Devoir maison n°2

Corrigé

⋆ ⋆ ⋆

1 Graphe de flot et flot maximal

Question 1 Il n'existe pas de tel flot : un débit de 13 entraînerait $f(s,1) = 4$ et $f(s,2) = 9$. Or, sachant que $f(2,4) \leqslant 8$, on aurait : $\sum_{v \in S} f(2,v) = f(2,s) + f(2,1) + f(2,4) \leqslant -1 + f(2,1) < 0$ car $f(2,1) \leqslant c(2,1) = 0$. On en déduit que la propriété de conservation n'est pas respectée.

Question 2 On a, pour $u \in S \setminus \{s,t\}$:

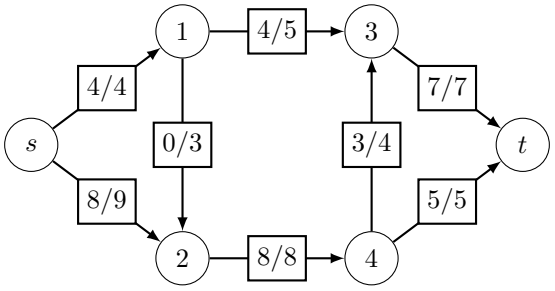
$$\begin{aligned}\sum_{v \in S} f(u,v) = 0 &\Leftrightarrow \sum_{v \in S, f(u,v)>0} f(u,v) + \sum_{v \in S, f(u,v)<0} f(u,v) = 0 \\ &\Leftrightarrow \phi_+(u) - \phi_-(u) = 0 \\ &\Leftrightarrow \phi(u) = 0\end{aligned}$$

Question 3 Comme s est une source, il est clair que $\phi_-(s) = 0$. On en déduit que $|f| = \phi_+(s) = \phi(s)$. Par ailleurs, on a :

$$\begin{aligned}0 = \sum_{(u,v) \in S^2, f(u,v)>0} f(u,v) - \sum_{(u,v) \in S^2, f(v,u)>0} f(v,u) &= \sum_{u \in S} \phi_+(u) - \sum_{u \in S} \phi_-(u) \\ &= \sum_{u \in S} \phi(u) \\ &= \phi(s) + \phi(t)\end{aligned}$$

La dernière équivalence découle de la question précédente. On en déduit le résultat.

Question 4 On propose le flot suivant :

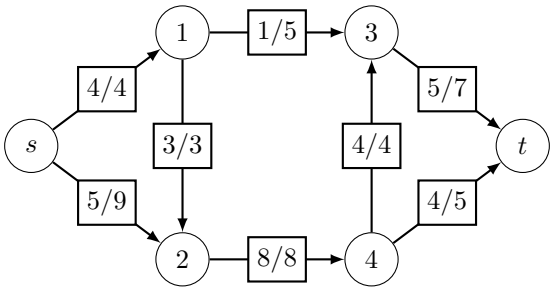


Il s'agit bien d'un flot maximal car $|f| = -\phi(t) = \sum_{u \in S} c(u,t) = 12$. Un flot de débit strictement supérieur entraînerait un non-respect de la capacité sur une des arêtes entrantes de t.

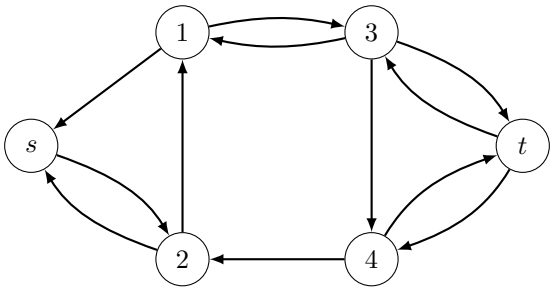
2 Algorithme de Ford-Fulkerson

Question 5 On peut lancer un parcours de graphe depuis s, en n'explorant que les arêtes non saturées, et en vérifiant si on atteint t. Pour récupérer le chemin de s à t, on peut utiliser un tableau de prédécesseur *pred* tel que $pred[u]$ est le prédécesseur de u dans le chemin de s à t. On peut alors reconstruire le chemin depuis t. La complexité d'un tel algorithme serait en $\mathcal{O}(|S| + |A|)$ si on peut accéder au flot et à la capacité en temps constant pour une arête donnée.

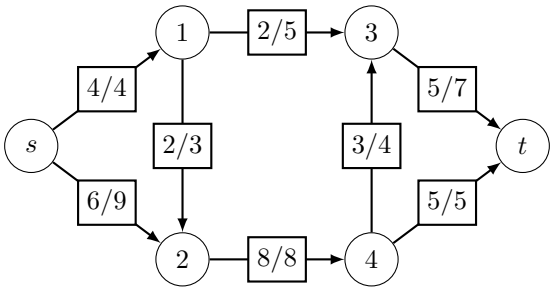
Question 6 Le chemin $(s, 1, 3, t)$ n'est pas saturé. Après saturation, on obtient le flot suivant. Avec ce flot, tous les chemins de s à t sont saturés, donc l'algorithme termine. Ce flot n'est clairement pas maximal car son débit est égal à 9 (et on a trouvé un flot de débit 12).



