Graphes

2. Problèmes usuels

Solen Quiniou

solen.quiniou@univ-nantes.fr

IUT de Nantes

Année 2023-2024 – BUT 1 (Semestre 2)

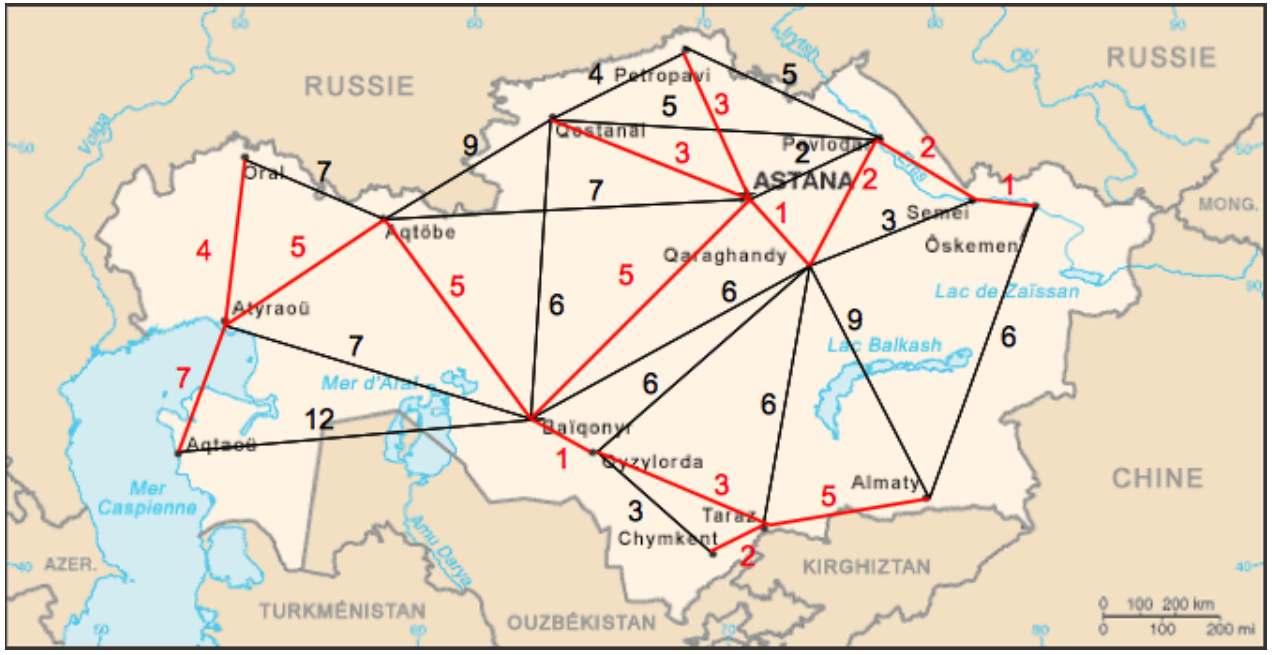
[Mise à jour du 17 janvier 2024]

Plan du cours

Plan du cours

Arbres : exemple de problème traité

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 fibre optique.
Après une étude préliminaire, vous estimez les coûts suivants de connexion entre les villes (en millions de KZT) :



Exemple tiré de `www.math.u-psud.fr/~montcouq/Enseignements/Apprentis/arbres.pdf`

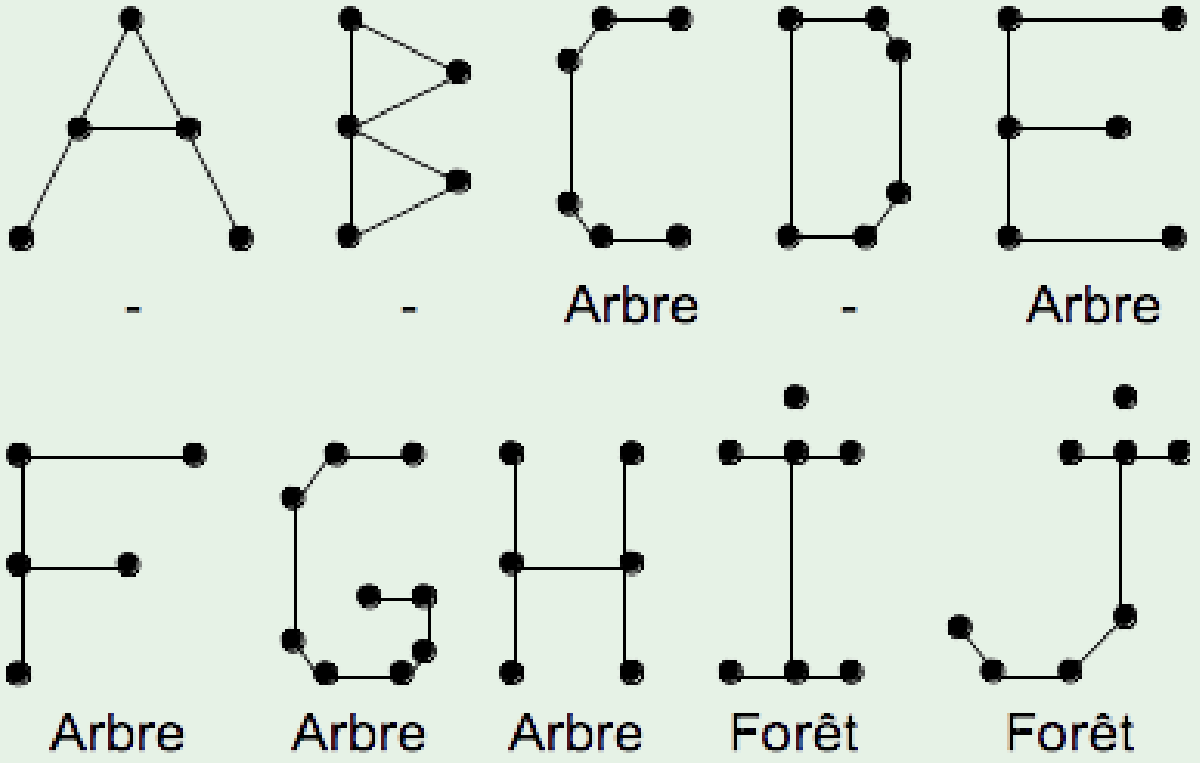
Quelles liaisons permettent de connecter toutes les villes à moindre coût ?

Arbres et forêts

Définitions

- Un **arbre** est un graphe connexe et sans cycle simple.
- Une **forêt** est un graphe sans cycle simple. Chacune de ses composantes connexes est un arbre.

Exemples



Caractérisation des arbres (1)

Le théorème fondamental suivant donne six caractérisations alternatives des arbres.

Théorème

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

1. Le graphe G est connexe et sans cycle simple ;
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 rend non connexe ;
4. Le graphe G est sans cycle simple et a $n-1$ arêtes ;
5. Le graphe G est sans cycle simple 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 (2)

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 simple à n sommets a toujours **au plus** $n-1$ arêtes.

Arbres enracinés

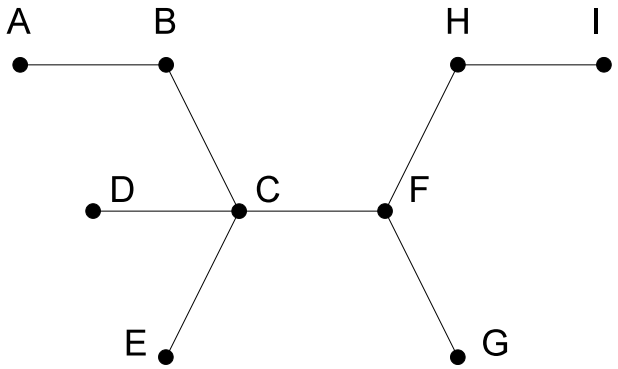
Les arbres utilisés en algorithmique ont le plus souvent une orientation et un sommet qui joue un rôle particulier, la racine.

Définition

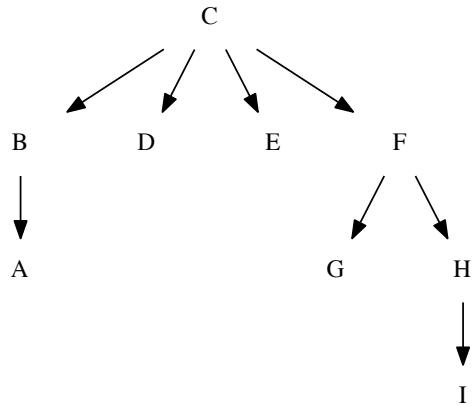
Un graphe non orienté est un **arbre enraciné** s'il est connexe, sans cycle simple et si un sommet particulier, la **racine**, a été distingué.

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 appelé **arborescence**.

