Graphes et algorithmes

José Neto
Institut Mines-Télécom - Télécom SudParis
Master Parisien de Recherche Opérationnelle
Jose.Neto@telecom-sudparis.eu

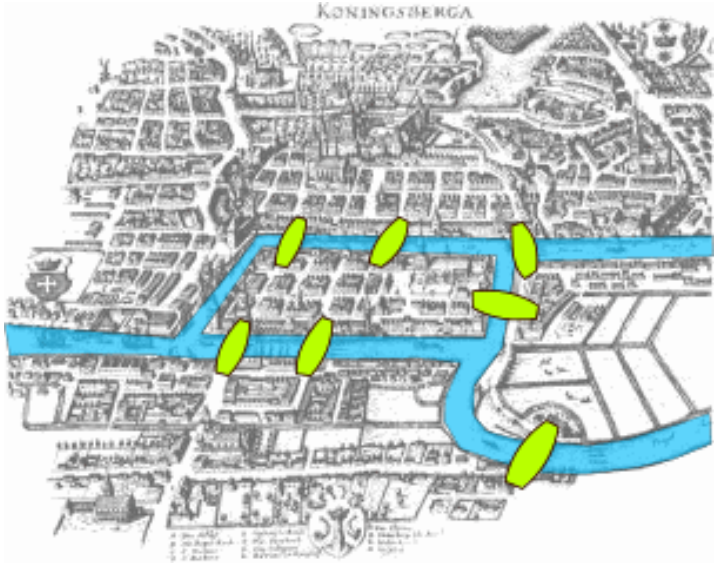
Septembre 2020
CNAM, Paris, France

Un peu d'histoire…

- Le problème des ponts de Königsberg.

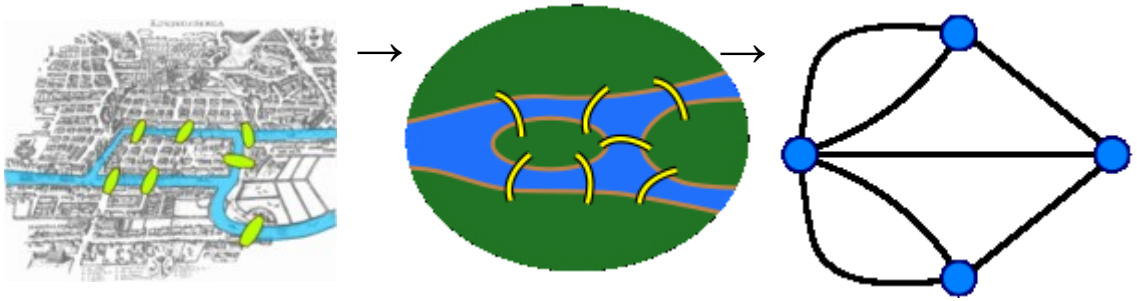
18ième siècle. La ville de Königsberg en Prusse (aujourd'hui Kaliningrad en Russie) est construite autour de deux îles situées sur le fleuve Pregolia et qui sont reliées entre elles et au reste de la ville par 7 ponts.

Le problème consiste à déterminer s'il existe une promenade dans la ville permettant de traverser chaque pont une fois exactement et de revenir à son point de départ.



Un peu d'histoire…

- Formuler le problème … avec un graphe



Solution: théorème d'Euler

Un graphe est eulérien (i.e. il existe un parcours du graphe empruntant chaque arête une fois exactement et se terminant à son point de départ), ssi tous ses sommets sont incidents à un nombre pair d' arêtes.

Leonhard Euler (1707-1783)

Portrait par Johann Georg Brucker

Motivations

- Aujourd'hui des graphes … Pour quoi faire ?
  - Modéliser des réseaux (de communications, sociaux…),
  - Traiter des problèmes de routage, d'ordonnancements, de localisations d'infrastructures…
  - Dimensionner des réseaux,
  - Représenter des automates,
  - …

Contenu du cours

1. Notions de base
2. Arbres
3. Problèmes de plus courts chemins
4. Flots dans les réseaux

PARTIE 1

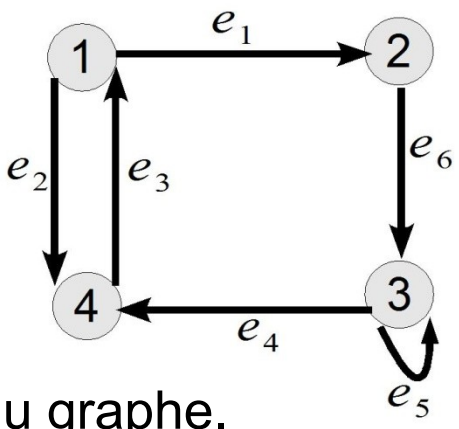
NOTIONS DE BASE

Graphe orienté

- Un graphe orienté *G=(X,E)* est défini par :
  - *X*: un ensemble de sommets (ou noeuds)
  - E: un ensemble d'arcs, i.e. de paires ordonnées de sommets

- Pour un arc *e=(i,j)* on appelle :
  - *i*: l'extrémité initiale de *e*
  - *j*: l'extrémité terminale de *e*

- Un exemple

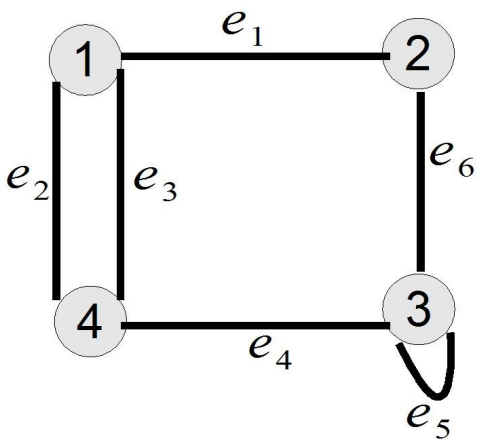


- |*X*| est appelé ordre du graphe.

Graphe non orienté

- Un graphe non orienté *G=(X,E)* est défini par :
  - *X*: un ensemble de sommets (ou noeuds)
  - E: un ensemble d'arêtes, i.e. de paires non ordonnées de sommets

- Un exemple



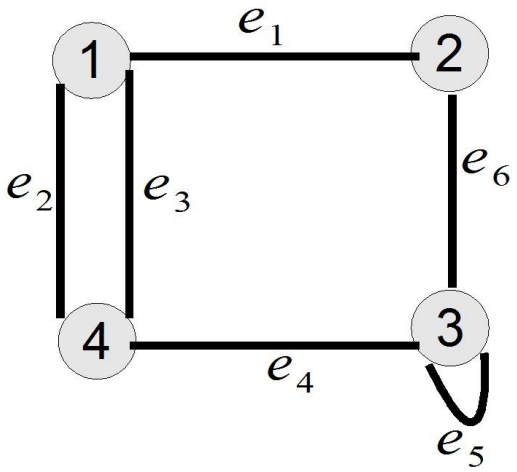
- A tout graphe orienté on peut associer un graphe non orienté en considérant son ensemble d'arcs comme un ensemble d'arêtes

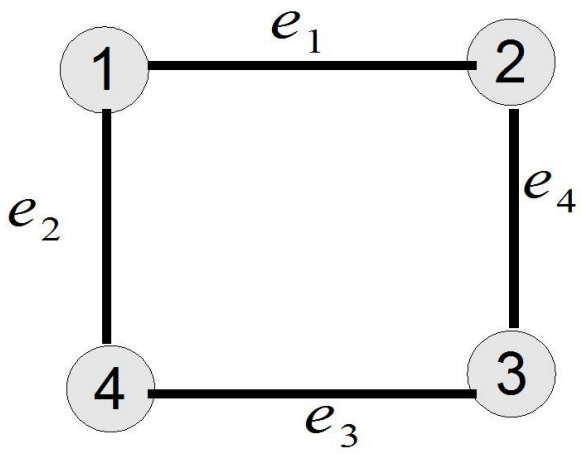
Boucle - graphe simple

- Une arête dont les deux extrémités coincident est appelée boucle.

- Un graphe (non orienté) est dit simple s'il est sans boucle et toute paire de sommets est reliée par au plus une arête. Un multigraphe est un graphe pour lequel il peut exister plusieurs arêtes reliant deux sommets donnés.

- Illustration

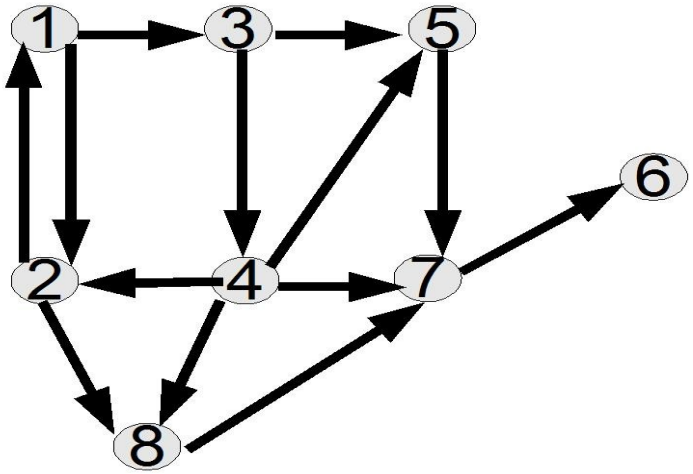




Adjacence

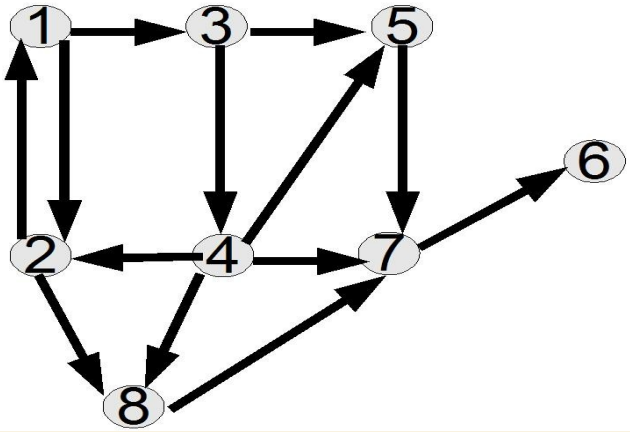
- Deux nœuds i et j sont dits adjacents ou voisins si l'arc (arête) (i,j) existe. Dans le cas d'un graphe orienté, on dit qu'un noeud *i* est un prédécesseur (resp. *successeur*) du noeud *j* s'il existe un arc de la forme *(i,j)* (resp. *(j,i)*).

- Deux arcs (arêtes) e1 and e2 sont dit(e)s adjacent(e)s s'ils (elles) ont au moins une extrémité en commun.



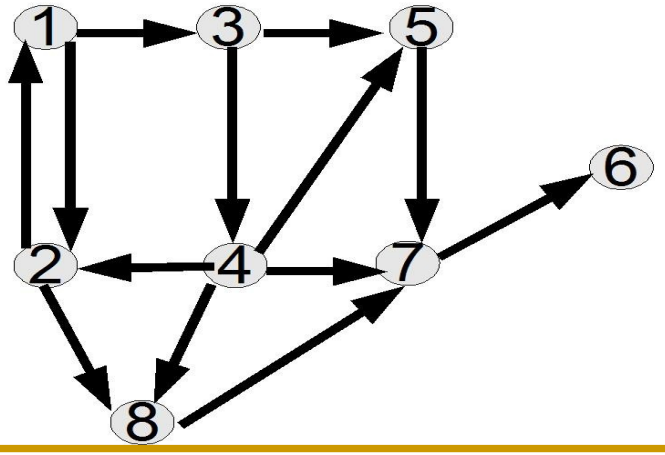
Degré – Demi-degré

- Dans un graphe orienté G,
  - le demi-degré extérieur $\mathbf{d_G^+(i)}$ du sommet *i* dans le graphe *G*, est le nombre d'arcs ayant *i* pour extrémité initiale.
  - le demi-degré intérieur $\mathbf{d_G^-(i)}$ du sommet *i* dans *G* est le nombre d'arcs ayant *i* pour extrémité terminale.
  - le degré $\mathbf{d_G(i)}$ du sommet *i* dans *G* est le nombre d'arcs ayant *i* pour extrémité: $\mathbf{d_G(i) = d_G^+(i) + d_G^-(i)}$.
- Dans un graphe non orienté G, le degré $\mathbf{d_G(i)}$ du sommet *i* est le nombre d'arêtes ayant *i* pour extrémité. (Attention : une boucle est comptabilisée deux fois.)
  - Illustration



Cocyle

- Dans un graphe orienté, étant donné un sous ensemble de sommets $\mathbf{A \subset X}$ on définit :
  - $\boldsymbol{\omega^{+}(A)}$: l'ensemble des arcs ayant leur extrémité initiale dans *A,* et leur extrémité terminale dans $\overline{A} = X \setminus A$.
  - $\boldsymbol{\omega^{-}(A)}$: l'ensemble des arcs ayant leur extrémité terminale dans *A,* et leur extrémité initiale dans $\overline{A}$.
  - $\boldsymbol{\omega(A) = \omega^{+}(A) \cup \omega^{-}(A)}$: le cocyle relatif à l'ensemble *A*.
- Dans un graphe non orienté $\boldsymbol{\omega(A)}$: le cocyle relatif à *A*, est défini comme l'ensemble des arêtes ayant exactement une extrémité dans *A*.

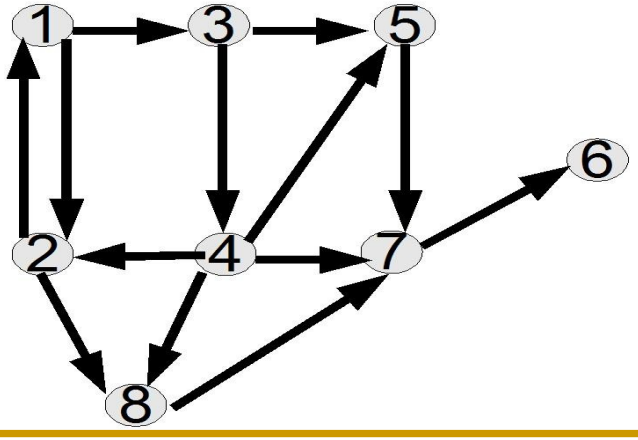


Chaîne (élémentaire)

- Une chaîne extraite d'un graphe $G = (X,E)$ est une séquence d'arêtes $L=\{u_1, u_2, \ldots, u_k\}$ telle que chaque arête u_i de la séquence (avec $2 \leq i \leq k\text{-}1$) a une extrémité commune avec l'arête précédente et l'autre extrémité commune avec l'arête suivante. On exige également que

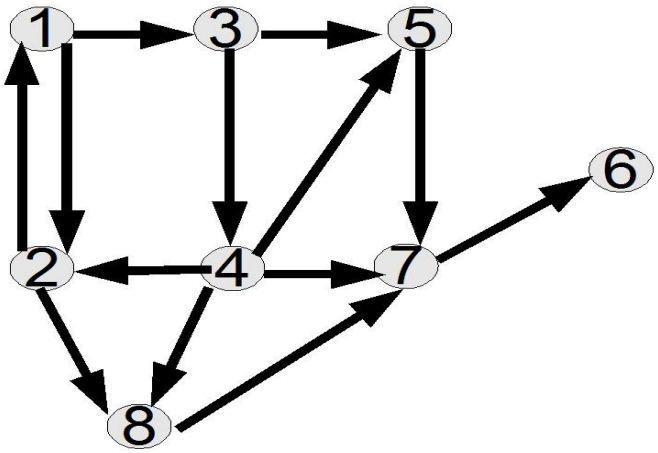
$$u_i \neq u_{i-1} \qquad (2 \leq i \leq k)$$

  Il s'agit d'une notion non orientée qu'on applique sur un graphe orienté en ne tenant pas compte de la direction des arcs (considérés alors comme des arêtes).
- Une chaîne est dite élémentaire si, en la parcourant, aucun sommet n'est rencontré plus d'une fois.



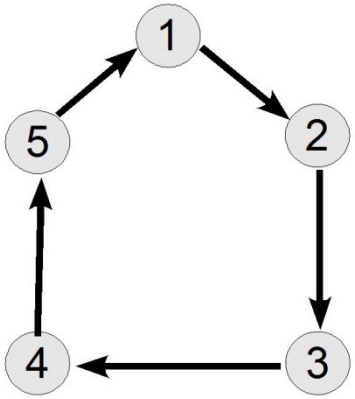
Chemin (simple/élémentaire)

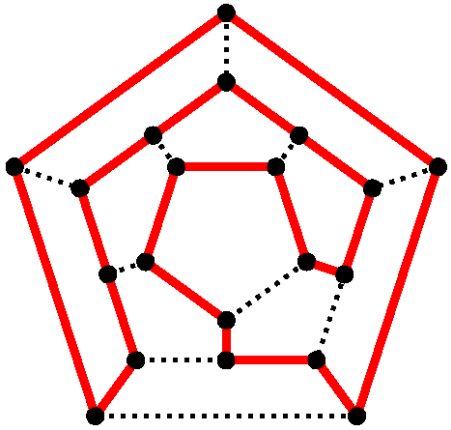
- Dans un graphe orienté, un chemin du sommet *s* au sommet *t* de longueur *q* est une séquence de *q* arcs : $P=\{e_1,e_2,\ldots,e_q\}$ telle que, pour chaque arc e_i de la séquence ($2 \leq i \leq q$-1) son extrémité initiale coïncide avec l'extrémité terminale de e_{i-1} et son extrémité terminale avec l'extrémité initiale de e_{i+1} ; aussi *s* correspond à l'extrémité initiale de e_1 et *t* à l'extrémité terminale de e_q.
- Un chemin est dit simple si, en le parcourant, aucun arc n'est parcouru plus d'une fois.
- Un chemin est dit élémentaire si, en le parcourant, aucun sommet n'est rencontré plus d'une fois.

