Chapitre 1 - partie 2
Flots et coupes

Cours RO202

Zacharie ALES
(zacharie.ales@ensta.fr)

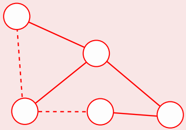
Adapté de cours de Marie-Christine Costa, Alain Faye et Sourour Elloumi

Créé le 21/01/2018
Modifié le 21/11/2022 (v6)

Programme

Optimisation dans les graphes

Chapitre 1

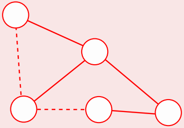


1.1 - Arbre couvrant

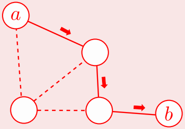
Programme

Optimisation dans les graphes

Chapitre 1



1.1 - Arbre couvrant

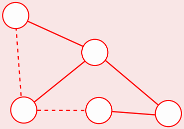


1.2 - Chemin

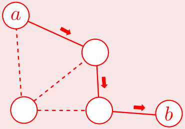
Programme

Optimisation dans les graphes

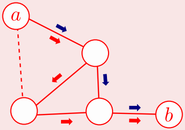
Chapitre 1



1.1 - Arbre couvrant



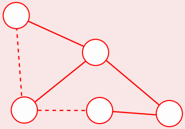
1.2 - Chemin



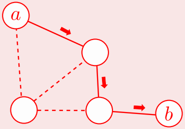
1.3 - Flot

Programme

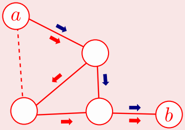
Optimisation dans les graphes
Chapitre 1



1.1 - Arbre couvrant

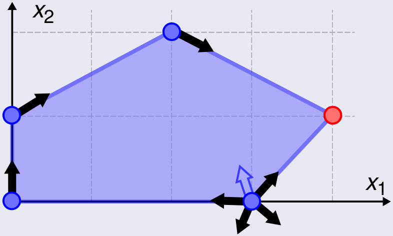


1.2 - Chemin



1.3 - Flot

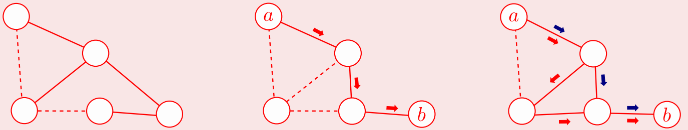
Programmation linéaire (PL)
Chapitre 2



Programme

Optimisation dans les graphes

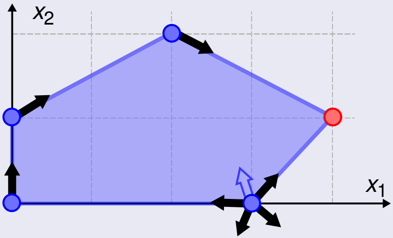
Chapitre 1



1.1 - Arbre couvrant 1.2 - Chemin 1.3 - Flot

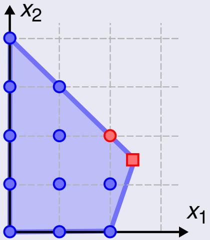
Programmation linéaire (PL)

Chapitre 2



PL en nombres entiers

Chapitre 3



Sommaire

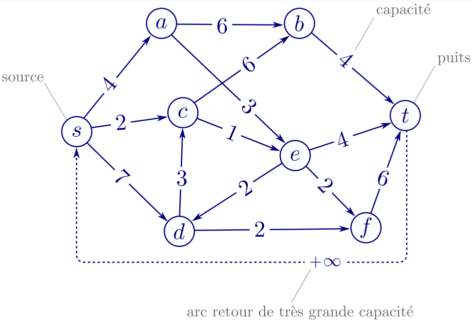
Réseau de transport

Capacité ≥ 0 des arcs

Graphe $G = (V, A, C)$ orienté tel qu'il existe :

- $s \in V$ une **source** $(\Gamma^-(s) = \emptyset)$
- $t \in V$ un **puits** $(\Gamma^+(t) = \emptyset)$

On ajoute $(ts) \in A$ un arc **fictif** de retour



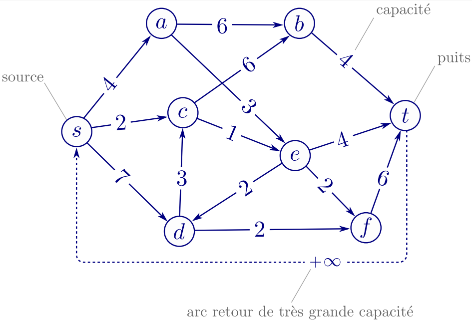
Réseau de transport

Capacité ≥ 0 des arcs

Graphe $G = (V, A, C)$ orienté tel qu'il existe :

- $s \in V$ une **source** $(\Gamma^-(s) = \emptyset)$
- $t \in V$ un **puits** $(\Gamma^+(t) = \emptyset)$

On ajoute $(ts) \in A$ un arc **fictif** de retour



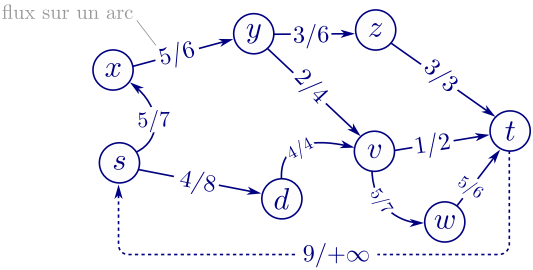
Problème de flot maximal

Comment transférer une quantité maximale de « matière » de s à t sans dépasser la capacité de chaque arc ?

Hypothèses

A chaque noeud :

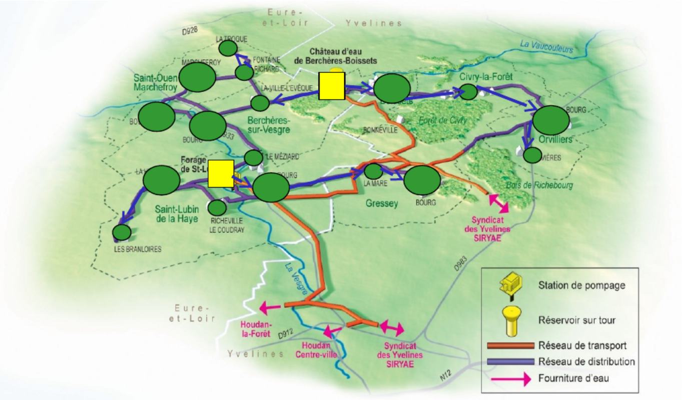
- ....
- ....



