

*Aucun document autorisé.*

*Une attention toute particulière sera portée à la présentation et à la rédaction de la copie.*

Dans tous les exercices, on désignera par  $V(G)$  et  $E(G)$  respectivement l'ensemble des sommets et l'ensemble des arêtes d'un graphe  $G$ . Les variables  $n$  et  $m$  désigneront respectivement le nombre de sommets et d'arêtes.

## 1 Plus court chemin

1.1) Rappeler à quelles conditions il est possible d'utiliser l'algorithme de Dijkstra pour calculer les plus courts chemins dans un graphe.

**Il faut et il suffit que les poids de tous les arcs soient positifs ou nuls.**

1.2) En utilisant l'algorithme de Dijkstra rappelé à la fin du document (Algorithme 1), trouver les plus courts chemins de  $s$  aux autres sommets du graphe  $G$  de la Figure 1. A chaque étape, on donnera le pivot utilisé et les modifications apportées aux distances à partir de  $s$ .

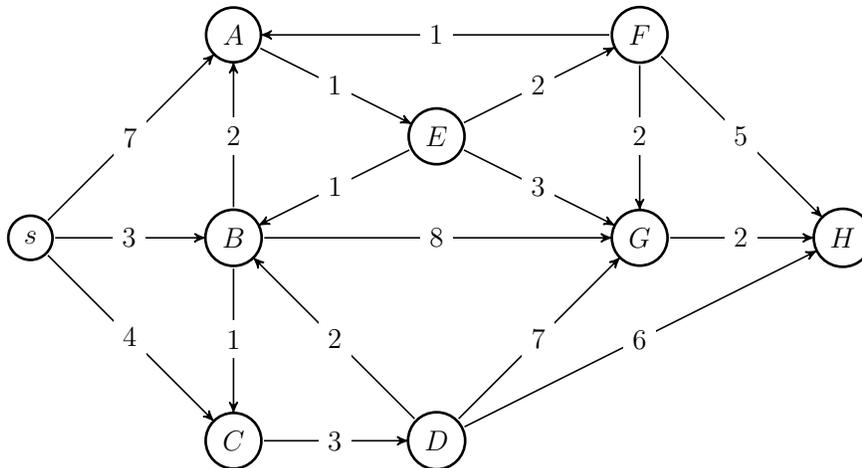


FIGURE 1 – Graphe G

Les lignes du tableau ci-dessous donnent à chaque étape la valeur du pivot dans la première colonne et les modifications de  $d[v]$  pour chaque sommet  $v$  (*Attention de ne pas*

mettre le poids de l'arc entre le pivot et le sommet  $v$ , mais bien la distance de  $s$  à  $v$ ). Les croix sont dans la colonne correspondant au sommet pivot pour signifier que celui-ci ne sera pas modifié par la suite.

<i>pivot</i>	$s$	$A$	$B$	$C$	$D$	$E$	$F$	$G$	$H$
	0	$\infty$							
$s$	X	7	3	4					
$B$		5	X					11	
$C$				X	7				
$A$		X				6			
$E$						X	8	9	
$D$					X				13
$F$							X		
$G$								X	11
$H$									X

1.3) Dessiner l'arborescence des plus courts chemins d'origine  $s$ .

L'arborescence des plus courts chemins est donné par la Figure 2 qui donne l'ensemble des arcs utilisés par les plus courts chemins dans  $G$  depuis  $s$ .

Pour rappel, cet ensemble forme une arborescence, c'est à dire un arbre orienté depuis une racine vers les feuilles, la racine étant ici le sommet  $s$ . Le prédécesseur d'un sommet  $v$  sera le sommet pivot lors de la dernière modification de  $d[v]$ .