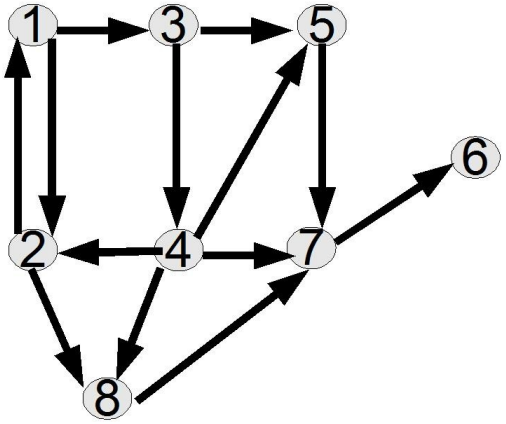


Circuit-cycle (élémentaire)

- Un circuit (resp. cycle) est un chemin (resp. chaîne) pour lequel origine et destination coincident, i.e. *s=t*.
- Un cycle hamiltonien est un cycle passant une et une seule fois par tous les sommets.
- Un circuit (ou cycle) est dit élémentaire si, en le parcourant, aucun sommet n'est rencontré plus d'une fois à l'exception du sommet de départ (=sommet d'arrivée) rencontré deux fois exactement.
- Illustrations





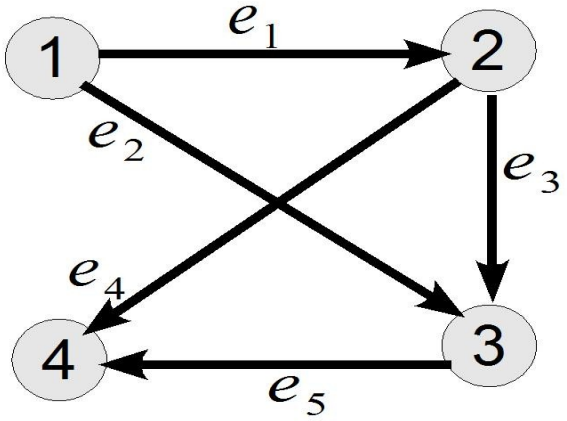


Représentations matricielles de graphes

- Matrice d'incidence « Sommets-arcs »
- Matrice d'incidence « Sommets-arêtes »
- Matrice d'adjacence

Matrice d'incidence « Sommets-arcs »

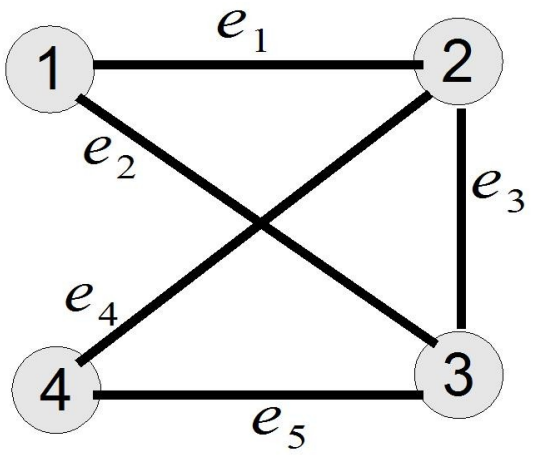
- Matrice d'incidence « Sommets-arcs » *A* du graphe *G=(X,E):*
  - chaque colonne correspond à un arc de *G*,
  - chaque ligne correspond à sommet de *G*.
  - pour un arc donné *e=(i,j),* la colonne correspondant à *e* a toutes ses entrées nulles sauf : $\mathbf{A}_{ie} = +1$ et $\mathbf{A}_{je} = -1$
- Illustration



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

Matrice d'incidence « Sommets-arêtes »

- Matrice d'incidence « Sommets-arêtes » *A* du graphe *G=(X,E):*
  - chaque colonne correspond à une arête de *G*,
  - chaque ligne correspond à un sommet de *G*.
  - Pour une arête donnée *e=(i,j),* la colonne correspondant à *e* a toutes ses entrées nulles sauf : $\mathbf{A}_{ie} = +1$ et $\mathbf{A}_{je} = +1$
- Illustration



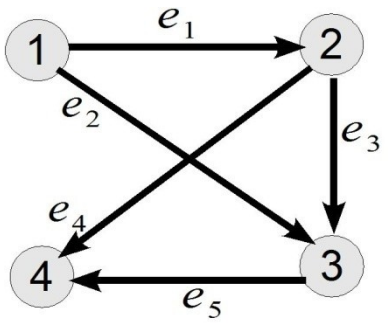
$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Matrice d'adjacence

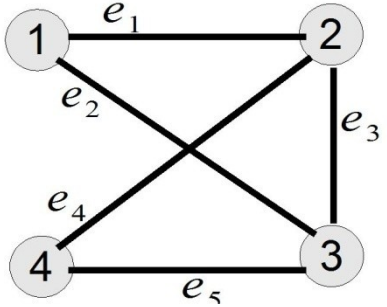
- Pour un graphe G=(X,E) avec |*X*|=*N*, la matrice d'adjacence *A* est une matrice carrée d'ordre *N* avec :

  $\mathbf{A_{ij}}$ = nombre d'arcs de type (i,j) dans E.

- Remarque : pour un graphe non orienté la matrice d'adjacence est symétrique
- Illustration



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

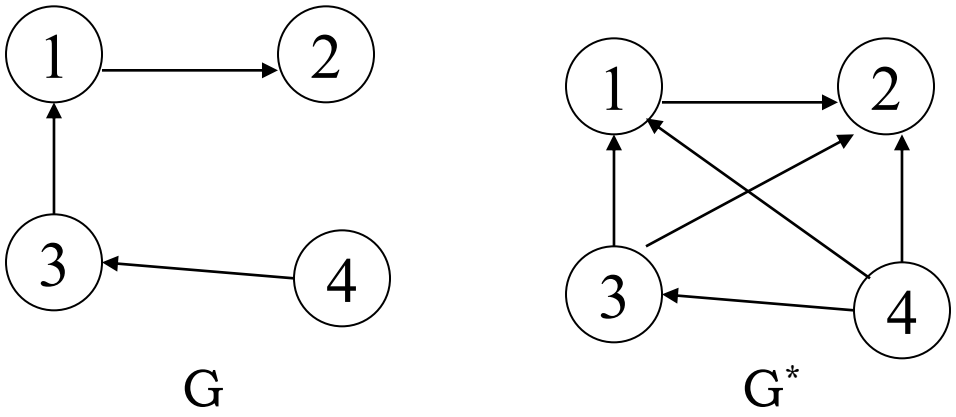


$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Fermeture transitive

- On appelle fermeture transitive du graphe G=(V,E) le graphe G*=(V,U*) tel que $(i,j) \in U^* \Leftrightarrow \exists \text{ chemin de i vers j dans G}$

- Exemple



Fermeture transitive

- Algorithme matriciel

Somme booléenne de matrices ⟹ Somme algébrique de matrices avec + remplacé par ‘v’ (« ou » logique)

Produit booléen de matrices ⟹ Produit algébrique de matrices avec (+,x) remplacé par (‘V’,’Λ’) (avec ’Λ’ : « et » logique)

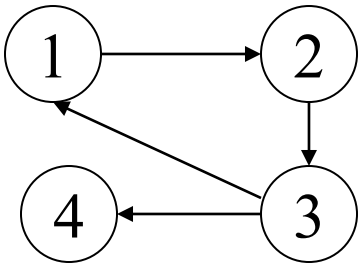
• Il existe un chemin de i vers j et de longueur p (en nombre d’arcs) $\Leftrightarrow (A^p)_{i,j} = 1$

• Un chemin élémentaire reliant deux sommets de G possède au plus N-1 arcs ; la fermeture transitive est obtenue par sommation (disjonction) des N-1 premières puissances de la matrice d’adjacence :

$$\exists \text{ chemin de i vers j} \Leftrightarrow \left(\sum_{p=1}^{N-1} A^p\right)_{i,j} = 1$$

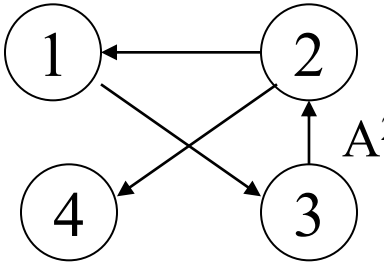
Fermeture transitive

- Exemple



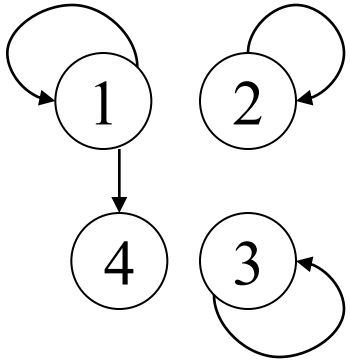
$A=$

0	1	0	0
0	0	1	0
1	0	0	1
0	0	0	0



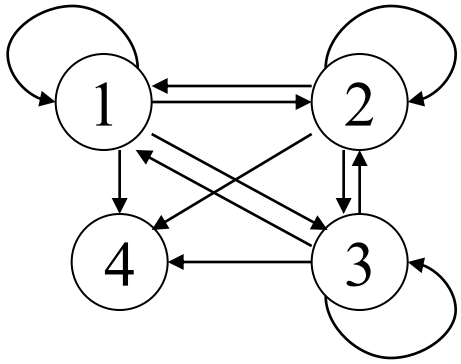
$A^2=$

0	0	1	0
1	0	0	1
0	1	0	0
0	0	0	0



$A^3=$

1	0	0	1
0	1	0	0
0	0	1	0
0	0	0	0

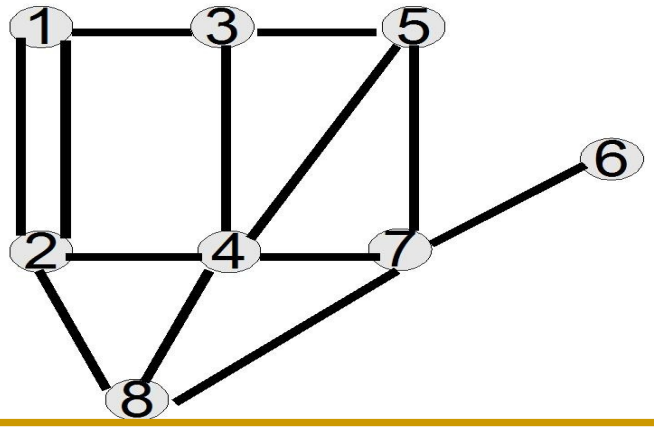


$\Sigma_{p=1..3}A^p=$

1	1	1	1
1	1	1	1
1	1	1	1
0	0	0	0

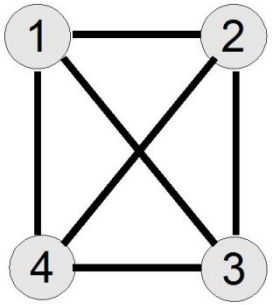
Sous-graphe, graphe partiel

- Etant donné un graphe *G=(X,E)*
  - le sous-graphe $\mathbf{G_A}$ engendré par $\mathbf{A \subset X}$ est le graphe avec pour ensemble de sommets *A* et pour ensemble d'arcs : ceux de G qui ont leurs deux extrémités dans *A*.
  - le graphe partiel engendré par $\mathbf{F \subset E}$ est le graphe avec pour ensemble de sommets *X*, et pour ensemble d'arcs *F* (i.e. les arcs de *E\F* sont retirés de G).
  - Le sous-graphe partiel engendré par *A* et *F* est le graphe partiel de $\mathbf{G_A}$ qui est engendré par *F*
- Illustration

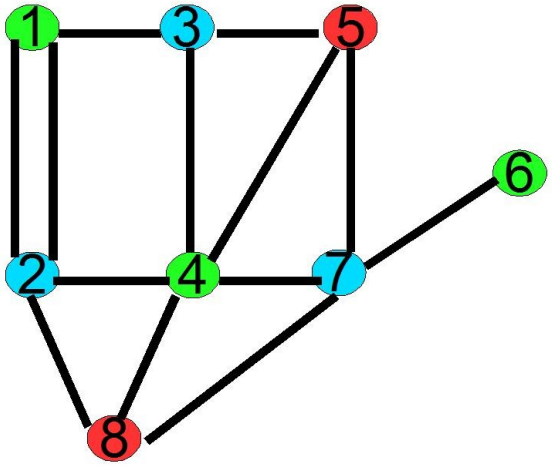


Graphe complet, clique, stable

- Un graphe *G=(X,E)* est dit complet si, pour chaque paire de sommets *{i,j},* il existe un arc de la forme *(i,j)* ou *(j,i)*. Un graphe simple et complet d'ordre *N* est noté $\mathbf{K_N}$.
- Une clique est un ensemble de sommets $\mathbf{C \subseteq X}$ tel que $\mathbf{G_C}$ (le sous-graphe engendré par C) est complet
- Un stable est un ensemble de sommets $\mathbf{S \subseteq X}$ tel que $\mathbf{G_S}$ n'a aucune arête. La cardinalité maximale d'un stable est appelée nombre de stabilité.
- Illustrations

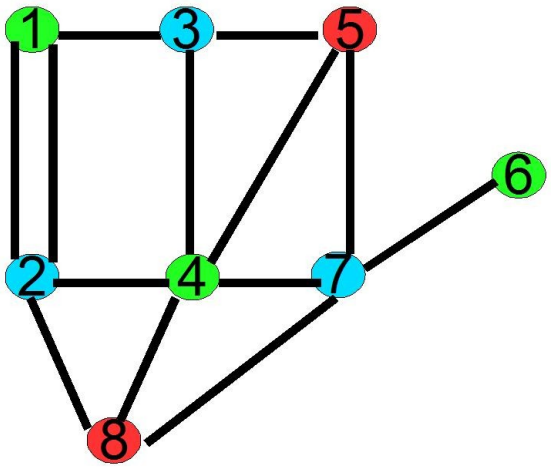


K_4



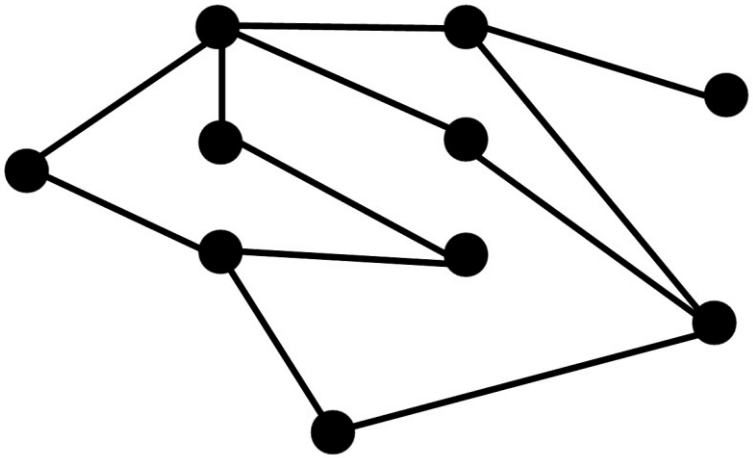
Coloration, nombre chromatique

- Une coloration de G est une partition de *X* en stables.
- La cardinalité minimale d'une coloration est appelée nombre chromatique.
- Illustration



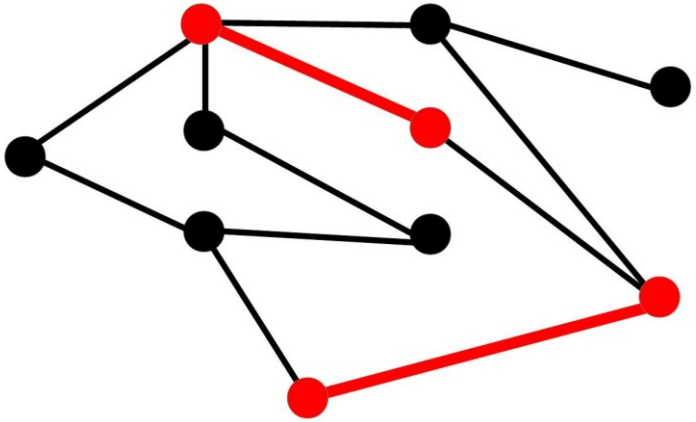
Couplage

- Définition. Dans un graphe *G=(V,E)* non orienté, on appelle couplage un ensemble d'arêtes qui n'ont pas d'extrémité en commun.



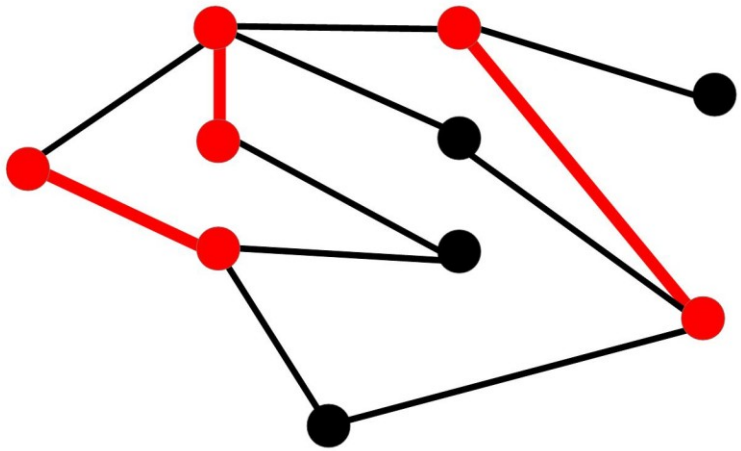
Couplage

- Définition. Dans un graphe *G=(V,E)* non orienté, on appelle couplage un ensemble d'arêtes qui n'ont pas d'extrémité en commun.