Applications

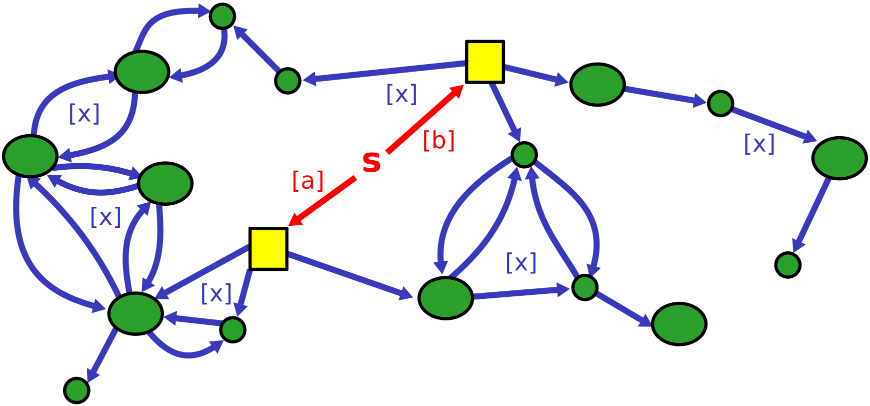
- Réseaux routiers
- Distribution d'eau
- Réseau internet
- ...

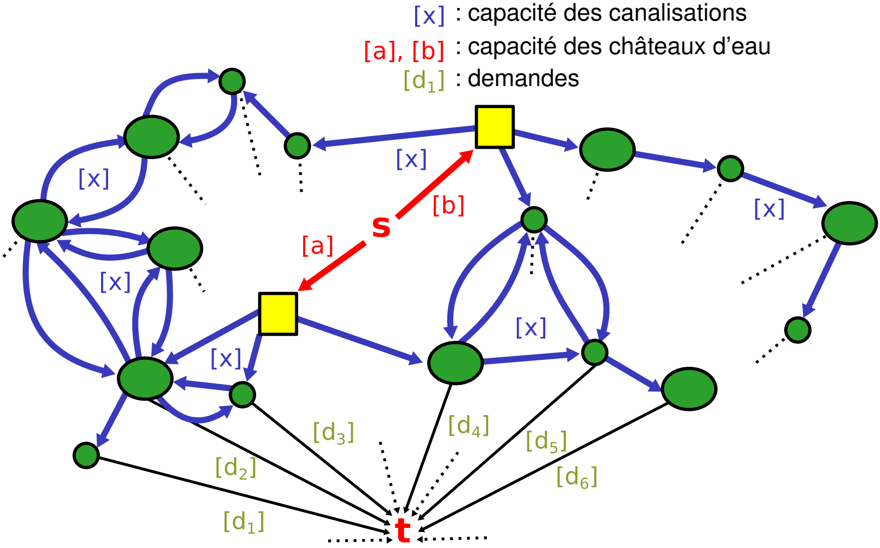
Un réseau de distribution de l'eau



[x] : capacité des canalisations

[a], [b] : capacité des châteaux d'eau




[x] : capacité des canalisations
[a], [b] : capacité des châteaux d'eau
[d_1] : demandes
[x]
[x]
[x]
[x]
[x]
[x]
[a]
[b]
s
t
[d_1]
[d_2]
[d_3]
[d_4]
[d_5]
[d_6]

Définition - Flux φ_{ij}

Flux sur l'arc (ij)
Quantité de matière circulant sur l'arc

Définition - Flot sur G

Vecteur $\varphi = \{\varphi_{ij}\}_{(ij)\in A\cup(t,s)}$
Flux sur chaque arc de G

Définition - Flot réalisable φ

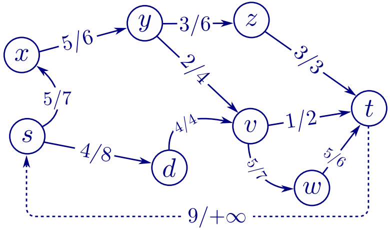
Flot vérifiant en chaque arc (ij) :

- la contrainte de capacité

$$0 \leq \varphi_{ij} \leq c_{ij}$$

- la loi de conservation
  $\approx$ loi de Kirchhoff en électricité

$$\sum_{i\in\Gamma^-(j)} \varphi_{ij} = \sum_{i\in\Gamma^+(j)} \varphi_{ji} \quad \forall j \in V$$



Définition - **Flux** φ_{ij}

Flux sur l'arc (ij)
Quantité de matière circulant sur l'arc

Définition - **Flot sur** G

Vecteur $\varphi = \{\varphi_{ij}\}_{(ij)\in A\cup(t,s)}$
Flux sur chaque arc de G

Définition - **Flot réalisable** φ

Flot vérifiant en chaque arc (ij) :

- la contrainte de capacité

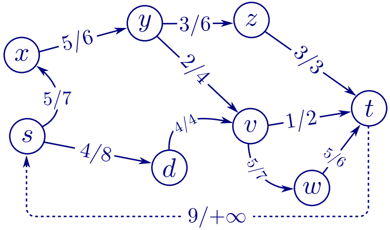
$$0 \leq \varphi_{ij} \leq c_{ij}$$

- la loi de conservation
  $\approx$ loi de Kirchhoff en électricité

$$\sum_{i\in\Gamma^-(j)} \varphi_{ij} = \sum_{i\in\Gamma^+(j)} \varphi_{ji} \quad \forall j \in V$$

Définition - Arc **saturé**

(ij) est dit **saturé** si $\varphi_{ij} = c_{ij}$
Exemple (dv)