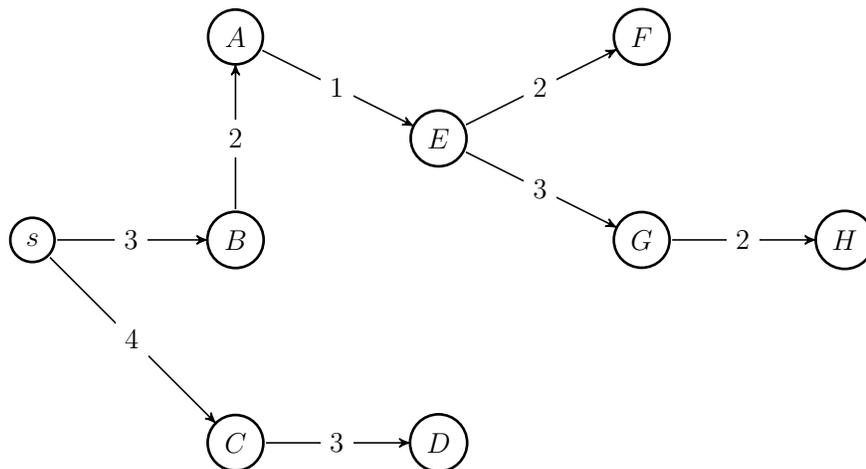


FIGURE 2 – Arborescence des plus courts chemins dans  $G$  depuis  $s$

## 2 Arbre de poids maximum

On considère le problème suivant :

un réseau routier reliant un ensemble de villes est représenté par un graphe  $G$  simple, non orienté, chaque ville étant représentée par un sommet et deux sommets  $a$  et  $b$  étant reliés par une arête s'il existe une route directe reliant la ville  $a$  à la ville  $b$ .

Chaque arête  $e$  est munie d'un poids  $h(e)$  donnant la hauteur maximum (en cm) d'un véhicule pouvant emprunter cette route. On souhaite calculer les itinéraires entre les villes maximisant la hauteur possible d'un véhicule.

On considère  $T_{max}$  l'arbre couvrant de  $G$  de poids maximum.

**2.1)** Montrer que si une arête  $e = \{u, v\}$  n'appartient pas à  $T_{max}$ , et si  $C_{uv}$  désigne la chaîne reliant  $u$  et  $v$  dans  $T_{max}$ , alors  $\forall e' \in C_{uv}, h(e) \leq h(e')$ .

Si  $e = \{u, v\}$  n'appartient pas à  $T_{max}$ , alors son ajout crée un cycle et un seul dans  $T_{max}$ , constitué de  $C_{uv} + e$ . S'il existe une arête  $e'$  dans  $C_{uv}$  de poids  $h(e') < h(e)$ , si on supprime  $e'$  de  $T_{max} + e$ , on obtient à nouveau un arbre  $T'$  de poids total  $\sum_{f \in T_{max}} h(f) + h(e) - h(e')$ . Si  $h(e') < h(e)$ , alors le poids total de  $T'$  sera strictement supérieur à celui de  $T_{max}$ , ce qui est contradictoire. Donc  $\forall e' \in C_{uv}, h(e') \geq h(e)$ .

**2.2)** Que pouvez-vous en conclure si les poids sont tous différents ?

Si tous les poids sont différents, il est impossible de remplacer une arête de  $T_{max}$  par une autre arête de même poids n'appartenant pas à  $T_{max}$ . Donc l'arbre de poids maximum sera unique.

**2.3)** Que faut-il modifier à l'algorithme de Prim, rappelé à la fin du document (Algorithme 2), pour calculer un arbre de poids maximum ?

On remplace

la ligne 3 par :

$cle(v) \leftarrow -\infty$

la ligne 8 par :

$u \leftarrow \text{EXTRAIRE\_MAX}(F)$

et la ligne 10 par :

**si**  $v \in F$  et  $w(u, v) > cle(v)$  **alors**

**2.4)** Dessiner l'ensemble des routes à emprunter dans le réseau routier de la Figure 3 pour maximiser la hauteur des camions utilisables pour tout transport entre deux villes.

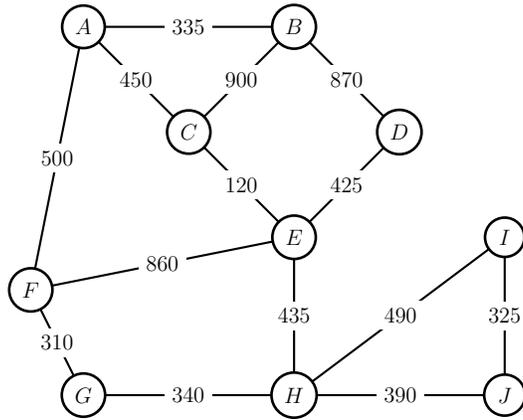


FIGURE 3 – Réseau routier

L'exécution de Prim modifié est donné par le tableau ci-dessous. Le graphe obtenu est donné par la Figure 4. Les valeurs mises dans le tableau sont les modifications de la clé, le père étant la valeur de  $u$  au moment de la modification. Le sommet choisi pour  $r$  est le sommet  $A$ .

