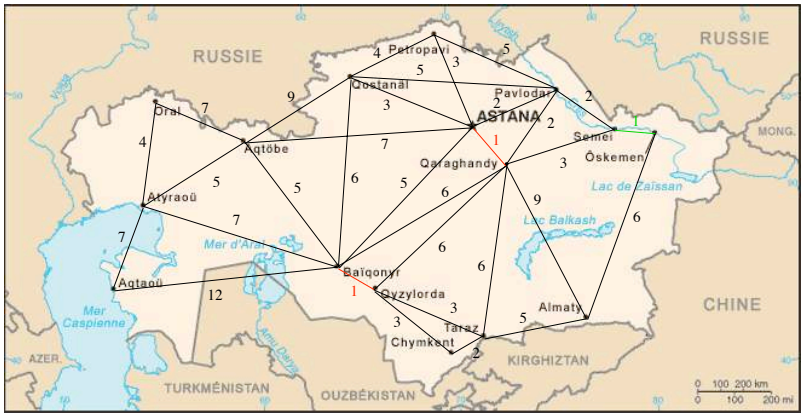
Exemple



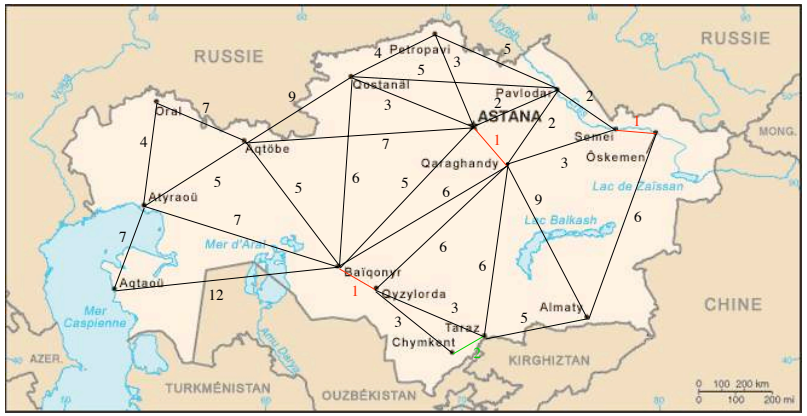
Exemple



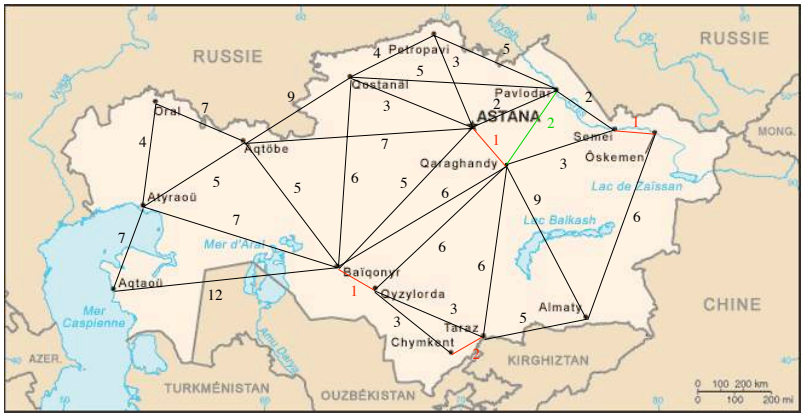
Exemple



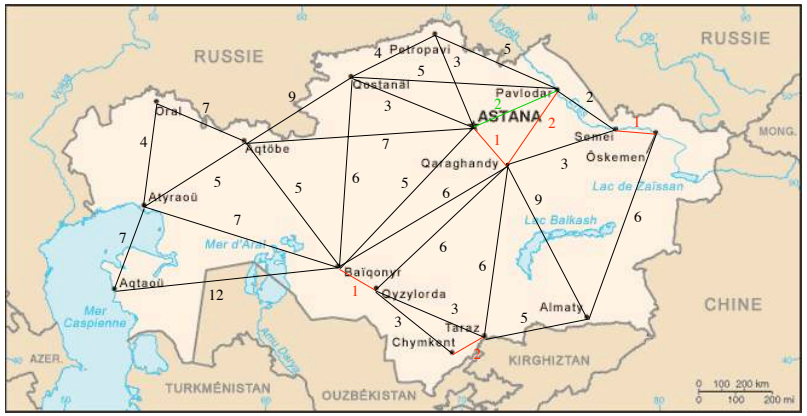
Exemple



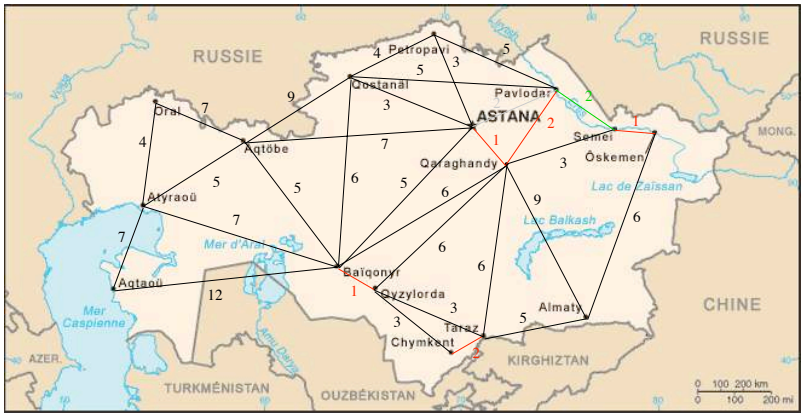
Exemple



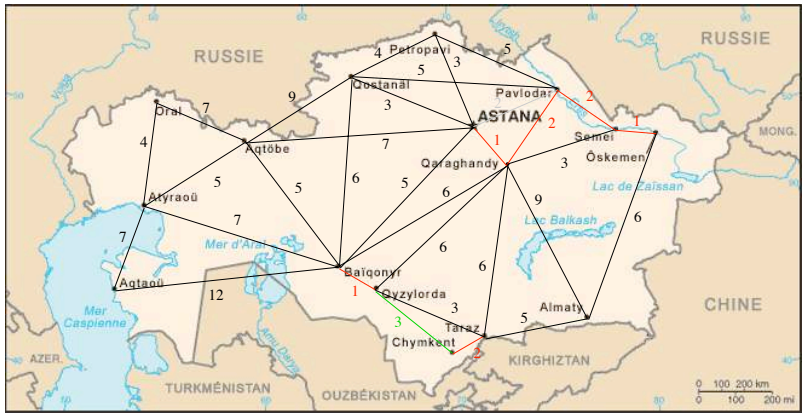
