

Partie VII – Réseaux de flots

Graphes et Algorithmes – GRAAL

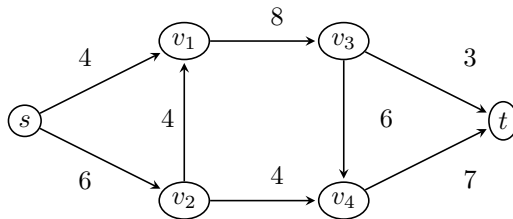
Gaëtan Staquet
gaetan.staquet@ec-nantes.fr

École Centrale de Nantes – LS2N
S508

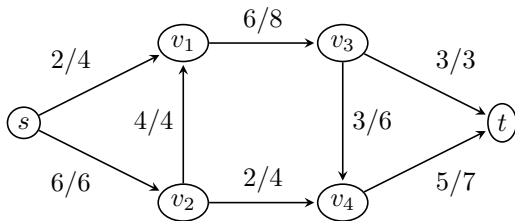
Janvier à mars 2026

1. Réseaux de flot
2. Algorithme de Ford-Fulkerson

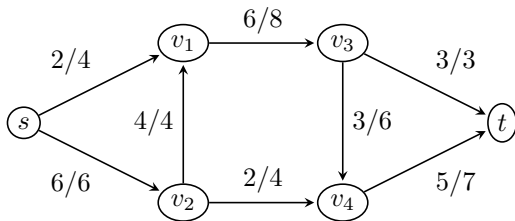
Des flots



Des flots



Des flots



Remarque VII.1. Pour simplifier la présentation, les exemples seront acycliques. En revanche, les arguments et les algorithmes s'appliquent également sur les graphes contenant des circuits.

Source pour cette partie :

<https://people.irisa.fr/Francois.Schwarzentruer/algo1/ALG01-all.pdf>.

1. Réseaux de flot

2. Algorithme de Ford-Fulkerson

Réseau de flot et flot

Définition VII.2 (Réseau de flot). Un **réseau de flot** est un tuple $\mathcal{G} = (V, E, c, s, t)$ où

- ▶ (V, E, c) est un graphe orienté pondéré avec $c : V^2 \rightarrow \mathbb{R}^{\geq 0}$ tel que, pour tout $(u, v) \notin E$, $c((u, v)) = 0$;
- ▶ $s \in V$ est un sommet appelé **source**, sans arcs entrants ; et
- ▶ $t \in V$ est un sommet appelé **puits**, sans arcs sortants.

La valeur $c((u, v))$ est la **capacité** de l'arc (u, v) .

Flot

Définition VII.3 (Flot). Un **flot** d'un réseau de flot $\mathcal{G} = (V, E, c, s, t)$ est une fonction $f : V^2 \rightarrow \mathbb{R}^{\geq 0}$ telle que

1. pour tous sommets u, v dans V , $0 \leq f((u, v)) \leq c((u, v))$; et
2. pour tout sommet $u \in V \setminus \{s, t\}$,

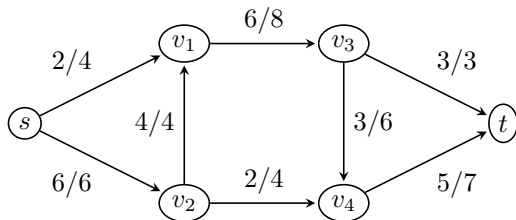
$$\sum_{v \in V} f((v, u)) = \sum_{w \in V} f((u, w)).$$

La **valeur** d'un flot f est

$$|f| = \sum_{v \in V} f((s, v)).$$

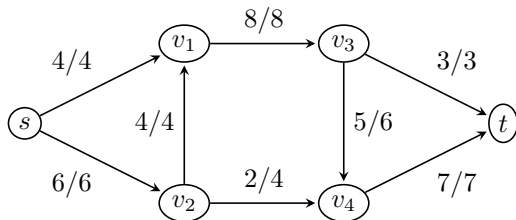
Remarque VII.4. Si $(u, v) \notin E$, alors $c((u, v)) = 0 = f((u, v))$.