$u$	$A$	$B$	$C$	$D$	$E$	$F$	$G$	$H$	$I$	$J$
	0	$-\infty$								
$A$	$X$	335	450			500				
$F$					860	$X$	310			
$E$				425	$X$			435		
$C$		900	$X$							
$B$		$X$		870						
$D$				$X$						
$H$							340	$X$	490	390
$I$									$X$	
$J$										$X$
$G$							$X$			

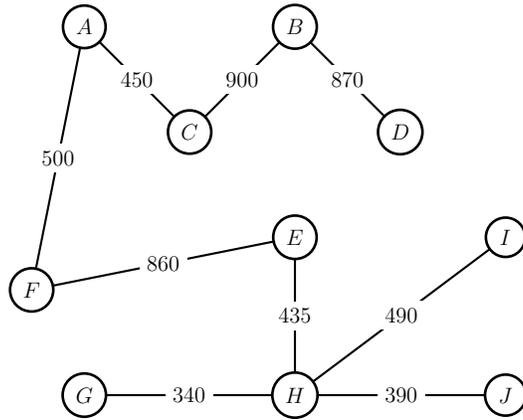


FIGURE 4 – Arbre de poids maximum

### 3 Flot Maximum

**3.1)** En utilisant l'algorithme de Ford-Fulkerson rappelé à la fin du document (Algorithme 3), trouver le flot maximum entre les sommets  $s$  et  $t$  du réseau de la Figure 5. Le flot possède déjà une valeur initiale donnée par le premier nombre porté sur les arcs, le second étant la capacité de l'arc. Par exemple, l'arc  $(s, A)$  possède un flot initial de 8 et une capacité de 13. *Vous devez améliorer ce flot existant, sans repartir d'un flot nul.*

On donnera pour chaque exécution de la procédure de marquage le chemin augmentant obtenu et l'augmentation correspondante.

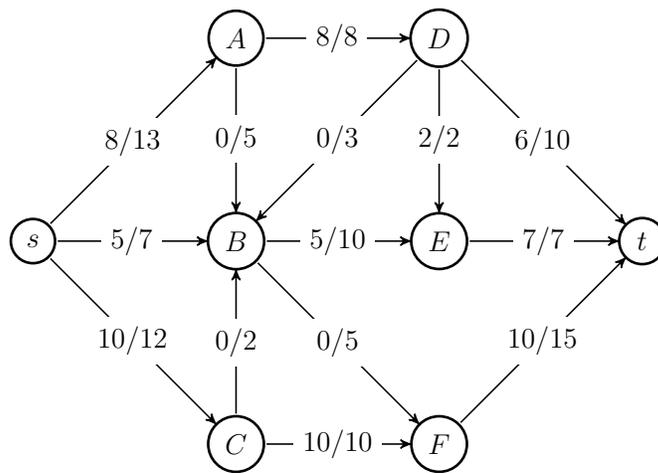


FIGURE 5 – Réseau

Les chemins augmentant sont les suivants (suivis de la valeur de leur augmentation respective :

sABEDt : +2

sABFt : + 3

sBFt : +2

ce qui nous donne un flot d'une valeur de  $f(sA) + f(sB) + f(sC) = 13 + 7 + 10 = 30$ .

**3.2)** Donner la coupe minimum obtenue par l'algorithme de Ford-Fulkerson, sa valeur et redessiner le graphe avec le flot maximum.

Les sommets visités lors du dernier marquage (donc l'ensemble  $Y$  qui nous donne la coupe minimum) sont  $Y = \{s, A, B, C, E\}$ .  $val(Y) = c(AD) + c(Et) + c(BF) + c(CF) = 8 + 7 + 5 + 10 = 30$ . Donc le flot est bien maximum avec une coupe minimum de même valeur.