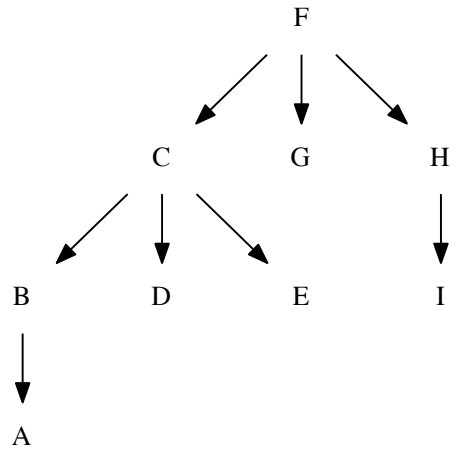
Exemple



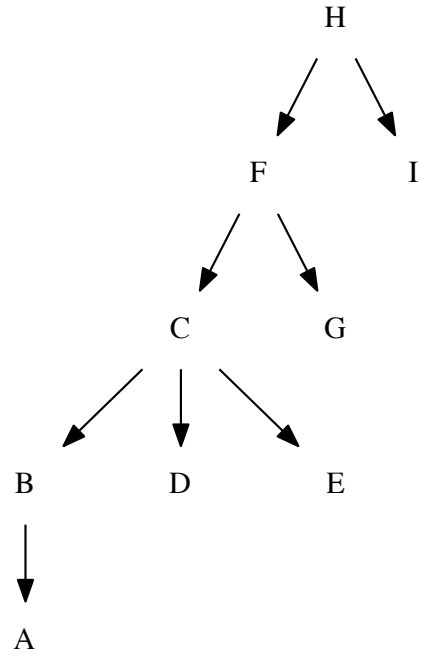
Racine en *C*



Racine en *F*



Racine en *H*

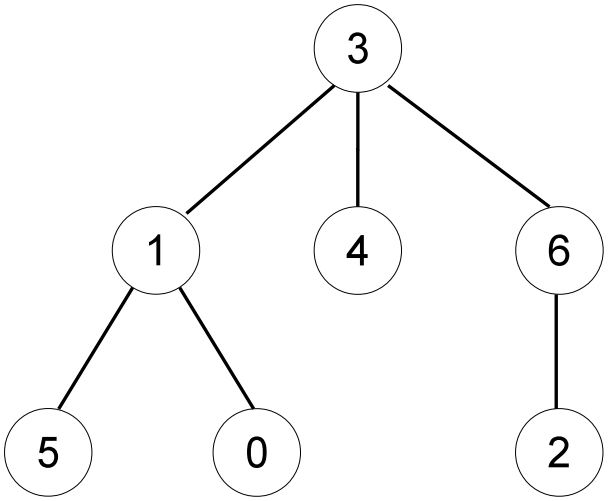


Terminologie

Définitions

- Si x est un sommet distinct de la racine r, les sommets suivants sont les **fils** de x et l'unique antécédent de x est son **père**. Les ascendants de x sont aussi appelés ses **ancêtres**.
- Les sommets sont appelés des **nœuds**. Les nœuds qui n'ont pas de sommet suivant sont appelés des **feuilles**.
- Le **degré** d'un sommet est égal au nombre de ses fils (une feuille a un degré nul).
- La **profondeur** (ou hauteur) d'un sommet est égale au nombre de ses ancêtres.

Exemple



- Racine r : sommet 3
- Fils du sommet 1 : sommets 0 et 5
- Père du sommet 1 : sommet 3
- Ancêtres du sommet 5 : sommets 1 et 3

- Feuilles : sommets 5, 0 et 2

- Degré du sommet 1 : 2
- Degré du sommet 5 : 0

- Profondeur du sommet 1 : 1

Arbres couvrants de poids minimum

Définition

Soit G un graphe valué non orienté et connexe.

- Un **arbre couvrant** est un sous-graphe couvrant de G (c'est-à-dire contenant tous les sommets), connexe et sans cycle simple. Son **poids** est la somme des valuations de ses arêtes.

- Un **arbre couvrant de poids minimum** est un arbre couvrant dont le poids est le plus petit parmi les arbres couvrants de G. Si toutes les arêtes ont des valuations positives, son poids est le plus petit parmi tous les sous-graphes connexes couvrants de G.

Recherche d'arbres couvrants de poids minimum

Nous allons étudier deux algorithmes classiques permettant de trouver un arbre couvrant de poids minimum dans un graphe valué G.
Nous supposerons toujours que le graphe est non orienté et que les valuations des arêtes sont toutes positives.

Dans les deux cas, nous allons construire l'arbre couvrant petit à petit, en s'assurant à chaque étape

- que l'arbre reste couvrant et sans cycle (algorithme de Kruskal) ;
- ou que l'arbre reste connexe et sans cycle (algorithme de Prim).

Ce sont deux exemples d'**algorithmes gloutons** : à chaque étape, nous faisons le meilleur choix courant, dans le but d'obtenir la meilleure solution globale.

Algorithme de Kruskal (1)

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 cycle.

Algorithme de Kruskal (2)

Données : Graphe $G = (S, A)$ non orienté et valué, sans valuations négatives

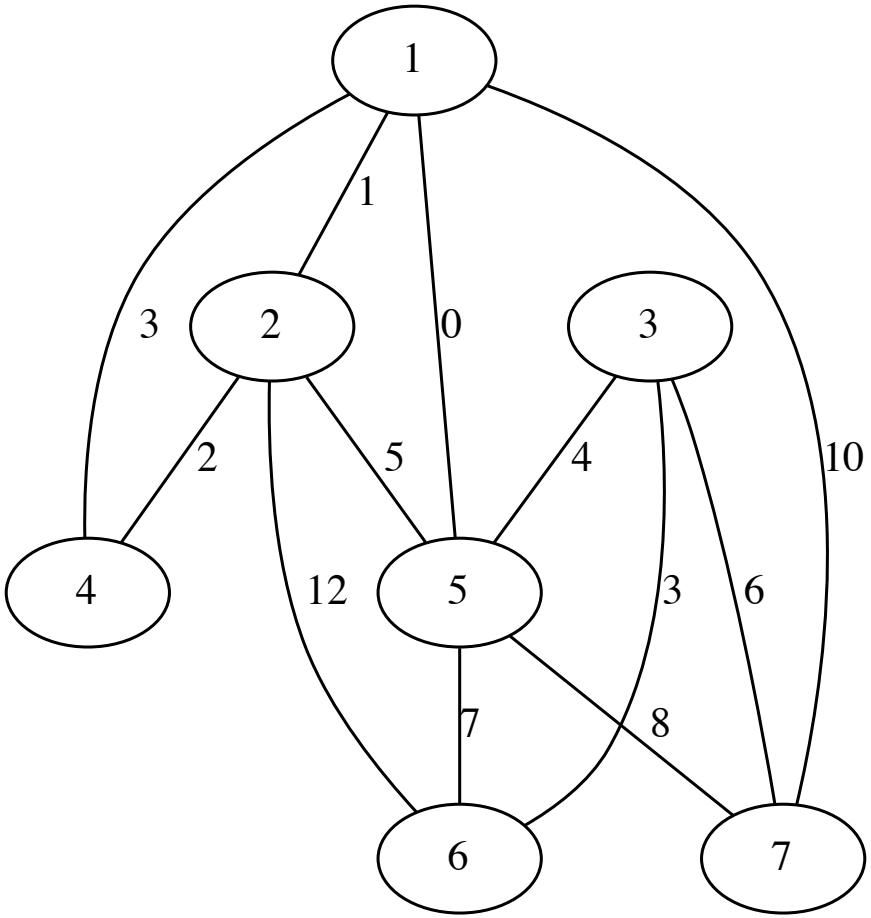
Résultat : Arbre couvrant de poids minimum $G_a = (S, F)$

$F = \emptyset$; `/* ensemble des arêtes de l'arbre couvrant */`

T = ensemble A des arêtes de G, triées par valuations croissantes ;

pour chaque $a \in T$ *(arêtes parcourues dans l'ordre du tri)* **faire**
  si $F \cup \{a\}$ *est sans cycle* **alors**
    $F = F \cup \{a\}$;
  fin
fin

Exemple d'utilisation de l'algorithme de Kruskal



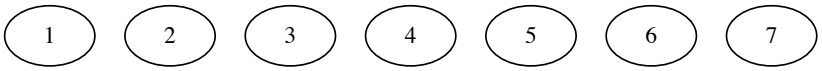
Graphe de départ

Exemple - choix des arêtes

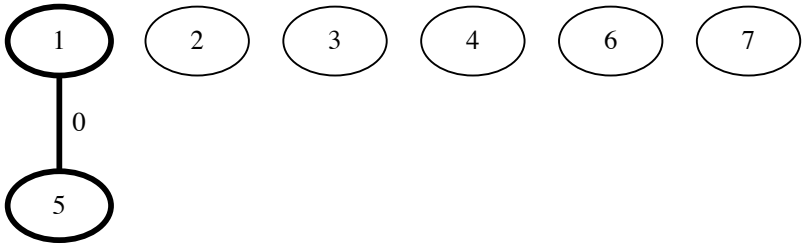
- Tri des arêtes par valuation croissante : $\{1,5\}$, $\{1,2\}$, $\{2,4\}$, $\{1,4\}$, $\{3,6\}$, $\{3,5\}$, $\{2,5\}$, $\{3,7\}$, $\{5,6\}$, $\{5,7\}$, $\{1,7\}$, $\{2,6\}$

- Boucle principale
  1. Ajout de $\{1,5\}$
  2. Ajout de $\{1,2\}$
  3. Ajout de $\{2,4\}$
  4. Pas d'ajout de $\{1,4\}$ car cela créerait le cycle simple $[1,2,4,1]$
  5. Ajout de $\{3,6\}$
  6. Ajout de $\{3,5\}$
  7. Pas d'ajout de $\{2,5\}$ car cela créerait le cycle simple $[1,2,5,1]$
  8. Ajout de $\{3,7\}$
  9. Pas d'ajout de $\{5,6\}$ car cela créerait le cycle simple $[3,5,6,3]$
  10. Pas d'ajout de $\{5,7\}$ car cela créerait le cycle simple $[3,5,7,3]$
  11. Pas d'ajout de $\{1,7\}$ car cela créerait le cycle simple $[1,5,3,7,1]$
  12. Pas d'ajout de $\{2,6\}$ car cela créerait le cycle simple $[1,2,6,3,5,1]$

