AL5
TD n° 09 : Flots

Comment trouver du flot

On rappelle qu'un *flot* sur un graphe G orienté pondéré par une pondération $\mathcal{C}$ et entre deux sommets spéciaux $e, s \in V$ est le fait d'attribuer à chaque arc (u, v) une valeur $\varphi(u, v)$ telle que :

1. Pour tout arc (u, v), $0 \leq \varphi(u, v) \leq \mathcal{C}(u, v)$ (où $\mathcal{C}(u, v)$ est la capacité de l'arc) ;
2. Pour tout sommet u (tel que $u \neq e$ et $u \neq s$), on a $\sum_v \varphi(v, u) = \sum_w \varphi(u, w)$ (loi des nœuds) ;

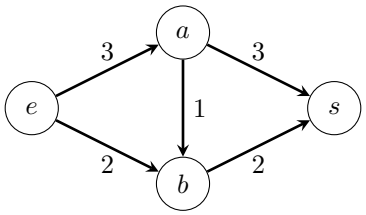
Les sommets e et s sont l'*entrée* et la *sortie*. La *charge* du flot est $\sum_v \varphi(e, v)$ $(= \sum_w \varphi(w, s)$).

Un flot maximum est un flot de charge maximum. L'algorithme de Ford-Fulkerson est le suivant :

Soit φ un flot vide (pour tout arc (u, v), $\varphi(u, v) = 0$);
tant que *Il existe un chemin améliorant* $\mathcal{P}$ **faire**
  Augmenter le flot φ le long de $\mathcal{P}$ de la charge de $\mathcal{P}$;
Renvoyer φ

Exercice 1 :

Une définition naturelle de *chemin améliorant* serait :
$\mathcal{P}$ est un chemin de e à s tel que le long de tout arc (u, v) de ce chemin, $\mathcal{C}(u, v) - \varphi(u, v) > 0$.
La charge de $\mathcal{P}$ est alors le minimum de $\mathcal{C}(u, v) - \varphi(u, v)$ le long du chemin.
Appliquer l'algorithme *pseudo* Ford-Fulkerson utilisant cette notion de *chemin améliorant* sur le graphe ci-dessous. Montrer que le flot renvoyé par cet algorithme est saturé, mais pas forcément maximal. Conclure



que cette version "simplifiée" de l'algorithme peut échouer.

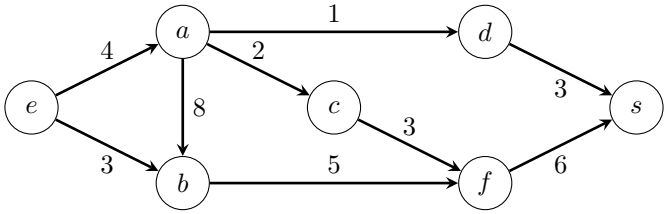
Dans l'algorithme de Ford-Fulkerson on s'autorise une définition plus large de "chemin améliorant" : c'est un chemin dans le **graphe des augmentations** $G_\varphi = (V_\varphi, E_\varphi, w)$ de $G = (V, E, \mathcal{C})$.

- G_φ a les même sommets que le graphe de départ, i.e. $V_\varphi = V$;
- Les arcs de G_φ sont $E_\varphi = \{a \in E \mid \varphi(a) < \mathcal{C}(a)\} \cup \{(v, u) \mid \varphi((u, v)) > 0\}$. Rappelons que les arcs (v, u) de E_φ pour lesquels $\varphi(u, v) > 0$ sont appelés arcs *arrières* et que les autres sont appelés arcs *avants*.
- La fonction de capacité w est définie par

$$w(u, v) := \begin{cases} \mathcal{C}(u, v) - \varphi(u, v) & \text{si } (u, v) \text{ est un arc avant}, \\ \varphi(v, u) & \text{si } (u, v) \text{ est un arc arrière et } (u, v) \notin E, \\ \varphi(v, u) + \mathcal{C}(u, v) & \text{si } (u, v) \text{ est un arc arrière et } (u, v) \in E. \end{cases}$$

Exercice 2 :

Appliquer l'algorithme de Ford-Fulkerson sur le graphe suivant, en donnant à chaque étape le graphe des augmentations.



Exercice 3 : covoiturage

Un groupe de n personnes $P = \{p_1, \ldots, p_n\}$ partagent leurs véhicules pour aller travailler pendant m jours. Le jour i, un sous-ensemble S_i de P utilisent un véhicule commun pour aller travailler. Le conducteur ce jour-là sera un élément de S_i. Etant donné P et les sous-ensembles $S_1, \ldots, S_m$, le problème est de choisir pour chaque i le conducteur du jour i de façon que la répartition soit équitable, c'est-à-dire qu'une personne j ne conduise pas plus de $\lceil \sum_{i:j \in S_i} \frac{1}{|S_i|} \rceil$ jours au total. Par exemple, s'il y a quatre personnes et trois jours avec $S_1 = \{p_1, p_2\}$, $S_2 = \{p_1, p_3, p_4\}$ et $S_2 = \{p_1, p_4\}$ alors la personne p_1 doit conduire au maximum $\lceil 1/2 + 1/3 + 1/2 \rceil = 2$ de ces trois jours alors que les autres conducteurs doivent conduire au maximum un de ces trois jours.

1. Modéliser ce problème comme un problème de flot maximal.

2. Monter qu'il existe toujours une répartition équitable.

Exercice 4 : Tournoi

On organise un tournoi de belotte au cours duquel toutes les équipes joueront plusieurs matchs, et même auront joué plusieurs fois contre la même équipe. L'équipe qui aura gagné le plus de matchs sera désignée championne.

A un moment donné du tournoi (avant la fin), on désire savoir quelle équipe ont encore une chance de gagner. Pour cela pour l'équipe numéro i, on dénote par :

- w_i le nombre de parties gagnées à ce moment du tournoi.
- $\forall j < i, r_{i,j}$ le nombre de parties restantes a jouer contre l'équipe j.

En utilisant une modélisation par un problème de flot, proposer un algorithme qui permette de décider si une équipe donnée peut encore gagner le tournoi ou pas.

Question Facultative : Peut-on adapter l'algorithme pour un championnat de tarot à 3 ? On note donc $\forall i < j < k, r_{i,j,k}$ le nombre de parties auxquels participent les joueurs i, j, k ?