Exemple



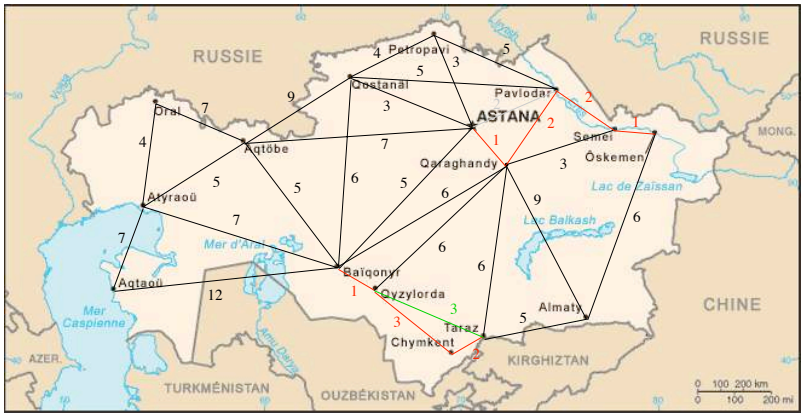
Exemple



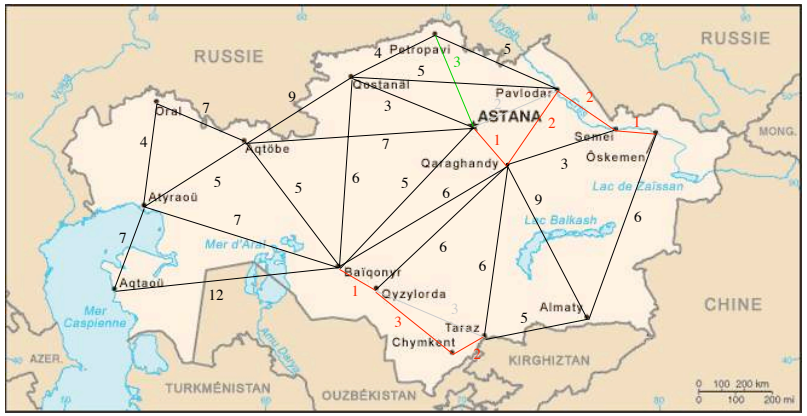
Exemple



Exemple



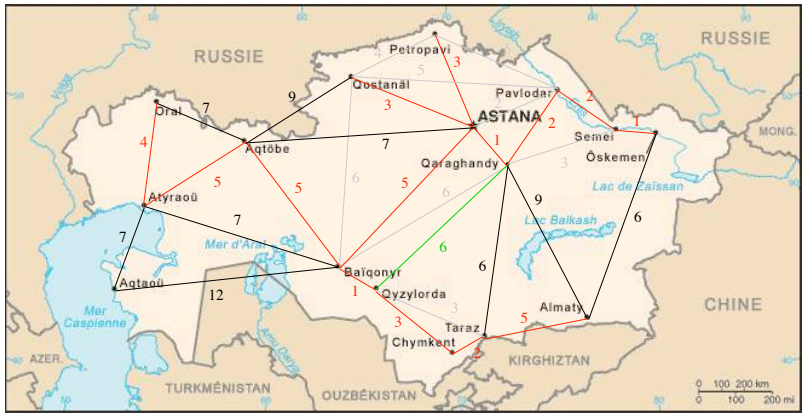
Exemple



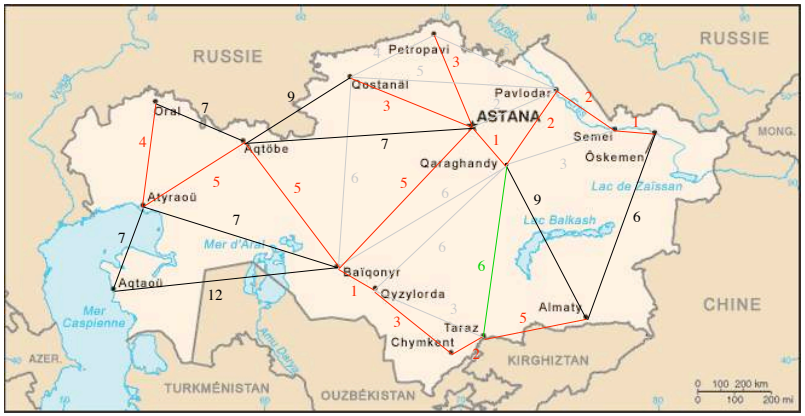
Exemple

(On saute quelques étapes. . .)