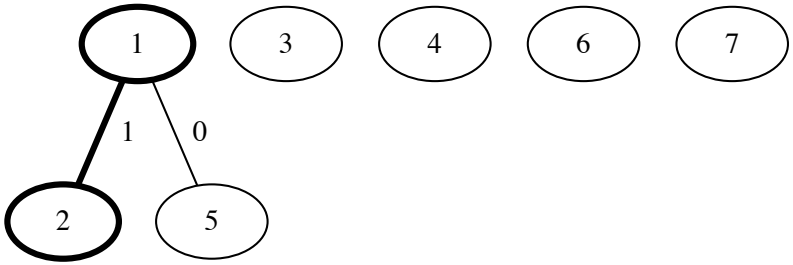
Exemple - arbre obtenu à chaque étape (1)



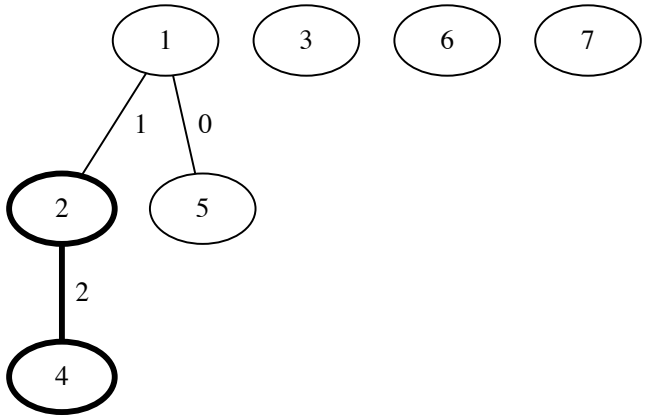
Initialisation de l'arbre



1. Ajout de l'arête $\{1, 5\}$

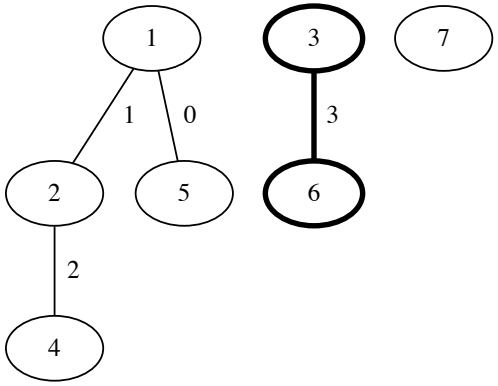


2. Ajout de l'arête $\{1, 2\}$

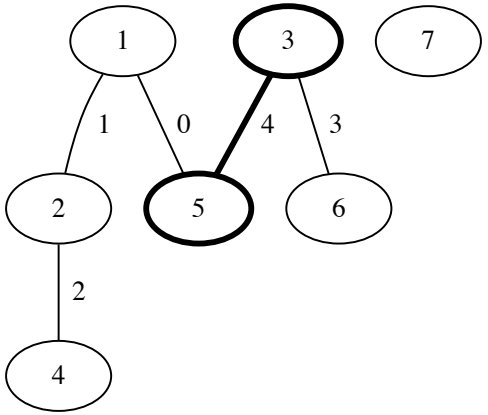


3. Ajout de l'arête $\{2, 4\}$

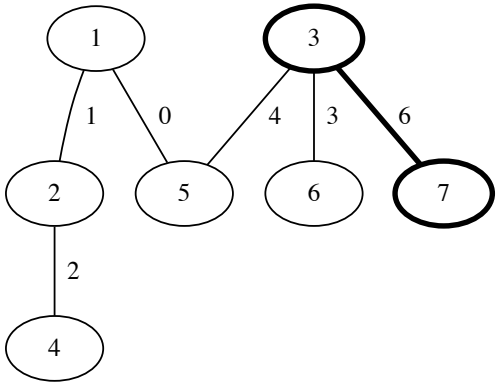
Exemple - arbre obtenu à chaque étape (2)



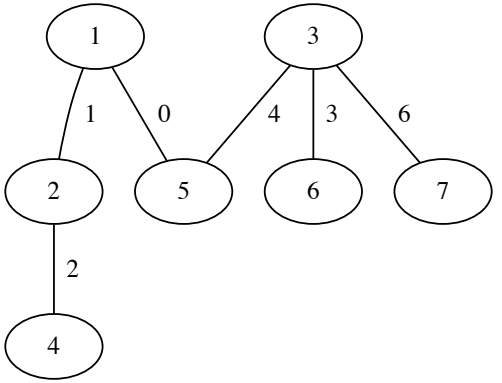
5. Ajout de l'arête $\{3, 6\}$



6. Ajout de l'arête $\{3, 5\}$



8. Ajout de l'arête $\{3, 7\}$



Arbre de poids couvrant minimal (poids = 16)

Algorithme de Kruskal (3)

Commentaires

- A chaque étape de l'algorithme, on obtient une forêt couvrante (c'est-à-dire 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 (1)

Principe

- On construit un sous-graphe en ajoutant des arêtes et des sommets les uns 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 avec un sommet qui n'est pas dans le sous-graphe. On ajoute alors, au sous-graphe, l'arête sortante et le sommet qu'elle joint.
- On termine dès que l'on a sélectionné $n - 1$ arêtes.

Algorithme de Prim (2)

Données : Graphe $G = (S, A)$ non orienté et valué, sans valuations négatives

Résultat : Arbre couvrant de poids minimum $G_a = (S, F)$

$F = \emptyset$; `/* 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* **faire**

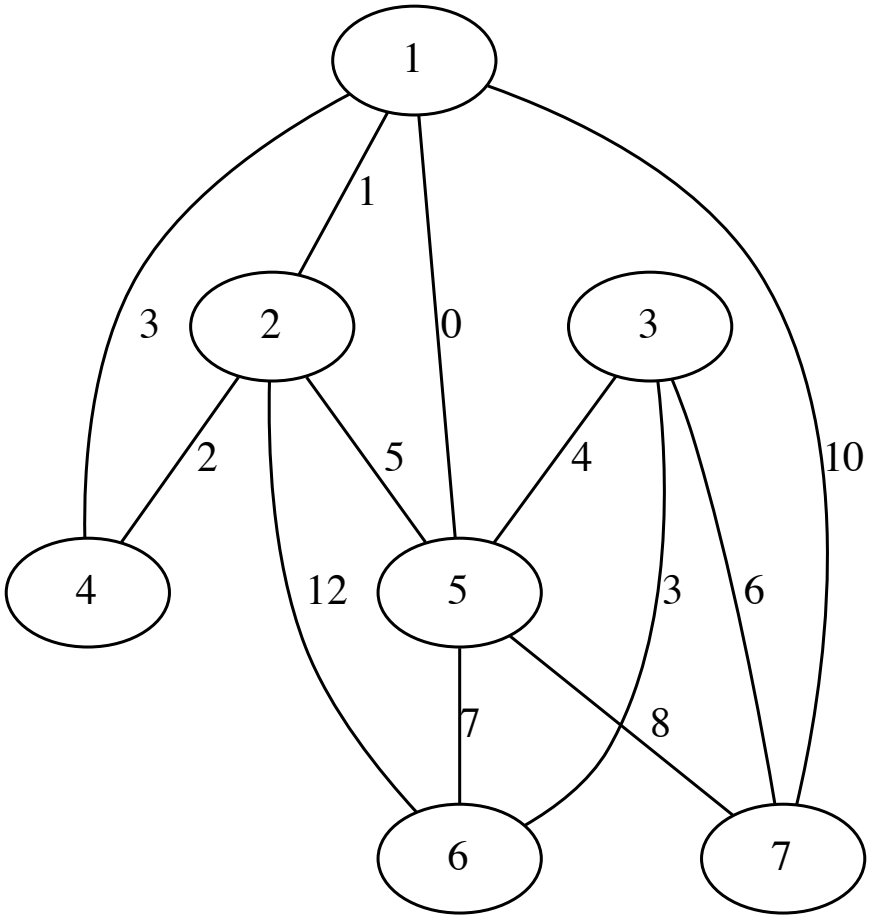
  Chercher l'arête sortante $a = \{x, y\}$ de plus petite valuation ; `/* a arête sortante :` $x \in M$ `et` $y \notin M$ `*/`

  $M = M \cup \{y\}$;

  $F = F \cup \{a\}$;

fin

Exemple d'utilisation de l'algorithme de Prim

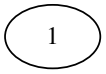


Graphe de départ

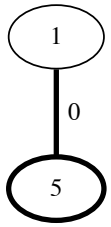
Exemple - choix des arêtes

a	M
-	$\{1\}$
$\{1,5\}$	$\{1,5\}$
$\{1,2\}$	$\{1,2,5\}$
$\{2,4\}$	$\{1,2,4,5\}$
$\{3,5\}$	$\{1,2,3,4,5\}$
$\{3,6\}$	$\{1,2,3,4,5,6\}$
$\{3,7\}$	$\{1,2,3,4,5,6,7\}$

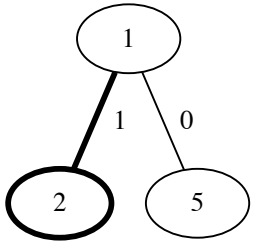
Exemple - arbre obtenu à chaque étape (1)



