ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

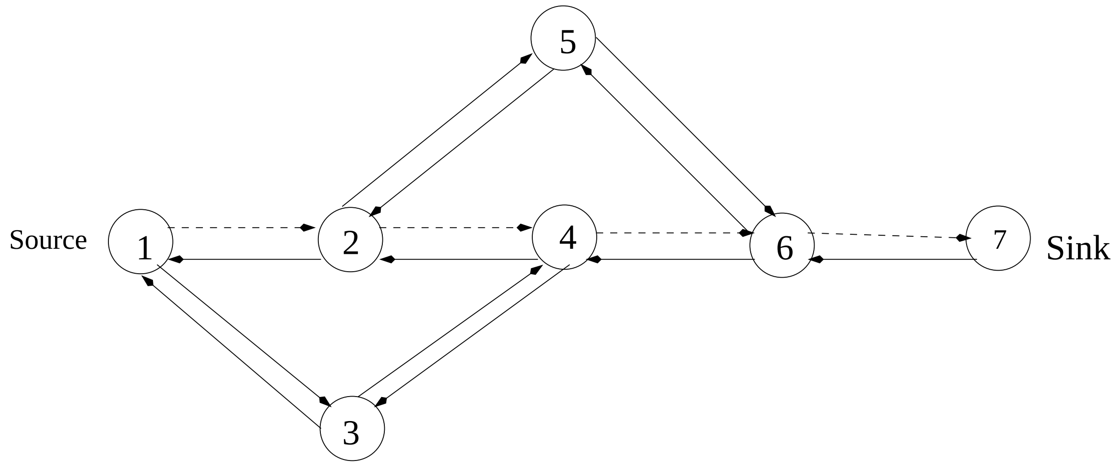
Section d'Informatique et de Systèmes de Communication