Flot sur un réseau de transport

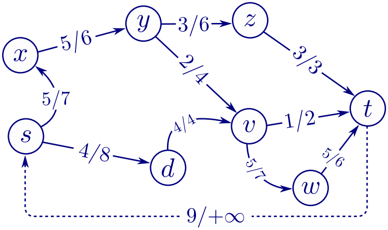
Remarque

La loi de conservation est vérifiée en *s* et *t* grâce à l'arc de retour

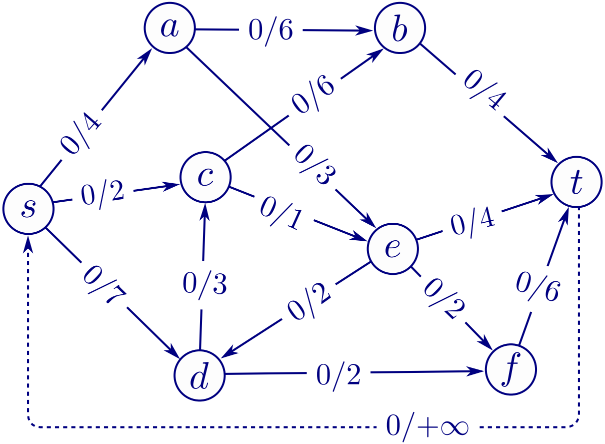
Défintion - **Flot complet** φ

φ est dit **complet** si et seulement si

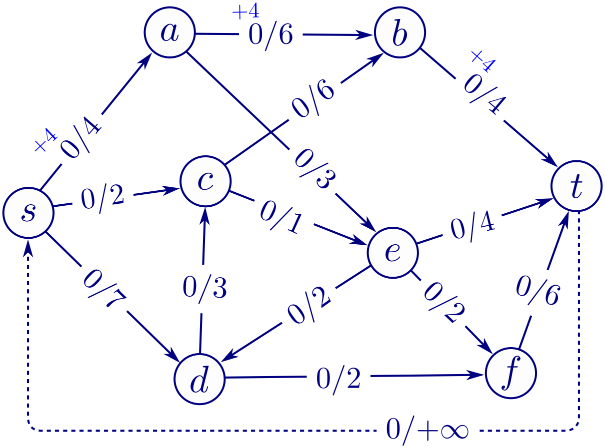
- ....



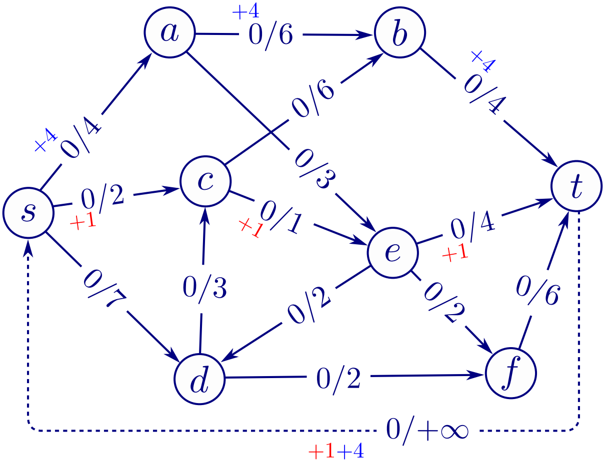
Exemple de flot complet



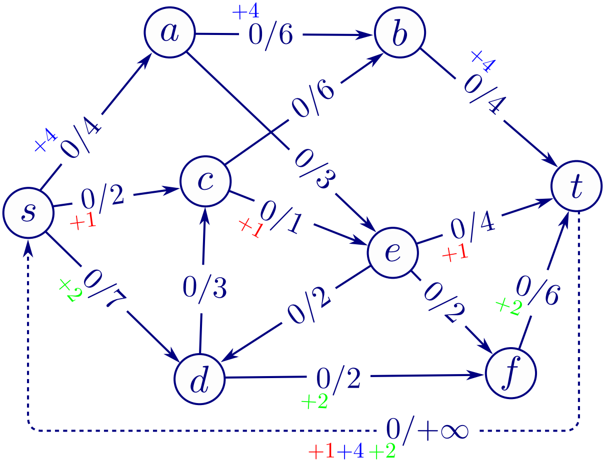
Exemple de flot complet



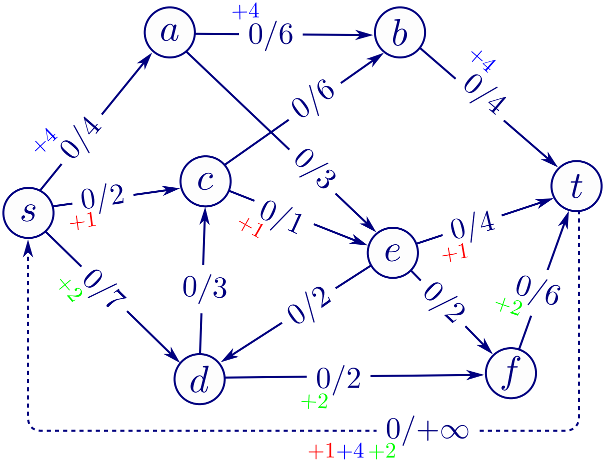
Exemple de flot complet



Exemple de flot complet



Exemple de flot complet



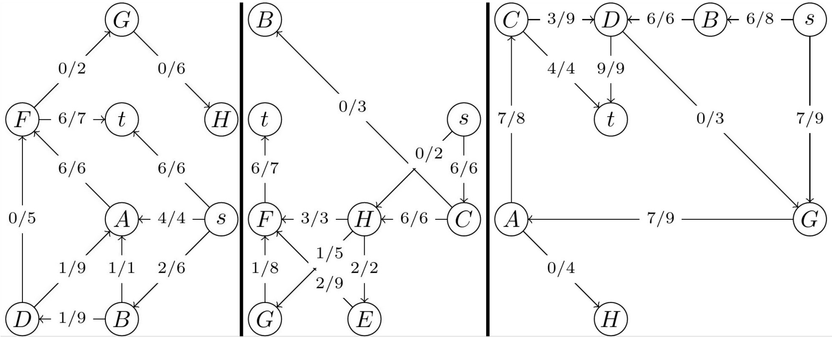
Flot complet

- v(φ)=7

Quiz !

Question 1

Quel flot est complet ?

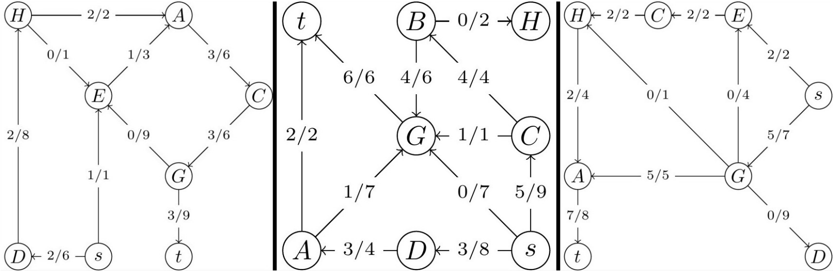


Graphe 1 Graphe 2 Graphe 3

Quiz !

Question 2

Quel flot est complet ?