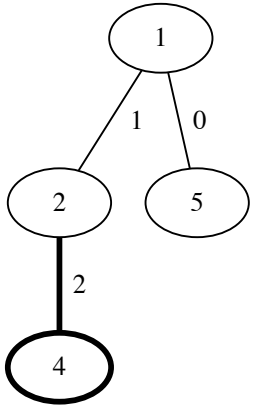
Initialisation de l'arbre



1. Ajout de l'arête $\{1, 5\}$

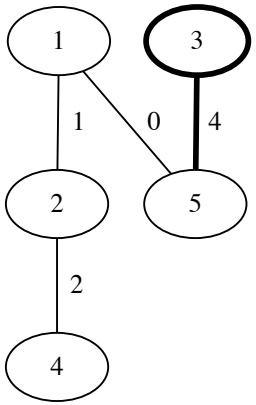


2. Ajout de l'arête $\{1, 2\}$

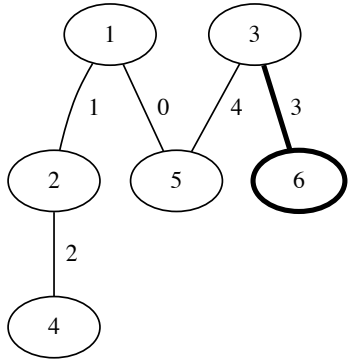


3. Ajout de l'arête $\{2, 4\}$

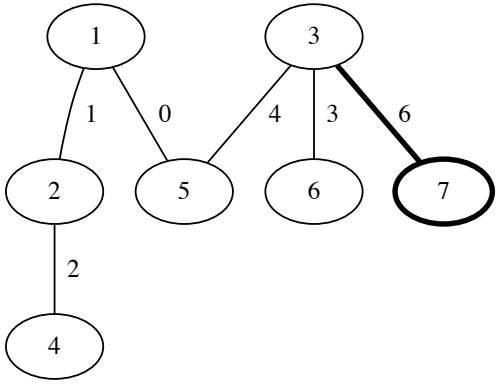
Exemple - arbre obtenu à chaque étape (2)



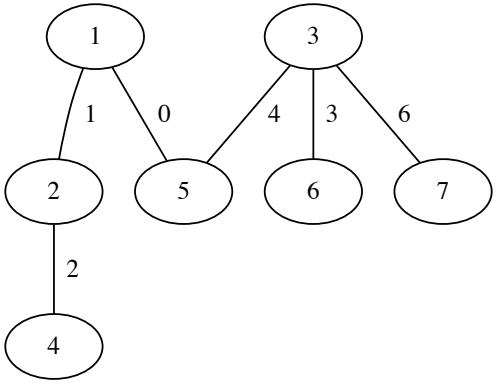
4. Ajout de l'arête $\{3, 5\}$



5. Ajout de l'arête $\{3, 6\}$



7. Ajout de l'arête $\{3, 7\}$



Arbre de poids couvrant minimal (poids = 16)

Algorithme de Prim (3)

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.

Plan du cours

Coloration de graphes : exemples

- **Allouer des fréquences GSM**
  - *Sommets* : les émetteurs radio ;
  - *Arête* entre x et y : le signal de x perturbe y ou réciproquement ;

  $\rightarrow$ *Couleurs* : les fréquences radio.

- **Organiser une session d'examens** en un minimum de jours sans créer de conflits pour des étudiants inscrits à plusieurs examens
  - *Sommets* : les examens ;
  - *Arête* entre x et y : les examens x et y ne peuvent pas avoir lieu le même jour ;

  $\rightarrow$ *Couleurs* : les jours des examens.

Stables et cliques

Définitions

Soit $G = (S, A)$ un graphe.

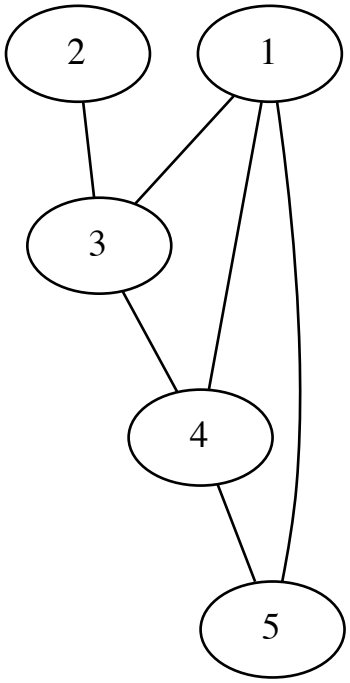
- Un sous-ensemble $V \subset S$ est un **stable** de G s'il ne comporte que des sommets non adjacents entre eux. Autrement dit, V est un sous-graphe sans arête / arc.
- Le **nombre de stabilité** de G est le cardinal du plus grand stable ; il est noté $\alpha(G)$.

Définitions

Soit $G = (S, A)$ un graphe.

- Une **clique d'ordre** k de G est un sous-graphe simple et complet composé de k sommets, c'est-à-dire que tous les sommets sont adjacents entre eux.
- L'ordre de la plus grande clique de G est noté $\omega(G)$.

Exemple



- Exemples de stables : $\{1,2\}$, $\{2,4\}$, $\{2,5\}$, $\{3\}\ldots$

$\longrightarrow$ Nombre de stabilité : $\alpha(G) = 2$

- Exemples de cliques d'ordre 3 : $\{1,3,4\}$ et $\{1,4,5\}$

$\longrightarrow$ Ordre de la plus grande clique : $\omega(G) = 3$

Coloration des sommets d'un graphe

Définitions

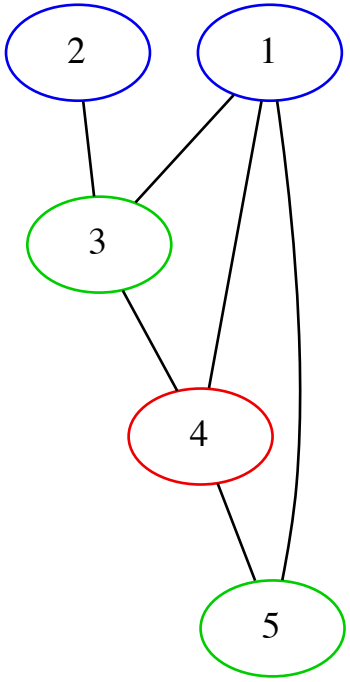
- **Colorier les sommets** d'un graphe G consiste à affecter une couleur à chaque sommet du graphe de telle sorte que deux sommets adjacents quelconques aient des couleurs différentes. Une coloration avec k couleurs est ainsi une partition de l'ensemble des sommets en k stables.
- Le **nombre chromatique** de G est le nombre minimal de couleurs nécessaires pour colorier les sommets du graphe ; il est noté $\gamma(G)$.

Encadrement du nombre chromatique

Soit $G = (S, A)$ un graphe. On a alors :

$$\omega(G) \leq \gamma(G) \leq n + 1 - \alpha(G).$$

Exemple



- Partition des sommets en 3 stables : $\{1, 2\}$, $\{3, 5\}$ et $\{4\}$.

$\longrightarrow$ Comme l'ordre de la (des) plus grandes cliques est $\omega(G) = 3$, on ne peut pas utiliser moins de 3 couleurs.

$\longrightarrow$ Remarque : on aurait aussi pu colorier le sommet 2 en rouge ; ainsi, il n'existe pas forcément une coloration minimale unique.

Algorithme de coloration des sommets

Remarque

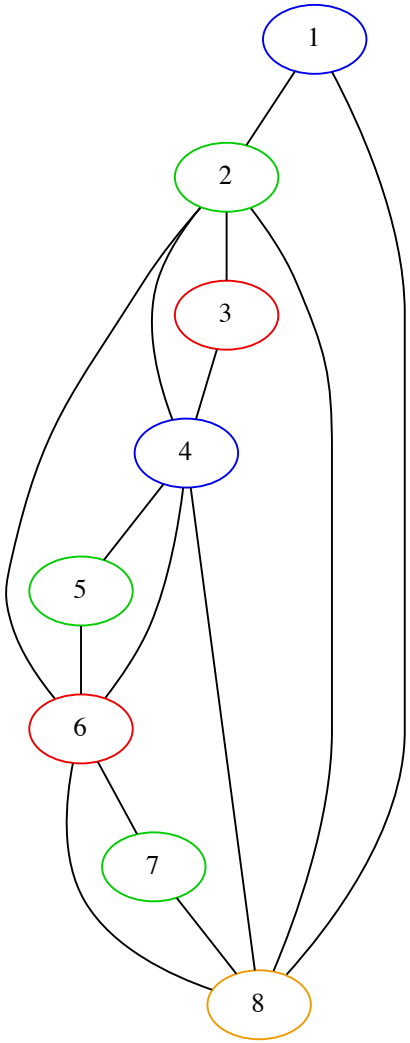
La coloration des sommets d'un graphe est un « problème difficile » en ce sens que, si le nombre de sommets est grand et si le graphe comporte beaucoup d'arêtes, on ne connaît pas d'algorithme performant pour déterminer la solution minimale. En revanche, l'algorithme suivant donne une solution satisfaisante même s'il ne donne pas forcément la solution minimale.

