Algorithmes de flots

Quentin Fortier

October 5, 2021

Définitions

On considère un graphe orienté $\overrightarrow{G} = (V, \overrightarrow{E})$ avec une **capacité** $c : \overrightarrow{E} \longrightarrow \mathbb{R}^+$.

Définitions

On considère un graphe orienté $\vec{G} = (V, \vec{E})$ avec une **capacité** $c : \vec{E} \longrightarrow \mathbb{R}^+$.

Notations

Si $A \subseteq V$, on note:

- $A^+ = \{(u, v) \in \vec{E} \mid u \in A, v \notin A\}$ (« arcs sortants de A »)

Définitions

On considère un graphe orienté $\vec{G} = (V, \vec{E})$ avec une **capacité** $c : \vec{E} \longrightarrow \mathbb{R}^+$.

Notations

Si $A \subseteq V$, on note:

- $A^+ = \{(u, v) \in \vec{E} \mid u \in A, v \notin A\}$ (« arcs sortants de A »)
- $A^- = \{(u, v) \in \vec{E} \mid u \notin A, v \in A\}$ (« arcs rentrants dans A »)

Définitions

On considère un graphe orienté $\vec{G} = (V, \vec{E})$ avec une **capacité** $c : \vec{E} \longrightarrow \mathbb{R}^+$.

Notations

Si $A \subseteq V$, on note:

- $A^+ = \{(u, v) \in \vec{E} \mid u \in A, v \notin A\}$ (« arcs sortants de A »)
- $A^- = \{(u, v) \in \vec{E} \mid u \notin A, v \in A\}$ (« arcs rentrants dans A »)
- Si $v \in V$, $v^+ = \{v\}^+$ et $v^- = \{v\}^-$

Définitions

On considère un graphe orienté $\overrightarrow{G} = (V, \overrightarrow{E})$ avec une **capacité** $c : \overrightarrow{E} \longrightarrow \mathbb{R}^+$.

Notations

Si $A \subseteq V$, on note:

- $A^+ = \{(u, v) \in \overrightarrow{E} \mid u \in A, v \notin A\}$ (« arcs sortants de A »)
- $A^- = \{(u, v) \in \overrightarrow{E} \mid u \notin A, v \in A\}$ (« arcs rentrants dans A »)
- Si $v \in V$, $v^+ = \{v\}^+$ et $v^- = \{v\}^-$

Définition

Si $f : \overrightarrow{E} \longrightarrow \mathbb{R}^+$ et $B \subseteq \overrightarrow{E}$:

$$f(B) = \sum_{\vec{e} \in \overrightarrow{E}} f(\vec{e})$$

Définitions

Flot

Soit $s, t \in V$.
Un **s-t flot** est une fonction $f : \vec{E} \longrightarrow \mathbb{R}^+$ telle que :

- $\forall \vec{e} \in \vec{E}$: $\boxed{0 \leq f(\vec{e}) \leq c(\vec{e})}$

Définitions

Flot

Soit $s, t \in V$.
Un **s-t flot** est une fonction $f : \vec{E} \longrightarrow \mathbb{R}^+$ telle que :

- $\forall \vec{e} \in \vec{E}$: $\boxed{0 \leq f(\vec{e}) \leq c(\vec{e})}$
- $\forall v \in V - \{s, t\}$: $\boxed{f(v^-) = f(v^+)}$

Définition

$f(s^+)$ est la **valeur** du flot f

Définitions

Flot

Soit $s, t \in V$.

Un **s-t flot** est une fonction $f : \vec{E} \longrightarrow \mathbb{R}^+$ telle que :

- $\forall \vec{e} \in \vec{E}$: $\boxed{0 \leq f(\vec{e}) \leq c(\vec{e})}$
- $\forall v \in V - \{s, t\}$: $\boxed{f(v^-) = f(v^+)}$

Définition

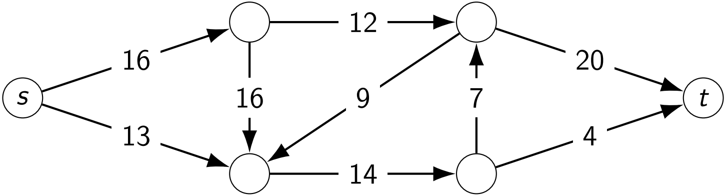
$f(s^+)$ est la **valeur** du flot f

Problème

Trouver un flot dont la valeur est maximum.