Flot – Exemple



On a un flot de valeur $2 + 6 = 8$.

Flot – Exemple



On a un flot de valeur $4 + 6 = 10$.

Flot maximum

Définition VII.5 (Flot maximum). Un flot f est maximum de \mathcal{G} si la valeur $|f|$ est maximum, *i.e.*, pour tout flot g de \mathcal{G} , $|g| \leq |f|$.

Flot maximum

Définition VII.5 (Flot maximum). Un flot f est maximum de \mathcal{G} si la valeur $|f|$ est maximum, *i.e.*, pour tout flot g de \mathcal{G} , $|g| \leq |f|$.

On veut un algorithme qui trouve un flot maximum, étant donné un réseau de flot \mathcal{G} .

Une méthode gloutonne ?

```

1 : procédure FLOTMAXIMUMGLOUTON( $\mathcal{G} = (V, E, c, s, t)$ )
2 :    $f \leftarrow$  flot nul
3 :   tant que il existe un chemin simple de  $s$  à  $t$  qui améliore le flot faire
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$   $\triangleright$  Combien d'eau peut-on ajouter le long du chemin ?
5 :     pour chaque  $e$  dans ce chemin faire
6 :        $f(e) \leftarrow f(e) + \Delta$   $\triangleright$  On met à jour le flot le long du chemin

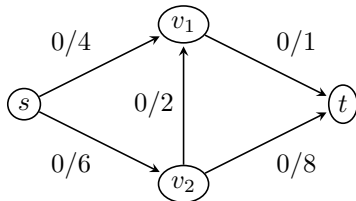
```

Une méthode gloutonne ?

```

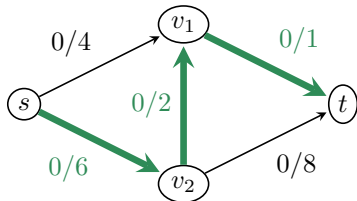
1 : procédure FLOTMAXIMUMGLOUTON( $\mathcal{G} = (V, E, c, s, t)$ )
2 :    $f \leftarrow$  flot nul
3 :   tant que il existe un chemin simple de  $s$  à  $t$  qui améliore le flot faire
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$  ▷ Combien d'eau peut-on ajouter le long du chemin ?
5 :     pour chaque  $e$  dans ce chemin faire
6 :        $f(e) \leftarrow f(e) + \Delta$  ▷ On met à jour le flot le long du chemin

```



Une méthode gloutonne ?

```
1 : procédure FLOTMAXIMUMGLOUTON( $\mathcal{G} = (V, E, c, s, t)$ )
2 :    $f \leftarrow$  flot nul
3 :   tant que il existe un chemin simple de  $s$  à  $t$  qui améliore le flot faire
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$  ▷ Combien d'eau peut-on ajouter le long du chemin ?
5 :     pour chaque  $e$  dans ce chemin faire
6 :        $f(e) \leftarrow f(e) + \Delta$  ▷ On met à jour le flot le long du chemin
```

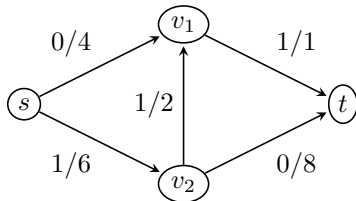


Une méthode gloutonne ?

```

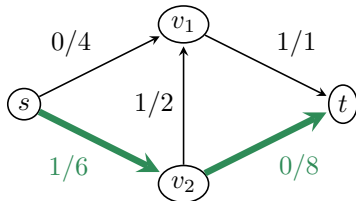
1 : procédure FLOTMAXIMUMGLOUTON( $\mathcal{G} = (V, E, c, s, t)$ )
2 :    $f \leftarrow$  flot nul
3 :   tant que il existe un chemin simple de  $s$  à  $t$  qui améliore le flot faire
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$  ▷ Combien d'eau peut-on ajouter le long du chemin ?
5 :     pour chaque  $e$  dans ce chemin faire
6 :        $f(e) \leftarrow f(e) + \Delta$  ▷ On met à jour le flot le long du chemin