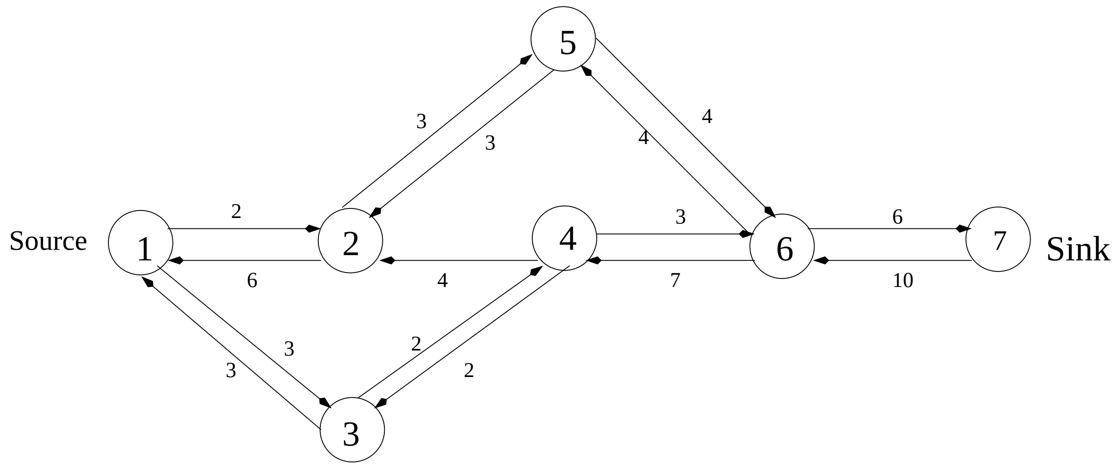
Corrigé de la série 11 7 Decembre 2009

1. *MaxFlowMinCut*

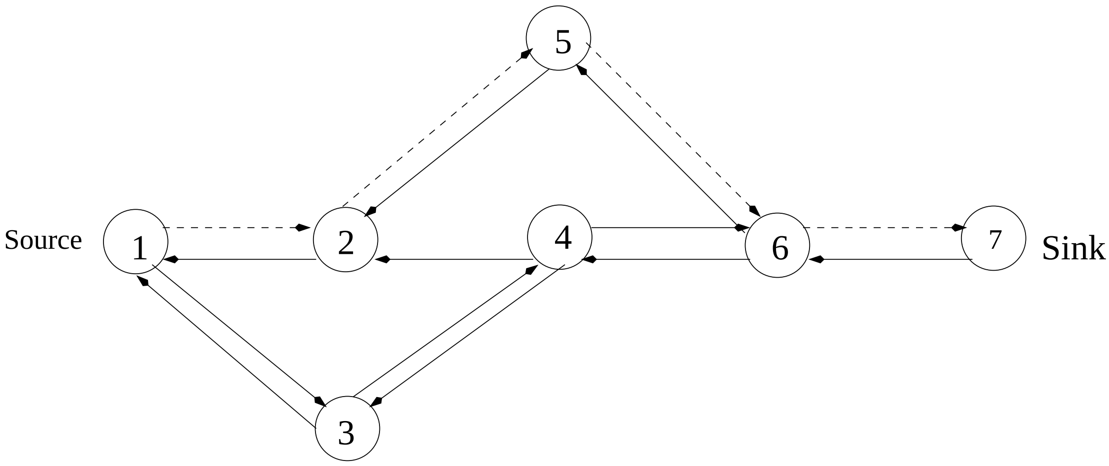
Au début, chaque arête est remplacée par deux arcs de sens opposé et de capacité égale à celle de l'arête de départ.

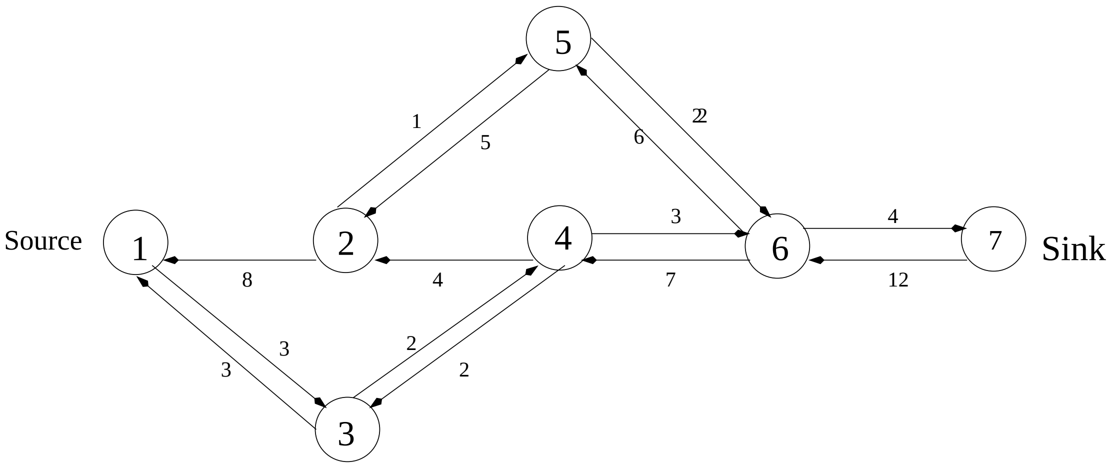
L'algorithme construit un chemin f-augmentant. On augmente le flux de deux unités le long de ce chemin et on met à jour le réseau résiduel. Pour chacune des itérations (dans cet exercice, l'algorithme trouve le flux maximal au bout de 3 itérations), les graphes indiquent le chemin f-augmentant de la source au sink trouvé et le graphe résiduel résultant.