Le flot maximum est donné par la Figure 6.

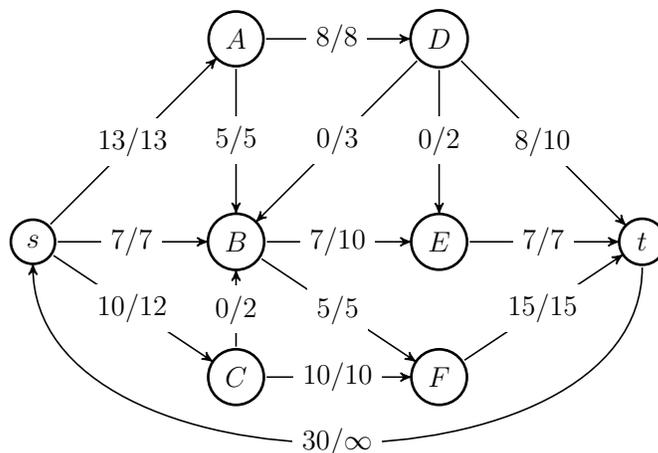


FIGURE 6 – Flot maximum

## 4 Interrupteurs

Soit  $G$  un graphe simple non orienté à  $n > 1$  sommets. Soit  $s$  un sommet de  $G$ , appelé *sommet initial*. Sur chaque sommet se trouve un interrupteur. Dans ce problème, on se déplace sur le graphe en changeant la position des interrupteurs. Si on se place au sommet  $s_1$  et qu'on franchit l'arête  $\{s_1, s_2\}$ , l'interrupteur de  $s_2$  change de valeur. Plus formellement, on modélise la situation comme suit.

Une *configuration* est la donnée d'un couple  $(t, in)$  où  $t$  est un sommet (le sommet courant qui représente où on est dans le graphe) et  $in$  un tableau représentant l'état des

interrupteurs (0 si l'interrupteur est éteint, 1 sinon). On peut passer de la configuration  $(s_1, \text{in}_1)$  à  $(s_2, \text{in}_2)$  si et seulement si  $\{s_1, s_2\} \in E(G)$  et pour tout sommet  $t$  de  $G$ , on a :

$$\text{in}_2[t] = \begin{cases} 1 - \text{in}_1[t] & \text{si } t = s_2 \\ \text{in}_1[t] & \text{sinon} \end{cases}$$

Dans ce cas, on notera  $(s_1, \text{in}_1) \rightarrow (s_2, \text{in}_2)$ .

Une configuration  $(t, \text{in}')$  est *accessible* depuis  $(s, \text{in})$  si on peut trouver un entier  $k \geq 1$  et des configurations telles que :

$$(s, \text{in}) = (s_0, \text{in}_0) \rightarrow (s_1, \text{in}_1) \rightarrow \cdots \rightarrow (s_k, \text{in}_k) = (t, \text{in}')$$

Une configuration  $\mathcal{C}$  est *résoluble* s'il existe un sommet  $t$  tel que  $(t, [1; 1; \dots; 1])$  est accessible depuis  $\mathcal{C}$ . Le but de cet exercice est de résoudre le problème suivant : étant donné un graphe  $G$ , on veut pouvoir déterminer si une configuration  $\mathcal{C}$  est résoluble. Autrement dit, on veut trouver un chemin dans  $G$  qui commence en un sommet fixé et qui allume tous les interrupteurs.

Par exemple, si on considère le graphe  $P3$  si dessous, la configuration  $(0, [1; 1; 0])$  est résoluble car :

$$(0, [1; 1; 0]) \rightarrow (1, [1; 0; 0]) \rightarrow (2, [1; 0; 1]) \rightarrow (1, [1; 1; 1])$$

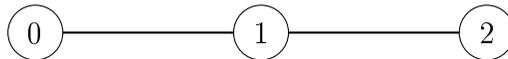
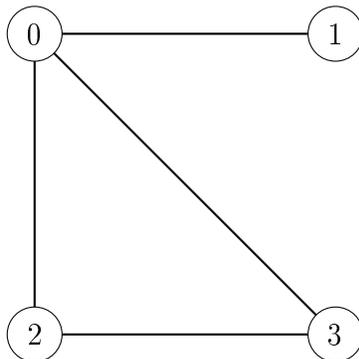