Graphe 1 Graphe 2 Graphe 3

Définition - **Valeur d'un flot** $v(\varphi)$

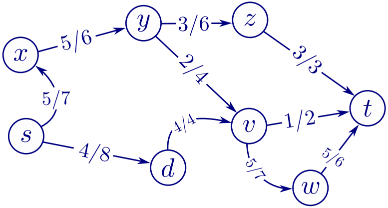
Flux des arcs entrants en t

= Flux des arcs sortants de s

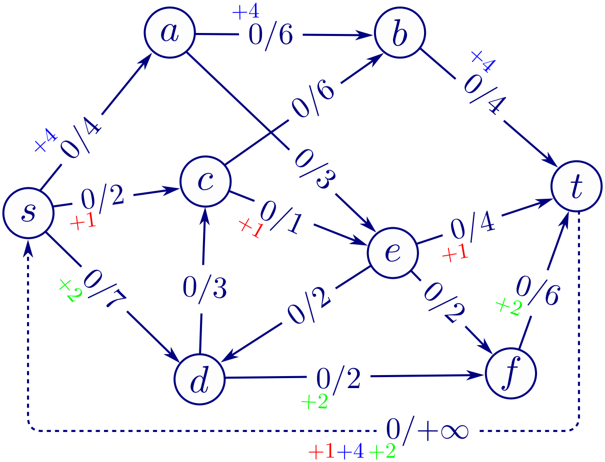
$$v(\varphi) = \sum_{i \in \Gamma^-(t)} \varphi_{it}$$

Définition - **Flot maximal**

Flot de valeur maximale



Retour à notre réseau de transport



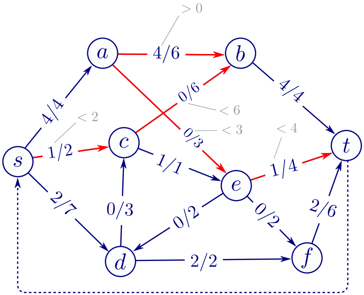
Comment améliorer le flot ?

Chaîne améliorante

Définition - Chaîne améliorante μ pour un flot φ

Chaîne de *s* à *t* vérifiant que :

- .... pour tout arc (ij) de μ dans le "bon sens"
  de *s* vers *t*
- .... pour tout arc (ij) de μ dans le "mauvais sens"
  de *t* vers *s*

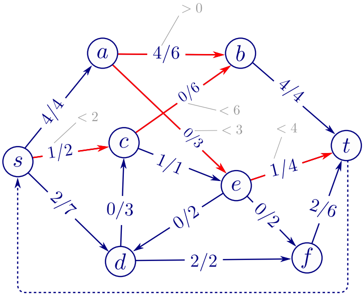


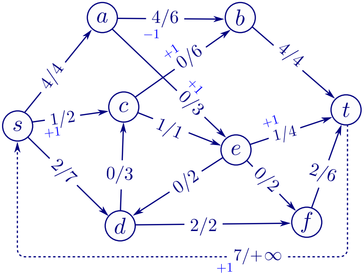
Chaîne améliorante

Définition - Chaîne améliorante μ pour un flot φ

Chaîne de *s* à *t* vérifiant que :

- .... pour tout arc (ij) de μ dans le "bon sens"
  de *s* vers *t*
- .... pour tout arc (ij) de μ dans le "mauvais sens"
  de *t* vers *s*



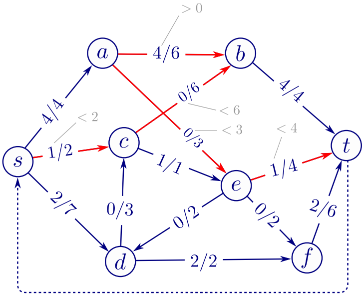


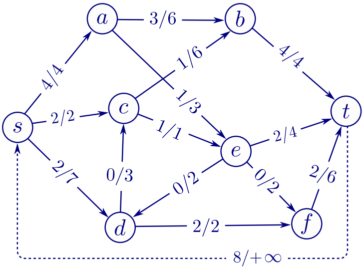
Chaîne améliorante

Définition - Chaîne améliorante μ pour un flot φ

Chaîne de *s* à *t* vérifiant que :

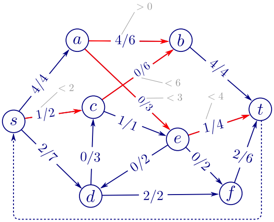
- .... pour tout arc (ij) de μ dans le "bon sens"
  de *s* vers *t*
- .... pour tout arc (ij) de μ dans le "mauvais sens"
  de *t* vers *s*





Chaîne améliorante

Soit μ une chaîne améliorante



Notations

- μ^+ = arcs de μ dans le bon sens
- μ^- = arcs de μ dans le mauvais sens

Augmentation de la valeur du flot de α

$\alpha =$ ....

- dans μ^+ : on augmente les flux de α
- dans μ^- : on diminue les flux de α

Exemple

φ_{ij} c_{ij} Peut être diminué de 4

$$s \underset{\in \mu^+}{\xrightarrow{1/2}} c \underset{\in \mu^+}{\xrightarrow{0/6}} b \underset{\in \mu^-}{\xleftarrow{4/6}} a \underset{\in \mu^+}{\xrightarrow{0/3}} e \underset{\in \mu^+}{\xrightarrow{1/4}} t$$

Peut être augmenté de 6-0

$\alpha = \min\left[\min(1, 6, 3, 3), \min(4)\right]$

Sommaire

L'algorithme de Ford-Fulkerson

Problème

Trouver un flot maximal

Principe

- Trouver un flot initial
  De préférence complet
- Tant qu'une chaîne améliorante est trouvée
  - Améliorer le flot le long de cette chaîne

$\Rightarrow$ un flot optimal

Preuve plus loin

L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

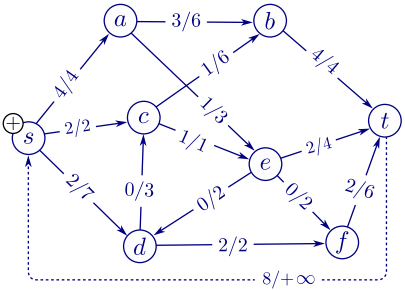
Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