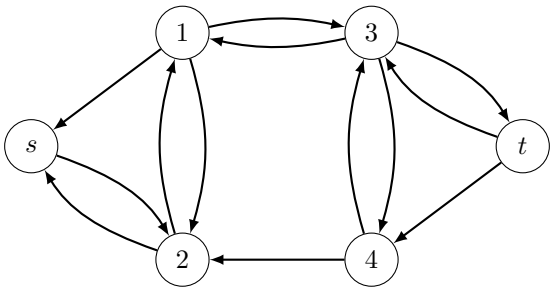
Question 7 On obtient le graphe résiduel suivant :



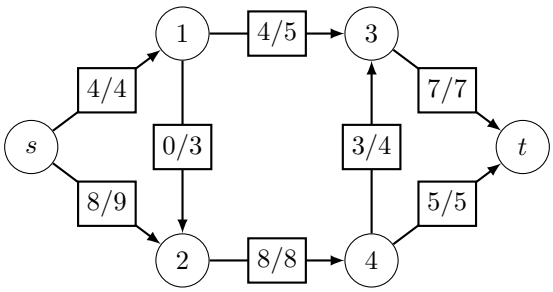
Question 8 Dans le graphe précédent, il existe deux chemins améliorant : $(s, 2, 1, 3, t)$ et $(s, 2, 1, 3, 4, t)$. Si on sature ce dernier, on obtient le flot :



Le graphe résiduel est alors :



Il ne reste qu'un seul chemin améliorant, $(s, 2, 1, 3, t)$. En saturant ce chemin, on obtient :



Il s'agit bien d'un flot maximal.

Question 9 Une version simple consiste à parcourir les matrices `f` et `c` et comparer les valeurs :

```
let compatible f (g, c) =
    let n = Array.length f in
    let b = ref true in
    for u = 0 to n - 1 do
        for v = 0 to n - 1 do
            if f.(u).(v) > c.(u).(v) then
               b := false
        done
    done;
    !b;;
```

La complexité est en $\mathcal{O}(|S|^2)$. On peut améliorer la complexité en vérifiant chaque liste d'adjacence :

```
let compatible f (g, c) =
    let n = Array.length f in
    let b = ref true in
    for u = 0 to n - 1 do
        let respect_capa v =
            if f.(u).(v) > c.(u).(v) then
               b := false in
        List.iter respect_capa g.(u)
    done;
    !b;;
```

On obtient une complexité en $\mathcal{O}(|S| + |A|)$.

Question 10 L'idée est de faire un parcours en profondeur depuis s. Plutôt que de créer explicitement le graphe résiduel, on peut faire la vérification qu'une arête existe dans G_f au cours de l'exploration. Pendant le parcours, on met à jour un tableau de prédécesseurs. La fonction `dfs : int -> int -> unit` prend en argument deux sommets u et v et, le cas échéant, marque u comme prédécesseur de v, puis relance un parcours depuis tous les voisins de v dans G_f.

Une fois le parcours terminé, on peut reconstruire le chemin en partant de t. On a posé -2 comme prédécesseur de s pour détecter le début du chemin.

```
let chemin_ameliorant f (g, c) =
    let n = Array.length f in
    let pred = Array.make n (-1) in
    let rec dfs u v =
        if pred.(v) = - 1 then begin
           pred.(v) <- u;
           let dfs_filtre w =
               if c.(v).(w) > f.(v).(w) then
                  dfs v w in
           List.iter dfs_filtre g.(v)
        end in
    dfs (-2) 0;
    if pred.(n - 1) = -1 then [] else begin
       let chemin = ref [n - 1] and u = ref (n - 1) in
       while pred.(!u) <> -2 do
           u := pred.(!u);
           chemin := !u :: !chemin
       done;
       !chemin
    end;;
```

La complexité du parcours est $\mathcal{O}(|S| + |A|)$. La reconstruction du chemin est $\mathcal{O}(|S|)$. On obtient bien la bonne complexité.

Question 11 On écrit une fonction `saturer : int list -> flot -> graphe_flot -> unit` qui prend en argument un chemin non vide, un flot f et un graphe de flot G et sature le chemin pour f et G. Pour ce faire, il faut trouver la capacité résiduelle $(c(a) - f(a))$ minimale le long du chemin. Ensuite, on augmente le flot de cette valeur le long de chaque arête du chemin.

```
let saturer chemin f (g, c) =
    let rec capa_resi = function
        | [] | [_]    -> failwith "chemin trop court"
        | [u; t]      -> c.(u).(t) - f.(u).(t)
        | u :: v :: q -> min (c.(u).(v) - f.(u).(v)) (capa_resi (v :: q)) in
    let cr = capa_resi chemin in
    let rec saturer_aux = function
        | [] | [_]    -> ()
        | u :: v :: q -> f.(u).(v) <- f.(u).(v) + cr;
                         f.(v).(u) <- f.(v).(u) - cr;
                         saturer_aux (v :: q) in
    saturer_aux chemin;;
```