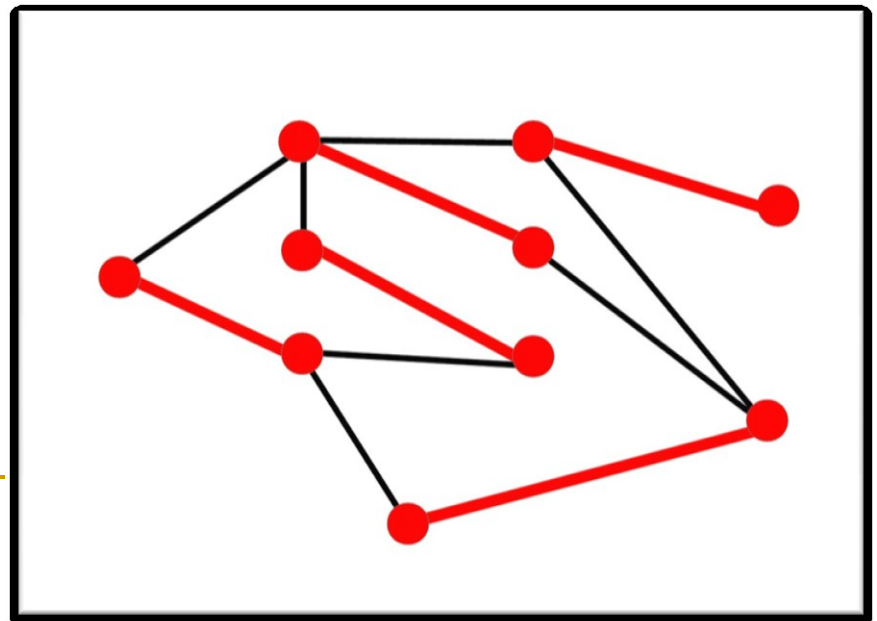
Couplage

- Définition. Dans un graphe *G=(V,E)* non orienté, on appelle couplage un ensemble d'arêtes qui n'ont pas d'extrémité en commun.
- Un couplage maximal est un couplage *M* du graphe tel que toute arête du graphe possède au moins une extrémité commune avec une arête de *M.*



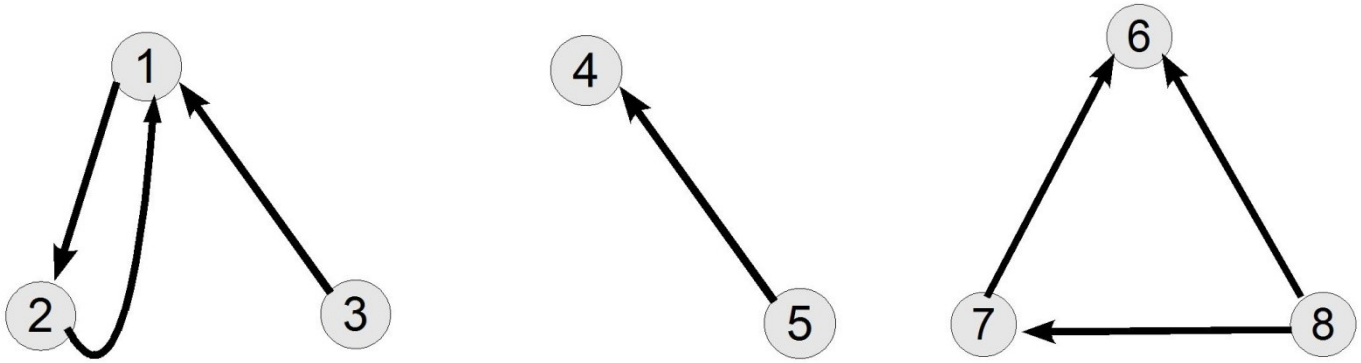
Couplage

- Définition. Dans un graphe *G=(V,E)* non orienté, on appelle couplage un ensemble d'arêtes qui n'ont pas d'extrémité en commun.
- Un couplage maximal est un couplage *M* du graphe tel que toute arête du graphe possède au moins une extrémité commune avec une arête de *M.*
- Un couplage maximum est un couplage contenant le plus grand nombre possible d'arêtes.
- Un couplage parfait ou couplage complet est un couplage *M* du graphe tel que tout sommet du graphe est incident à exactement une arête de *M*.



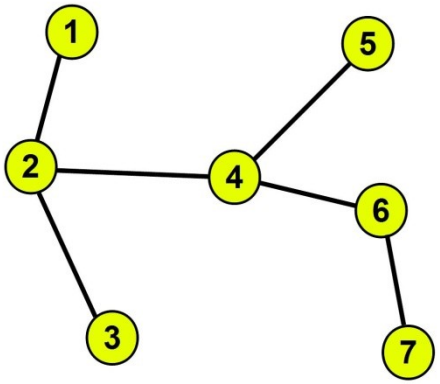
Connexité

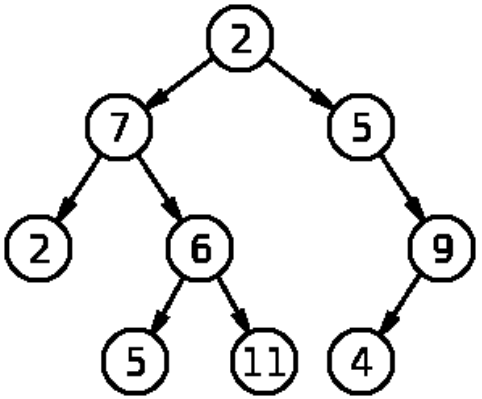
- Un graphe est dit connexe, si pour toute paire de sommets *(i,j),* il existe une chaîne reliant *i* et *j*.
- Une composante connexe est un sous-graphe connexe maximal (pour l'inclusion).
- Un graphe est dit fortement connexe si, pour toute paire ordonnée de sommets *(i,j),* il existe un chemin de *i* vers *j*.
- On appelle composante fortement connexe un sous-graphe fortement connexe maximal (pour l'inclusion).
- Illustrations



Arbre, arborescence

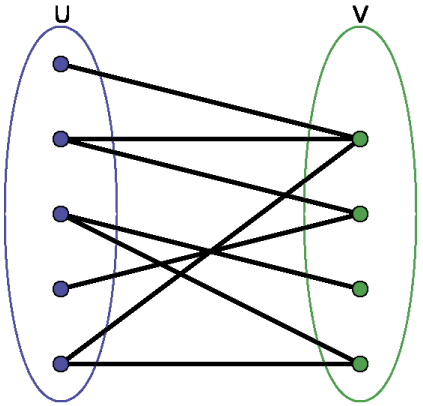
- Un arbre est un graphe non orienté, connexe et sans cycle.
- Une forêt est un graphe sans cycle
- Une arborescence (ou arbre enraciné) est un graphe orienté *G* sans cycle avec un sommet particulier appelé racine et tel qu'il existe un chemin depuis la racine vers tout autre sommet de *G*.
- Illustrations

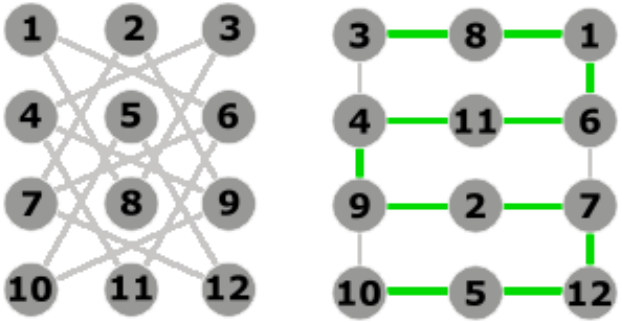




Graphe biparti, graphe planaire

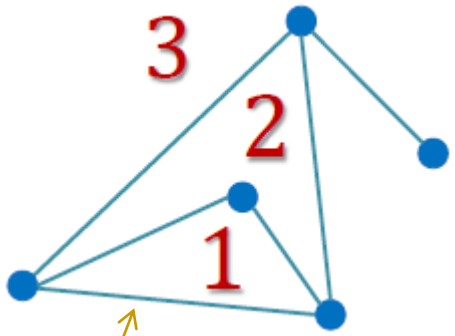
- Un graphe est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles U et V telle que chaque arête ait une extrémité dans U et l'autre dans V.
- Un graphe *G* est dit planaire s'il admet une représentation sur un plan *P* par des points distincts figurant les sommets et des courbes simples figurant les arêtes, deux telles courbes ne se rencontrant pas en dehors de leurs extrémités. Une telle représentation, notée R(G) dans la suite, est appelée graphe planaire topologique.
- Illustrations



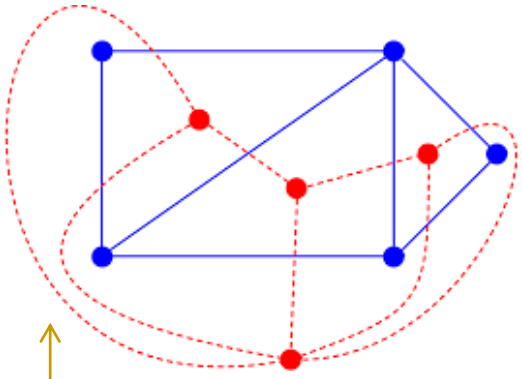


Graphes planaires : face, dual

- Les parties connexes (pour la topologie du plan) de *P-R(G)* sont appelées les *faces*. Une face est une région du plan délimitée par des arêtes et dont l'ensemble constitue la *frontière*. Deux faces sont dites *adjacentes* lorsque leurs frontières ont au moins une arête commune.
- Soit *G* un graphe planaire, connexe et sans sommet isolé. On lui fait correspondre un graphe planaire *G** comme suit. A l'intérieur de chaque face *s* de *R(G)*, on place un sommet *s** de *G** ; à toute arête *e* de *G*, on fait correspondre une arête *e** de *G** reliant les sommets *s** et *t** correspondant aux faces *s* et *t* qui se trouvent de part et d'autre de l'arête *e* dans *R(G)*. Le graphe *G** ainsi défini est planaire, connexe, sans sommet isolé. On l'appelle le *graphe dual* (topologique) de *G*.
- Illustration



Faces d'un graphe planaire



Un graphe planaire (traits continus) et son dual (en pointillés)

Graphes planaires : Euler, Kuratowski

- Théorème [Formule d'Euler]

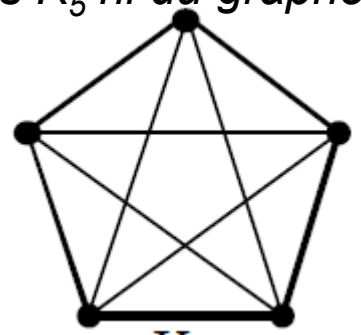
  Si G est un graphe planaire topologique connexe avec N sommets, M arêtes et F faces, alors on a : N-M+F=2.

  (Preuve par récurrence sur le nombre de faces)

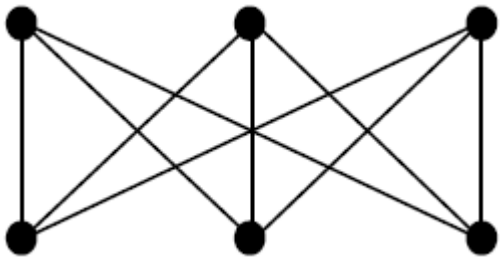
- Appelons subdivision (ou expansion) d'un graphe, tout graphe obtenu en ajoutant un ou plusieurs sommets sur une ou plusieurs arêtes.

- Théorème [Kuratowski,1930]

  Un multigraphe G est planaire si et seulement si il ne contient pas (comme sous graphe partiel) de subdivision du graphe complet à 5 sommets K_5 ni du graphe biparti complet $K_{3,3}$.



K_5

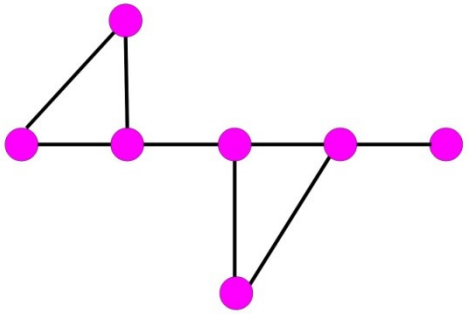


$K_{3,3}$

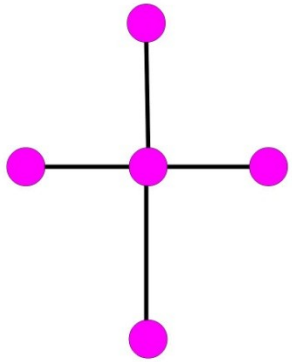
Mineur

- Un graphe *H* est un mineur d'un graphe non orienté *G* si *H* peut être obtenu à partir de *G* en effectuant un nombre quelconque d'opérations parmi les suivantes:
  - suppression d'un sommet isolé *x* : le sommet *x* est supprimé du graphe ;
  - suppression d'une arête {*x,y*} : l'arête {*x,y*} est supprimée, mais ses extrémités sont inchangées ;
  - contraction d'une arête {*x,y*} : l'arête {*x,y*} est supprimée, les deux sommets *x* et *y* sont fusionnés en un sommet *z*. Toute arête {*t,x*} ou {*t,y*} est remplacée par une nouvelle arête {*t,z*}. Une même arête n'est pas ajoutée deux fois (on ne crée pas d'arêtes parallèles).

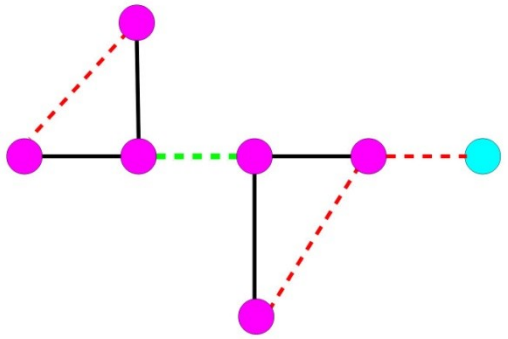
- Illustration



Graphe *G*



un mineur *H* de *G*



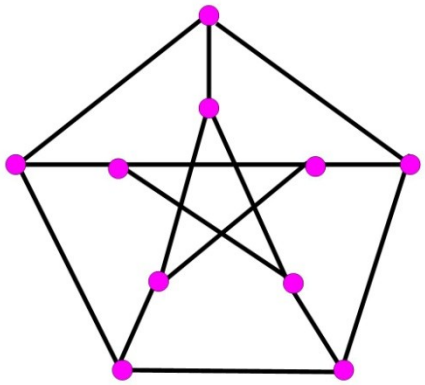
passage de *G* à *H*

Graphes planaires : Wagner

- Théorème [Wagner]

  Un graphe G est planaire si et seulement si il ne compte pas K_5 ni $K_{3,3}$ parmi ses mineurs.

- Illustration



Exploration d'un graphe, concepts

- Explorer un graphe : étant donné un sommet *s*, il s'agit d'atteindre le plus grand nombre d'autres sommets du graphe en parcourant un chemin ayant s pour point de départ.

- Une exploration : liste L ordonnée de sommets qui appartiennent à une chaîne ayant s pour extrémité et telle que :
  - s est le premier élément de la liste L ,
  - chaque sommet de X est représenté au plus une fois dans L
  - pour chaque sommet i dans L, i≠s, il existe un prédécesseur (voisin dans le cas non orienté) de i dans L .

Un algorithme d'exploration générique

- 1. Marquer le sommet *s*; Ouvrir le sommet *s;*
- 2. Tant que il existe un sommet ouvert faire
  - 2.1 Choisir un sommet ouvert *i*;
  - 2.2 Si tous les successeurs (voisins dans le cas non orienté) de *i* sont marqués alors
    - Fermer le sommet *i*;
  - Sinon
    - Choisir un sommet *j* qui est un successeur non marqué (voisin dans le cas non orienté) de *i*;
    - Marquer le sommet *j*;
    - Ouvrir le sommet *j*;

  Fin_si;

Fin_Tant_que;

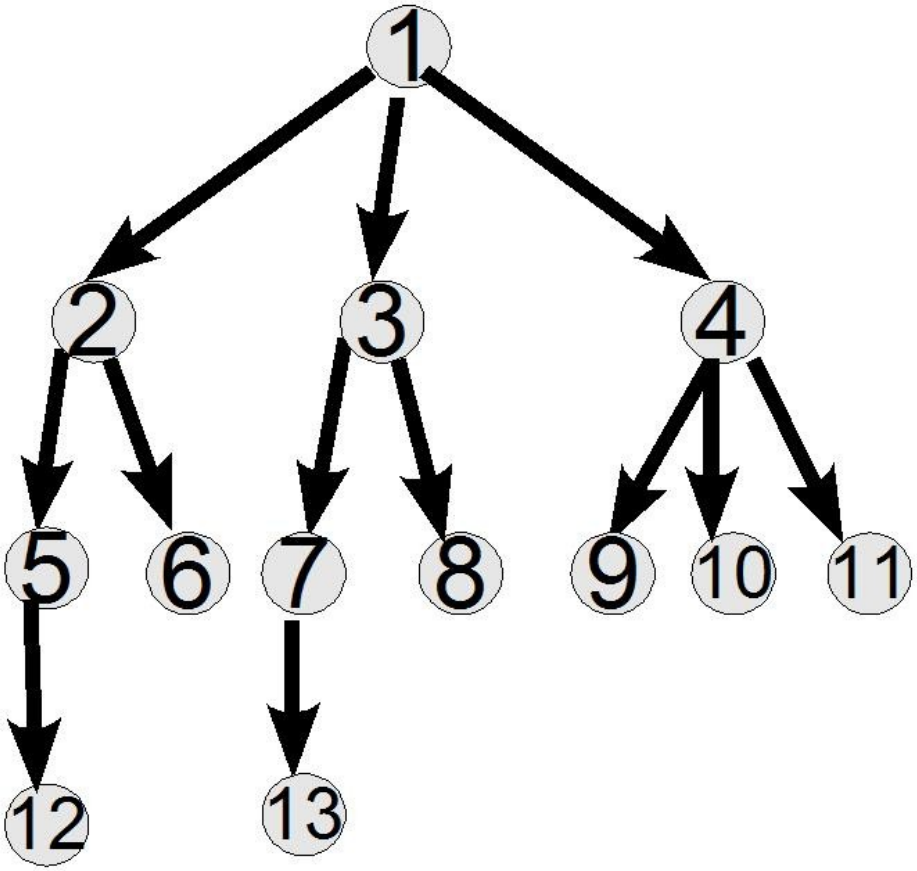
Terminologie. On dit que l'on marque un sommet quand on le rencontre pour la première fois. On dit que l'on ferme un sommet quand tous ses successeurs ont été marqués. Un sommet qui est à la fois marqué et non fermé est dit ouvert.

L'insertion dans la liste d'exploration correspond à l'étape de marquage.