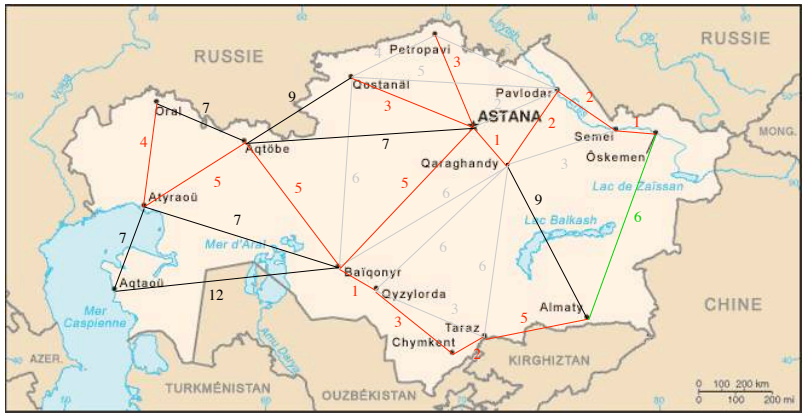
Exemple



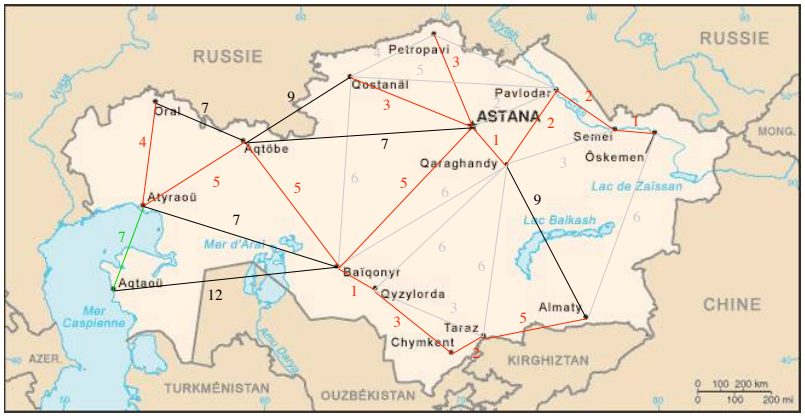
Exemple



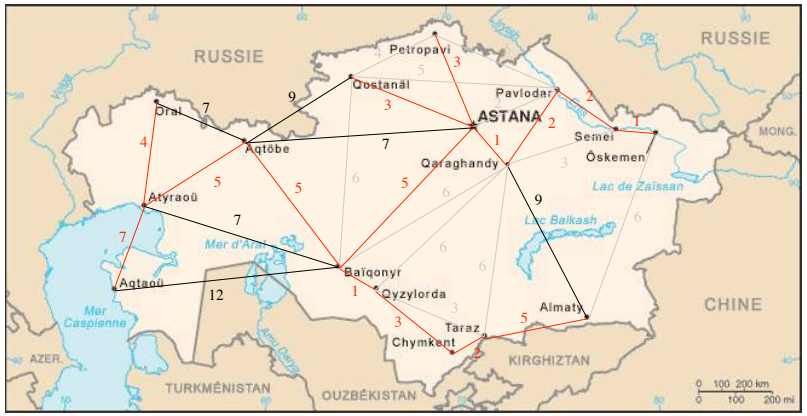
Exemple



Exemple



Exemple



Commentaires

- 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

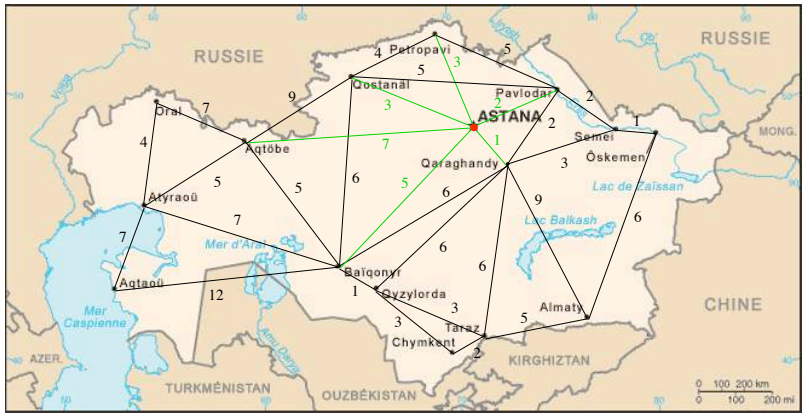
Principe :
On construit un sous-graphe en ajoutant arêtes et sommets les un après les autres. A chaque étape, on cherche l'arête *sortante* de plus petite valuation. Une arête est sortante si elle joint un sommet du sous-graphe à un sommet qui n'est pas dans le sous-graphe. On ajoute alors l'arête et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

Algorithme de Prim

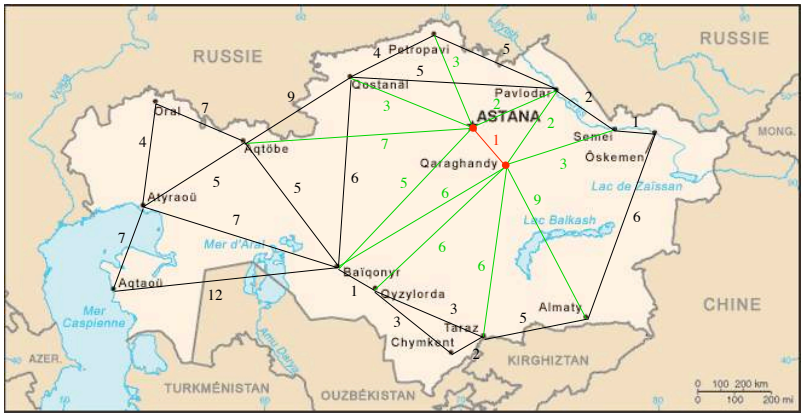
Initialisation :
$A =$ ensemble des arêtes du graphe G
$F = \emptyset$ (ensemble des arêtes de l'arbre couvrant)
$M = \{x_0\}$ (On **marque** un sommet quelconque de G)