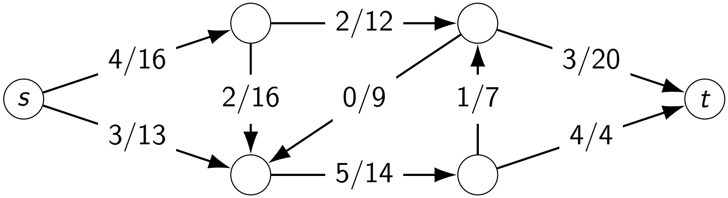
Exemple

Exemple de graphe $\vec{G}$ avec une capacité c sur les arcs :



Définitions

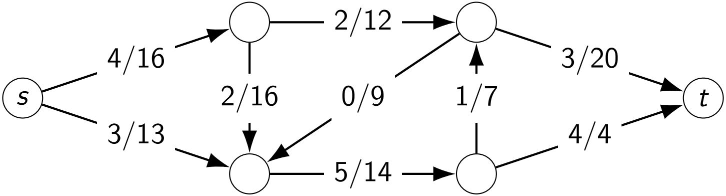
Exemple de flot :



Valeur du flot :

Définitions

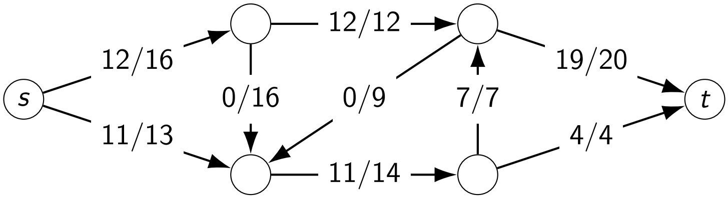
Exemple de flot :



Valeur du flot : $4 + 3 = 7$

Définitions

Exemple de flot de valeur maximum :



Valeur de ce flot : 23

Algorithme de Ford-Fulkerson

Idée : partir d'un flot f non optimal (souvent le flot nul partout) et l'améliorer petit à petit.

Algorithme de Ford-Fulkerson

Idée : partir d'un flot f non optimal (souvent le flot nul partout) et l'améliorer petit à petit.

Graphe résiduel

La **capacité résiduelle** d'un arc est le flot que l'on peut encore y ajouter (capacité initiale moins flot)

Algorithme de Ford-Fulkerson

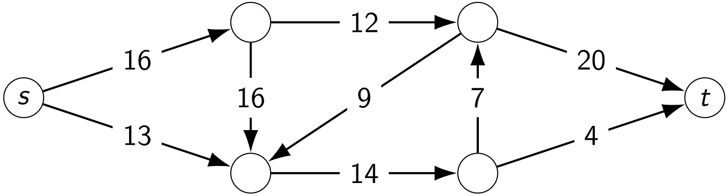
Idée : partir d'un flot f non optimal (souvent le flot nul partout) et l'améliorer petit à petit.

Graphe résiduel

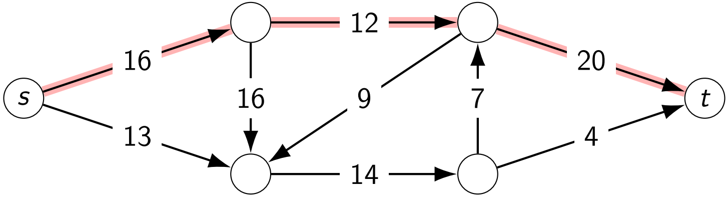
La **capacité résiduelle** d'un arc est le flot que l'on peut encore y ajouter (capacité initiale moins flot)
Le **graphe résiduel** est obtenu en conservant la capacité résiduelle de chaque arc. Si un arc a une capacité résiduelle nulle, il est supprimé.

Si on trouve un chemin $\overrightarrow{P}$ de s à t dans le graphe résiduel, on peut augmenter le flot du minimum des capacités de $\overrightarrow{P}$.