Exploration en profondeur/largeur

- Exploration en profondeur (DFS : Depth-first search)
  - A l'étape 2.1, choisir le dernier sommet ouvert : utiliser une pile pour structure de données
  - Applications: trouver les composantes (fortement) connexes, tri topologique, …
- Exploration en largeur (BFS : Breadth-first search)
  - A l'étape 2.1, choisir le sommet ouvert en premier : utiliser une file pour structure de données
  - Applications: trouver les composantes connexes, tester si un graphe est biparti, trouver un plus court chemin, …
- Complexité temps en O(nombre d'arêtes)

BFS & DFS: un exemple



PARTIE 2

ARBRES COUVRANTS

Contenu

- Notions de base
- Caractérisations des arbres couvrants optimaux
- Algorithme de Kruskal
- Algorithme de Prim

Notions de base

- Un arbre est un graphe connexe sans cycle
- Une forêt est un graphe sans cycle: chaque composante connexe d'une forêt est un arbre

- Les propositions suivantes sont équivalentes :
  - T =(X,U) est un arbre
  - T est connexe et sans cycle
  - T est connexe et minimal (pour l'inclusion) avec cette propriété
  - T est connexe et a |X|-1 arêtes
  - T est acyclique et maximal (pour l'inclusion) avec cette propriété
  - T est acyclique et a |X|-1 arêtes
  - pour chaque paire de sommets il existe une unique chaîne qui les relie

- Un arbre couvrant ou arbre de recouvrement dans un graphe *G=(X,E)* est un graphe partiel de *G* correspondant à un arbre avec *X* pour ensemble de sommets

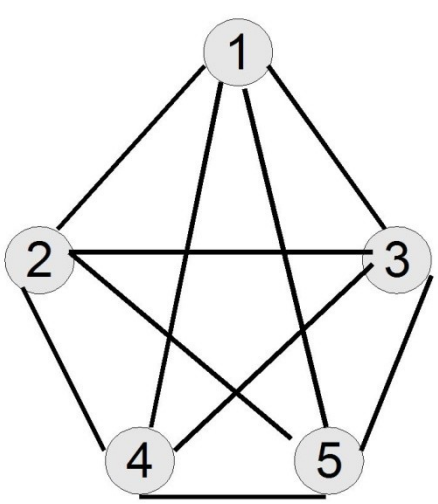
Arbres couvrants optimaux

- Formulation du problème
  - Extraire d'un graphe valué : G=(X,E) avec des coûts $d_e, \forall e \in E$

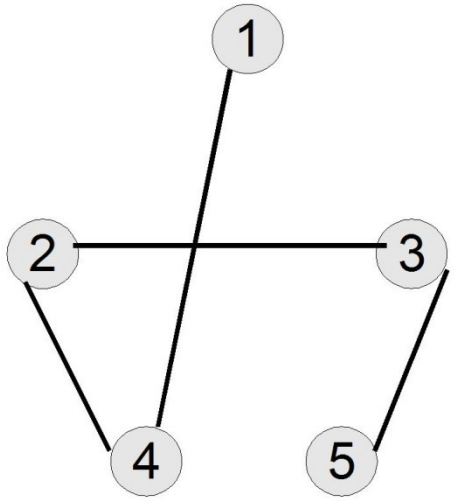
    sur les arêtes, un arbre couvrant T de coût $\sum_{e \in T} d_e$ minimum/maximum.
- Les problèmes de Maximisation/Minimisation sont duaux : multiplier les coûts par '-1' pour résoudre l'autre problème
- Dans la suite on considère le problème de minimisation

Propriétés des arbres couvrants optimaux

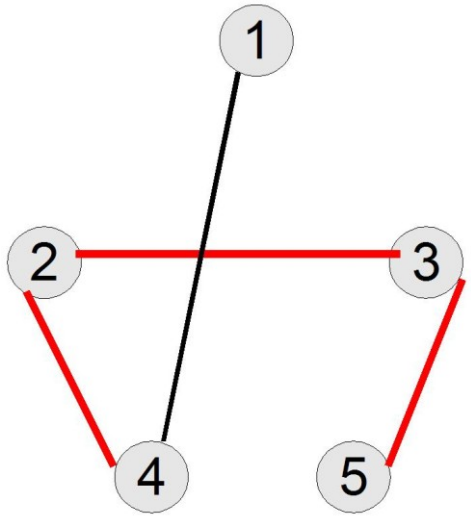
- Soit T=(X,U) un arbre dans un graphe G=(X,E)
- Pour une arête donnée $e \in E \setminus U$, on note c(e) la chaîne reliant les deux extrémités de e dans T



G=(X,E)



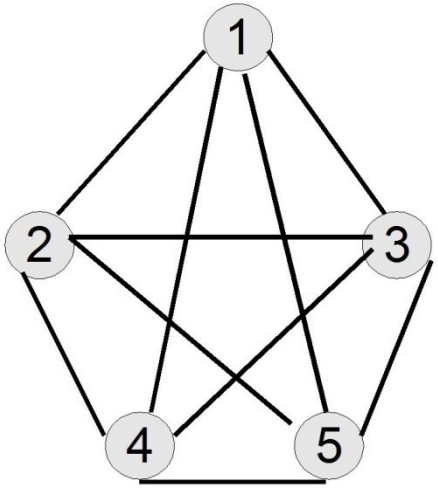
T =*(X,U)*



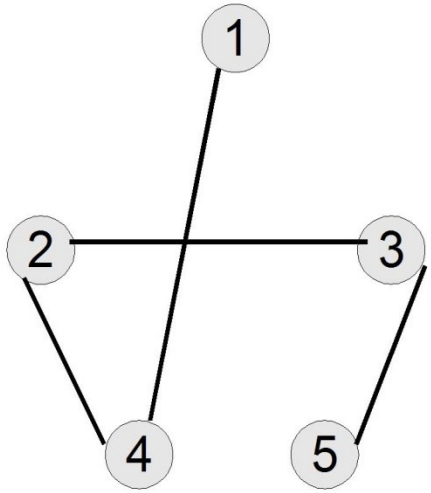
c(4,5)

Propriétés des arbres couvrants optimaux

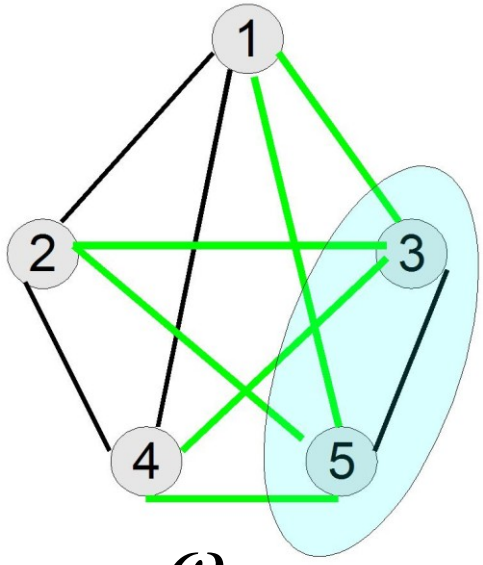
- Soit une arête $v \in U$. En la retirant de **T** on obtient deux composantes connexes.
- Soit ω_v l'ensemble des arêtes qui ont exactement une extrémité dans chacune de ces composantes.



G=(X,E)



T =*(X,U)*



$\omega_{(2,3)}$

Propriétés des arbres couvrants optimaux

$T = (X, U)$ est un arbre couvrant de poids minimum.

(si et seulement si)

$\forall e \in E \setminus U,$ on a : $d_e \geq d_w, \forall w \in c(e)$

(si et seulement si)

$\forall v \in U,$ on a : $d_v \leq d_w, \forall w \in \omega_v$

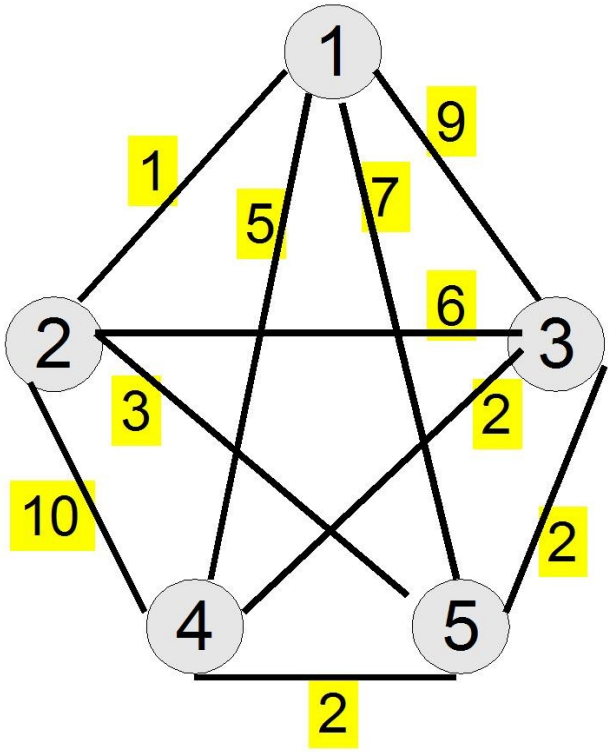
Algorithme de Kruskal

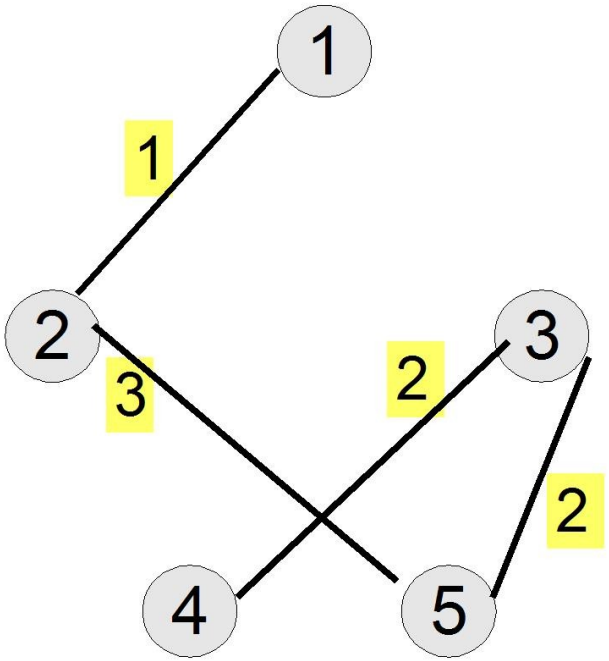
1. Trier les arêtes de *G* par valeurs de coûts croissantes :
   $d_{e_1} \leq d_{e_2} \leq \cdots \leq d_{e_{|E|}}$;
   U←Ø ; i←1 ;
2. **Tant que** |U|<|X|-1 **faire**
   Si $(X, U \cup \{e_i\})$ est sans cycle **alors**
   $U := U \cup \{e_i\}$;
   fin_si ;
   i←i+1 ;
   fin_tantque ;

Remarque. Cet algorithme peut être implémenté avec une complexité en $O(|X|^2 \log_2 |X|)$, avec |X| = nombre de sommets du graphe.

Algorithme de Kruskal

- Un exemple





Algorithme de Prim

1. A←{i} ; (i: un sommet quelconque du graphe) ;
   U←Ø ;
   $\omega \leftarrow \omega_A$;
2. **Tant que** A≠X **faire**

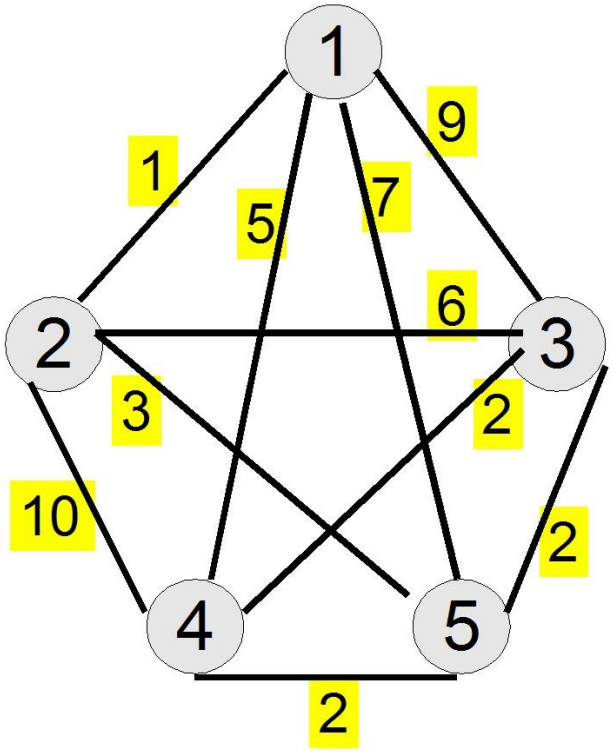
   Choisir dans ω une arête e=(i,j) de coût minimum
   $A \leftarrow A \cup \{ j \}$;
   $U \leftarrow U \cup \{ e \}$;
   $\omega \leftarrow \omega \, \Delta \, \omega_{\{j\}}$;

   fin_tantque ;

Remarque. Il existe une implémentation simple de cet algorithme avec une complexité en $O(|X|^2)$.

Algorithme de Prim

- Un exemple



PARTIE 3

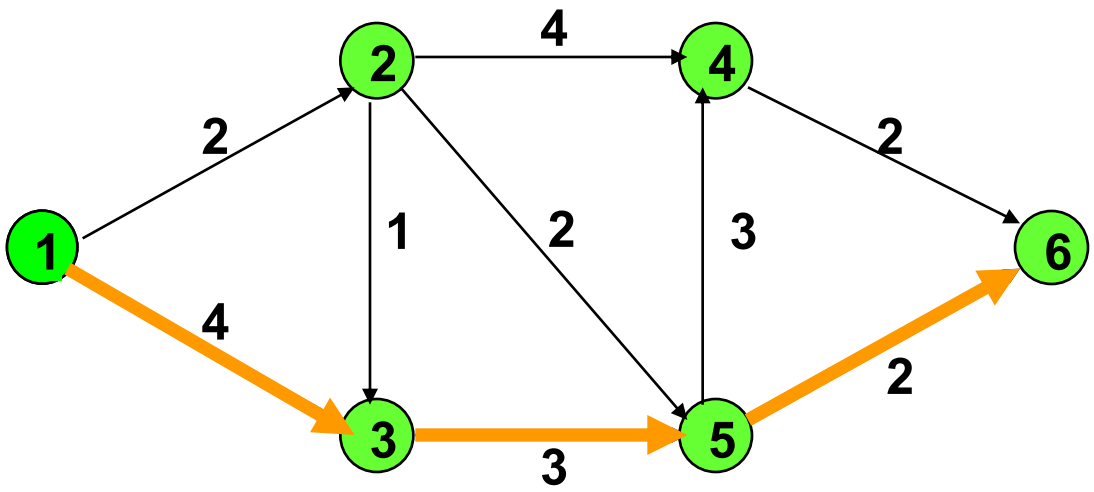
PLUS COURTS CHEMINS

Contenu

- Introduction et motivations
- Algorithmes
  - Cas de longueurs positives ou nulles
  - Cas de longueurs uniformes
  - Cas de longueurs générales
  - Cas d'un graphe sans circuit

Préliminaires

- Illustration
  - Graphe orienté *G=(V,E)*
  - Longueurs (poids, …) sur les arcs : $c_e, \forall e \in E$



<u>Définition</u>
Longueur d'un chemin *P* :

$$l(P) = \sum_{e \in P} c_e$$

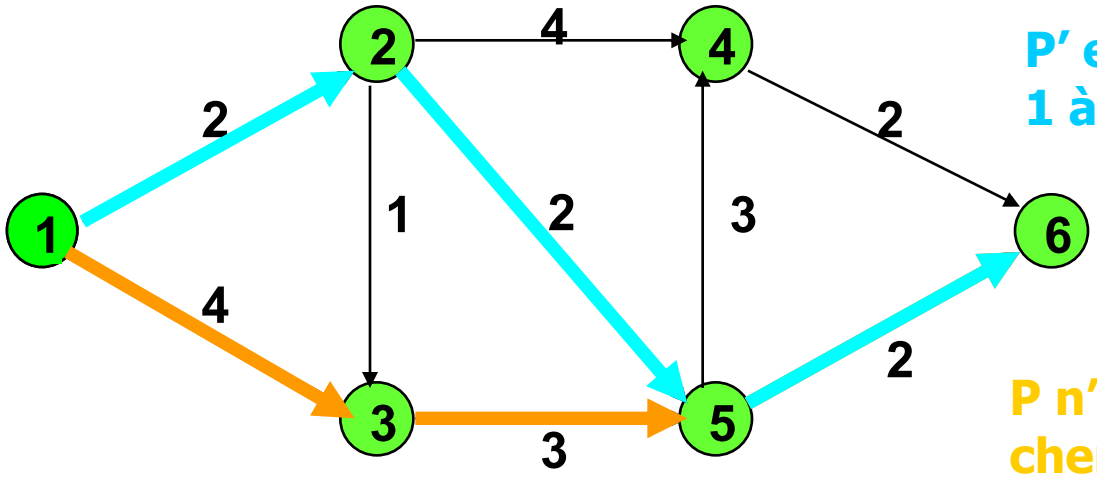
<u>Exemple</u>
Chemin ***P*** de 1 à 6 :

$$l(P) = c_{13} + c_{35} + c_{56}$$
$$= 4 + 3 + 2 = 9$$

Plus court chemin

- Définition

  On appelle **plus court chemin** du sommet s au sommet t un chemin de s à t de longueur minimum.



P' est un plus court chemin de 1 à 6 (de longueur 6)

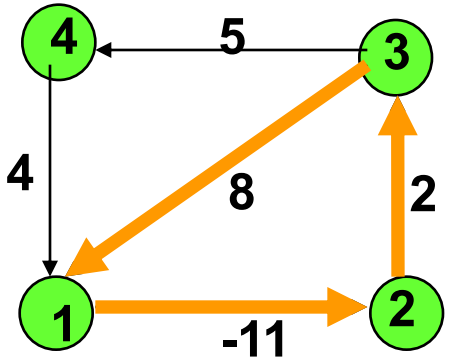
P n'est PAS un plus court chemin de 1 à 6 (de longueur 9)

Existence d'un plus court chemin

- Préliminaires: conditions pour l'existence d'un plus court chemin du sommet s au sommet t

Condition 1
Il existe au moins un chemin de *s* à *t*

Condition 2
Aucun chemin de *s* à *t* ne comporte de circuit de longueur<0



Il n'existe pas de plus court chemin de 3 vers 4

Problèmes de plus courts chemins

1. Etant donnés deux sommets *s* et *t*, trouver **un** plus court chemin de *s* à *t*,
2. Trouver les plus courts chemins depuis **un** sommet s vers **tous** les autres sommets,
3. Trouver les plus courts chemins pour **toutes les paires** (ordonnées) de sommets.