Algorithme 1 : Algorithme de Welsh et Powell

Classer les sommets dans l'ordre décroissant de leur degré;
Attribuer à chacun des sommets son numéro d'ordre dans la liste précédente;

tant que *tous les sommets du graphe ne sont pas coloriés* **faire**
    En parcourant la liste dans l'ordre, attribuer une couleur c_i non encore utilisée au premier sommet non encore colorié;
    Attribuer cette couleur c_i aux sommets non encore coloriés et non adjacents à un sommet de cette couleur;
fin

Exemple



Sommet	1	2	3	4	5	6	7	8
Degré	2	5	2	5	2	5	2	5
Ordre	5	1	6	2	7	3	8	4
Couleur	2	1	3	2	1	3	1	4

Coloration des arêtes d'un graphe

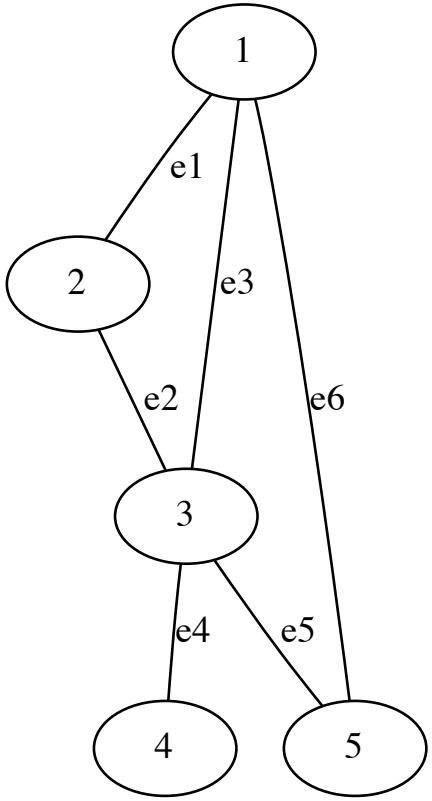
Définitions

- **Colorier les arêtes** d'un graphe G consiste à affecter une couleur à chaque arête du graphe de telle sorte que deux arêtes adjacentes quelconques aient des couleurs différentes.
- L'**indice chromatique** de G est le nombre minimal de couleurs nécessaires pour colorier les arêtes du graphe.
- Le **graphe aux arêtes** (ou **graphe adjoint**) de G, noté G', est défini comme suit :
  - chaque arête de G est représenté par un sommet dans G' ;
  - deux sommets sont reliés dans G' ssi les deux arêtes correspondantes dans G sont adjacentes.

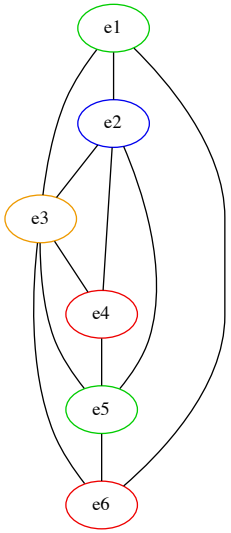
Algorithme de coloration des arêtes

L'algorithme de coloration des sommets peut être appliqué sur le graphe aux arêtes, G', pour colorier ses sommets.
Une fois la coloration des sommets réalisée sur G', il suffit de colorier les arêtes de G de la même couleur que les sommets correspondants de G'.

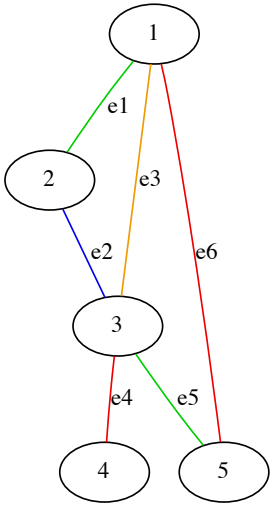
Exemple



- Graphe aux arêtes colorié



- Graphe initial avec arêtes coloriées



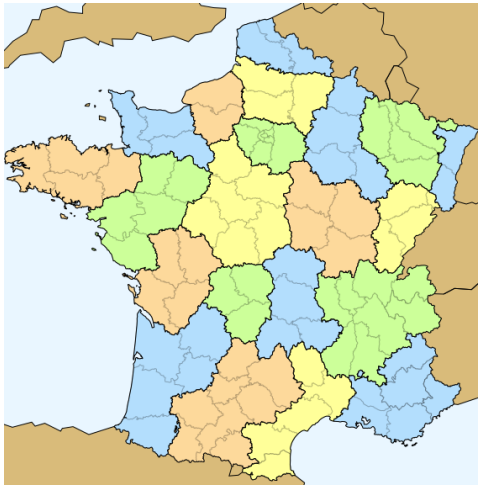
Théorème des quatre couleurs

Définition

Un **graphe planaire** est un graphe que l'on peut dessiner sans que ses arêtes ne se croisent.

Théorème

On peut colorier les sommets d'un graphe planaire (sans boucle) en utilisant au plus quatre couleurs, de telle sorte que toutes les arêtes aient des extrémités de couleurs différentes.



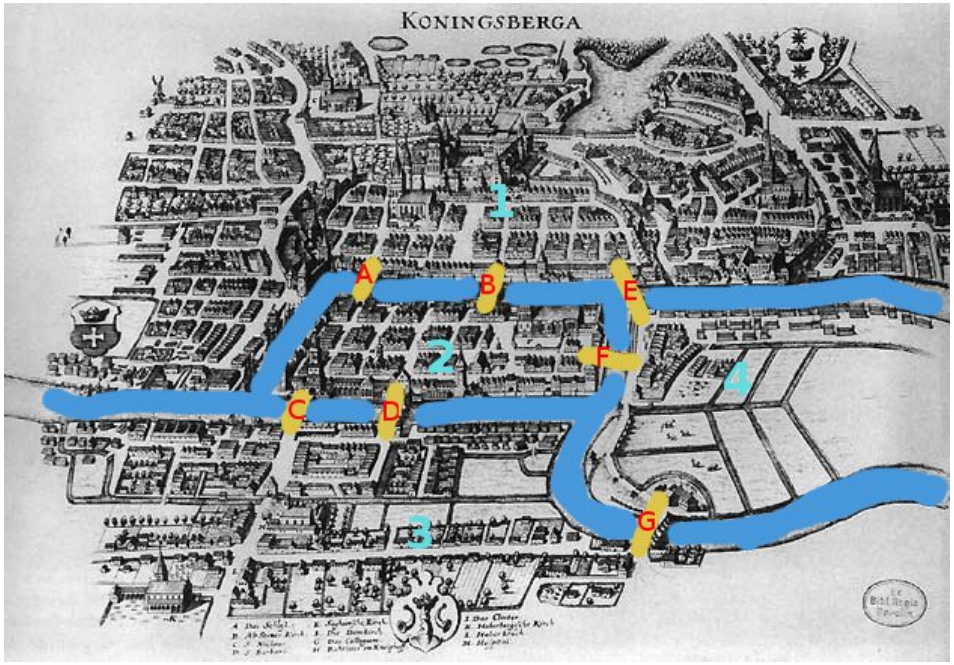
http://commons.wikimedia.org/wiki/File:Carte_France_geo_4_couleurs.png

- Ce théorème a été formulé par Guthrie en 1852 pour colorier une carte d'Angleterre.
- La preuve du théorème n'a été faite qu'en 1976 par Appel et Haken à partir de 1 478 cas critiques dont l'étude a nécessité l'utilisation d'un ordinateur, pour la première fois.

Plan du cours

Graphes eulériens : introduction

Au 18ème siècle, un casse-tête est populaire parmi les habitants de Königsberg : est-il possible de se promener dans la ville en ne passant qu'une seule fois par chacun des 7 ponts de Königsberg ?



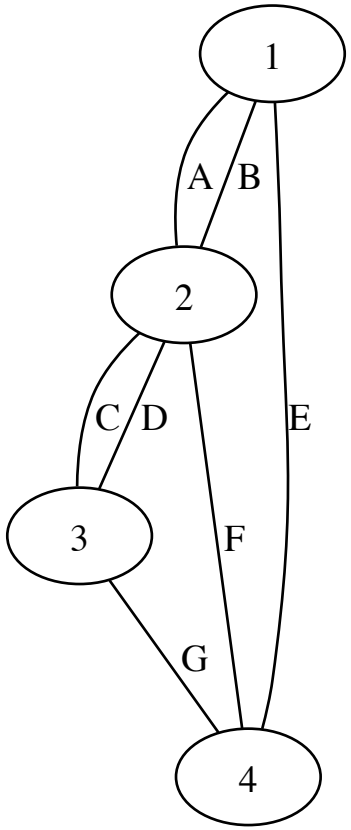
http://epik.scientifik.fr/wp-content/uploads/2010/08/koninsberg1.jpg

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.

Graphes eulériens : introduction

Le problème se reformule ainsi en terme de graphe :

existe-t-il un cycle qui passe exactement une fois par toutes les arêtes du graphe ci-dessous ?



Graphes eulériens

Définitions

Soit G un graphe non orienté.

- Un **cycle eulérien** de G est un cycle qui passe une et une seule fois par chaque arête de G. Un graphe est alors dit **eulérien** s'il possède un cycle eulérien.

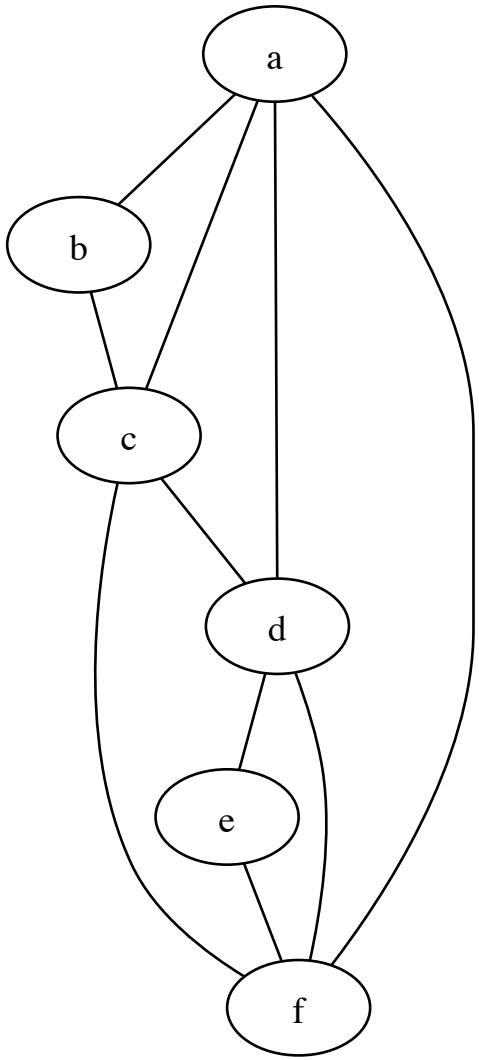
- Une **chaîne eulérienne** de G est une chaîne qui passe une et une seule fois par chaque arête de G. Un graphe est alors dit **semi-eulérien** s'il ne possède que des chaînes eulériennes.