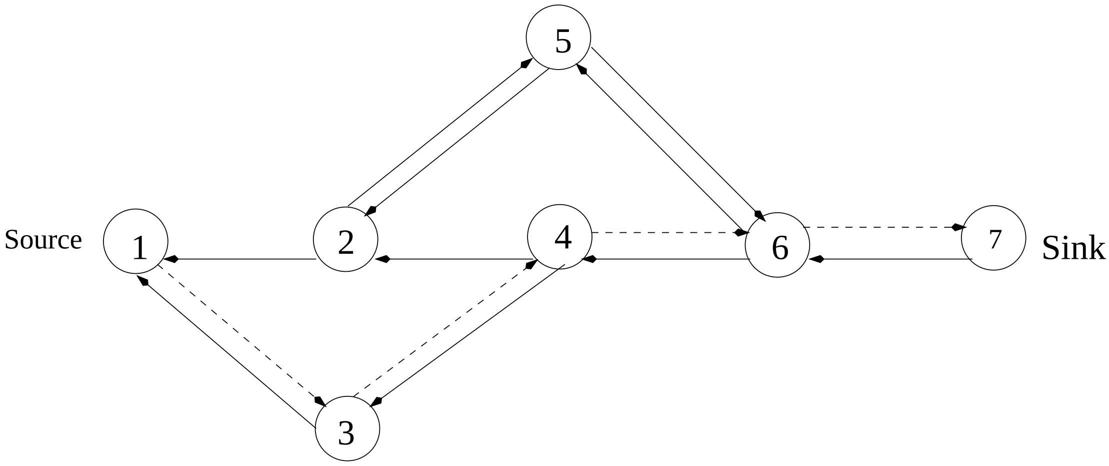


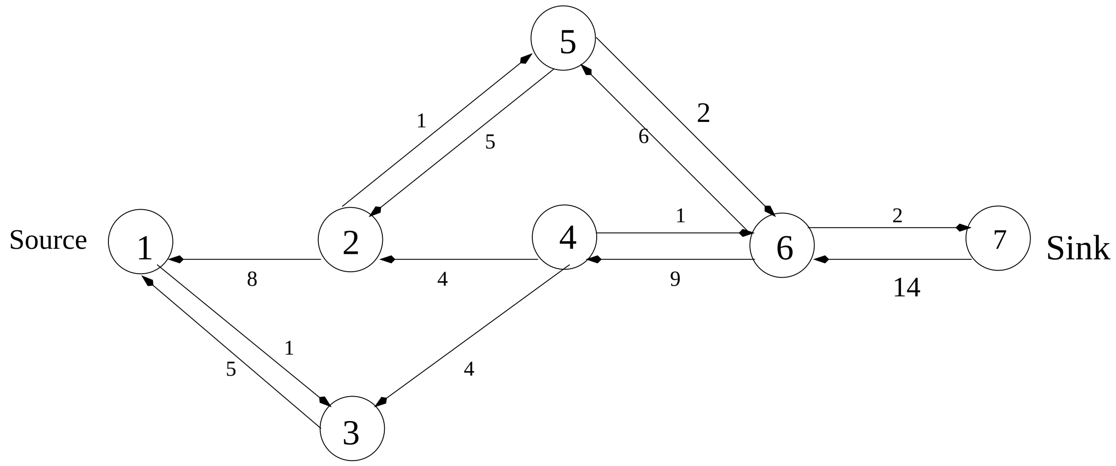
A l'iteration suivante, l'algorithme construit un chemin f-augmentant de la source au sink. On augmente le flux de deux unités le long de ce chemin et on met à jour le graphe résiduel.



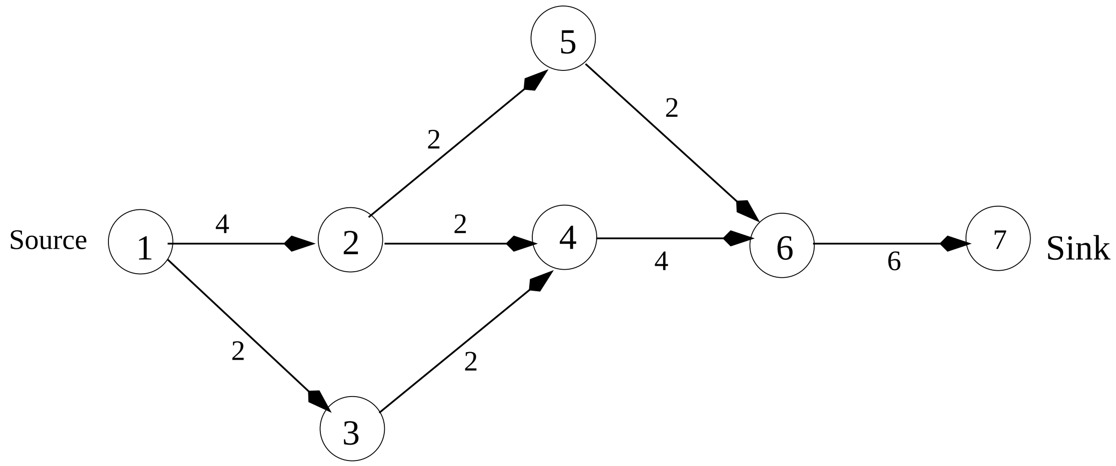


A l'iteration suivante, l'algorithme construit un chemin f-augmentant de la source au sink. On augmente le flux de deux unités le long de ce chemin et on met à jour le réseau résiduel.