Motivations

- Des applications très diverses
  - Problèmes d'optimisation dans les réseaux (télécommunications / routiers, …),
  - Problèmes d'investissements, de gestion de stocks,
  - Traitement du signal, (dé)codage de l'information…
- Méthodes de résolution « Efficaces »

Méthodes

1. Algorithme de Moore-Dijkstra
   - Cas de longueurs ≥0
2. Cas de longueurs uniformes (>0)
3. Algorithme de Bellman-Ford
   - Cas de longueurs générales
   - Détection de circuits de longueur < 0
4. Algorithme de Bellman
   - Cas de graphes sans circuit

Propriétés des plus courts chemins

Propriété 1
Si ***P=(1,2,…,h)*** est un plus court chemin de 1 vers h alors ***(1,2,…,q)*** est un plus court chemin de ***1*** vers ***q***, ***q=2,…,h-1***

Propriété 2
Soit ***d(i)*** la longueur d'un plus court chemin de ***1*** vers ***i.***

P est un plus court chemin de 1 à k ⇔

$$d(j) = d(i) + c_{ij}, \forall ij \in P$$

Propriété 3
Soit ***π(i)*** la longueur d'un **chemin quelconque** de ***1*** à *i*, alors

$$d(j) \leq \pi(i) + c_{ij}$$

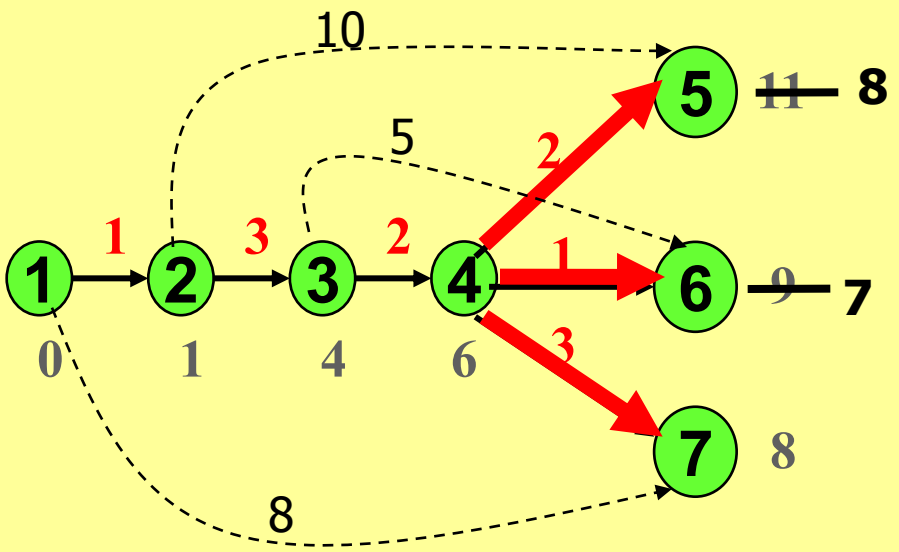
Etape basique (Moore-Dijkstra)

- Soit $\pi(i)$ la longueur d'un certain chemin de ***1*** à ***i***
  et ***pred(i)*** le prédécesseur de ***i*** sur ce chemin
- Pour un sommet ***i*** donné,

Update_Label(*i*)

Pour chaque arc *(i,j)* faire
Si $\pi(j) > \pi(i) + c_{ij}$ alors
$\pi(j) := \pi(i) + c_{ij}$;
pred(*j*) : = *i* ;
Fin_Si;

Exemple



Algorithme de Moore-Dijkstra

Hypothèse
Longueur des arêtes ≥0

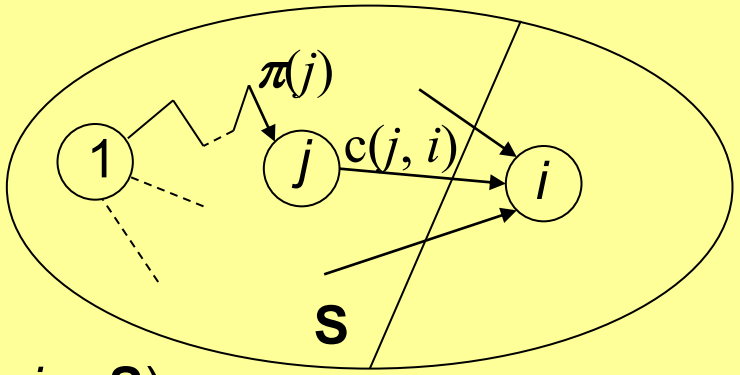
Notations
d(j) : longueur d'un plus court chemin de ***1*** à ***j***
π(j) : marquage (« étiquette ») temporaire du sommet ***j***

Principe
L'algorithme va construire un sous-ensemble **S** de sommets (au départ **S** = {1}) en s'appuyant sur le marquage suivant:

$\pi(i) = d(i)$ si $i \in \mathbf{S}$

$\pi(i) = \min_{j \in \mathbf{S} \cap \Gamma^-_i} (\pi(j) + c(j, i))$ sinon

($\pi(i) = +\infty$ s'il n'y a pas d'arc (j, i) avec $j \in \mathbf{S}$)



Algorithme de Moore-Dijkstra

La validité de l'algorithme repose sur le lemme suivant.

Lemme : Soit $k \in \mathbf{X}\backslash\mathbf{S}$ tel que $\pi(k) = \min_{i \in \mathbf{X}\backslash\mathbf{S}} \pi(i)$. Alors $\pi(k) = d(k)$.

Preuve.

Il existe évidemment un chemin de 1 à *k* de longueur $\pi(k)$, montrons qu'il n'en existe pas de plus court.

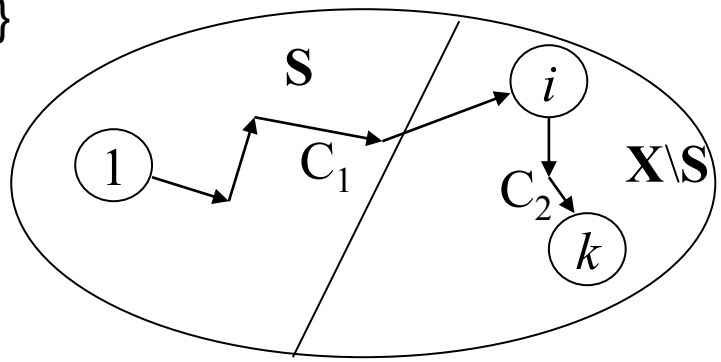
Soit C un chemin de 1 à *k* et soit *i* le premier sommet de ce chemin appartenant à **X\S**.

$C = C_1\ C_2$ avec $C_1 = \{1,\ldots,i\}$ et $C_2 = \{i,\ldots,k\}$

On a: $L(C_1) \geq \pi(i)$ et $L(C_2) \geq 0$

(d'où positivité des poids nécessaire)

$L(C) = L(C_1) + L(C_2) \geq \pi(i) \geq \pi(k)$.



Algorithme de Moore-Dijkstra

Recherche d'un plus court chemin de 1 à *t*

Moore_Dijkstra

$S := \{1\}$;
$pred(j) := 0$, j = 1,…, |V| ;
$\pi(1) := 0$;
$\pi(j) := \infty$, j = 2,…, |V| ;
Update_Label(1);

Tant que $t \notin S$ faire

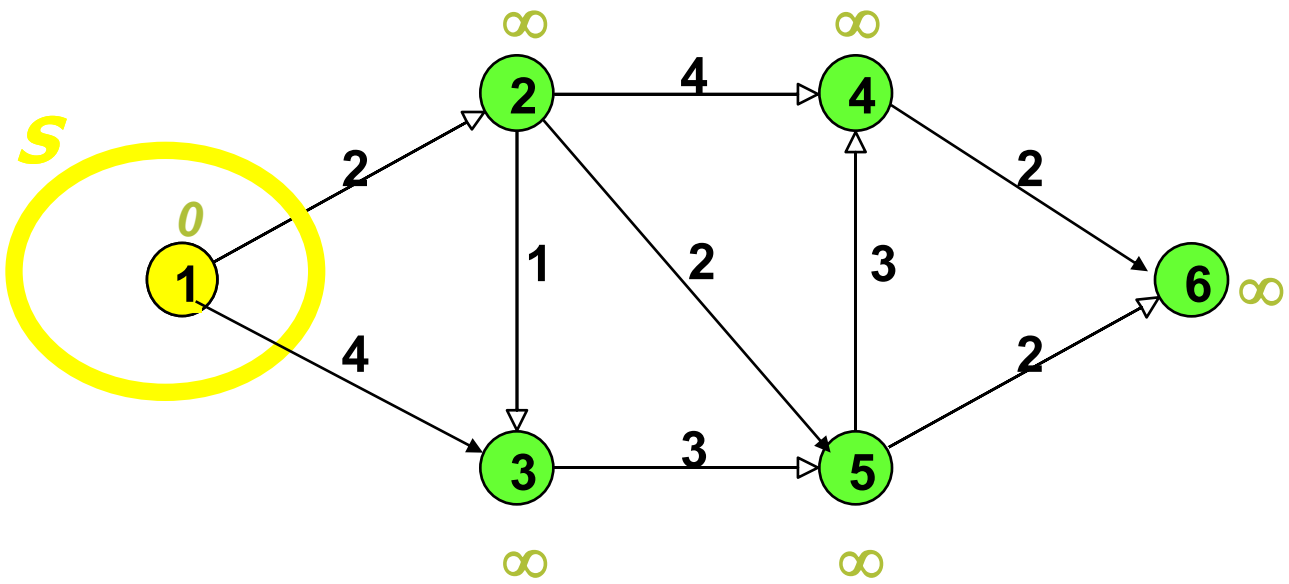
Choisir *k* tel que $\pi(k) = \min_{i \in V \setminus S} \pi(i)$;
$S := S \cup k$
Update_Label(*k*) ;

Fin_tantque;

Pour trouver les plus courts chemins de 1 vers tous les autres sommets : remplacer le critère d'arrêt par : $S \neq V$

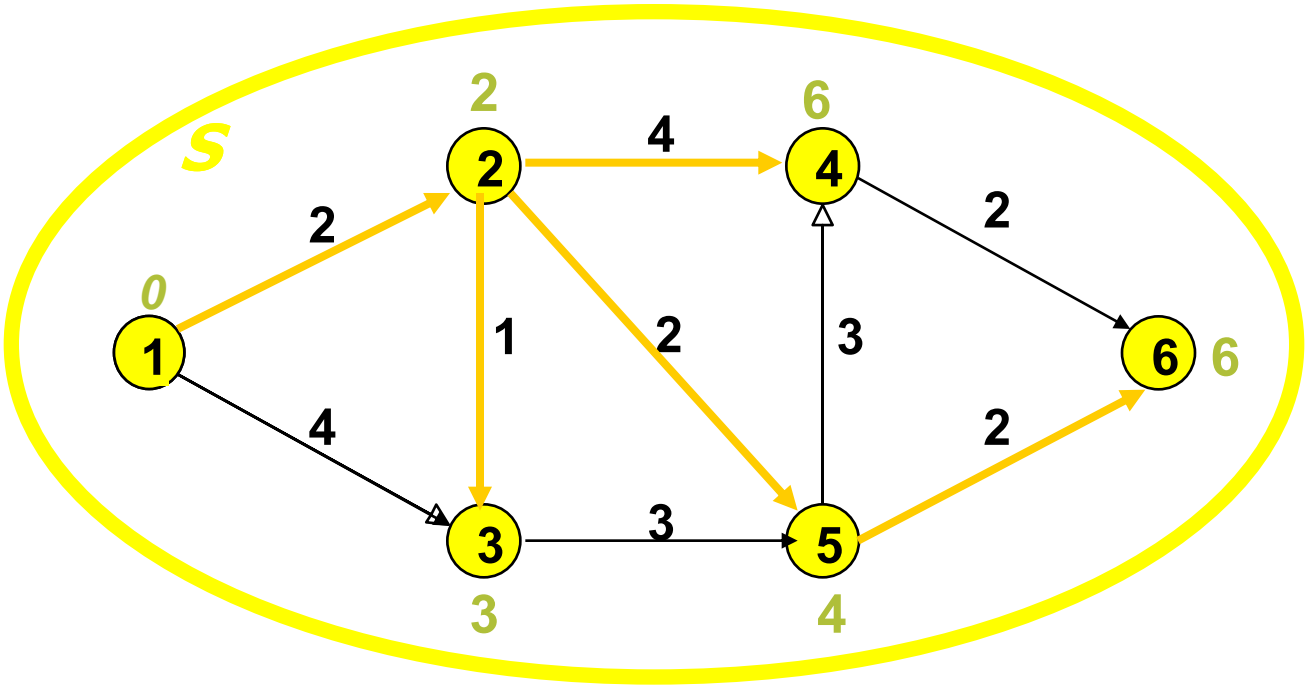
Exemple

- Initialisations



Exemple

- $S=V$



Complexité (Moore-Dijkstra)

- Marquage des sommets
  - Chaque arc est parcouru une fois au plus

    ⟶ $m=|E|$ opérations
- Sélection du sommet à insérer S
  - A chaque itération |V\S| comparaisons

    ⟶ $(n-1)+(n-2)+\ldots+1=n(n-1)/2$, avec $n=|V|$

    $O(n^2)$ opérations

Complexité de l'algorithme : $O(n^2)$

Cas de longueurs unitaires

- Notation: $\pi(i)$ désigne une valeur de marquage du sommet i, qui, lorsqu'elle prendra une valeur finie correspondra à la longueur d'un plus court chemin de *1* à *i*

Unit_Lengths

$\mathbf{S} := \mathbf{S}_0 := \{1\};$

k := 0 ;

$\pi(1) := 0$;

$\pi(j) := \infty$, j = 2,..., |X| ;

Tant que $\mathbf{S} \neq \mathbf{X}$ faire

$\mathbf{S}_{k+1} := \Gamma(\mathbf{S}_k) \setminus \mathbf{S};$

$\pi(i) := k+1, \ \forall i \in S_{k+1};$

$\mathbf{k} := \mathbf{k}+1;$

$\mathbf{S} := \mathbf{S} \cup \mathbf{S}_k;$

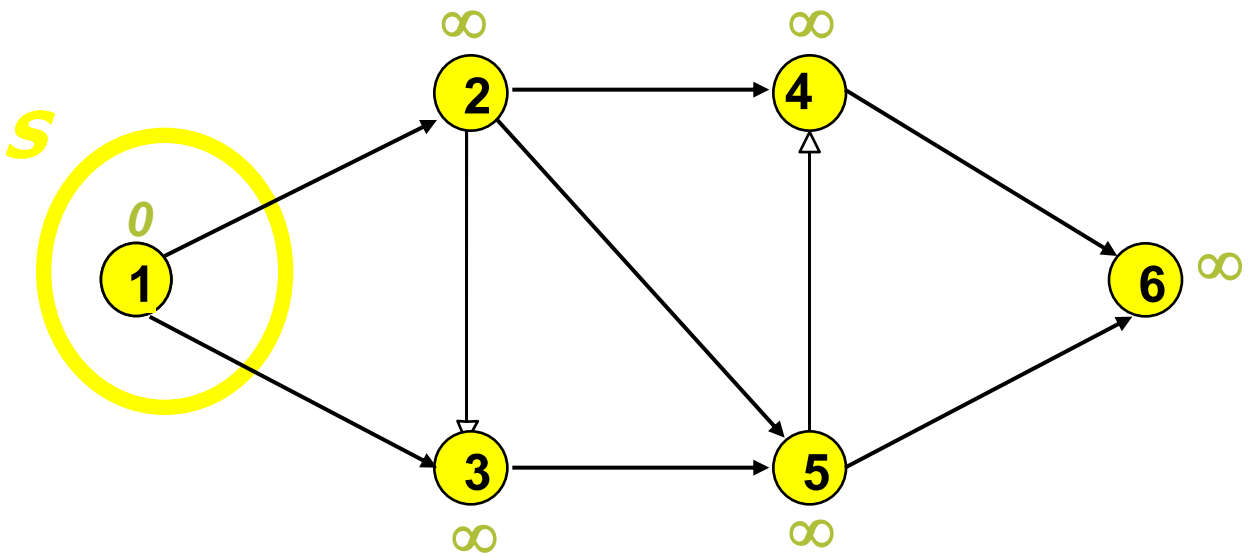
Fin_tantque;

Notation

Γ(S): ensemble des sommets de *G* qui sont adjacents à un sommet (au moins) dans *S*.

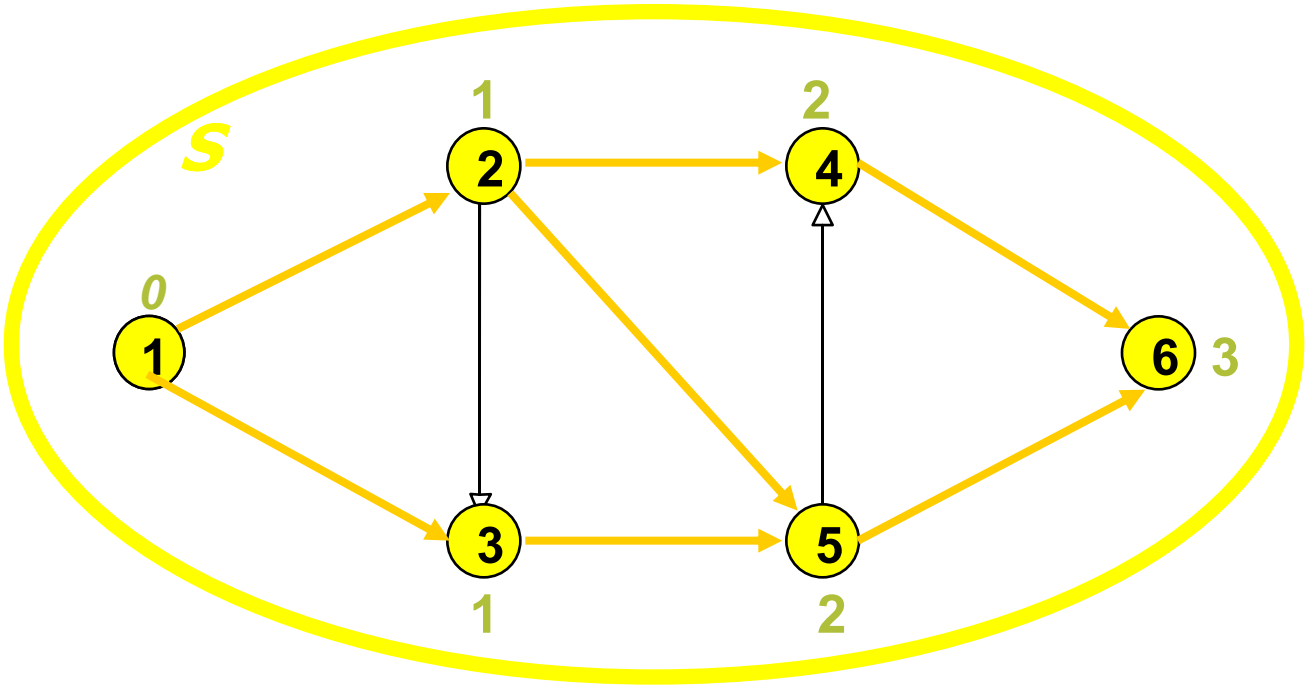
Exemple

- $\mathbf{S}_0 = \{1\}$



Exemple

- **S = X**



Cas de longueurs unitaires : complexité

- Marquage des sommets
  - Chaque arc est parcouru une fois

    → $m=|E|$ opérations

Complexité de l'algorithme : O(m)

Algorithme de Bellman-Ford

- Préliminaires

Propriété

Soit P_k un plus court chemin de *1* à *t* avec au plus k arcs.
Si x est le prédécesseur de *t* dans P_k alors le sous-chemin de 1 à x est un plus court chemin avec au plus k-1 arcs.