Dès lors, on peut appliquer l'algorithme de Ford-Fulkerson : on crée un flot nul, et on cherche successivement des chemins améliorants.

```
let ford_fulkerson (g, c) =
    let n = Array.length g in
    let f = Array.make_matrix n n 0 in
    let chemin = ref (chemin_ameliorant f (g, c)) in
    while !chemin <> [] do
        saturer !chemin f (g, c);
        chemin := chemin_ameliorant f (g, c)
    done;
    f;;
```

Question 12 On remarque que la saturation d'un chemin améliorant augmente le débit du flot d'une valeur entière strictement positive. Le nombre de saturations effectuées est donc majoré par $|f^*|$ où f^* est un flot maximal. Sachant que chaque saturation et recherche de chemin améliorant ont une complexité en $\mathcal{O}(|S|+|A|)$, on en déduit la terminaison et une complexité totale en $\mathcal{O}(|S|^2 + |f^*|(|S| + |A|))$ (la création de la matrice `f` est nécessaire).

3 Flot maximal, coupe minimale

Question 13 On a $C(X) = c(s,2) + c(1,2) + c(3,t) = 15$.

Question 14 On a :

$$\begin{aligned}
|f| =& \phi(s) = \sum_{u\in X} \phi(u) = \sum_{u\in X} (\phi_+(u) - \phi_-(u)) \\
&= \sum_{u\in X} \left(\sum_{v\in X, f(u,v)>0} f(u,v) + \sum_{v\in \overline{X}, f(u,v)>0} f(u,v) - \sum_{v\in X, f(v,u)>0} f(v,u) - \sum_{v\in \overline{X}, f(v,u)>0} f(v,u) \right) \\
&= \sum_{u\in X} \left(\sum_{v\in \overline{X}, f(u,v)>0} f(u,v) - \sum_{v\in \overline{X}, f(v,u)>0} f(v,u) \right) \\
&= \sum_{(u,v)\in X\times\overline{X}} f(u,v)
\end{aligned}$$

En effet :

$$\sum_{u\in X} \left(\sum_{v\in X, f(u,v)>0} f(u,v) - \sum_{v\in X, f(u,v)<0} f(v,u) \right) = \sum_{(u,v)\in X^2, f(u,v)>0} f(u,v) - \sum_{(u,v)\in X^2, f(v,u)>0} f(v,u) = 0$$

L'inégalité est claire car le flot respecte la capacité.

Question 15 On montre par implications successives :

- 1. $\Rightarrow$ 2. : Par contraposée. S'il existe un chemin améliorant, l'action de saturer ce chemin augmente strictement la valeur du flot qui n'était donc pas maximal.
- 2. $\Rightarrow$ 3. : On pose X l'ensemble des sommets accessibles depuis s dans G_f. Comme il n'existe pas de chemin améliorant, $t \notin X$, donc X est une coupe. Par ailleurs, il n'existe aucune arête dans G_f de la forme (u, v), $u \in X$ et $v \in \overline{X}$ (sinon v est accessible depuis s). On en déduit que pour $(u, v) \in X \times \overline{X}$, $f(u, v) = c(u, v)$. Finalement, par la question précédente, $|f| = \sum_{(u,v)\in X\times\overline{X}} f(u, v) = \sum_{(u,v)\in X\times\overline{X}} c(u, v) = C(X)$.
- 3. $\Rightarrow$ 1. : soit f' un flot. Par la question précédente, $|f'| \leqslant C(X) = |f|$. On en déduit que f est un flot maximal.

Question 16 L'implication 2. $\Rightarrow$ 1. de la question précédente montre clairement la correction de l'algorithme.

4 Réduction du couplage maximum

Question 17 Soit $G = (S = X \cup Y, A)$ un graphe biparti. On pose $G' = (S \cup \{s, t\}, A', c, s, t)$ un graphe de flot où :

- $A' = \{(x, y) \in X \times Y \mid \{x, y\} \in A\} \cup \{(s, x) \mid x \in X\} \cup \{(y, t) \mid y \in Y\}$;
- pour tout $a \in A'$, $c(a) = 1$.

Si f est un flot de G', on définit $C_f = \{\{x, y\} \in A \mid f(x, y) = 1\}$. Alors f est un flot maximal pour G' si et seulement si C_f est un couplage maximum pour G. En effet, il est relativement facile à montrer que $(u_0, u_1, \ldots, u_k)$ est un chemin augmentant pour C_f si et seulement si $(s, u_0, u_1, \ldots, u_k, t)$ est un chemin améliorant pour f.

La construction du graphe G' se fait en temps $\mathcal{O}(|S|+|A|)$. Si on applique l'algorithme de Ford-Fulkerson, on a une complexité totale en $\mathcal{O}(|S|^2 + |S|(|S| + |A|)) = \mathcal{O}(|S|^2 + |S||A|)$, car le débit du flot est au plus $|S|$ (le nombre d'arêtes sortantes de s).

$\star\star\star$