- Retirer toutes les marques
- Marquer '+' le sommet s
- **répéter**
  - Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :
    - i est marqué
    - j est non marqué
    - (ij) non saturé
  - Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :
    - i est non marqué
    - j est marqué
    - (ij) a un flux non nul
- **tant que** un nouveau sommet a été marqué **et** t n'est pas marqué
- **si** t est marqué **alors**
  - Améliorer le flux via une chaîne améliorante

tant que t est marqué



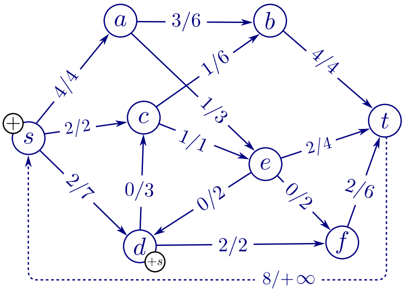
L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

Données : $G = (V, A, C)$
Établir un flot admissible (complet de préférence)
répéter
  Retirer toutes les marques
  Marquer '+' le sommet s
  répéter
    Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :
    - i est marqué
    - j est non marqué
    - (ij) non saturé

    Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :
    - i est non marqué
    - j est marqué
    - (ij) a un flux non nul

  tant que un nouveau sommet a été marqué **et** t n'est pas marqué
  si t est marqué **alors**
    Améliorer le flux via une chaîne améliorante
tant que t est marqué



L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

  Retirer toutes les marques

  Marquer '+' le sommet *s*

  répéter

    Marquer '$+i$' le sommet terminal *j* de tout arc (ij) tel que :

    - *i* est marqué
    - *j* est non marqué
    - (ij) non saturé

    Marquer '$-j$' le sommet initial *i* de tout arc (ij) tel que :

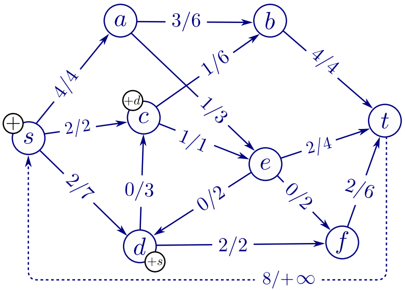
    - *i* est non marqué
    - *j* est marqué
    - (ij) a un flux non nul

  tant que un nouveau sommet a été marqué **et** *t* n'est pas marqué

  si *t* est marqué **alors**

    Améliorer le flux via une chaîne améliorante

tant que *t* est marqué



L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

Retirer toutes les marques

Marquer '+' le sommet s

répéter

Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :

- i est marqué
- j est non marqué
- (ij) non saturé

Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :

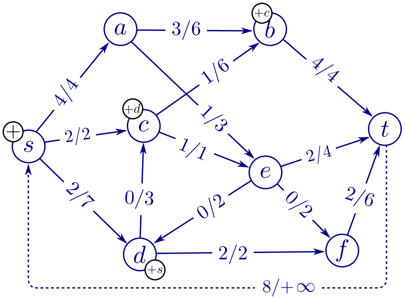
- i est non marqué
- j est marqué
- (ij) a un flux non nul

tant que un nouveau sommet a été marqué **et** t n'est pas marqué

si t est marqué **alors**

Améliorer le flux via une chaîne améliorante

tant que t est marqué



L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

Retirer toutes les marques

Marquer '+' le sommet s

répéter

Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :

- i est marqué
- j est non marqué
- (ij) non saturé

Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :

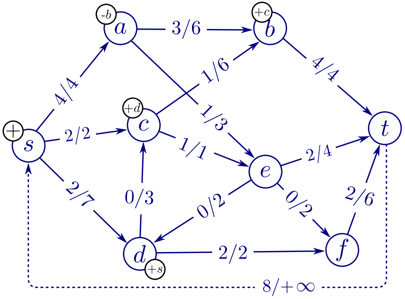
- i est non marqué
- j est marqué
- (ij) a un flux non nul

tant que un nouveau sommet a été marqué **et** t n'est pas marqué

si t est marqué **alors**

Améliorer le flux via une chaîne améliorante

tant que t est marqué



L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

Retirer toutes les marques

Marquer '+' le sommet s

répéter

Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :

- i est marqué
- j est non marqué
- (ij) non saturé

Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :

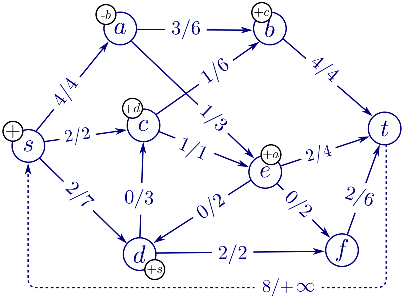
- i est non marqué
- j est marqué
- (ij) a un flux non nul

tant que un nouveau sommet a été marqué **et** t n'est pas marqué

si t est marqué **alors**

Améliorer le flux via une chaîne améliorante

tant que t est marqué



L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

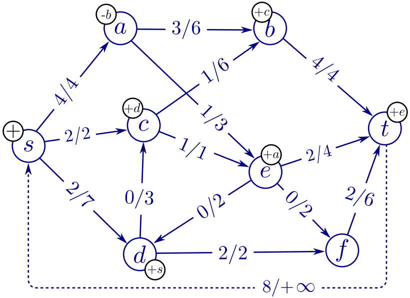
Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

- Retirer toutes les marques
- Marquer '+' le sommet s
- **répéter**
  - Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :
    - i est marqué
    - j est non marqué
    - (ij) non saturé
  - Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :
    - i est non marqué
    - j est marqué
    - (ij) a un flux non nul
- **tant que** un nouveau sommet a été marqué **et** t n'est pas marqué
- **si** t est marqué **alors**
  - Améliorer le flux via une chaîne améliorante

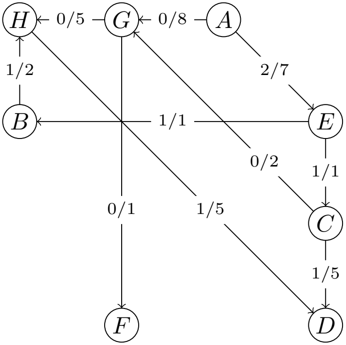
tant que t est marqué

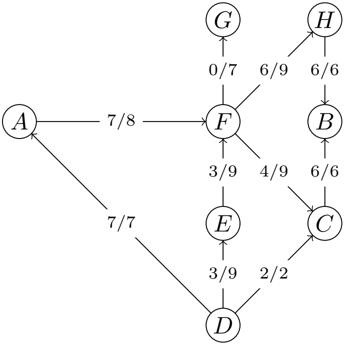


Quiz !

Question 3 et 4

Appliquer le marquage de l'algorithme de Ford-Fulkerson en vue de trouver le flot maximal entre les sommets A et D.
A chaque étape si vous avez la possibilité de marquer plusieurs sommets marquer celui qui est le premier dans l'ordre alphabétique.