Les mêmes notions peuvent être définies pour un graphe orienté G : **circuit eulérien** et **chemin eulérien**.

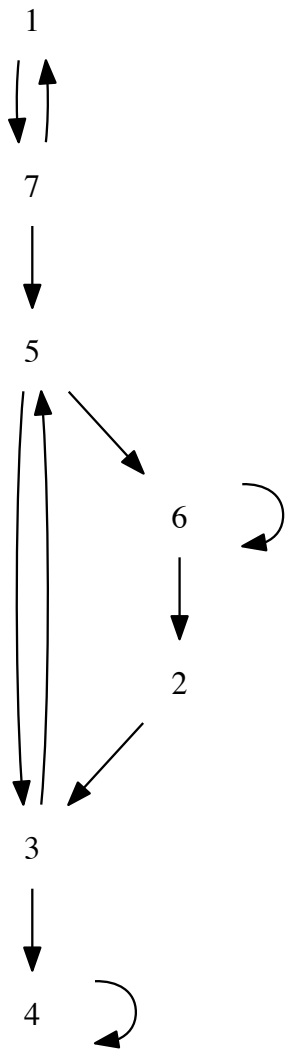
Remarque

On peut dire qu'un graphe est eulérien s'il est possible de dessiner le graphe sans lever le stylo et sans passer deux fois sur la même arête.

Exemples de graphes eulériens



[*d*, *c*, *b*, *a*, *c*, *f*, *a*, *d*, *f*, *e*, *d*] cycle eulérien donc graphe eulérien.



[7, 1, 7, 5, 6, 6, 2, 3, 5, 3, 4, 4] chemin eulérien mais pas de circuit eulérien donc graphe semi-eulérien.

Théorème d'Euler

Théorème

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

1. le graphe est connexe ;
2. $\forall x \in S$, $d(x)$ est pair (cas non orienté)

   $\forall x \in S$, $d^+(x) = d^-(x)$ (cas orienté).

Calcul d'un circuit eulérien

Données : Graphe connexe $G = (S, A)$ vérifiant les conditions du théorème d'Euler
Résultat : C_e le circuit eulérien construit
On choisit un sommet quelconque s de S;
On détermine un circuit simple C_1 de s à s;

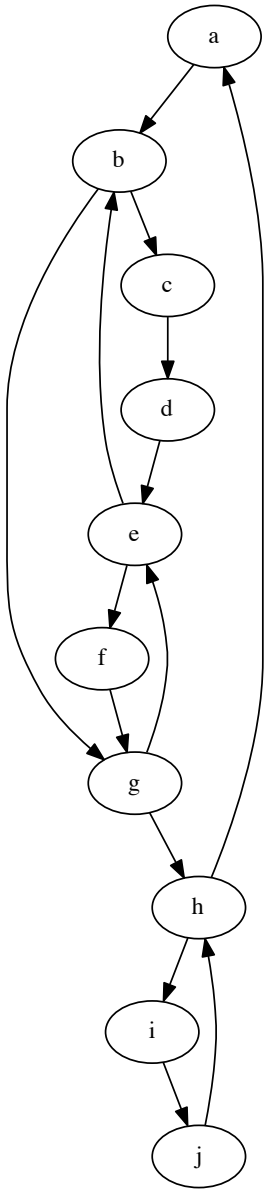
$A_1 = A \setminus E_1$; /* E_1 est l'ensemble des arcs utilisés dans C_1 */
$k = 1$;

tant que $A_k \neq \emptyset$ **faire**
    On choisit dans $G_k = (S, A_k)$ un sommet quelconque s_k de C_k;
    On détermine un circuit simple C_{k+1} de s_k à s_k;
    On insère les arcs du circuit C_{k+1} au circuit C_k, au niveau du sommet s_k;

    $A_{k+1} = A_k \setminus E_{k+1}$; /* E_{k+1} est l'ensemble des arcs utilisés dans C_{k+1} */
    $C_{k+1} = C_k$;
    $k = k + 1$;
fin

$C_e = C_k$;

Exemple de calcul de circuit eulérien



- $s_1 = a$, $C_1 = [a, b, g, h, a]$
- $s_2 = b$, $C_2 = [a, \mathbf{b,c,d,e,b}, g, h, a]$
- $s_3 = e$, $C_3 = [a, b, c, d, \mathbf{e,f,g,e}, b, g, h, a]$
- $s_4 = h$, $C_4 = [a, b, c, d, e, f, g, e, b, g, \mathbf{h,i,j,h}, a]$

- Circuit eulérien :
  $[a, b, c, d, e, f, g, e, b, g, h, i, j, h, a]$

Chaînes et chemins eulériens

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 d'un graphe G admettant un cycle eulérien, le nouveau graphe admet encore une chaîne eulérienne.

Chaînes et chemins eulériens

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)** ssi les deux conditions suivantes sont vérifiées :

1. le graphe 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, $d^+(x) = d^-(x)$ et les deux derniers sommets vérifient $d^+(x) = d^-(x) \pm 1$ (cas orienté).

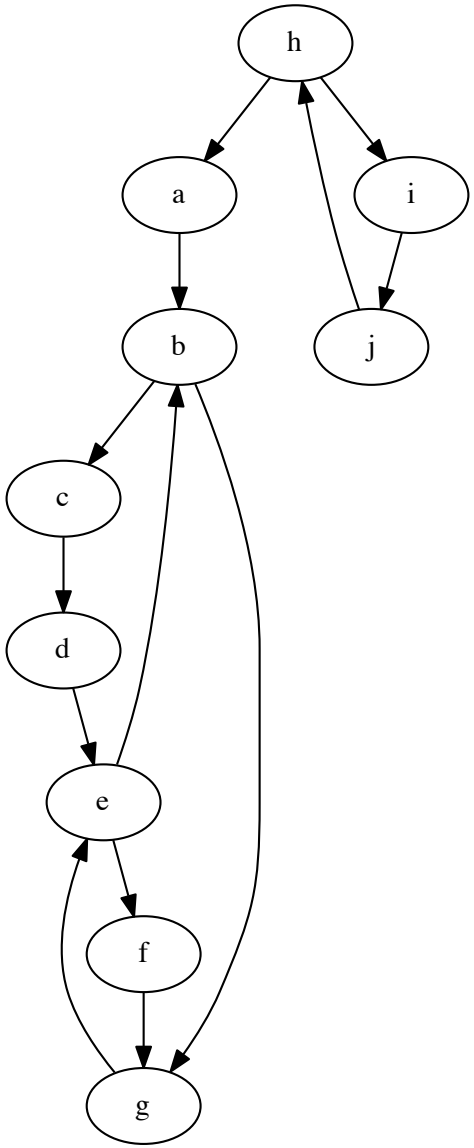
Calcul d'un chemin eulérien

Données : Graphe connexe $G = (S, A)$ vérifiant les conditions du théorème d'Euler
Résultat : C_e le chemin eulérien construit
si *tous les sommets sont de degré pair* **alors**
    On choisit un sommet quelconque s de S;
    On détermine un circuit simple C_1 de s à s;
sinon
    `/* le graphe admet 2 sommets` a `et` b `de degré impair */`
    On détermine un chemin simple C_1 de a à b;
fin

$A_1 = A \setminus E_1$; `/* ` E_1 `est l'ensemble des arcs utilisés dans` C_1 `*/`
$k = 1$;

tant que $A_k \neq \emptyset$ **faire**
    On choisit dans $G_k = (S, A_k)$ un sommet quelconque s_k de C_k;
    On détermine un circuit simple C_{k+1} de s_k à s_k;
    On insère les arcs du circuit C_{k+1} au chemin C_k, au niveau du sommet s_k;
    $A_{k+1} = A_k \setminus E_{k+1}$; `/* ` E_{k+1} `est l'ensemble des arcs utilisés dans` C_{k+1} `*/`
    $C_{k+1} = C_k$;
    $k = k + 1$;
fin
$C_e = C_k$;

Exemple de calcul de chemin eulérien



- Sommet(s) de degré impair : g et h

- $s_1 = h$, $C_1 = [h, a, b, g]$
- $s_2 = h$, $C_2 = [h, i, j, h, a, b, g]$
- $s_3 = b$, $C_3 = [h, i, j, h, a, b, c, d, e, b, g]$
- $s_4 = e$, $C_4 = [h, i, j, h, a, b, c, d, e, f, g, e, b, g]$

- Chemin eulérien :
  $[h, i, j, h, a, b, c, d, e, f, g, e, b, g]$

Graphes hamiltoniens

Définitions

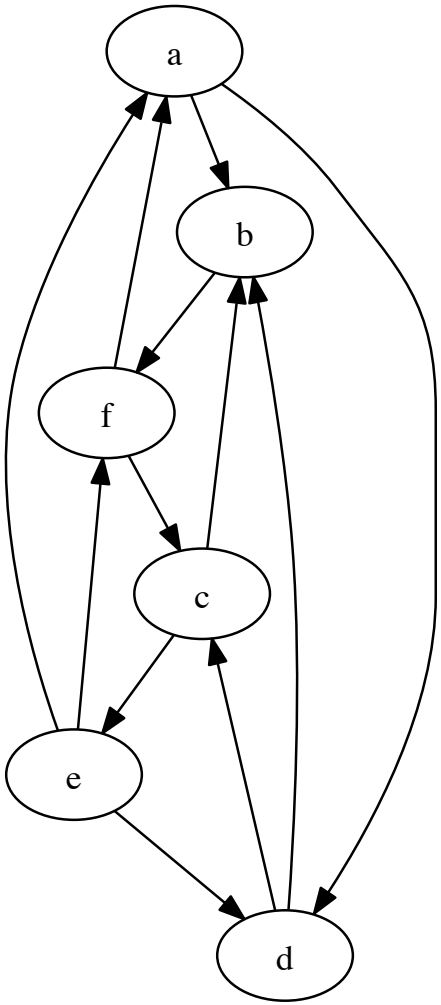
Soit G un graphe non orienté.