Propriété

En notant $d^k(t)$ la longueur minimale d'un chemin de *1* à *t* avec au plus k arcs, on a :

$$d^k(t) = \min\left\{ d^{k-1}(t),\ \min_{y:yt\in E}(d^{k-1}(y) + c_{yt})\right\}$$

Etape basique (Bellman-Ford)

- Soit $\pi^k(i)$ la longueur d'un certain chemin de ***1*** à ***i*** avec au plus *k* arcs et $\boldsymbol{pred}^k(\boldsymbol{i})$ le prédécesseur du sommet ***i*** sur ce chemin

Update_Label2(i)

$$\pi^k(i) := \pi^{k-1}(i)$$

Pour chaque arc *(j,i)* faire

Si $\pi^k(i) > \pi^{k-1}(j) + c_{ji}$ **alors**

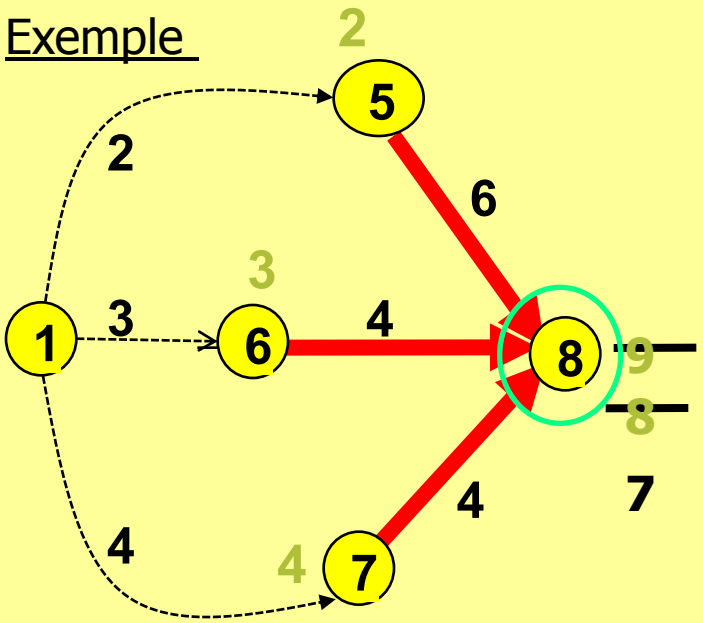
$$\pi^k(i) := \pi^{k-1}(j) + c_{ji}$$

$$\boldsymbol{pred}^k(\boldsymbol{i}) := \boldsymbol{j};$$

Fin_si;

Fin_pour;

Exemple



Algorithme de Bellman-Ford

Ford

$k := 0$;

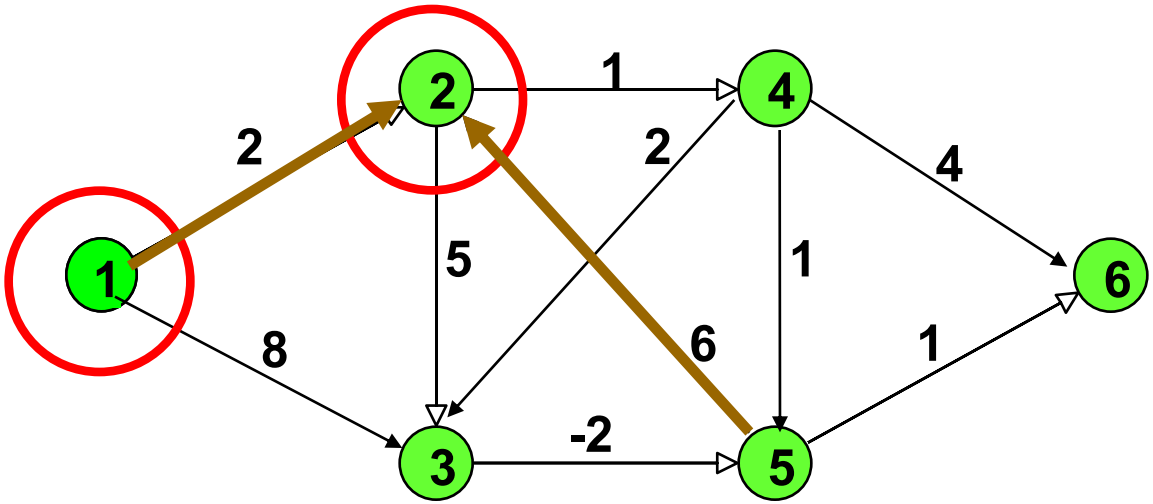
$\pi^0(1) := 0$; $pred^0(1) := 0$;

$\pi^0(j) := \infty$, $pred^0(j) := 0$, $j = 2,...,|V|$

Répéter
  k:=k+1 ;
  Pour chaque sommet x faire
    Update_Label2(x) ;
  Fin_pour;
jusqu'à k=|V| ou aucune étiquette modifiée sur une itération;

Si une étiquette a été modifiée à l'itération |V| alors il existe un circuit de longueur <0.

Exemple



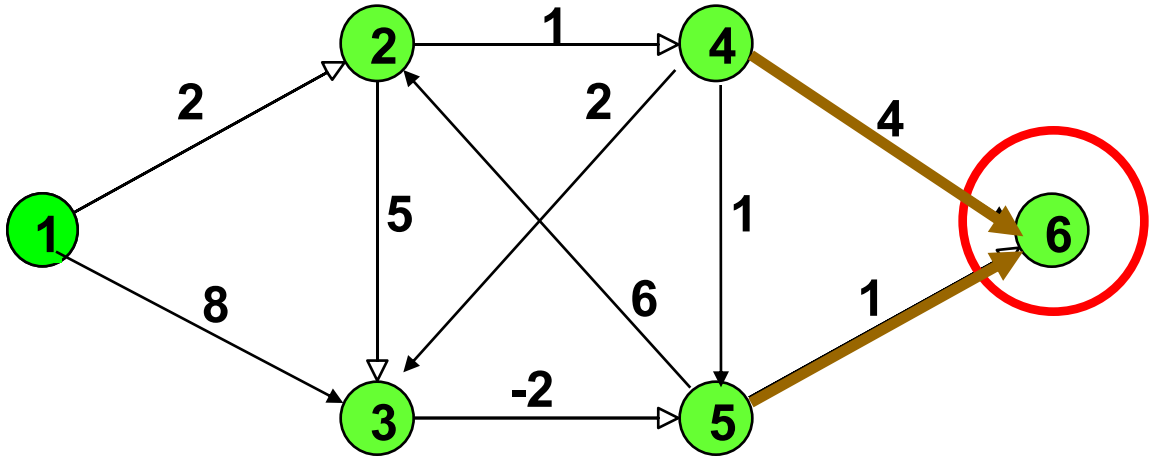
K=0

π	0	∞	∞	∞	∞	∞
pred	-	-	-	-	-	-
	1	2	3	4	5	6

K=1

π	0	2				
pred	-	1				
	1	2	3	4	5	6

Exemple



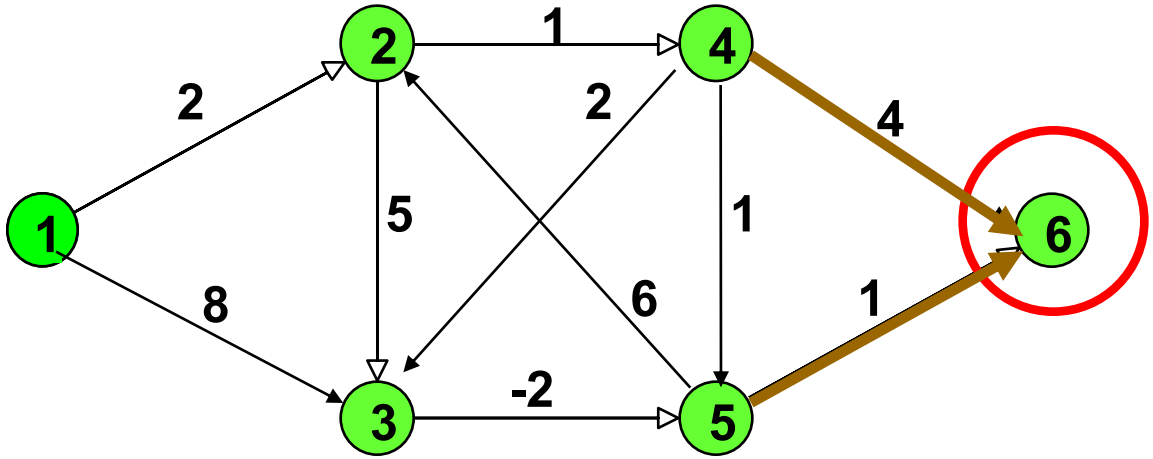
K=0

π	0	∞	∞	∞	∞	∞
pred	-	-	-	-	-	-
	1	2	3	4	5	6

K=1

π	0	2	8	∞	∞	∞
pred	-	1	1	-	-	-
	1	2	3	4	5	6

Exemple



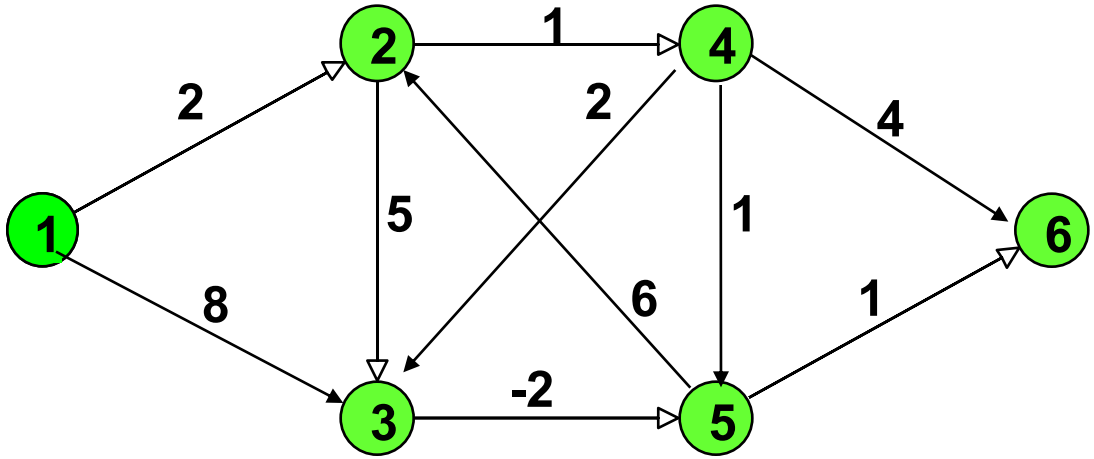
K=1

π	0	2	8	∞	∞	∞
pred	-	1	1	-	-	-
	1	2	3	4	5	6

K=2

π	0	2	7	3	6	∞
pred	-	1	2	2	3	-
	1	2	3	4	5	6

Exemple



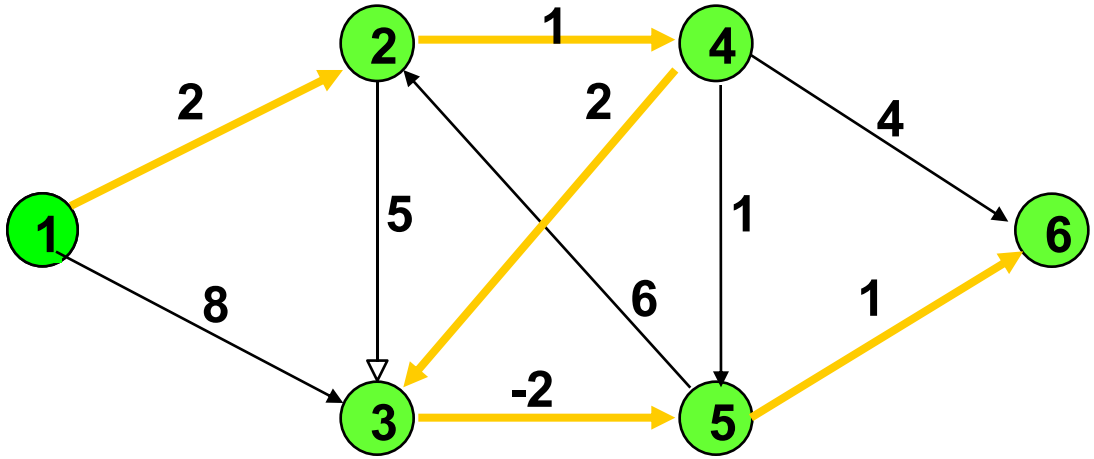
K=3

π	0	2	5	3	4	7
pred	-	1	4	2	4	4
	1	2	3	4	5	6

K=4

π	0	2	5	3	3	5
pred	-	1	4	2	3	5
	1	2	3	4	5	6

Exemple



K=5

π	0	2	5	3	3	4
pred	-	1	4	2	3	5
	1	2	3	4	5	6

K=6
Pas de modification

Complexité (Bellman-Ford)

- Marquage des sommets
  - A chaque itération chaque arc est parcouru une fois exactement

    ⟹ m=|E| opérations

- Nombre d'itérations: au plus n=|V|

Complexité de l'algorithme : O($n.m$)

Graphes sans circuit

Propriété
Un graphe sans circuit a au moins un sommet sans prédécesseur. Un tel sommet est appelé **racine** du graphe.

Définition
On appelle **ordre topologique** d'un graphe sans circuit une numérotation des sommets du graphe *G=(X,E),* telle que $(i, j) \in E \Rightarrow i < j$.

Définition
La **fonction rang** *r* associée à un graphe G=(X,E) avec '1' pour racine vérifie :

- *r(1)=0*,
- *r(i)*=longueur maximum (en nombres d'arcs) d'un chemin de 1 à i.

Ordre topologique

Ordre topologique

$\pi(i) := \left|\{j \in X : ji \in E\}\right|, \forall i \in X; S_0 := \{1\}; k := 0;$

Tant que $S_k \neq \emptyset$ faire

$S_{k+1} := \emptyset;$

Pour chaque sommet $i \in S_k$ faire

r(i):=k;

Pour chaque sommet *j* tel que $ij \in E$ faire

$\pi(j) := \pi(j) - 1;$

Si $\pi(j) = 0$ alors

$S_{k+1} := S_{k+1} \cup \{j\};$

Fin_si;

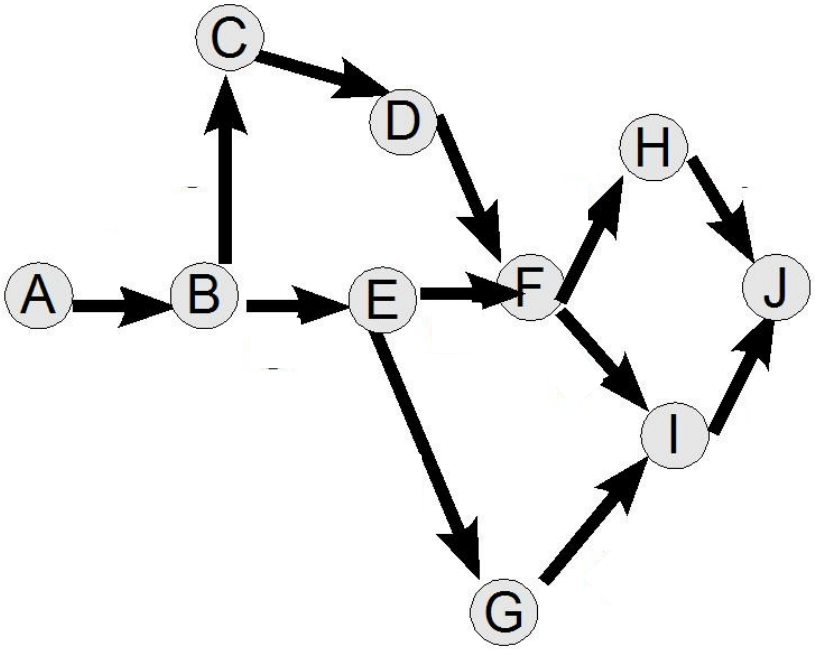
Fin_pour;

Fin_pour;

k:=k+1;

Fin_tantque;

Exemple



Algorithme de Bellman

Bellman

1. **Initialisations**

$$\pi(1) := 0;$$

$$\pi(i) := +\infty,\ \forall i \in X \setminus \{1\};$$

2. **Renumérotation des sommets dans un ordre topologique ;**

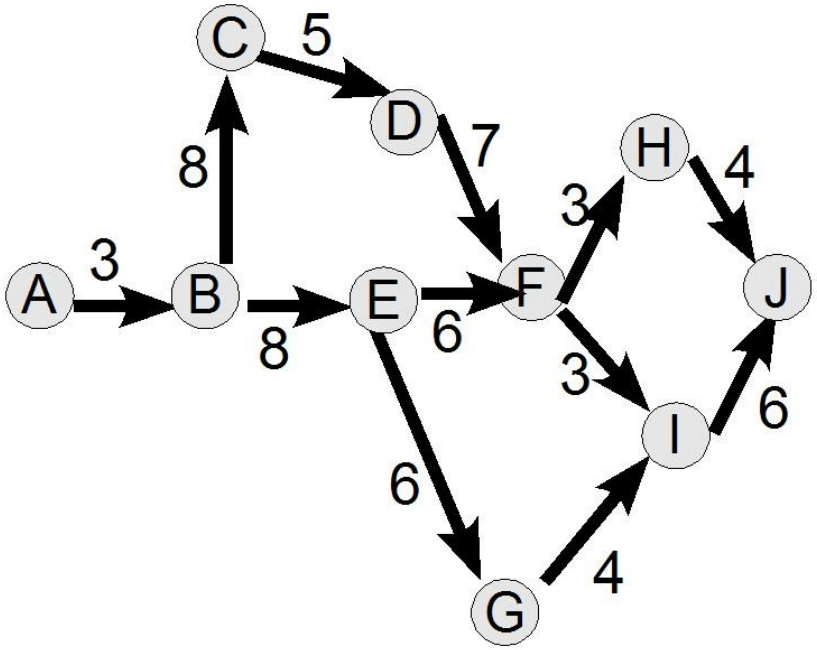
3. Pour k de 1 à |X| faire

$$\pi(k) := \min_{j} \{ \pi(j) + l_{jk} \mid jk \in E \};$$

Fin_pour;

Complexité de l'algorithme : O(|E|)

Exemple



Conclusion

- Plusieurs méthodes pour la résolution de problèmes de plus courts chemins mais attention …

aux conditions de validité

à la complexité

PARTIE 4

FLOTS DANS LES RESEAUX

Contenu

- Préliminaires
  - Problème de transport
  - Propriétés des coupes dans un graphe
  - Graphe d'écart
- Algorithme de Ford & Fulkerson
- Extensions

Préliminaires – Notion de flot

- Soit G=(X, U, c) un graphe orienté connexe et valué sur les arcs, avec |U| = M. Un **flot** dans G est un vecteur à M composantes:

$$\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_M)^T \in R^M,$$

tel que **pour tout sommet i du graphe**, la première loi de Kirchhoff (loi de conservation) est vérifiée:

$$\Sigma_{u \in \omega+(i)} \varphi_u = \Sigma_{u \in \omega-(i)} \varphi_u \ .$$

- Soit A la matrice d'incidence sommets-arcs représentative du graphe. L'égalité précédente peut s'écrire:

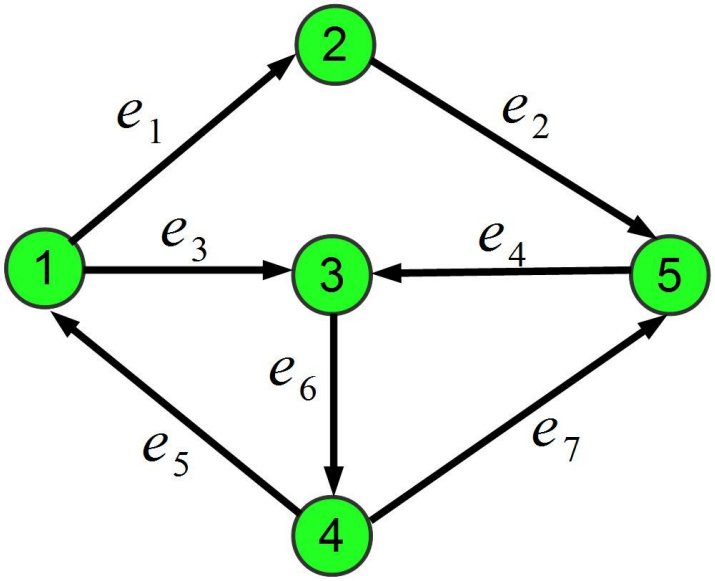
$$A \varphi = 0.$$

- La composante φ_u du flot φ est appelée **quantité de flot** ou **flux** sur l'arc u.