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

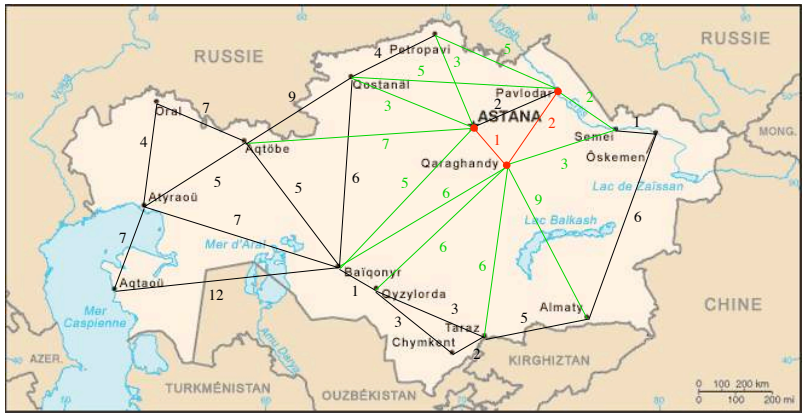
Exemple



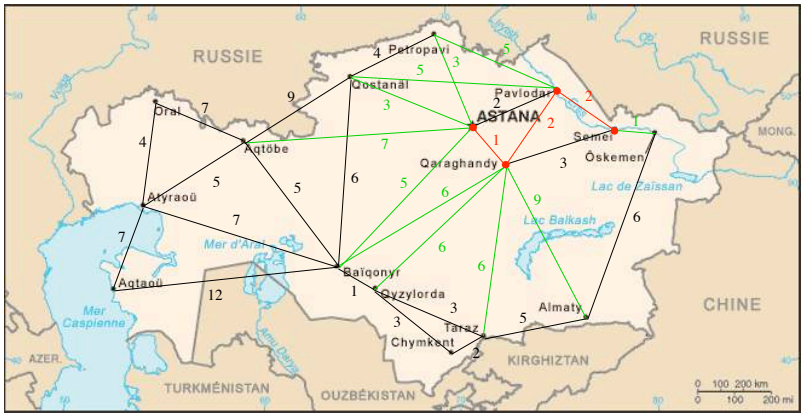
Exemple



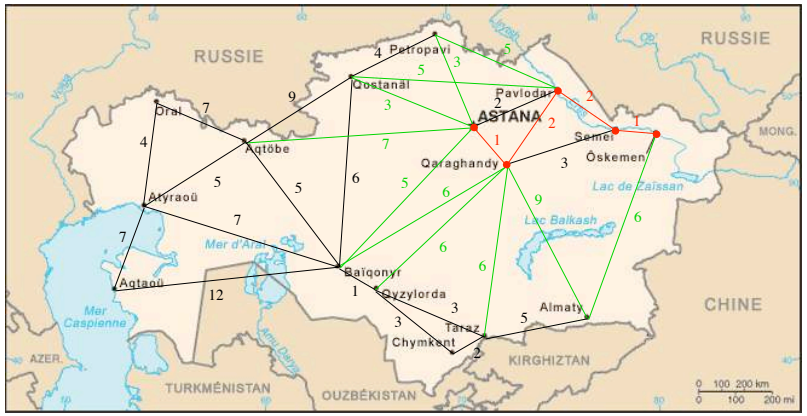
Exemple



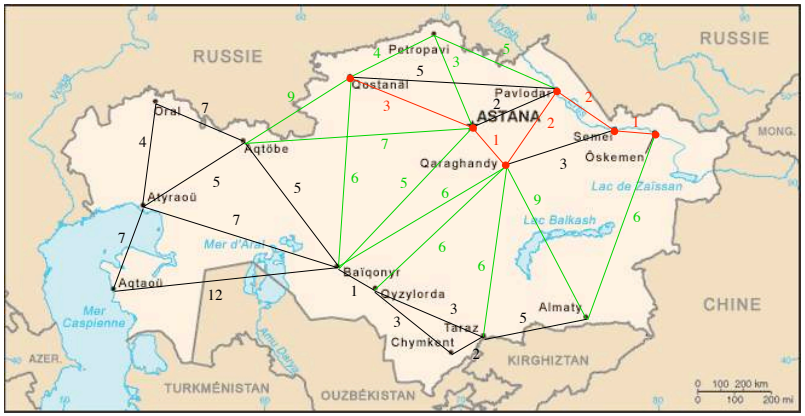
Exemple



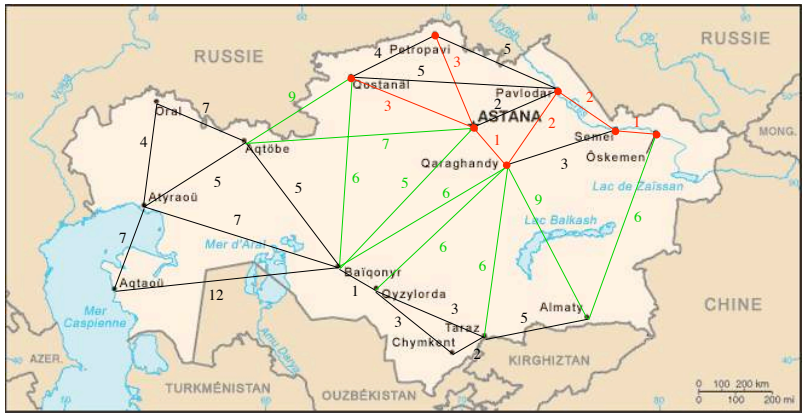
Exemple



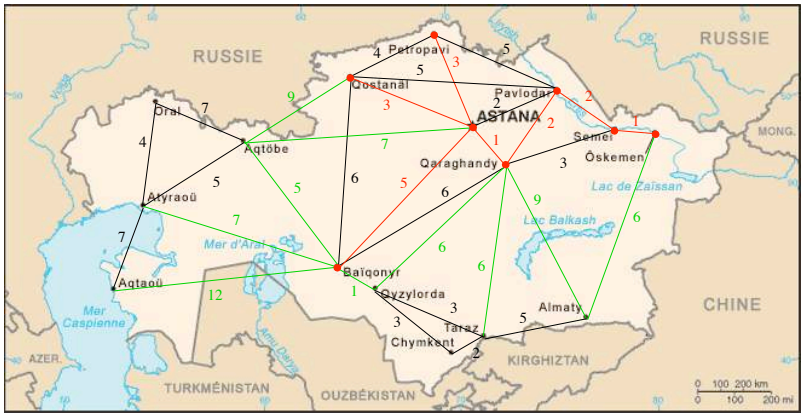
Exemple



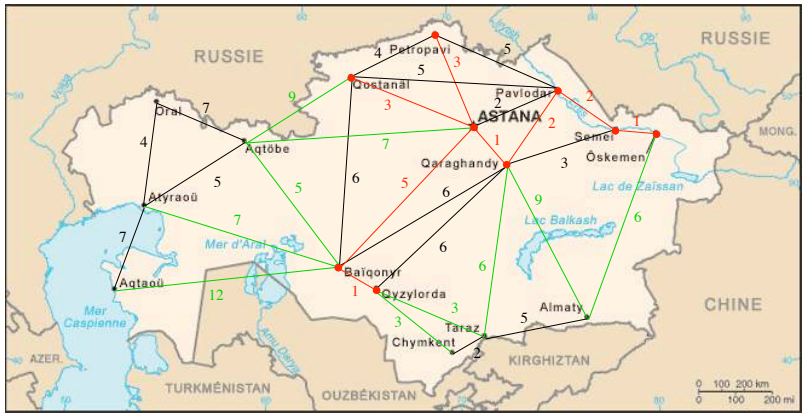
Exemple



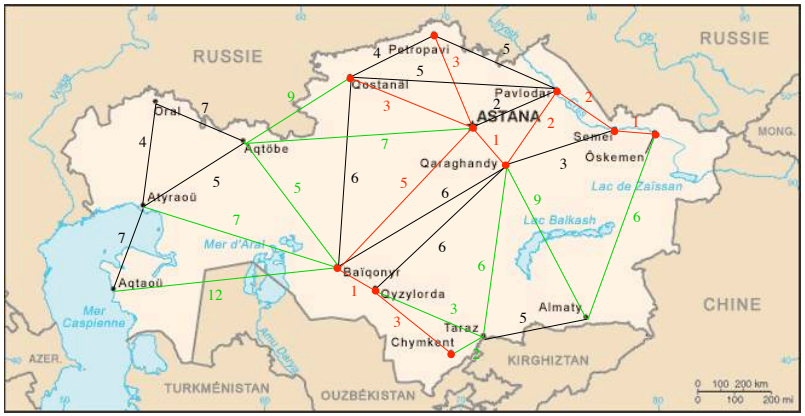
Exemple



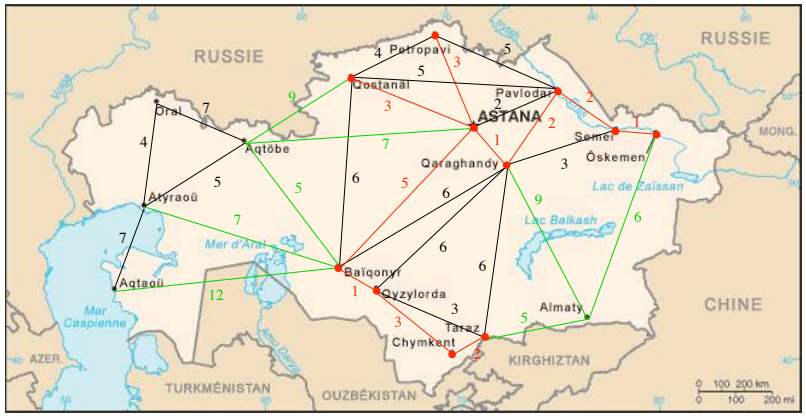
Exemple



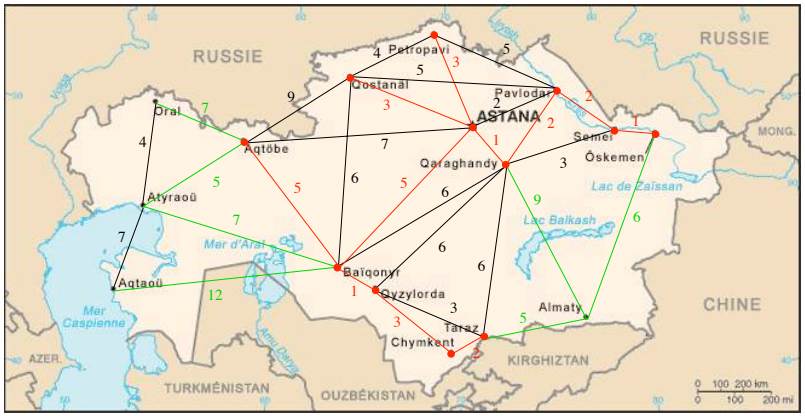
Exemple



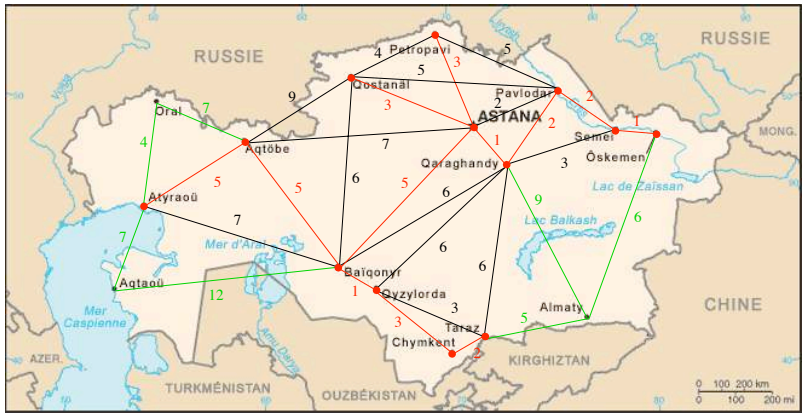