L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

  Retirer toutes les marques

  Marquer '+' le sommet s

  répéter

    Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :

    - i est marqué
    - j est non marqué
    - (ij) non saturé

    Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :

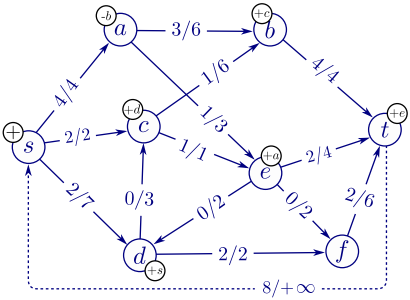
    - i est non marqué
    - j est marqué
    - (ij) a un flux non nul

  tant que un nouveau sommet a été marqué **et** t n'est pas marqué

  si t est marqué **alors**

    Améliorer le flux via une chaîne améliorante

tant que t est marqué



L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

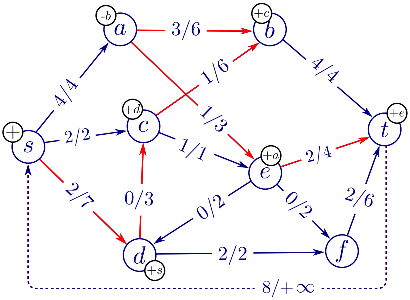
Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

- Retirer toutes les marques
- Marquer '+' le sommet s
- **répéter**
  - Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :
    - i est marqué
    - j est non marqué
    - (ij) non saturé
  - Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :
    - i est non marqué
    - j est marqué
    - (ij) a un flux non nul
- **tant que** un nouveau sommet a été marqué **et** t n'est pas marqué
- **si** t est marqué **alors**
  - Améliorer le flux via une chaîne améliorante

tant que t est marqué



Chaîne améliorante μ

$s \xrightarrow{5} d \xrightarrow{3} c \xrightarrow{5} b \xleftarrow{3} a \xrightarrow{2} e \xrightarrow{2} t$

- Amélioration de ...

L'algorithme de Ford-Fulkerson : une procédure de marquage

Algorithme

Données : $G = (V, A, C)$

Établir un flot admissible (complet de préférence)

répéter

  Retirer toutes les marques

  Marquer '+' le sommet s

  répéter

    Marquer '$+i$' le sommet terminal j de tout arc (ij) tel que :

    - i est marqué
    - j est non marqué
    - (ij) non saturé

    Marquer '$-j$' le sommet initial i de tout arc (ij) tel que :

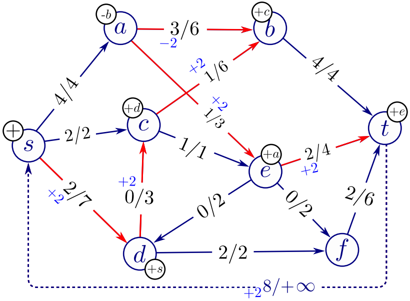
    - i est non marqué
    - j est marqué
    - (ij) a un flux non nul

  tant que un nouveau sommet a été marqué **et** t n'est pas marqué

  si t est marqué **alors**

    Améliorer le flux via une chaîne améliorante

tant que t est marqué



Chaîne améliorante μ

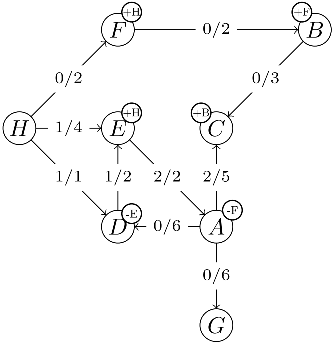
$s \xrightarrow{5} d \xrightarrow{3} c \xrightarrow{5} b \xleftarrow{3} a \xrightarrow{2} e \xrightarrow{2} t$

- Amélioration de ...

Quiz !

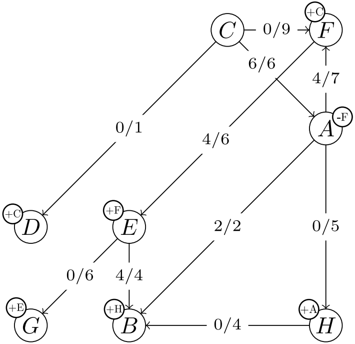
Question 5

De combien d'unités ce marquage permet-il d'augmenter le flot entre H et C ?



Question 6

De combien d'unités ce marquage permet-il d'augmenter le flot entre C et B ?

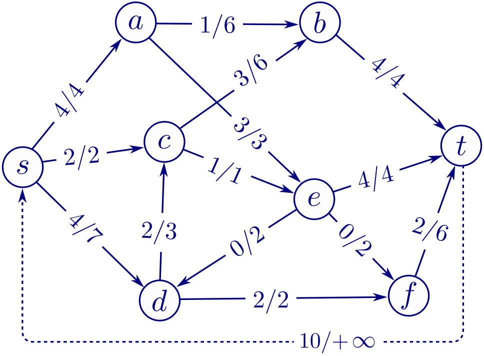


Un réseau de transport

Le flot obtenu est-il optimal ?

1. Oui
2. Non

. . . .



Preuve de l'algorithme

Problème de coupe minimale

Comment séparer s de t en supprimant un ensemble d'arcs de valeur totale minimale ?

"Séparer" signifie qu'il n'existe plus de chemin de s à t après suppression des arcs

Preuve de l'algorithme

Problème de coupe minimale

Comment séparer s de t en supprimant un ensemble d'arcs de valeur totale minimale ?

"Séparer" signifie qu'il n'existe plus de chemin de s à t après suppression des arcs

Définition - **Coupe (S,T)**

Partition de V en deux sous-ensembles S et T telle que

- $s \in S$
- $t \in T$

Preuve de l'algorithme

Problème de coupe minimale

Comment séparer s de t en supprimant un ensemble d'arcs de valeur totale minimale ?

"Séparer" signifie qu'il n'existe plus de chemin de s à t après suppression des arcs

Définition - **Coupe (S,T)**

Partition de V en deux sous-ensembles S et T telle que

- $s \in S$
- $t \in T$

Notations

- $\omega^-(T) =$ arcs entrant dans T
  $\{(i,j) \in A \mid i \in S, j \in T\}$
- $\omega^+(T) =$ arcs sortant de T
  $\{(i,j) \in A \mid i \in T, j \in S\}$

Remarque

Par définition $(ts) \notin \omega^+(T)$
Car $(ts) \notin A$

Preuve de l'algorithme

Problème de coupe minimale

Comment séparer s de t en supprimant un ensemble d'arcs de valeur totale minimale ?