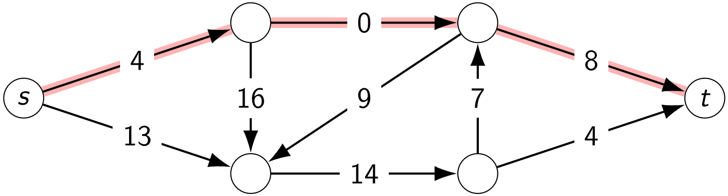
Algorithme de Ford-Fulkerson



Algorithme de Ford-Fulkerson



Algorithme de Ford-Fulkerson



Flot augmenté de 12 (et capacité résiduelle diminuée de 12)

Algorithme de Ford-Fulkerson

Algorithme de Ford-Fulkerson

Tant que $\exists$ un chemin $\overrightarrow{P}$ de s à t dans le graphe résiduel :

  $c \longleftarrow$ minimum des capacités de $\overrightarrow{P}$

  Diminuer de c la capacité des arcs de $\overrightarrow{P}$

Algorithme de Ford-Fulkerson

Algorithme de Ford-Fulkerson

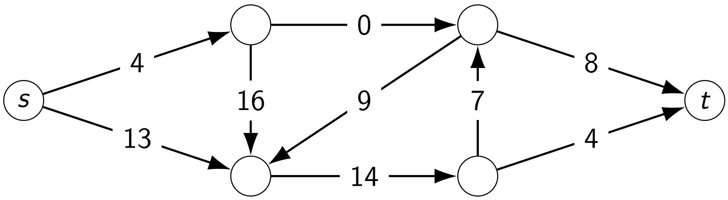
Tant que $\exists$ un chemin $\overrightarrow{P}$ de s à t dans le graphe résiduel :

  $c \longleftarrow$ minimum des capacités de $\overrightarrow{P}$

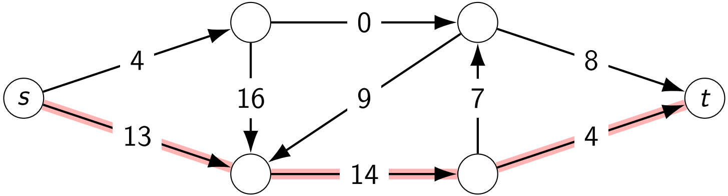
  Diminuer de c la capacité des arcs de $\overrightarrow{P}$

À la fin, on peut connaître le flot sur chaque arc en retranchant la capacité résiduelle à la capacités initiale.