- Un **cycle hamiltonien** de G est un cycle qui passe une et une seule fois par chaque sommet de G. Un graphe est alors dit **hamiltonien** s'il possède un cycle hamiltonien.

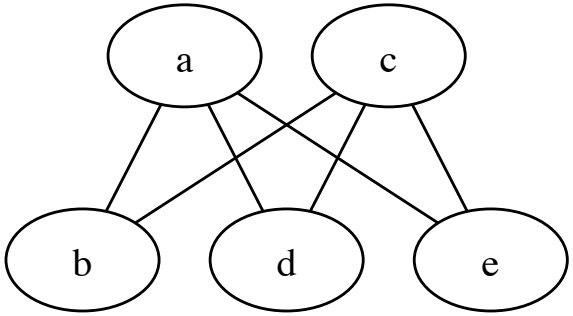
- Une **chaîne hamiltonienne** de G est une chaîne qui passe une et une seule fois par chaque sommet de G. Un graphe est alors dit **semi-hamiltonien** s'il ne possède que des chaînes hamiltoniennes.

Les mêmes notions peuvent être définies pour un graphe orienté G : **circuit hamiltonien** et **chemin hamiltonien**.

Exemples de graphes hamiltoniens



$[a, d, b, f, c, e, a]$ circuit hamiltonien donc graphe hamiltonien.



$[e, c, d, a, b]$ chaîne hamiltonienne mais pas de cycle hamiltonien donc graphe semi-hamiltonien.

Conditions nécessaires et suffisantes ?

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

Contrairement au cas des cycles eulériens, il n'existe aucune propriété générale (c'est-à-dire des conditions nécessaires et suffisantes) permettant de conclure si un graphe est ou non hamiltonien : ce problème est **algorithmiquement difficile**.

Conditions suffisantes sur les graphes hamiltoniens

Théorèmes

- Un graphe possédant un sommet de degré 1 ne peut pas être hamiltonien.
- Si un sommet dans un graphe est de degré 2, alors les deux arêtes incidentes à ce sommet doivent faire partie du cycle hamiltonien.

Théorème (Ore)

Soit G un graphe simple d'ordre $n \geq 3$. Si, pour toute paire $\{x, y\}$ de sommets non adjacents, on a $d(x) + d(y) \geq n$ alors G est hamiltonien.

Corollaire (Dirac)

Soit G un graphe simple d'ordre $n \geq 3$. Si, pour toute sommet x de G, on a $d(x) \geq \frac{n}{2}$ alors G est hamiltonien.

Plan du cours

Recherche de plus courts chemins

Exemples d'applications

- Recherche de l'itinéraire le plus rapide en voiture, entre deux villes
- Routage dans des réseaux de communication
- Problèmes d'ordonnancement...

Algorithmes et cas considérés

- Algorithmes étudiés pour résoudre le problème suivant :

  Étant donné un sommet x, on veut déterminer, pour chaque sommet y, la distance et un plus court chemin de x à y

- Étant donné deux sommets x et y, il existe plusieurs cas :
  1. Il n'existe pas de chemins de x à y
  2. Il existe un ou plusieurs plus courts chemins de x à y
  3. Il existe des chemins de x à y mais pas de plus courts

$\rightarrow$ Problématiques similaires pour les graphes non orientés

Distance et plus court chemin

Définitions : distance et plus court chemin

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

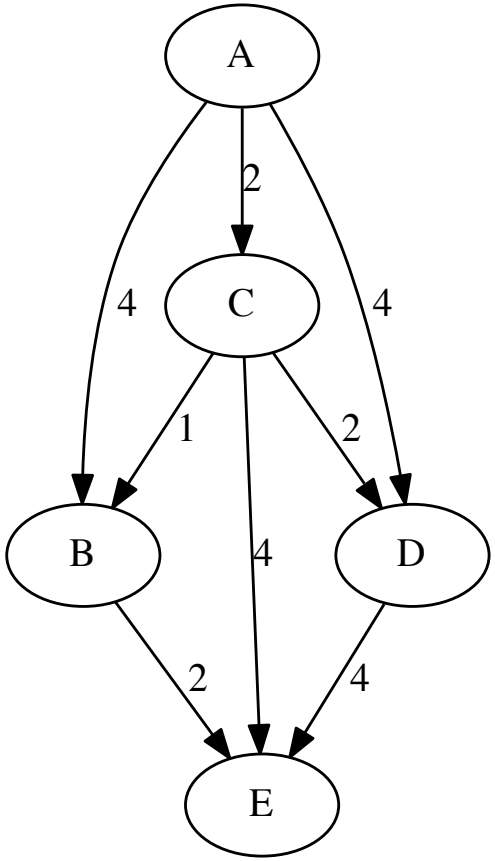
- **Distance de x à y**, notée $d(x, y)$: minimum des valuations des chemins allant de x à y
- **Plus court chemin de x à y** : tout chemin dont la valuation est égale à $d(x, y)$

$\rightarrow$ Définitions similaires pour les graphes non orientés

Remarque

Distance en nombre d'arcs : cas particulier du calcul de distance, qui correspond au cas où tous les arcs sont de valuation 1

Exemples de plus courts chemins



- Distance de A à E : $d(A, E) = 5$

$\longrightarrow$ Plus court(s) chemin(s) : $[A, C, B, E]$

- Distance de A à D : $d(A, D) = 4$

$\longrightarrow$ Plus court(s) chemin(s) : $[A, D]$ et $[A, C, D]$

- Distance de E à A : non définie

$\longrightarrow$ Plus court(s) chemin(s) : aucun

Circuit absorbant

Définition : circuit absorbant

Circuit absorbant : circuit de valuation négative

$\rightarrow$ Si un graphe possède un circuit absorbant alors il n'existe pas de plus courts chemins entre certains de ses sommets

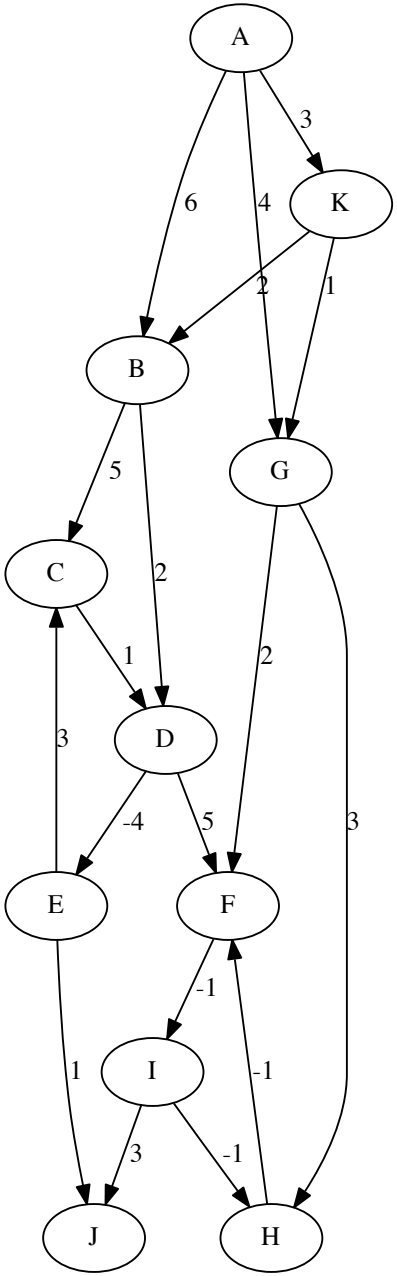
Théorème

Soit G un graphe orienté valué sans circuit absorbant et x et y deux sommets de G
S'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

$\rightarrow$ Définition et théorème similaires pour les graphes non orientés

Graphes considérés considérés dans la suite : sans circuits (ou cycles) absorbants

Exemple



- De A à B : il existe **un unique plus court chemin**, $[A, K, B]$
- De A à G : il existe **2 plus courts chemins**, $[A, K, G]$ et $[A, G]$
- De E à A : il n'existe pas de chemin donc **aucun plus court chemin**
- De A à E : il existe une **infinité de plus courts chemins**, tels que $[A, K, B, D, E]$, $[A, K, B, D, E, C, D, E]$, à-cause du circuit $[C, D, E, C]$ de valuation nulle
- De A à J : il existe des chemins mais **aucun plus court chemin**, à-cause du circuit absorbant $[F, I, H, F]$ qui réduit la valuation du chemin de A à J à chaque passage dans le circuit

Principaux algorithmes de recherche de plus courts chemins

- Algorithme de Floyd-Warshall-Roy
  - L'algorithme peut être utilisé sur tout type de graphe ;
  - Les distances sont calculées entre tout couple de sommets ;
  - L'algorithme consomme beaucoup de mémoire (2 matrices de taille $n \times n$) ;
  - Le temps de calcul est relativement long ($O(n^3)$).