Exemple



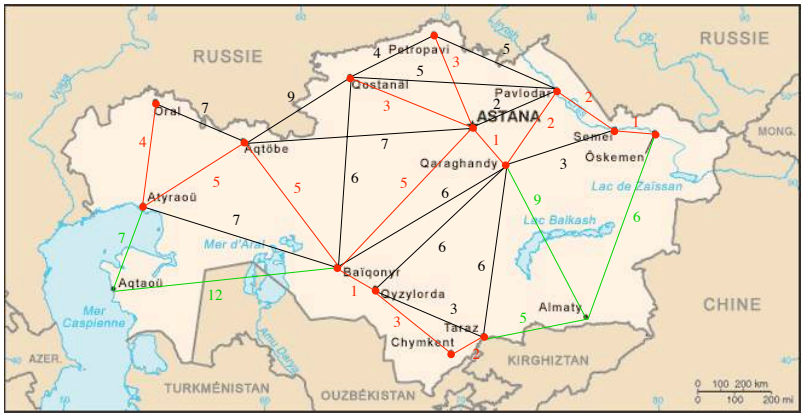
Exemple



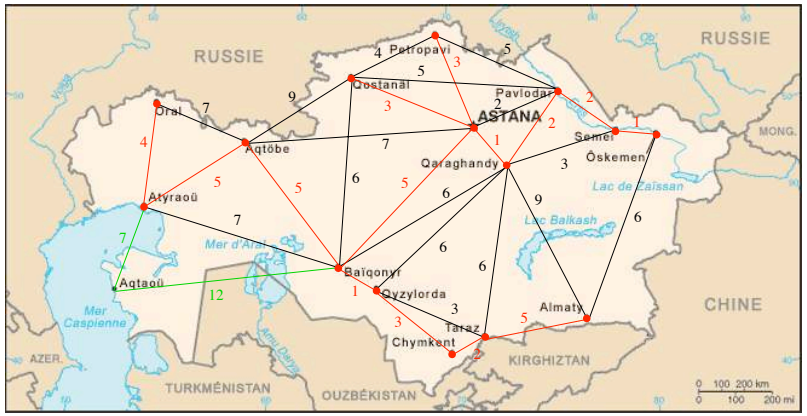
Exemple



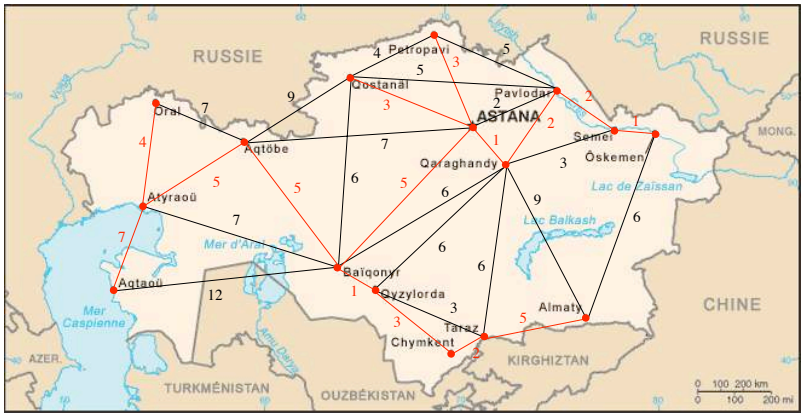
Exemple



Exemple



Exemple



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 visités 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.

Un jeu très simple

Voici un jeu qui se joue sur à deux, sur un graphe orienté.

- On place un pion sur un sommet du graphe.
- A tour de rôle, chaque joueur doit déplacer le pion en suivant un arc du graphe.
- Le premier joueur qui ne peut pas déplacer le pion a perdu.

Peut-on prévoir à l'avance qui va gagner ?