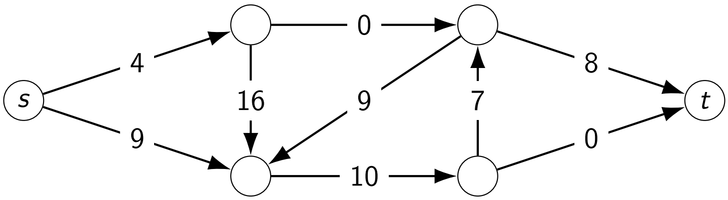
Algorithme de Ford-Fulkerson



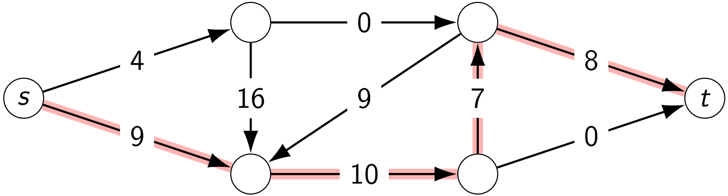
Algorithme de Ford-Fulkerson



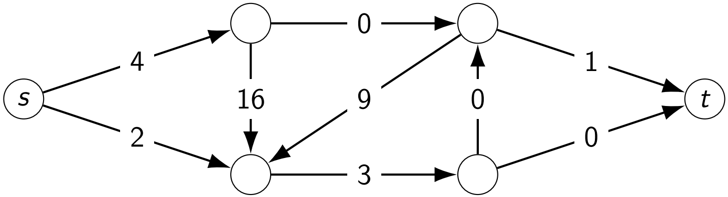
Algorithme de Ford-Fulkerson



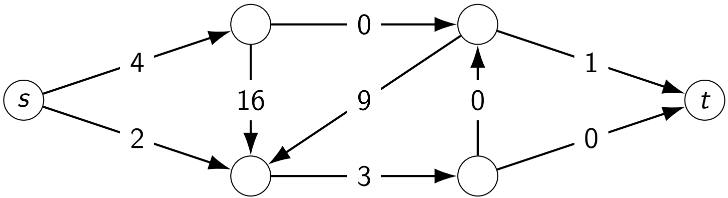
Algorithme de Ford-Fulkerson



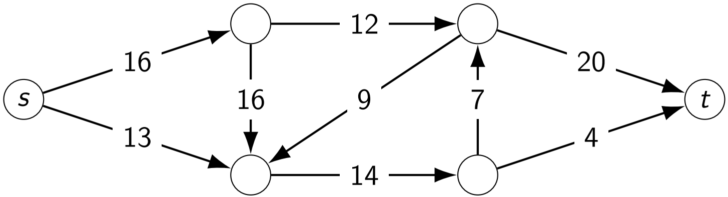
Algorithme de Ford-Fulkerson



Algorithme de Ford-Fulkerson



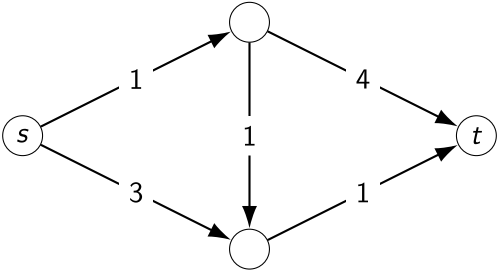
La comparaison avec le graphe initial permet de connaître le flot sur chaque arc :



Valeur du flot obtenu : 23

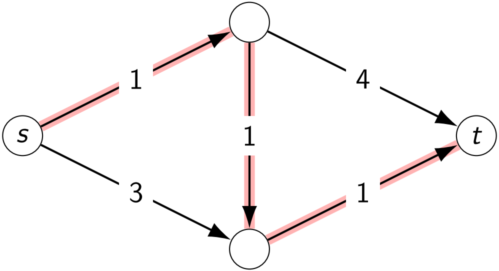
Algorithme de Ford-Fulkerson

Lorsqu'on ajoute du flot sur un arc, on créé dans le graphe résiduel un arc dans le sens inverse (un **arc arrière**) dans la direction opposée :



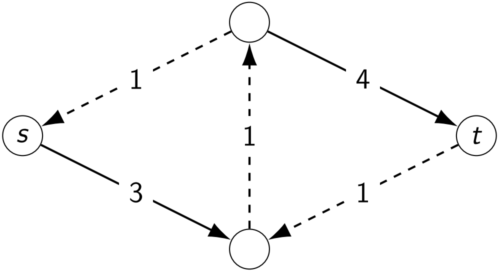
Algorithme de Ford-Fulkerson

Lorsqu'on ajoute du flot sur un arc, on créé dans le graphe résiduel un arc dans le sens inverse (un **arc arrière**) dans la direction opposée :



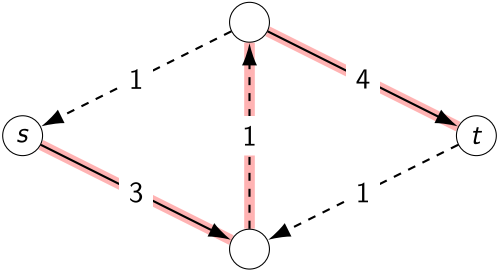
Algorithme de Ford-Fulkerson

Lorsqu'on ajoute du flot sur un arc, on créé dans le graphe résiduel un arc dans le sens inverse (un **arc arrière**) dans la direction opposée :



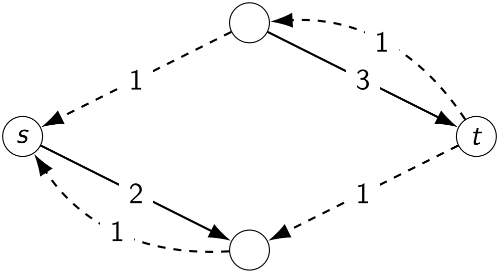
Algorithme de Ford-Fulkerson

Lorsqu'on ajoute du flot sur un arc, on créé dans le graphe résiduel un arc dans le sens inverse (un **arc arrière**) dans la direction opposée :



Algorithme de Ford-Fulkerson

Lorsqu'on ajoute du flot sur un arc, on créé dans le graphe résiduel un arc dans le sens inverse (un **arc arrière**) dans la direction opposée :



Algorithme de Ford-Fulkerson

Théorème

Si les capacités initiales sont toutes entières, l'algorithme de Ford-Fulkerson termine (ne fait pas boucle infinie).