```



Une méthode gloutonne ?

```
1 : procédure FLOTMAXIMUMGLOUTON( $\mathcal{G} = (V, E, c, s, t)$ )  
2 :    $f \leftarrow$  flot nul  
3 :   tant que il existe un chemin simple de  $s$  à  $t$  qui améliore le flot faire  
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$   $\triangleright$  Combien d'eau peut-on ajouter le long du chemin ?  
5 :     pour chaque  $e$  dans ce chemin faire  
6 :        $f(e) \leftarrow f(e) + \Delta$   $\triangleright$  On met à jour le flot le long du chemin
```

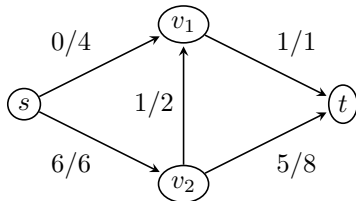


Une méthode gloutonne ?

```

1 : procédure FLOTMAXIMUMGLOUTON( $\mathcal{G} = (V, E, c, s, t)$ )
2 :    $f \leftarrow$  flot nul
3 :   tant que il existe un chemin simple de  $s$  à  $t$  qui améliore le flot faire
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$  ▷ Combien d'eau peut-on ajouter le long du chemin ?
5 :     pour chaque  $e$  dans ce chemin faire
6 :        $f(e) \leftarrow f(e) + \Delta$  ▷ On met à jour le flot le long du chemin

```



On obtient un flot de valeur 6.

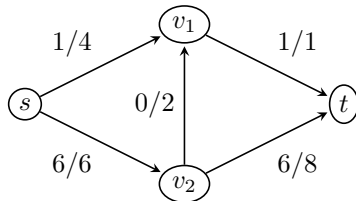
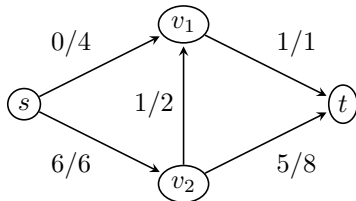
Une méthode gloutonne ?

```

1 : procédure FLOTMAXIMUMGLOUTON( $\mathcal{G} = (V, E, c, s, t)$ )
2 :    $f \leftarrow$  flot nul
3 :   tant que il existe un chemin simple de  $s$  à  $t$  qui améliore le flot faire
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$   $\triangleright$  Combien d'eau peut-on ajouter le long du chemin ?
5 :     pour chaque  $e$  dans ce chemin faire
6 :        $f(e) \leftarrow f(e) + \Delta$ 

```

\triangleright On met à jour le flot le long du chemin



On obtient un flot de valeur 6. Il existe un flot de valeur 7.

1. Réseaux de flot
2. Algorithme de Ford-Fulkerson

Graphe résiduel

On voudrait **annuler** des choix qui s'avèrent être non optimaux. Construisons un graphe! 😊

Graphe résiduel

On voudrait **annuler** des choix qui s'avèrent être non optimaux. Construisons un graphe ! 😊

Définition VII.6 (Graphe résiduel). Le **graphe résiduel** d'un réseau $\mathcal{G} = (V, E, c, s, t)$ et d'un flot f est le graphe pondéré $R_f = (V, E_f, r_f)$ avec

$$\forall (u, v) \in E : r_f((u, v)) = \begin{cases} c((u, v)) - f((u, v)) & \text{si } (u, v) \in E \\ f((v, u)) & \text{si } (v, u) \in E \\ 0 & \text{sinon} \end{cases}$$