Jeux combinatoires

Ce "jeu" simple permet en fait de modéliser toute une classe de jeux, les **jeux combinatoires à information parfaite à deux joueurs**.

Cette classe de jeu comprend par exemple :

- les échecs, les dames, le jeu de go
- othello/reversi, puissance 4
- morpion, tic-tac-toe, jeu de petits carreaux
- etc...

Jeux combinatoires - suite

Les jeux combinatoires à information parfaite à deux joueurs sont tous les jeux :

- combinatoire, c'est-à-dire de réflexion (pas de jeux d'habileté type fléchettes) et sans hasard (ce qui exclut quasiment tous les jeux de cartes)
- à information complète : pas d'éléments cachés
- à deux joueurs, jouant à tour de rôle (ce qui exclut des jeux type pierre-feuille-ciseau)

Modélisation

Etant donné un jeu combinatoire à infomation parfaite à deux joueurs, on lui associe un graphe orienté de la façon suivante (on laisse de côté la possibilité de parties nulles) :

- l'ensemble des sommets est l'ensemble des états possibles du jeu
- deux sommets sont reliés par un arc s'il existe un coup amenant de la première position à la deuxième

L'étude des stratégies des joueurs se ramène à un problème de théories des graphes : trouver un **noyau** dans un graphe orienté donné.