Preuve :

Algorithme de Ford-Fulkerson

Théorème

Si les capacités initiales sont toutes entières, l'algorithme de Ford-Fulkerson termine (ne fait pas boucle infinie).

<u>Preuve</u> :

- Les capacités résiduelles restent toujours entières

Algorithme de Ford-Fulkerson

Théorème

Si les capacités initiales sont toutes entières, l'algorithme de Ford-Fulkerson termine (ne fait pas boucle infinie).

Preuve :

- Les capacités résiduelles restent toujours entières
- À chaque itération, on diminue au moins de 1 la somme de toutes les capacités résiduelles. Donc il ne peut pas y avoir plus de $c(\overrightarrow{E})$ itérations.

Algorithme de Ford-Fulkerson

Théorème

Si les capacités initiales sont toutes entières, l'algorithme de Ford-Fulkerson termine (ne fait pas boucle infinie).

Preuve :

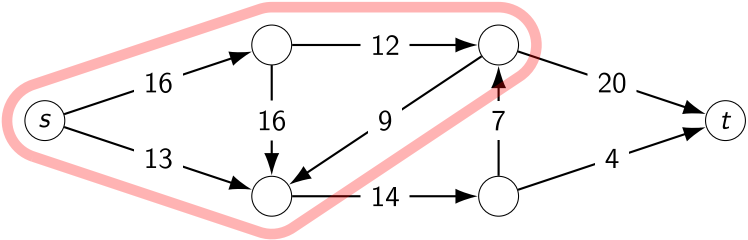
- Les capacités résiduelles restent toujours entières
- À chaque itération, on diminue au moins de 1 la somme de toutes les capacités résiduelles. Donc il ne peut pas y avoir plus de $c(\overrightarrow{E})$ itérations.

On a de plus un majorant grossier de la complexité : $\mathrm{O}(|c(\overrightarrow{E})|)$.
La complexité plus précise dépend de la façon dont on choisit les chemins.

Coupes

Définition

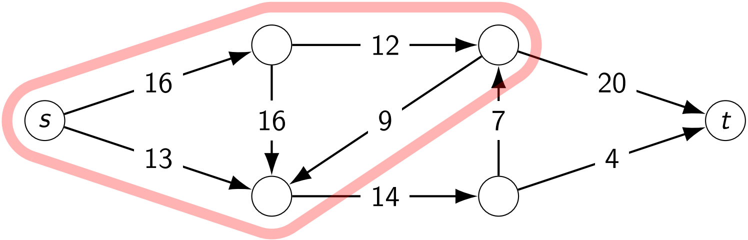
Une **coupe** de $\vec{G}$ est un ensemble $S \subseteq V$ contenant s mais pas t.



Coupes

Définition

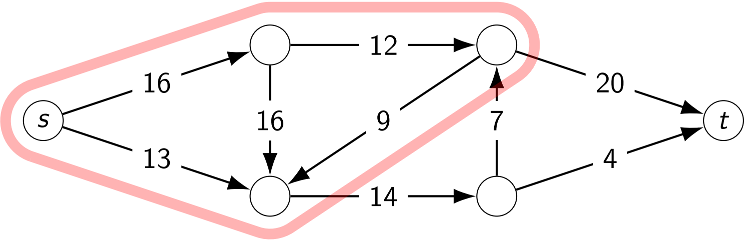
La capacité d'une coupe S est la somme $c(S^+)$ des capacités des arcs sortant de S :



Coupes

Définition

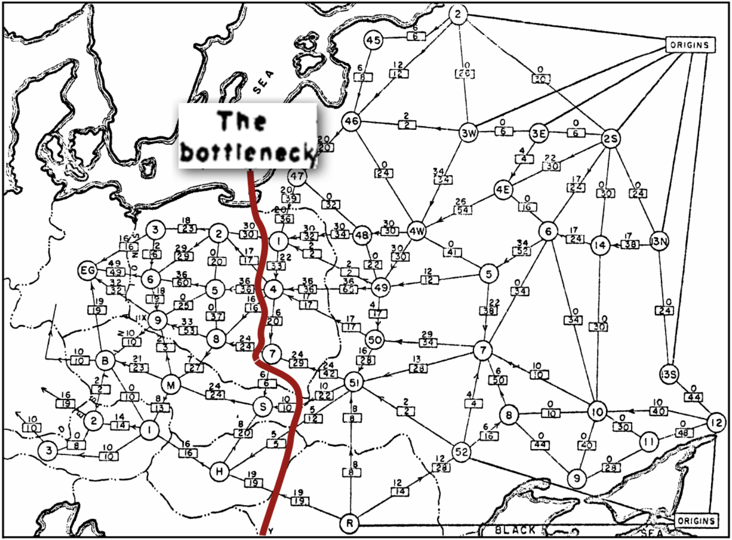
La capacité d'une coupe S est la somme $c(S^+)$ des capacités des arcs sortant de S :