On ajoute de l'eau→
On retire de l'eau - - ->
On ne fait rien

$$E_f = \{(u, v) \in V \times V \mid r_f((u, v)) > 0\}.$$

Graphe résiduel

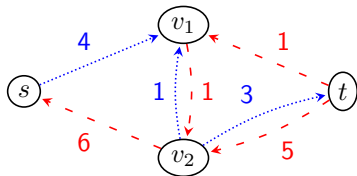
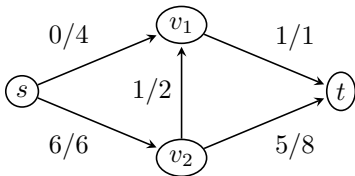
On voudrait **annuler** des choix qui s'avèrent être non optimaux. Construisons un graphe ! 😊

Définition VII.6 (Graphe résiduel). Le **graphe résiduel** d'un réseau $\mathcal{G} = (V, E, c, s, t)$ et d'un flot f est le graphe pondéré $R_f = (V, E_f, r_f)$ avec

$$\forall (u, v) \in E : r_f((u, v)) = \begin{cases} c((u, v)) - f((u, v)) & \text{si } (u, v) \in E \\ f((v, u)) & \text{si } (v, u) \in E \\ 0 & \text{sinon} \end{cases}$$

On ajoute de l'eau>
On retire de l'eau - - ->
On ne fait rien

$$E_f = \{(u, v) \in V \times V \mid r_f((u, v)) > 0\}.$$



Algorithme de Ford-Fulkerson

```

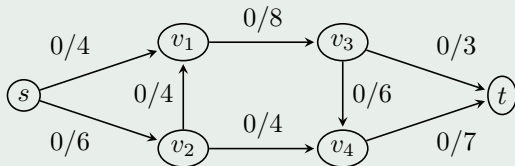
1 : procédure FORDFULKERSON( $\mathcal{G} = (V, E, c, s, t)$ )
2 :    $f \leftarrow$  flot nul
3 :   tant que il existe un chemin simple  $\gamma$  reliant  $s$  à  $t$  dans le graphe résiduel  $R_f$  faire
4 :      $\Delta \leftarrow \min\{r_f(e) \mid e \text{ est un arc emprunté le long de } \gamma\}$ 
5 :     pour chaque arc  $e$  dans  $\gamma$  faire
6 :       si  $e \in E$  alors
7 :          $f(e) \leftarrow f(e) + \Delta$ 
8 :       sinon
9 :          $f(e) \leftarrow f(e) - \Delta$ 
10 :   retourner  $f$ 

```

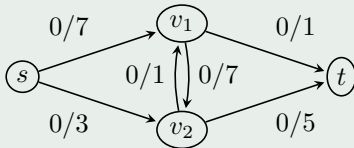
Un exemple au tableau.

Algorithme de Ford-Fulkerson – Exercices

Exercice VII.7. Appliquez l'algorithme de Ford-Fulkerson sur le graphe suivant.



Exercice VII.8. Appliquez l'algorithme de Ford-Fulkerson sur le graphe suivant. Existe-t-il un unique flot maximum ?



Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

Arrêt et complexité

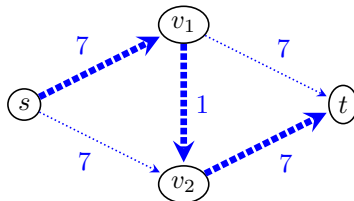
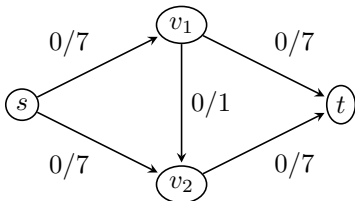
Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?

Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

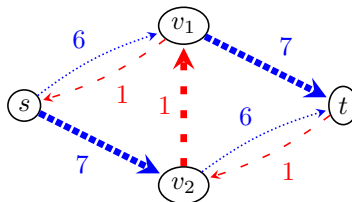
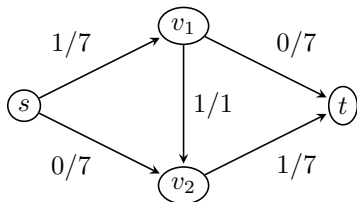
Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?



Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

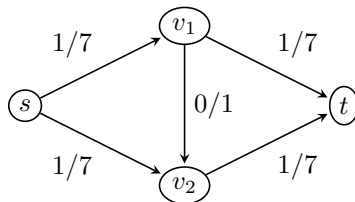
Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?



Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?

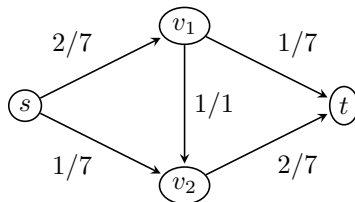


Et ainsi de suite.

Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

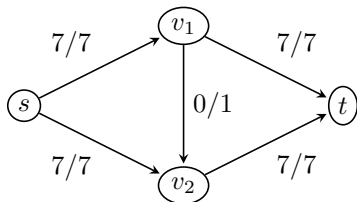
Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?



Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

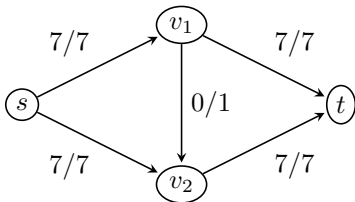
Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?



Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?

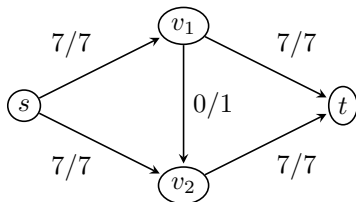


Sur cet exemple, il nous faut, dans le pire des cas, $2 * 7$ itérations. Autrement dit, $|f^*|$ itérations, où f^* est un flot maximum.

Arrêt et complexité

Lemme VII.9. Si les capacités sont des **entiers**, l'algorithme de Ford-Fulkerson termine.

Question VII.10. Est-ce que l'algorithme de Ford-Fulkerson a une complexité polynomiale en le nombre de sommets et d'arêtes ?



Sur cet exemple, il nous faut, dans le pire des cas, $2 * 7$ itérations. Autrement dit, $|f^*|$ itérations, où f^* est un flot maximum.

Proposition VII.11. L'algorithme de Ford-Fulkerson est en $\mathcal{O}(|f^*|(|V| + |E|))$.

Vers la correction

Question VII.12. Pourquoi est-ce que l'algorithme de Ford-Fulkerson calcule bien un flot maximum ?

On introduit un certificat d'optimalité : une coupe minimale du réseau de flot.

Correction – Coupe

Définition VII.13. Une **coupe** (S, T) d'un réseau de flots $\mathcal{G} = (V, E, c, s, t)$ est une partition de V avec $s \in S$ et $t \in T$.

La **capacité d'une coupe** (S, T) est

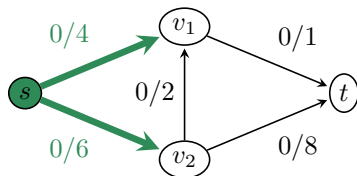
$$c(S, T) = \sum_{(u,v) \in S \times T} c((u, v)).$$

Correction – Coupe

Définition VII.13. Une **coupe** (S, T) d'un réseau de flots $\mathcal{G} = (V, E, c, s, t)$ est une partition de V avec $s \in S$ et $t \in T$.

La **capacité d'une coupe** (S, T) est

$$c(S, T) = \sum_{(u,v) \in S \times T} c((u, v)).$$



Toutes les coupes possibles de ce réseau et leur capacité :

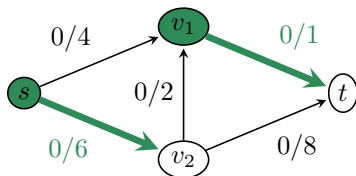
- Avec $S = \{s\}$ et $T = \{v_1, v_2, t\}$,
 $c(S, T) = 4 + 6 = 10$.

Correction – Coupe

Définition VII.13. Une **coupe** (S, T) d'un réseau de flots $\mathcal{G} = (V, E, c, s, t)$ est une partition de V avec $s \in S$ et $t \in T$.