Noyau d'un graphe

Définition

Un **noyau** *d'un graphe orienté $G = (S, A)$ est une partie N de l'ensemble des sommets S telle que :*

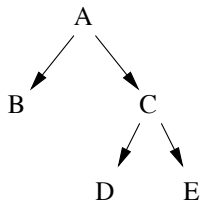
- *les sommets de N sont deux à deux non adjacents (on dit que N est* **stable***) :*

$$\forall x, y \in N,\ (x, y) \notin A$$

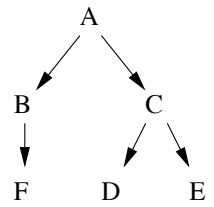
- *tout sommet qui n'est pas dans N a un successeur dans N :*

$$\forall x \in S \setminus N,\ \exists y \in N,\ (x, y) \in A$$

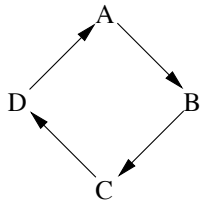
Exemples



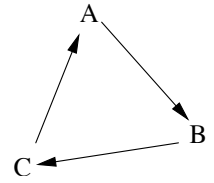
Noyau : $\{B, D, E\}$



Noyau : $\{A, D, E, F\}$



Noyau : $\{A, C\}$, et aussi $\{B, D\}$



Pas de noyau.

Existence d'un noyau

On a vu qu'un graphe ne possédait pas forcément de noyau. On a cependant le théorème suivant :

Théorème

Soit G un graphe orienté **sans circuit***. Alors G possède un unique noyau.*

Remarque : c'est un problème très difficile en général de trouver un noyau et s'il en existe dans un graphe donné (avec circuits).

Existence d'un noyau - suite

Démonstration :

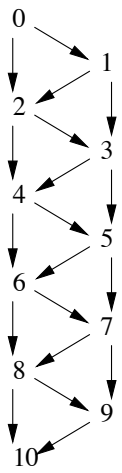
On commence par remarquer que :

- un graphe sans circuit possède au moins un **puits**, c'est-à-dire un sommet sans successeur,
- un noyau contient nécessairement tous les puits,
- et donc les prédécesseurs des puits ne peuvent pas être dans un noyau.

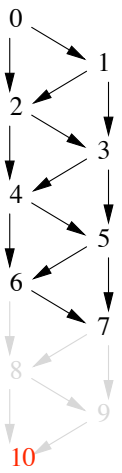
On construit le noyau N ainsi :

- on met tous les puits dans N,
- on supprime de G tous les puits et tous leurs prédécesseurs,
- on recommence avec le nouveau graphe obtenu (qui est toujours sans circuit).

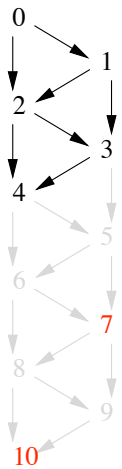
Exemple



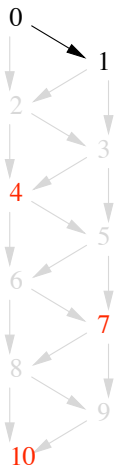
Exemple



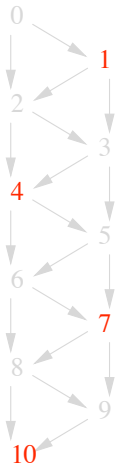
Exemple



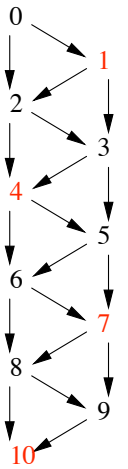
Exemple



Exemple



Exemple



Noyau : $\{1, 4, 7, 10\}$

Noyau et stratégie gagnante

Soit G un graphe orienté possédant un noyau N.

Théorème

Un joueur, dont la position au début de son tour n'est pas dans N, a une stratégie non-perdante.

Démonstration :

On va montrer qu'un tel joueur J ne peut pas perdre (s'il ne fait pas d'erreurs...).
Le joueur J joue un coup qui le place dans N (c'est toujours possible par définition du noyau).
Si cette nouvelle position est un puits, le joueur J a gagné ; sinon, le coup du joueur opposé ramène forcément en dehors du noyau, et le joueur J peut à nouveau jouer un coup qui le place dans N.

Existence de stratégies gagnantes

Corollaire

Soit un jeu combinatoire à information parfaite à deux joueurs, sans possibilité de partie nulle, et dont le graphe associé est sans circuit.

Alors un des deux joueurs a une stratégie gagnante.

Démonstration :

Le graphe associé possède un unique noyau N. On regarde si la position de départ du jeu est dans N : si c'est le cas, le deuxième joueur a une stratégie gagnante ; sinon, c'est le premier joueur.

Remarques

- La plupart des jeux sont sans circuit (pas de possibilité de partie sans fin).
  Exemples : règle sur les échecs perpétuel aux échecs, règle du ko au go...
- Il est facile d'adapter l'énoncé pour prendre en compte les nulles :
  un des deux joueurs a une stratégie non-perdante.
- Le graphe d'un jeu est en général tellement énorme qu'il est impossible de déterminer une stratégie gagnante (et heureusement).
  Un jeu est **résolu** quand une stratégie gagnante a été déterminée (exemple de jeu résolu : puissance 4).

Théorie des graphes

G. Montcouquiol

IUT Orsay

2006-2007