La capacité de cette coupe est $20 + 14 = 34$.

Problème

Trouver une coupe de capacité maximum dans un graphe.



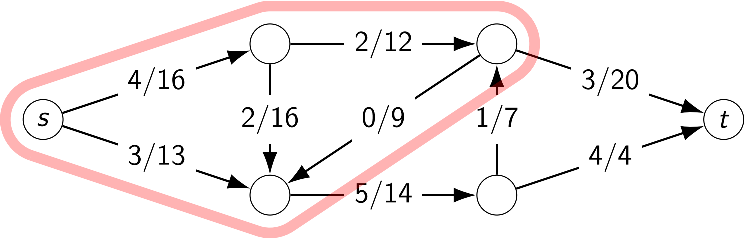
Plan américain de destruction d'un «min cut» des rails soviétiques

Coupes

Définition

Le flot sortant d'une coupe S est définie par :

$$f(S) = f(S^+) - f(S^-)$$

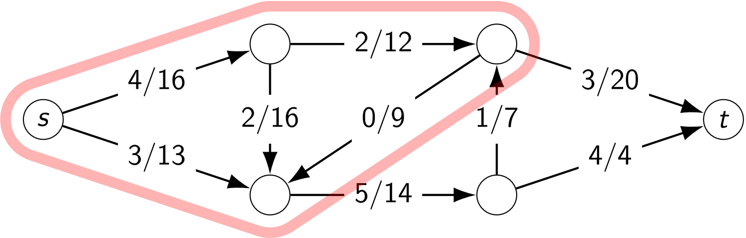


Coupes

Définition

Le flot sortant d'une coupe S est définie par :

$$f(S) = f(S^+) - f(S^-)$$



Le flot sortant de cette coupe est 5 + 3 - 1 = 7.

Coupes

Lemme 1

Si S est une coupe, $f(S) \leq c(S^+)$.

Coupes

Lemme 1

Si S est une coupe, $f(S) \leq c(S^+)$.

<u>Preuve</u> :

$$f(S) = f(S^+) - f(S^-) \leq f(S^+) \leq c(S^+)$$

Coupes

Lemme 2

Si S est une coupe, $f(S) = f(s^+)$.

Coupes

Lemme 2

Si S est une coupe, $f(S) = f(s^+)$.

Preuve :
Soit H_k : « Si S est une coupe avec k sommets alors $f(S) = f(s^+)$ ».

Coupes

Lemme 2

Si S est une coupe, $f(S) = f(s^+)$.

Preuve :
Soit H_k : « Si S est une coupe avec k sommets alors $f(S) = f(s^+)$ ».

- H_1 est vrai car alors $S = \{s\}$.

Coupes

Lemme 2

Si S est une coupe, $f(S) = f(s^+)$.

Preuve :

Soit H_k : « Si S est une coupe avec k sommets alors $f(S) = f(s^+)$ ».

- H_1 est vrai car alors $S = \{s\}$.
- Supposons H_k vraie pour un $k \geq 1$.

Coupes

Lemme 2

Si S est une coupe, $f(S) = f(s^+)$.

Preuve :

Soit H_k : « Si S est une coupe avec k sommets alors $f(S) = f(s^+)$ ».

- H_1 est vrai car alors $S = \{s\}$.
- Supposons H_k vraie pour un $k \geq 1$.
  Soit S une coupe avec $k+1$ sommets.

Coupes

Lemme 2

Si S est une coupe, $f(S) = f(s^+)$.

<u>Preuve</u> :
Soit H_k : « Si S est une coupe avec k sommets alors $f(S) = f(s^+)$ ».