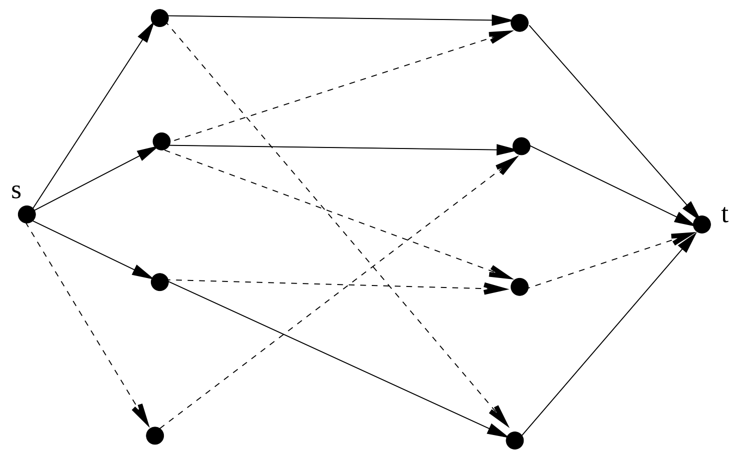
A cette étape de l'algorithme, depuis la source (sommet numeroté *1*), l'on ne peut atteindre que le sommet numeroté *3*. Le cut minimum est S= { 1,3 } de valeur 6. On trouve une solution optimale au problème du flux maximum de valeur 6.



2. Matching

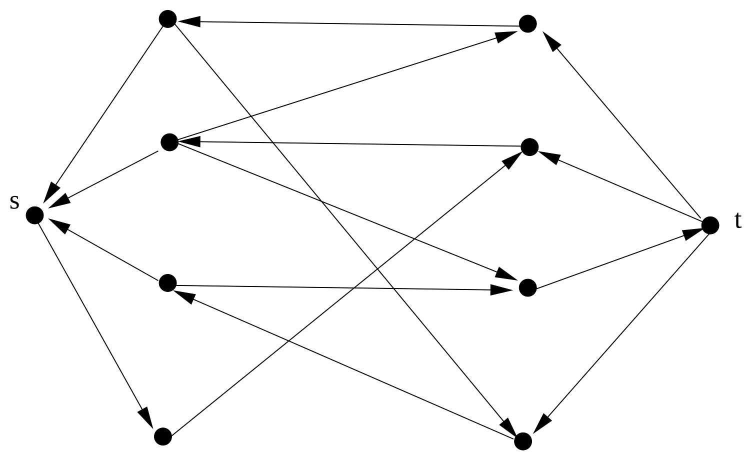
Nous utilisons l'algorithme MaxFlowMinCut comme décrit dans le cours.

Nous ajoutons 2 sommets s et t et transformons le graphe en réseau dans lequel toutes les arêtes ont une capacité de 1. Au matching de l'énnoncé correspond un flux (voir cours), nous dessinons ce flux ci-dessous:

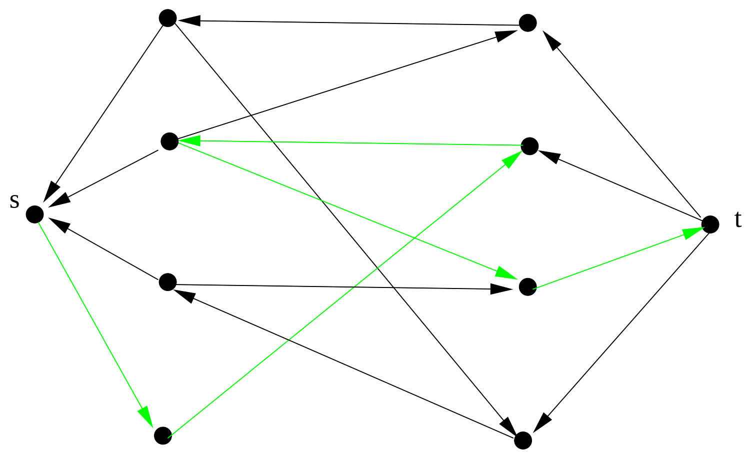


(Toutes les arêtes ont capacité 1, Celles en pointillé ont un flux de 0/1, alors que les autres ont un flux de 1/1). La valeur de ce flux est 3.

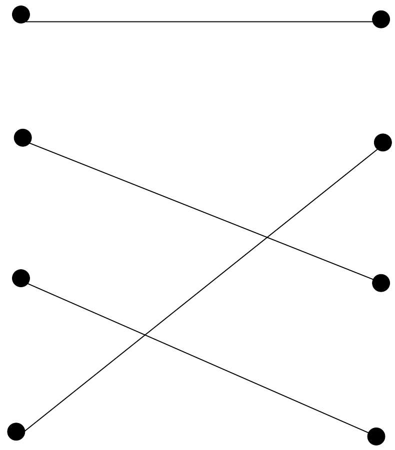
Nous dessinons ensuite le graphe résiduel par rapport à ce flux afin de voir s'il est possible de trouver un chemin augmentant:



Nous prenons le chemin augmentant ci-dessous:



Et obtenons ainsi le matching suivant:



Le matching proposé n'était donc pas maximal.

3. L'algorithme de Karp

a) Soit $s \to \cdots \to x$ un chemin de poids minimal et de longueur k. On suppose $k \geq 2$ (l'autre cas est immédiatement réglé), et alors on peut écrire ce chemin $s \to \cdots \to u \to x$, où u est l'avant-dernier sommet sur ce chemin. Le chemin $s \to \cdots \to u$ est de longueur $k-1$ et certainement minimal, parce que sinon on pourrait le remplacer par le chemin minimal et on aura aussi un meilleur chemin de s à x.

Donc par hypothèse d'induction le poids du morceau $s \to \cdots \to u$ est $F_{k-1}(u)$, et le chemin $s \to x$ a certainement poids

$$F_{k-1}(u) + w(u, x).$$

Le reste est évident.

b) Comme le graphe est connexe, certainement $|V| \leq |E|+1$, ce qui montre que F_0 peut bien être calculé en $O(|E|)$ opérations. La suite est par induction sur k.

Supposons qu'on a déjà calculé $F_{k-1}, \ldots, F_0$ en $O(|E|k)$ opérations. Si nous montrons que F_k peut alors être trouvé en $O(|E|)$, nous avons fini. Il s'agit d'appliquer la formule du point a). On procède comme suit:

for $x \in V$ **do**
  $F_k(x) \leftarrow \infty$
for $(x, y) \in E$ **do**
  $u \leftarrow F_{k-1}(x) + w(x, y)$
  if $u < F_k(y)$ **then**
    $F_k(y) \leftarrow u.$

Comme $|V| = O(|E|)$, cet algorithme utilise évidemment $O(|E|)$ opérations. Il est correct à cause du point précédent.

c) D'abord on calcule les F_k et ensuite le μ^*. Voir l'Algorithme 1 pour les détails.

Algorithme 1 Karp

```
Choisir s ∈ V arbitraire.
for x ∈ V do
  F_0(x) ← ∞
F_0(s) ← 0
for k = 1, . . . , n do
  for x ∈ V do
    F_k(x) ← ∞
  for (x, y) ∈ E do
    u ← F_{k−1}(x) + w(x, y)
    if u < F_k(y) then
      F_k(y) ← u.
μ* ← ∞
for x ∈ V do
  μ_tmp ← −∞
  for k = 0, · · · , n − 1 do
    u ← (F_n(x) − F_k(x)) / (n − k)
    if u > μ_tmp then
      μ_tmp ← u
  if μ_tmp < μ* then
    μ* ← μ_tmp
```