"Séparer" signifie qu'il n'existe plus de chemin de s à t après suppression des arcs

Définition - **Coupe (S,T)**

Partition de V en deux sous-ensembles S et T telle que

- $s \in S$
- $t \in T$

Notations

- $\omega^-(T) =$ arcs entrant dans T
  $\{(i,j) \in A \mid i \in S, j \in T\}$
- $\omega^+(T) =$ arcs sortant de T
  $\{(i,j) \in A \mid i \in T, j \in S\}$

Remarque

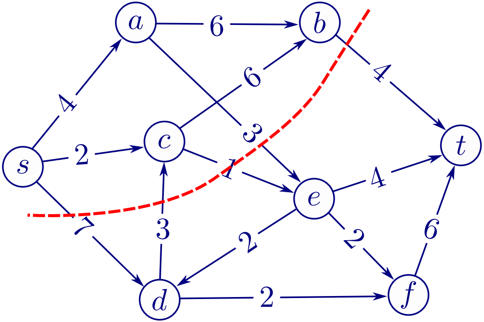
Par définition $(ts) \notin \omega^+(T)$
Car $(ts) \notin A$

Définition - **Capacité d'une coupe (S,T)**

$$c(S,T) =$$

....

Exemple de coupe



Coupe de valeur 15

- $S = \{s, a, b, c\}$
- $T = \{t, d, e, f\}$
- $C = \omega^-(T) = \{(b, t), (a, e), (c, e), (s, d)\}$

Relation flots / coupes

Propriété

Soit $G = (V, A)$ un réseau de transport

- $\forall \varphi$ flot admissible sur G
- $\forall (S, T)$ coupe de G

On a

....

Preuve

Soit (S, T) une coupe de G

- .... (loi de conservation)
- On sait que $\varphi_{ts} = v(\varphi)$
- .... (flux $\leq$ capacité)

Fin de l'algorithme de Ford-Fulkerson

Rappel

t non marqué $\Rightarrow$ flot maximal

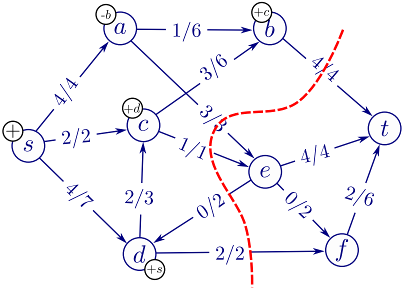
Propriété

La coupe minimale sépare les sommets marqués des non marqués

Exemple

- $S^* = \{s, a, b, c, d\}$
- $T^* = \{t, e, f\}$
- $C^* = \{(b, t), (a, e), (c, e), (d, f)\}$

$$v(\varphi^*) = 10 = v(C^*)$$



Théorème de Ford-Fulkerson

Théorème - Ford-Fulkerson, 1962

La valeur d'un flot maximal est égale ....

Propriété - CNS d'optimalité

Un flot φ de s à t est maximal si et seulement si ....
....

Preuve du théorème et de l'algorithme de Ford-Fulkerson

Notations

- φ^* : flot obtenu par l'algorithme
- S^* : ensemble des sommets marqués à la fin de l'algorithme
- T^* : ensemble des sommets non marqués à la fin de l'algorithme

Rappels

- $v(\varphi^*) = \varphi^*(t, s)$
- $(t, s) \notin \omega^+(T)$

Preuve

- Toute coupe (S,T) et tout flot φ vérifient : $v(\varphi) \leq c(S, T)$
- .... (loi de conservation des flux)

  .... (principe de marquage)

  ....
- ....

Convergence de l'algorithme

Théorème des valeurs entières

Dans un réseau de transport à capacités entières, il existe un flot maximal dont tous les flux sont entiers

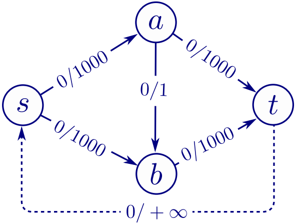
Convergence de l'algorithme

Si les capacités sont entières, l'algorithme de Ford-Fulkerson converge en un nombre fini d'itérations car :

- La valeur du flot max est bornée
  Par la capacité de n'importe quelle coupe
- À chaque itération, on augmente le flot d'une valeur entière

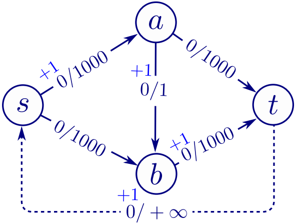
Nombre d'itérations

Un mauvais choix de chaînes améliorantes peut entraîner un nombre d'itérations égal à la valeur du flot maximal



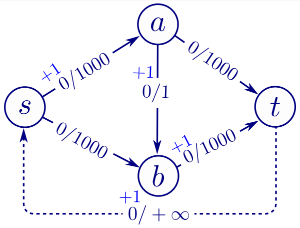
Nombre d'itérations

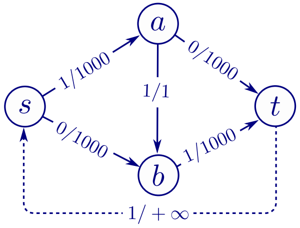
Un mauvais choix de chaînes améliorantes peut entraîner un nombre d'itérations égal à la valeur du flot maximal



Nombre d'itérations

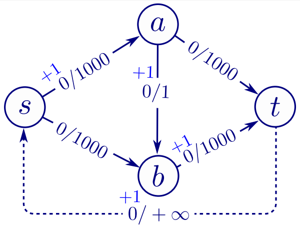
Un mauvais choix de chaînes améliorantes peut entraîner un nombre d'itérations égal à la valeur du flot maximal

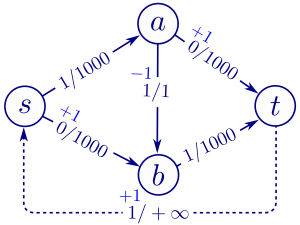




Nombre d'itérations

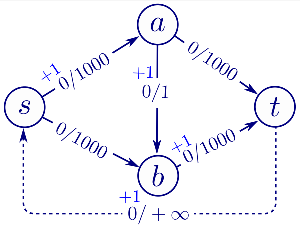
Un mauvais choix de chaînes améliorantes peut entraîner un nombre d'itérations égal à la valeur du flot maximal