- H_1 est vrai car alors $S = \{s\}$.
- Supposons H_k vraie pour un $k \geq 1$.
  Soit S une coupe avec $k+1$ sommets.
  Soit $v \in S$. Alors, en utilisant la définition :
  $f(S) = f(S - \{v\}) + f(\{v\})$.

Coupes

Lemme 2

Si S est une coupe, $f(S) = f(s^+)$.

Preuve :
Soit H_k : « Si S est une coupe avec k sommets alors $f(S) = f(s^+)$ ».

- H_1 est vrai car alors $S = \{s\}$.
- Supposons H_k vraie pour un $k \geq 1$.
  Soit S une coupe avec $k+1$ sommets.
  Soit $v \in S$. Alors, en utilisant la définition :
  $f(S) = f(S - \{v\}) + f(\{v\})$.Or $f(S \backslash \{v\}) = f(s^+)$ par hypothèse de récurrence et $f(\{v\}) = 0$.
  Donc $f(S) = f(s^+)$: H_{k+1} est vraie.

Coupes

Lemme 1

Si f est un flot et S une coupe, $f(S) \leq c(S^+)$.

Lemme 2

Si f est un flot et S une coupe, $f(S) = f(s^+)$.

Théorème max flow - min cut

Si un flot f et une coupe S vérifient $f(S) = c(S)$ alors :

- f est un flot de valeur maximum
- S une coupe de capacité minimum

Preuve :

Coupes

Lemme 1

Si f est un flot et S une coupe, $f(S) \leq c(S^+)$.

Lemme 2

Si f est un flot et S une coupe, $f(S) = f(s^+)$.

Théorème max flow - min cut

Si un flot f et une coupe S vérifient $f(S) = c(S)$ alors :

- f est un flot de valeur maximum
- S une coupe de capacité minimum

<u>Preuve</u> :Soit f^* un flot de valeur maximum.

Coupes

Lemme 1

Si f est un flot et S une coupe, $f(S) \leq c(S^+)$.

Lemme 2

Si f est un flot et S une coupe, $f(S) = f(s^+)$.

Théorème max flow - min cut

Si un flot f et une coupe S vérifient $f(S) = c(S)$ alors :

- f est un flot de valeur maximum
- S une coupe de capacité minimum

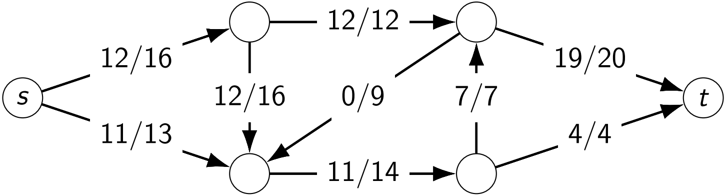
Preuve :Soit f^* un flot de valeur maximum.

$$f^*(s^+) \underset{2}{=} f^*(S) \underset{1}{\leq} c(S) = f(S) \underset{2}{=} f(s^+)$$

Coupes

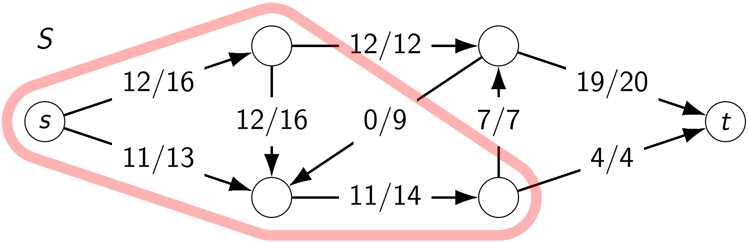
Question

Comment prouver que ce flot est maximum ?



Question

Comment prouver que ce flot est maximum ?



$c(S) = 23 = f(S)$

Donc f est un flot maximum et S une coupe minimum.

Coupes

Théorème

Si l'algorithme de Ford-Fulkerson termine, le flot obtenu est un flot maximum

Preuve :

Coupes

Théorème

Si l'algorithme de Ford-Fulkerson termine, le flot obtenu est un flot maximum

Preuve :
Soit S l'ensemble des sommets accessibles depuis s dans le graphe résiduel.
Tout arc $\vec{e}$ sortant de S a une capacité résiduelle nulle, donc $c(\vec{e}) = f(\vec{e})$.
D'où:

$$c(S) = \sum_{\vec{e} \in \vec{E}} c(\vec{e}) = \sum_{\vec{e} \in \vec{E}} f(\vec{e}) = f(S)$$