FIGURE 7 – Graphe  $P3$

1. Soit  $\mathcal{C}$  la configuration  $(0, [1; 0; 1; 0])$ .
  - (a) Donner les configurations  $(s, \text{in})$  telles que  $\mathcal{C} \rightarrow (s, \text{in})$  dans le graphe suivant :



On a :

$$(0, [1; 0; 1; 0]) \rightarrow (1, [1, 1, 1, 0])$$

$$(0, [1; 0; 1; 0]) \rightarrow (2, [1, 0, 0, 0])$$

$$(0, [1; 0; 1; 0]) \rightarrow (3, [1, 0, 1, 1])$$

(b)  $\mathcal{C}$  est-elle résoluble ?

Oui,  $\mathcal{C}$  est résoluble avec le parcours suivant :

$$(0, [1; 0; 1; 0]) \rightarrow (1, [1, 1, 1, 0]) \rightarrow (0, [0, 1, 1, 0]) \rightarrow (3, [0, 1, 1, 1]) \rightarrow (0, [1, 1, 1, 1])$$

2. Dans cette question, on s'intéresse aux configurations résolubles d'un arbre. On va montrer par récurrence sur  $n \geq 2$  la propriété  $P_n$  suivante : "Pour tout arbre  $A$  à  $n$  sommets et toute configuration  $(s, \text{in})$ ,  $(s, \text{in})$  est résoluble".

(a) Montrer que la proposition  $P_2$  est vraie, c'est à dire que toute configuration dans un arbre à deux sommets est résoluble.

Le seul arbre à deux sommets est le graphe composé de deux sommets reliés par une arête. Les deux sommets seront numérotés respectivement 0 et 1.

On considère d'abord les 4 configurations possibles en partant de 0.

$$(0, [0, 0]) \rightarrow (1, [0, 1]) \rightarrow (0, [1, 1])$$

$$(0, [0, 1]) \rightarrow (1, [0, 0]) \rightarrow (0, [1, 0]) \rightarrow (1, [1, 1])$$

$$(0, [1, 0]) \rightarrow (1, [1, 1])$$

$$(0, [1, 1])$$

Pour chacune des 4 configurations, on a un parcours montrant sa résolution. D'autre part, si on considère les 4 configurations en partant de 1, après la première étape, on arrivera à une configuration partant de 0. Comme toutes les configurations partant de 0 sont résolubles, il en sera de même pour celles partant de 1.

(b) Soit  $n > 2$  un entier tel que  $P_{n-1}$  est vraie. Soit  $A$  un arbre à  $n$  sommets et  $(s, \text{in})$  une configuration.

On rappelle que dans tout arbre ayant au moins deux sommets, il existe au moins deux feuilles distinctes. On peut donc toujours supposer qu'il existe une feuille  $f$  avec  $f \neq s$ .

Montrer qu'il existe une configuration  $(s, \text{in}')$  accessible depuis  $(s, \text{in})$  telle que  $\text{in}'[f] = 1$ . (On pourra distinguer les cas  $\text{in}[f] = 0$  et  $\text{in}[f] = 1$  et exhiber un chemin bien choisi.)

Si  $\text{in}[f] = 0$ , soit  $s, v_1, \dots, v_k, f$  la chaîne reliant  $s$  à  $f$ . On parcourt cette chaîne de  $s$  à  $f$ , puis de  $f$  à  $s$ . On peut remarquer que chaque sommet situé strictement entre  $s$  et  $f$  a été parcouru deux fois. On obtient ainsi une configuration  $(s, \text{in}')$  telle que  $\forall v \notin \{s, f\}, \text{in}'[v] = \text{in}[v], \text{in}'[s] = 1 - \text{in}[s]$  et  $\text{in}'[f] = 1 - \text{in}[f] = 1$ .

Si  $\text{in}[f] = 1$ , alors on prend  $\text{in}' = \text{in}$ .

Dans les deux cas, on a bien  $\text{in}'[f] = 1$ .