La **capacité d'une coupe** (S, T) est

$$c(S, T) = \sum_{(u,v) \in S \times T} c((u,v)).$$



Toutes les coupes possibles de ce réseau et leur capacité :

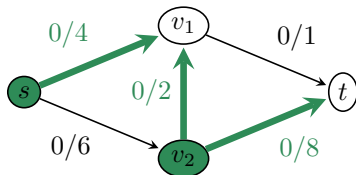
- ▶ Avec $S = \{s\}$ et $T = \{v_1, v_2, t\}$,
 $c(S, T) = 4 + 6 = 10$.
- ▶ $c(\{s, v_1\}, \{v_2, t\}) = 1 + 6 = 7$.

Correction – Coupe

Définition VII.13. Une **coupe** (S, T) d'un réseau de flots $\mathcal{G} = (V, E, c, s, t)$ est une partition de V avec $s \in S$ et $t \in T$.

La **capacité d'une coupe** (S, T) est

$$c(S, T) = \sum_{(u,v) \in S \times T} c((u,v)).$$



Toutes les coupes possibles de ce réseau et leur capacité :

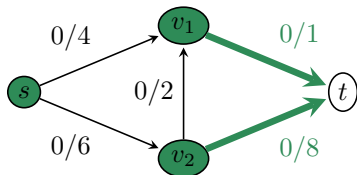
- ▶ Avec $S = \{s\}$ et $T = \{v_1, v_2, t\}$,
 $c(S, T) = 4 + 6 = 10$.
- ▶ $c(\{s, v_1\}, \{v_2, t\}) = 1 + 6 = 7$.
- ▶ $c(\{s, v_2\}, \{v_1, t\}) = 4 + 2 + 8 = 14$.

Correction – Coupe

Définition VII.13. Une **coupe** (S, T) d'un réseau de flots $\mathcal{G} = (V, E, c, s, t)$ est une partition de V avec $s \in S$ et $t \in T$.

La **capacité d'une coupe** (S, T) est

$$c(S, T) = \sum_{(u,v) \in S \times T} c((u,v)).$$



Toutes les coupes possibles de ce réseau et leur capacité :

- ▶ Avec $S = \{s\}$ et $T = \{v_1, v_2, t\}$,
 $c(S, T) = 4 + 6 = 10$.
- ▶ $c(\{s, v_1\}, \{v_2, t\}) = 1 + 6 = 7$.
- ▶ $c(\{s, v_2\}, \{v_1, t\}) = 4 + 2 + 8 = 14$.
- ▶ $c(\{s, v_1, v_2\}, \{t\}) = 1 + 8 = 9$.

Correction – Flot maximum \leq coupe minimum

Soient f un flot d'un réseau $\mathcal{G} = (V, E, c, s, t)$ et $S, T \subseteq V$. On note

$$f(S, T) = \sum_{(u, v) \in S \times T} f((u, v)).$$

Lemme VII.14. Soient f un flot et (S, T) une coupe. On a $|f| = f(S, T) - f(T, S)$.

Corollaire VII.15. Pour tout flot f et toute coupe (S, T) , on a $|f| \leq c(S, T)$.

Correction – Flot maximum \geq coupe minimum

Théorème VII.16. Soient un réseau $\mathcal{G} = (V, E, c, s, t)$ et un flot f . Les affirmations suivantes sont équivalentes.

- ▶ Le flot f est maximum.
- ▶ Il n'existe pas de chemin de s à t dans le graphe résiduel R_f .
- ▶ Il existe une coupe (S, T) telle que $|f| = c(S, T)$.

Corollaire VII.17. Pour un flot maximum f et une coupe minimum (S, T) , on a $|f| \geq c(S, T)$.

Correction – Flot maximum = coupe minimum

Corollaire VII.18. Pour un flot maximum f et une coupe minimum (S, T) , on a $|f| = c(S, T)$.

Corollaire VII.19. L'algorithme de Ford-Fulkerson est correct.

Démonstration. L'algorithme s'arrête lorsqu'il n'y a plus de chemin dans le graphe résiduel. Par le théorème précédent, le flot construit est maximum. □