✓ *Remarque* : un courant circulant dans un réseau est un exemple classique de flot.

Préliminaires – Notion de flot

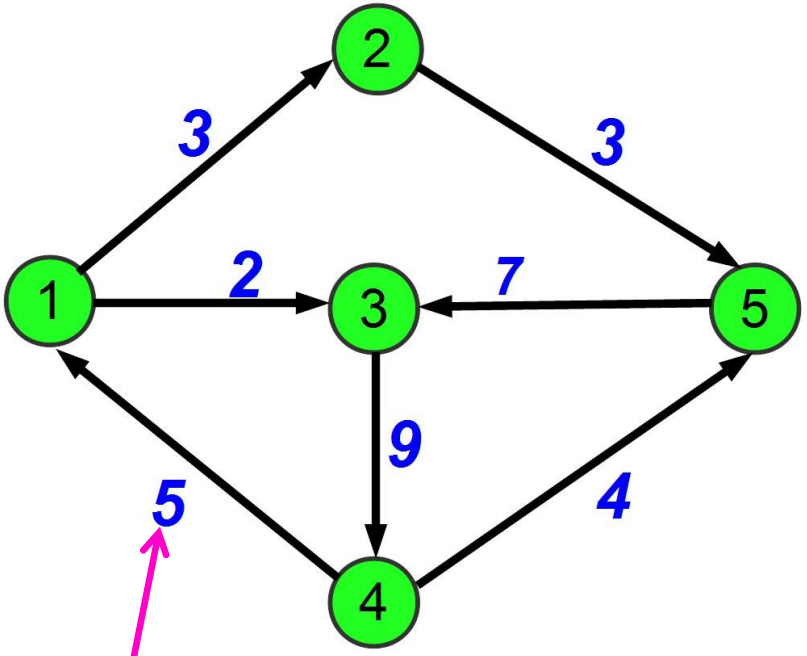
- Exemple



$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

Préliminaires – Notion de flot

- Exemple de flot



$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

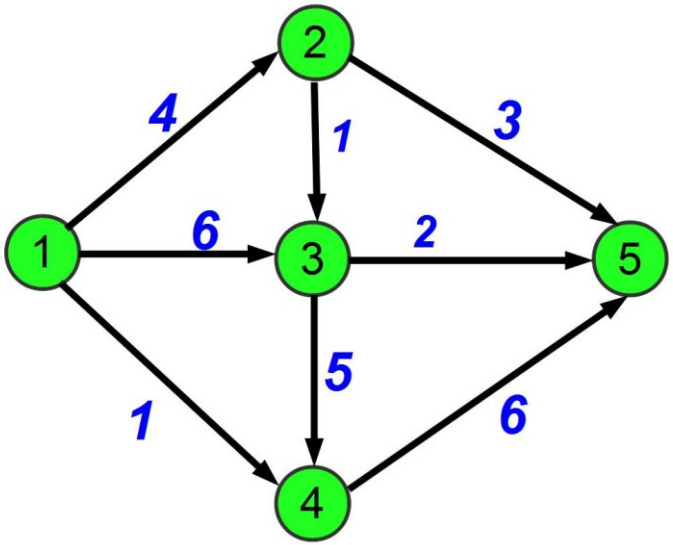
Flot φ=(3,3,2,7,5,9,4)

Préliminaires – Flot de *s* à *t*

- Etant donnés deux sommets particuliers dans un graphe orienté G =(X,U) : un sommet « source » s et un sommet « puits » t, un vecteur φ est un **flot de *s* à *t*** dans G si et seulement si les lois de conservations aux nœuds sont vérifiées en tous les sommets de G **SAUF aux sommets *s* et *t*** où on a :

$$\Sigma_{u\in\omega+(s)}\ \varphi_u = \Sigma_{u\in\omega-(t)}\ \varphi_u = \varphi_0$$

- La quantité φ_0 est appelée la **valeur du flot** φ.
- Exemple

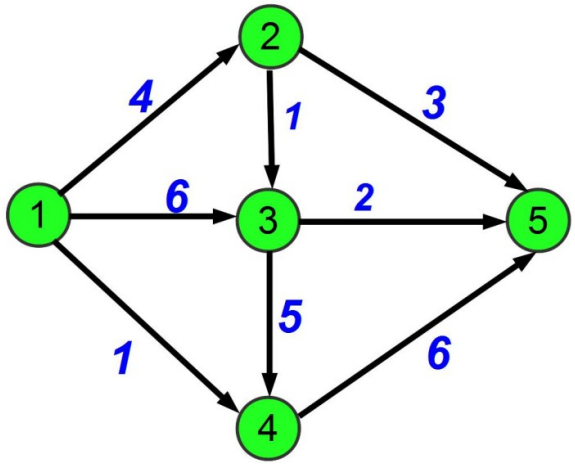


Flot de 1 à 5 de valeur 11

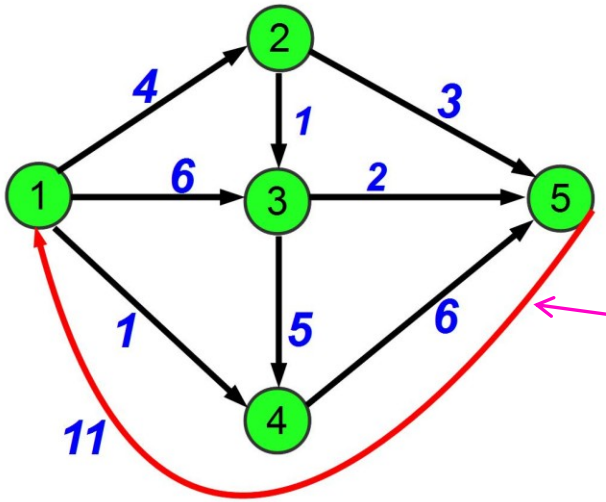
Préliminaires – Flot et flot de *s* à *t*

- Etant donnés deux sommets particuliers dans un graphe orienté $G=(X,U)$: un sommet « source » s et un sommet « puits » t, on considère le graphe G^0 déduit de G en ajoutant l'arc (t,s) dit « **arc de retour** » de capacité infinie auquel on attribue l'indice *0*.
- Si φ est un flot de s à t dans G de valeur φ_0, alors $\varphi'=(\varphi_0, \varphi)$ est simplement un flot dans G^0.

- Exemple



Flot de *1* à *5* de valeur 11 dans *G*.



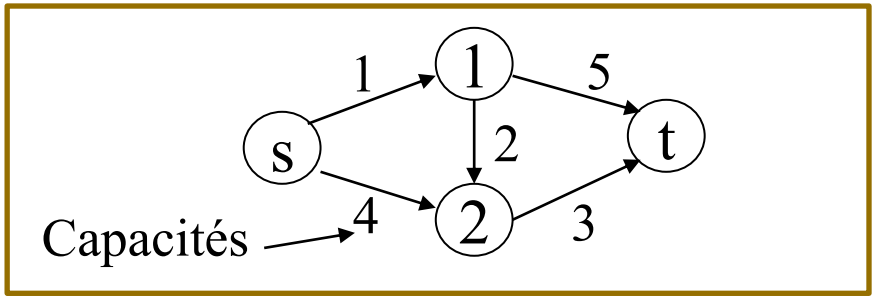
Arc de retour

Flot dans G^0.

Problème de flot maximum

- Un **réseau de transport** est un graphe orienté $G=(X,U,c)$ dans lequel chaque arc u est muni d'une valeur $c_u \geq 0$, appelée **capacité** de l'arc u ; elle représente le flux maximum admissible sur cet arc.
- Un **flot** φ dans G est dit **réalisable** si et seulement si : $0 \leq \varphi_u \leq c_u$, $\forall u \in U$.

- Le **problème du flot maximum de *s* à *t* dans *G*** consiste à déterminer dans G un flot réalisable de s à t et de valeur maximum.
- De manière équivalente, il s'agit de déterminer dans G^0 un flot réalisable, et tel que la valeur du flux φ_0 sur l'arc de retour soit maximum.
- Illustration – Une instance du problème de flot maximum (dans G).



Propriétés dans les réseaux de transports

- Une **coupe** qui sépare deux sommets *s* et *t* est un ensemble d'arcs de la forme $\omega^+(A)$ avec $s \in A, t \notin A$.

- La **capacité d'une coupe** désigne la quantité : $\sum_{e \in \omega^+(A)} c_e$.

- Proposition. Etant donné un flot Φ du sommet *s* au sommet *t* et un ensemble *A* avec $s \in A, t \notin A$, la quantité :

$$\sum_{e \in \omega^+(A)} \phi_e - \sum_{e \in \omega^-(A)} \phi_e$$

ne dépend pas de l'ensemble *A* et vaut ϕ_0.

Propriétés dans les réseaux de transports

- Proposition. Soit Φ un flot réalisable et $\omega^+(A)$ une coupe qui sépare les sommets *s* et *t*. Alors :

  - $\phi_0 \leq \sum_{e \in \omega^+(A)} c_e$
  - Si $\phi_0 = \sum_{e \in \omega^+(A)} c_e$ alors $\begin{cases} \phi_0 \text{ est maximum et} \\ \omega^+(A) \text{ a une capacité minimum} \end{cases}$
  - $\phi_0 = \sum_{e \in \omega^+(A)} c_e \Leftrightarrow \begin{cases} \text{pour chaque arc dans } \omega^+(A),\ \phi_e = c_e, \text{ et} \\ \text{pour chaque arc dans } \omega^-(A),\ \phi_e = 0. \end{cases}$

Théorème du flot maximum et de la coupe minimum

Le théorème fondamental qui suit est une conséquence de l'algorithme de Ford et Fulkerson présenté dans la suite.

Théorème [« FLOT MAX=COUPE MIN »].

Soit G=(X,U,c) un graphe (orienté) avec des valeurs de capacité sur les arcs, toutes rationnelles (et positives). Soit *s* la source et *t* le puits. Alors

- un flot (de valeur) maximum de *s* à *t* peut être calculé en temps polynomial.
- valeur du flot maximum = capacité minimum d'une coupe séparant *s* de *t*.

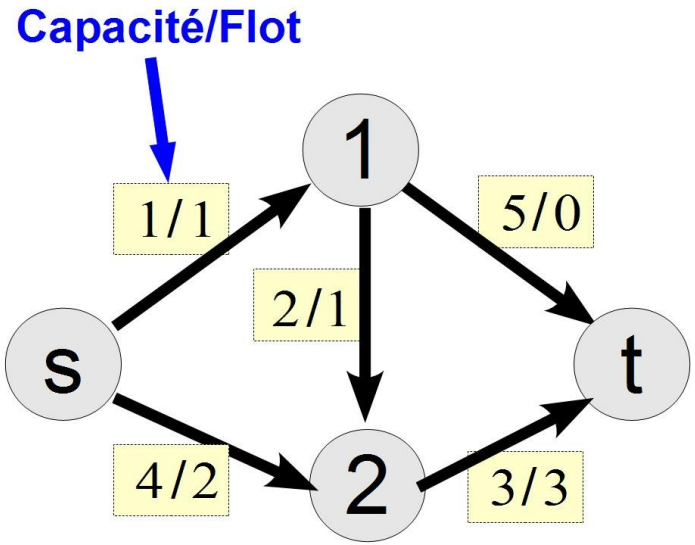
De plus, si les capacités sont entières il est possible de trouver un flot maximum entier (en temps polynomial).

Graphe d'écart

- Etant donné un chemin *P*, sa capacité résiduelle correspond à :

$$\delta(P) = \min_{e \in P}(c_e - \phi_e)$$

- Une « première idée » pour calculer un flot de valeur maximum :
  - Ajouter itérativement du flot sur des chemins de s à t qui ont une capacité résiduelle >0
  - Cela est-il correct ?



Graphe d'écart

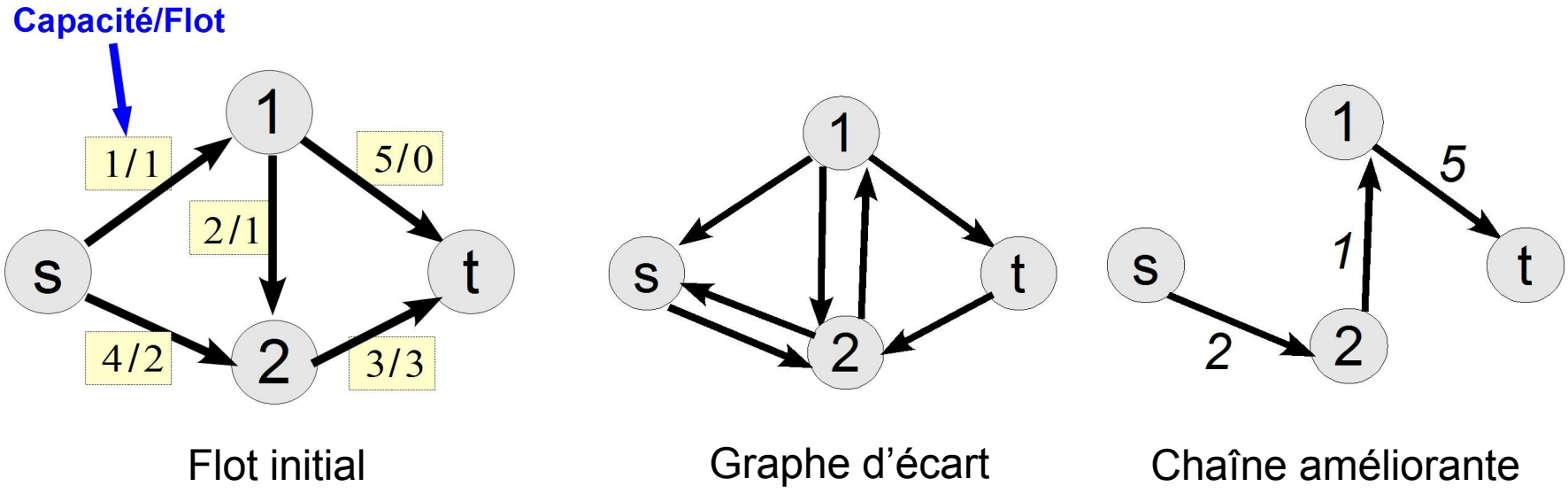
- Soit $\phi = (\phi_1, \phi_2, \cdots, \phi_M)^T$ un flot de *s* à *t* dans le graphe *G=(X,E).*

  Le graphe d'écart G'_ϕ associé à *Φ* et *G* a pour
  - Ensemble de sommets : *X*
  - Ensemble d'arcs *E':*
    - A chaque arc *e=(i,j)* de G on fait correspondre au plus 2 arcs dans G'_ϕ

      - $e^+ = (i,j)$ si $\phi_e < c_e$ avec pour capacité $c_e - \phi_e > 0$,
      - $e^- = (j,i)$ si $\phi_e > 0$ avec pour capacité ϕ_e.

Graphe d'écart

- Théorème. Le flot Φ est de valeur maximum ssi il n'existe pas de chemin de s à t dans le graphe d'écart associé G'_{ϕ}
- Un chemin de s à t dans le graphe d'écart est appelé chaîne améliorante
- Illustration



Algorithme de Ford & Fulkerson

1. Initialisations: $k \leftarrow 0$; $\phi^k \leftarrow 0$;
2. Chercher un chemin π^k de s à t dans le graphe d'écart ;

Si un tel chemin existe alors

$$\phi_e^{k+1} = \begin{cases} \phi_e^k + \delta(\pi^k) \text{ if } e^+ \in \pi^k, \\ \phi_e^k - \delta(\pi^k) \text{ if } e^- \in \pi^k \\ \phi_0^k + \delta(\pi^k) \text{ if } e = (t,s) \end{cases}$$

$k \leftarrow k+1$;

Aller à Etape 2

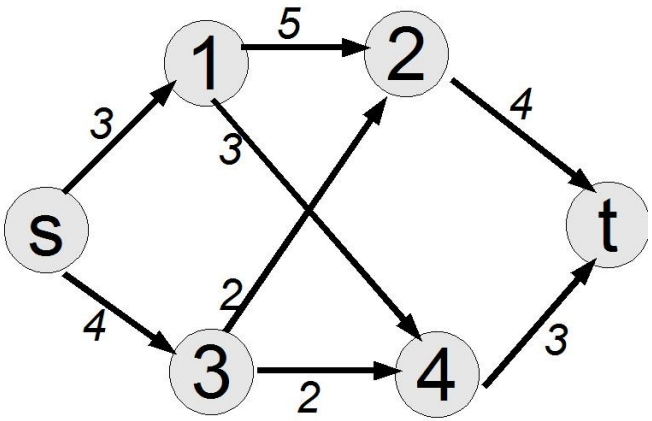
Sinon

Le flot ϕ^k est maximum ;

Fin_si ;

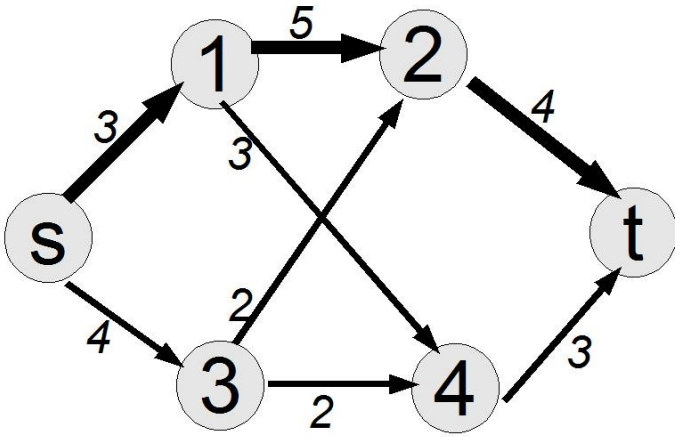
$\delta(\pi_k)$: minimum des capacités des arcs de π_k dans le graphe d'écart.