- Algorithme de Bellman-Ford-Kalaba
  - L'algorithme peut être utilisé sur tout type de graphe ;
  - Les distances sont calculées entre un sommet et tous les autres sommets ;
  - L'algorithme consomme moins de mémoire (2 tableaux de taille n) ;
  - Le temps de calcul est relativement long ($O(n^3)$) mais peut être réduit pour une version simplifiée de l'algorithme ($O(n^2)$).

- **Algorithme de Dijkstra**
  - L'algorithme ne peut être utilisé que sur les graphes à valuations positives ;
  - Les distances sont calculées entre un sommet et tous les autres sommets ;
  - L'algorithme consomme moins de mémoire (2 tableaux de taille n) ;
  - Le temps de calcul est plus court ($O(n^2)$).

Algorithme de Dijkstra (1)

Présentation de l'algorithme de Dijkstra

- Un des algorithmes les plus connus et efficaces pour la recherche de distance et de plus courts chemins
- Calcul des distances et d'un plus court chemin entre un sommet x_0 et tous les autres sommets du graphe
- Limitations : ne peut être utilisé que dans le cas où **toutes les valuations des arcs (ou arêtes) sont positives** (donc pas de circuits absorbants)

Principe de l'algorithme de Dijkstra

- Construction itérative d'un **ensemble de sommets marqués**, noté M
  - $\rightarrow$ Pour tout sommet marqué s : estimation $d(s)$ = distance $d(x_0, s)$
- À chaque étape
  1. Sélection d'un sommet non marqué x dont la distance estimée $d(x)$ est la plus petite parmi les distances estimées des sommets non marqués
  2. Ajout de x à l'ensemble M
  3. Mise à jour des distances estimées des successeurs non marqués de x

  $\rightarrow$ À recommencer jusqu'à ce que tous les sommets soient marqués

Algorithme de Dijkstra (2)

Données : Graphe valué $G = (S, A, v)$ sans valuations négatives ; sommet x_0

```
// Initialisation des tableaux d et P et de l'ensemble M
M = ∅ ;                                          // Aucun sommet marqué
pour chaque s ∈ S faire
    P[s] = nul;
    si s = x_0 alors
        d[s] = 0;
    fin
    sinon
        d[s] = +∞;
    fin
fin

// Boucle principale
tant que M ≠ S faire
    x = arg min_{s∈S, s∉M} d[s] ;        // Sommet non marqué de distance min.
    M = M ∪ {x} ;                          // Ajout de x aux sommets marqués
    pour chaque y ∈ Γ+(x), y ∉ M faire
        si d[x] + v(x, y) < d[y] alors
            d[y] = d[x] + v(x, y) ;              // Mise à jour de la distance
            P[y] = x ;                                  // Mise à jour du père
        fin
    fin
fin
```

Exemple (1)

Soit le graphe G dont les sommets sont nommés de A à D et défini par la matrice de valuation suivante :

$$G = \begin{pmatrix} +\infty & 8 & 6 & 2 \\ +\infty & +\infty & +\infty & +\infty \\ +\infty & 3 & +\infty & +\infty \\ +\infty & 5 & 1 & +\infty \end{pmatrix}$$

1 Initialisation de l'algorithme

x	M	$d[A]$ $_{P[A]}$	$d[B]$ $_{P[B]}$	$d[C]$ $_{P[C]}$	$d[D]$ $_{P[D]}$
$x_0 = A$	$\emptyset$	0 $_{nul}$	$+\infty$ $_{nul}$	$+\infty$ $_{nul}$	$+\infty$ $_{nul}$

2 Itération 1 : choix du sommet $x = A$, de distance 0

x	M	$d[A]$ $_{P[A]}$	$d[B]$ $_{P[B]}$	$d[C]$ $_{P[C]}$	$d[D]$ $_{P[D]}$
$x_0 = A$	$\emptyset$	0 $_{nul}$	$+\infty$ $_{nul}$	$+\infty$ $_{nul}$	$+\infty$ $_{nul}$
A	$\{A\}$	–	$0+8 < +\infty$? oui donc : 8 $_{A}$	$0+6 < +\infty$? oui donc : 6 $_{A}$	$0+2 < +\infty$? oui donc : 2 $_{A}$

Exemple (2)

3. Itération 2 : choix du sommet $x = D$, de distance 2

x	M	$d[A]$ $P[A]$	$d[B]$ $P[B]$	$d[C]$ $P[C]$	$d[D]$ $P[D]$
$x_0 = A$	$\emptyset$	0 *nul*	$+\infty$ *nul*	$+\infty$ *nul*	$+\infty$ *nul*
A	$\{A\}$	–	8 *A*	6 *A*	2 *A*
D	$\{A, D\}$	–	$2+5<8$? oui donc : 7 *D*	$2+1<6$? oui donc : 3 *D*	–

4. Itération 3 : choix du sommet $x = C$, de distance 3

x	M	$d[A]$ $P[A]$	$d[B]$ $P[B]$	$d[C]$ $P[C]$	$d[D]$ $P[D]$
$x_0 = A$	$\emptyset$	0 *nul*	$+\infty$ *nul*	$+\infty$ *nul*	$+\infty$ *nul*
A	$\{A\}$	–	8 *A*	6 *A*	2 *A*
D	$\{A, D\}$	–	7 *D*	3 *D*	–
C	$\{A, D, C\}$	–	$3+3<7$? oui donc : 6 *C*	–	–

5. Itération 4 : choix du sommet $x = B$, de distance 6

x	M	$d[A]$ $P[A]$	$d[B]$ $P[B]$	$d[C]$ $P[C]$	$d[D]$ $P[D]$
$x_0 = A$	$\emptyset$	0 *nul*	$+\infty$ *nul*	$+\infty$ *nul*	$+\infty$ *nul*
A	$\{A\}$	–	8 *A*	6 *A*	2 *A*
D	$\{A, D\}$	–	7 *D*	3 *D*	–
C	$\{A, D, C\}$	–	6 *C*	–	–
B	$\{A, D, C, B\}$	0 *nul*	6 *C*	3 *D*	2 *A*

Construction des plus courts chemins

Présentation de la construction des plus courts chemins

- À la fin de l'algo. de Dijkstra : distances calculées, de x_0 à tout sommet s
- On peut déterminer un plus court chemin de x_0 à s, pour tout sommet s

$\rightarrow$ Utilisation du tableau des pères pour construire les plus courts chemins

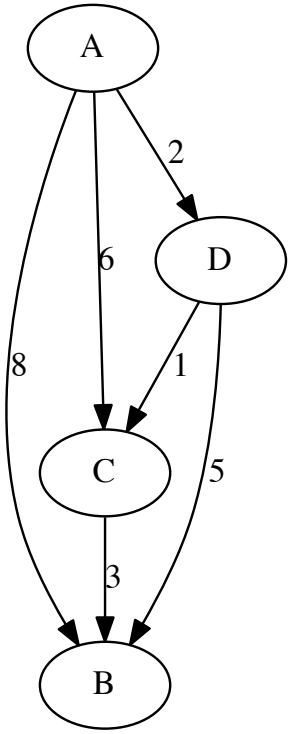
Principe de la construction du plus court chemin du sommet s

- Construction itérative du plus court chemin de x_0 à s, à partir de la fin

- Initialisation
  - Initialisation du sommet courant x avec le sommet s : $x \leftarrow s$
  - Initialiation du plus court chemin de x_0 à s avec le sommet x : $c \leftarrow [x]$
- À chaque étape
  1. Sélection du nouveau sommet courant x en choisissant son père : $x \leftarrow P[x]$
  2. Ajout de x au début du plus court chemin : $c \leftarrow x + c$

  $\rightarrow$ À recommencer jusqu'à ce que le sommet courant soit le sommet x_0

Exemple



$P[A]$	$P[B]$	$P[C]$	$P[D]$
nul	C	D	A

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

Propriétés

Propriété

A toutes les étapes de l'algorithme, chaque sommet s marqué vérifie $d(s) = d(x_0, s)$

Propriété

Après i itérations de la boucle principale de l'algorithme, ensemble des sommets marqués : $M = \{x_0, x_1, \ldots, x_{i-1}\}$

$\rightarrow$ Cela implique : à la fin de l'algorithme, on a bien calculé les distances de x_0 à s

Plan du cours

Conclusion

La **théorie des graphes** permet de représenter de nombreux problèmes courants en informatique mais également dans d'autres domaines.

Pour pouvoir utiliser ce formalisme, il est tout d'abord nécessaire de **modéliser** le problème à résoudre sous la forme d'un graphe. Il faut alors préciser si le graphe est **orienté ou non** et identifier les éléments à représenter par les **sommets** ainsi que ceux à représenter par les **arcs** (ou par les arêtes).

Une fois le graphe construit, il reste à résoudre le problème en identifiant, par exemple, les **propriétés** du chemin recherché ou l'**algorithme** à utiliser pour extraire une sous-partie du graphe.
Dans ce cours, nous avons présenté des **algorithmes** permettant de calculer la fermeture transitive d'un graphe, de calculer ses composantes connexes, de calculer des chemins eulériens, de calculer des plus courts chemins entre un sommet donné et les autres sommets du graphe, de calculer un arbre couvrant de poids minimum et d'effectuer la coloration des sommets d'un graphe ou de ses arêtes. Et il en existe encore d'autres. . .