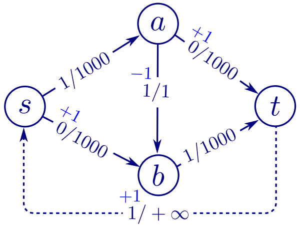


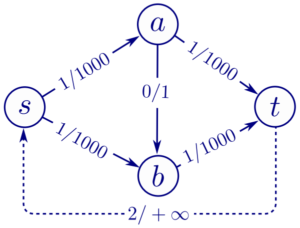


Nombre d'itérations

Un mauvais choix de chaînes améliorantes peut entraîner un nombre d'itérations égal à la valeur du flot maximal

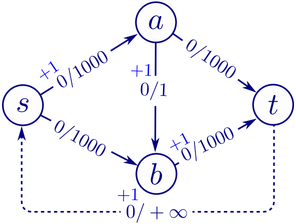


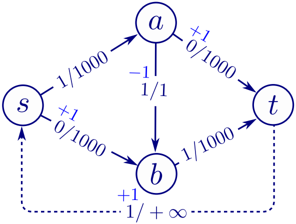


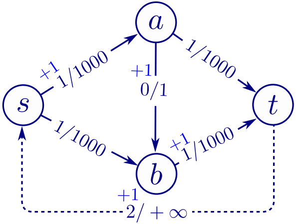


Nombre d’itérations

Un mauvais choix de chaînes améliorantes peut entraîner un nombre d’itérations égal à la valeur du flot maximal

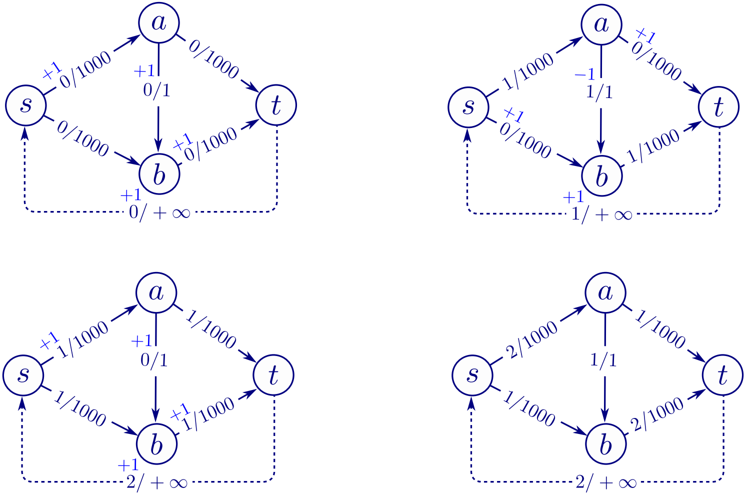






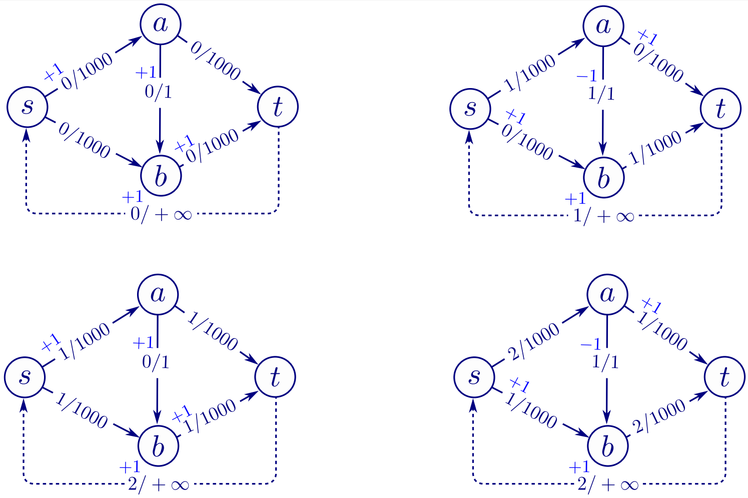
Nombre d'itérations

Un mauvais choix de chaînes améliorantes peut entraîner un nombre d'itérations égal à la valeur du flot maximal



Nombre d'itérations

Un mauvais choix de chaînes améliorantes peut entraîner un nombre d'itérations égal à la valeur du flot maximal



Complexité de l'algorithme (nombre d'«opérations »)

Théorème

Si chaque augmentation du flot est faite suivant une chaîne améliorante de longueur minimale, alors

- le flot maximal est obtenu après moins de $\frac{mn}{2}$ itérations.

Complexité

$$\mathcal{O}(\frac{m^2 n}{2})$$

D'après le théorème et le fait qu'il y ait au plus m marquages à chaque itération

Remarque

Il existe des algorithmes plus efficaces

Un modèle mathématique « Programmation linéaire »

Programme linéaire

$$\begin{cases} \max & cx \\ & Ax \leq b \\ & x \geq 0 \end{cases}$$

- A, b, c : données
- x : variables

Propriété

L'optimum d'un programme linéaire en variables continues peut être obtenu en temps polynomial
Voir cours suivant

Programme linéaire pour le flot maximal

$$\begin{cases} \max & \varphi_{ts} & & \\ & \varphi_{ij} \leq c_{ij} & \forall (ij) \in A & \text{(capacités)} \\ & \sum\limits_{i \in \Gamma^-(j)} \varphi_{ij} = \sum\limits_{i \in \Gamma^+(j)} \varphi_{ji} & \forall j \in V & \text{(conservation des flux)} \\ & \varphi_{ij} \geq 0 & \forall (ij) \in A & \end{cases}$$

Résumé

Notions abordées dans ce chapitre

- Définitions
  Réseau de transport
  Flot
  Flot maximal
  Coupe
  Capacité d'une coupe
  ...
- Algorithme de Ford-Fulkerson
  Calcul d'un flot maximal par détection de chaînes améliorantes via une procédure de marquage
- En fin d'algorithme, s et t sont séparés par une coupe de capacité égale à la valeur du flot
  Ces deux problèmes sont **duaux** (voir chapitre suivant)

Pistes d'approfondissement

- Flot maximal de coût minimal
  Problème "facile"
- Multiflots et multicoupes
  Problèmes "difficiles"
- Flot avec multiplicateurs
  Flux en entrée d'un arc multiplié à sa sortie
- Matrice totalement unimodulaire et programmation linéaire en nombres entiers
- Programmation linéaire et dualité

Q1 : 3
Q2 : 2
Q3 : B-H D+H E+A F+G G+A H+G

Q4 : A-F C+F E+D F+E G+F H+F
Q5 : 2
Q6 : 4