Algorithme de Ford & Fulkerson



Initialisation

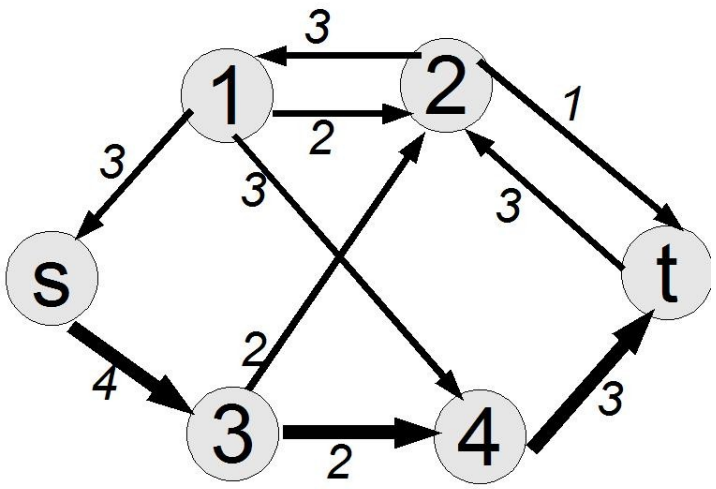
$\phi_0 = 0$



Chaîne améliorante : $\{s, 1, 2, t\}$

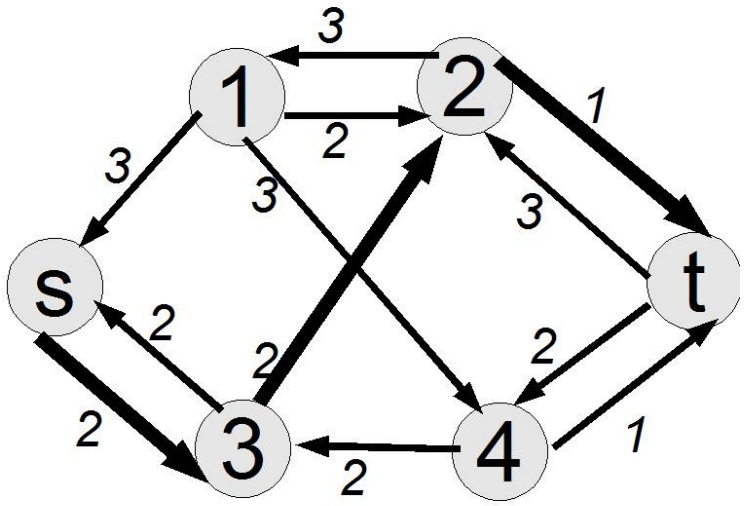
$\delta = 3, \phi_0 = 3$

Algorithme de Ford & Fulkerson



Chaîne améliorante : $\{s, 3, 4, t\}$

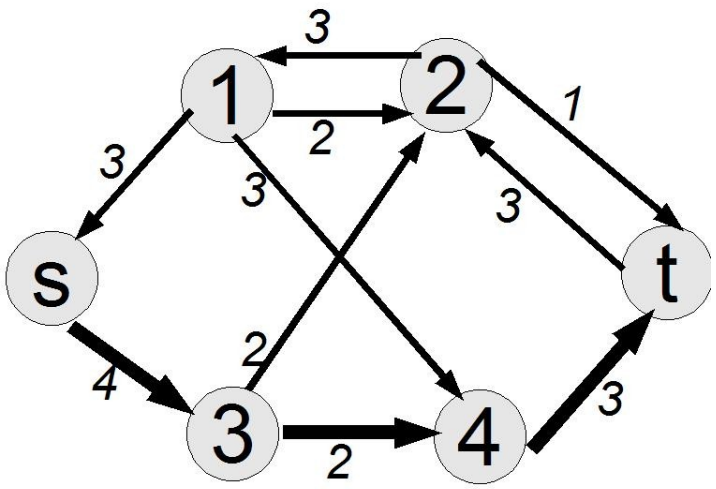
$\delta = 2, \phi_0 = 5$



Chaîne améliorante : $\{s, 3, 2, t\}$

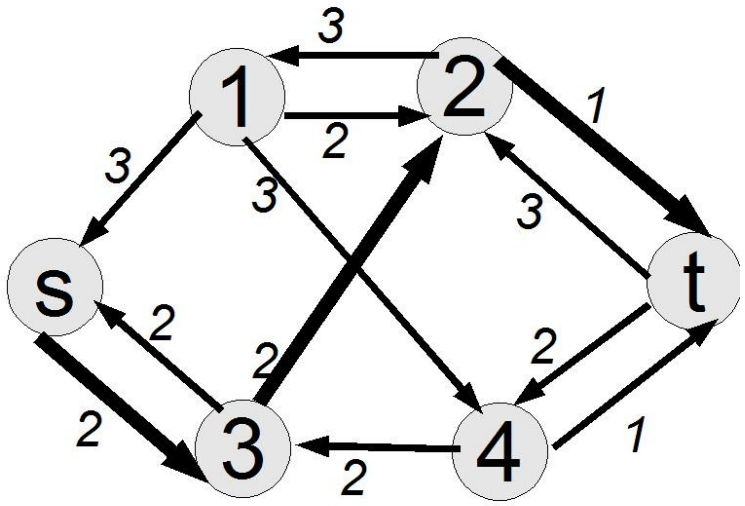
$\delta = 1, \phi_0 = 6$

Algorithme de Ford & Fulkerson



Chaîne améliorante : $\{s, 3, 4, t\}$

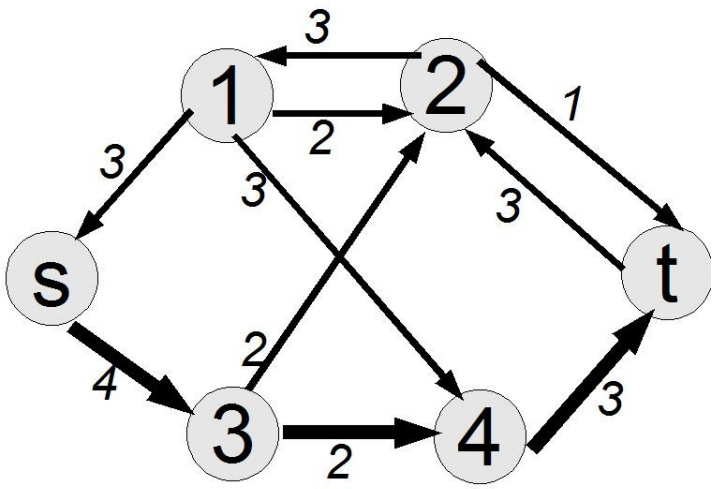
$\delta = 2, \phi_0 = 5$



Chaîne améliorante : $\{s, 3, 2, t\}$

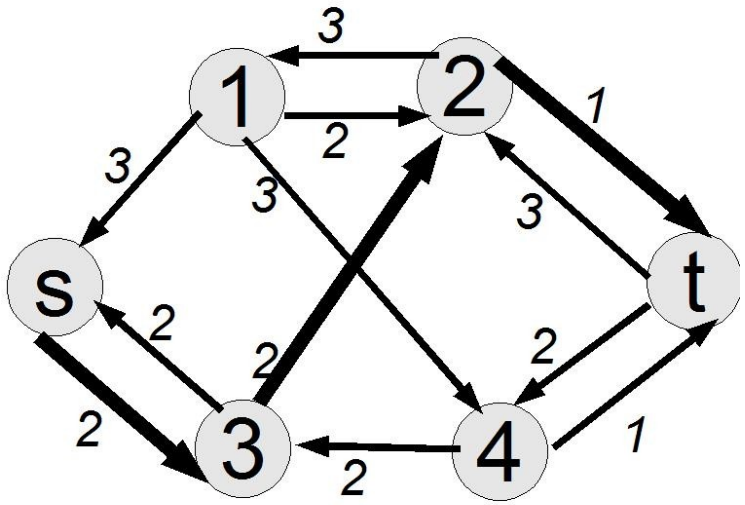
$\delta = 1, \phi_0 = 6$

Algorithme de Ford & Fulkerson



Chaîne améliorante : $\{s, 3, 4, t\}$

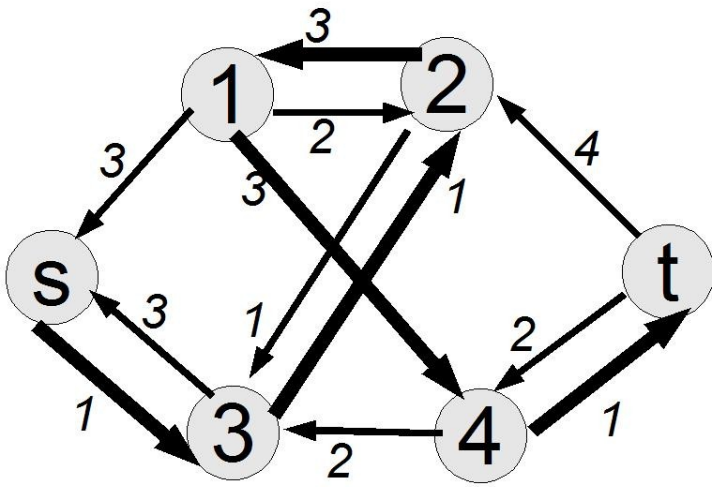
$\delta = 2, \phi_0 = 5$



Chaîne améliorante : $\{s, 3, 2, t\}$

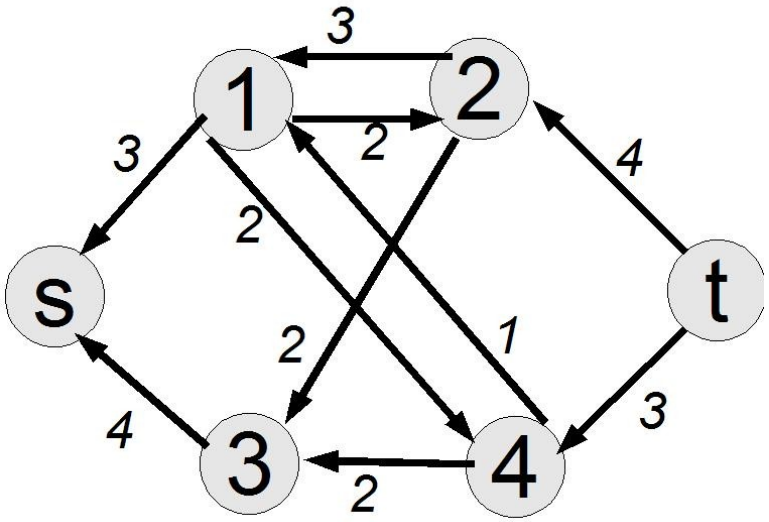
$\delta = 1, \phi_0 = 6$

Algorithme de Ford & Fulkerson



Chaîne améliorante : $\{s, 3, 2, 1, t\}$

$\delta = 1, \phi_0 = 7$



Pas de chaîne améliorante

Algorithme de Ford & Fulkerson

- Propriétés
  - Le flot calculé vérifie les contraintes de capacités
  - Si les capacités sont des valeurs entières et finies alors l'algorithme converge en un nombre fini d'itérations

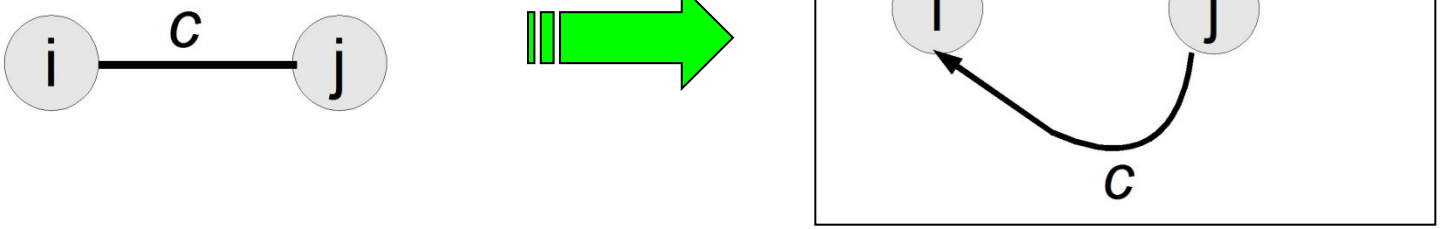
- Complexité
  - Variante de Edmonds & Karp : sélection d'un chemin dans le graphe d'écart avec un nombre d'arcs minimum
  - Complexité pour cette variante : $O(|E|^2.|X|)$

Obtention d'une coupe minimum

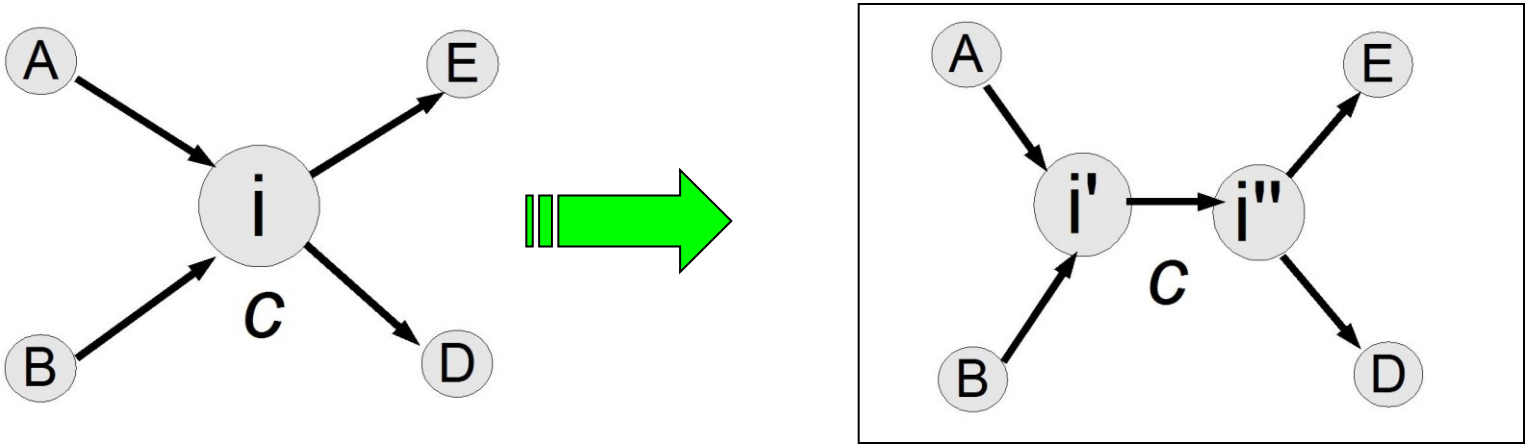
- Suite à l'application de l'algorithme de Ford et Fulkerson…
  - On note Φ le flot maximum trouvé par cet algorithme.
  - On note A l'ensemble des sommets tels que pour tout sommet x dans A il existe un chemin de s vers x dans le graphe d'écart associé à Φ.
  - Une coupe minimum est alors donnée par l 'ensemble $\omega^+(A)$

Quelques extensions…

- Graphes non orientés



- Contraintes de capacité sur les sommets



Algorithme de Busacker & Gowen

- L'algorithme de Busacker et Gowen permet de déterminer un **flot maximum de coût minimum** parmi les flots maximaux dans un réseau $G=(X, U, c, v)$, v étant une fonction de coût sur U.
- On suppose qu'il n'y a pas de circuit de coût strictement négatif dans le graphe de départ et que G est antisymétrique (i.e. pour chaque paire $\{i,j\}$ de sommets au plus un des deux arcs (i,j) ou (j,i) existe)
- Partant d'un flot nul, l'algorithme consiste à augmenter itérativement le flot dans le réseau en utilisant une chaîne améliorante de coût minimum.
- On utilise le graphe d'écart avec des coûts sur les arcs définis comme suit.

  Si $u = (i, j)$ est un arc de G, alors:
  - u^+ (arc d'orientation directe) a pour coût $v(u)$,
  - u^- (arc d'orientation inverse) a pour coût $-v(u)$.

Algorithme de Busacker & Gowen

- Rechercher une chaîne améliorante de coût minimum revient à rechercher un plus court chemin de *s* à *t* dans *G**. S'il n'existe aucun chemin de *s* à *t*, on arrête : le flot courant est de valeur maximum et de coût minimum. Sinon, on détermine dans *G** un plus court chemin de *s* à *t* et on augmente le flot dans *G* en utilisant la chaîne améliorante correspondant à ce chemin (et d'une valeur égale au minimum des capacités des arcs du chemin trouvé dans *G**).

Algorithme de Busacker & Gowen

- Sous les hypothèses mentionnées plus haut et si les capacités sont toutes des nombres entiers, alors en utilisant l'algorithme de Bellman-Ford pour la recherche des plus courts chemins, on obtient une implémentation de l'algorithme de Busacker et Gowen en $O(B \cdot N \cdot M)$ avec : B=valeur du flot maximum, N=nombre de sommets et M=nombre d'arcs de G.

- Il est possible d'implémenter l'algorithme avec une complexité en

$$O\big(N \cdot M + B \cdot (M + N \cdot \log(N))\big).$$

(L'idée consiste à se ramener à une recherche de plus court chemin avec des couts tous positifs ou nuls sur les arcs du graphe d'écart, sauf à la première itération.)

  - *Remarque. L'algorithme de Busacker et Gowen est aussi parfois appelé « algorithme des plus courts chemins successifs » (successive shortest path algorithm)*

Bibliographie

- Ahuja R., Magnanti T. and Orlin J., *Network flows: theory, algorithms, and applications*, Prentice Hall, 1993
- Diestel R., *Graph Theory* (Series-Graduate texts in mathematics), Springer, 2006
- Gondran M. and Minoux M., *Graphes et algorithmes* (4ème édition), EDF R&D,Tec & Doc, 2009
- Schrijver, A., *Combinatorial Optimization: Polyhedra and efficiency*, Springer, 2002
- West D.B., *Introduction to graph theory* – Second edition, Prentice Hall, 2001