(c) Conclure la récurrence.

Si  $P_{n-1}$  est vrai, soit  $A$  un arbre à  $n$  sommets et  $(s, \text{in})$  une configuration de  $A$ . Soit  $f$  une feuille de  $A$  différente de  $s$ . Il suffit d'appliquer la solution de la question précédente pour avoir  $f$  allumé. On considère ensuite l'arbre  $T' = T \setminus f$  et la configuration  $(s, \text{in}')$  obtenue après l'allumage de  $f$  dans  $T'$ . D'après  $P_{n-1}$  cette configuration est résoluble et sa résolution n'utilisant pas le sommet  $f$ , on a donc que  $(s, \text{in})$  est résoluble et donc que  $P_{n-1} \Rightarrow P_n$  est vraie. Comme  $P_2$  est vraie, pour tout  $n \geq 2$ ,  $P_n$  est vraie.

3. En déduire une condition nécessaire et suffisante sur un graphe à  $n \geq 2$  sommets pour que toutes ses configurations soient résolubles.

Pour que toute configuration d'un graphe  $G$  soit résoluble, il est nécessaire que  $G$  soit connexe. En effet, toute configuration ayant un sommet éteint dans un autre composante connexe que celle du sommet initial ne sera pas résoluble.

Pour qu'un graphe  $G$  ayant au moins deux sommets aient toutes ses configurations résolubles, il suffit que  $G$  possède un arbre couvrant, donc que  $G$  soit connexe.

Par conséquent, toutes les configurations d'un graphe  $G$  sont résolubles si et seulement si  $G$  est connexe.

## Algorithmes (Rappel)

---

**Algorithme 1** Dijkstra( $G, w, s$ )

---

```
1: pour tout  $v$  de  $V(G)$  faire
2:    $d(v) \leftarrow \infty$ 
3:    $pere(v) \leftarrow NIL$ 
4:    $couleur(v) \leftarrow BLANC$ 
5: fin pour
6:  $d(s) \leftarrow 0$ 
7:  $F \leftarrow FILE\_PRIORITE(\{s\}, d)$ 
8: tant que  $F \neq \emptyset$  faire
9:    $pivot \leftarrow EXTRAIRE\_MIN(F)$ 
10:  pour tout  $e = (pivot, v)$  arc sortant de  $pivot$  faire
11:    si  $couleur(v) = BLANC$  alors
12:      si  $d(v) = \infty$  alors
13:        INSERER( $F, v$ )
14:      fin si
15:      si  $d[v] > d[pivot] + w(e)$  alors
16:         $d[v] \leftarrow d[pivot] + w(e)$ 
17:         $pere[v] \leftarrow pivot$ 
18:      fin si
19:    fin si
20:  fin pour
21:   $couleur[pivot] \leftarrow NOIR$ 
22: fin tant que
```

---

---

**Algorithme 2** Prim( $G, w$ )

---

```
1:  $F \leftarrow FILE\_PRIORITE(V(G), cle)$ 
2: pour tout  $v$  de  $V(G)$  faire
3:    $cle(v) \leftarrow \infty$ 
4: fin pour
5:  $cle(r) \leftarrow 0$ 
6:  $pere(r) \leftarrow NIL$ 
7: tant que  $F \neq \emptyset$  faire
8:    $u \leftarrow EXTRAIRE\_MIN(F)$ 
9:   pour tout  $v \in Adj(u)$  faire
10:    si  $v \in F$  et  $w(u, v) < cle(v)$  alors
11:       $pere(v) \leftarrow u$ 
12:       $cle(v) \leftarrow w(u, v)$ 
13:    fin si
14:  fin pour
15: fin tant que
```

---

Dans l'Algorithme 3 (FlotMax), le paramètre  $c$  désigne les capacités du graphe  $G$ ,  $s$  la source et  $t$  la destination du flot  $f$  calculé.  $I(e)$  et  $T(e)$  désignent respectivement le sommet initial et le sommet terminal d'un arc  $e$ .

---

**Algorithme 3** FlotMax( $G, c, s, t$ )

---

```
1: pour tout  $e$  de  $E(G)$  faire
2:    $f[e] \leftarrow 0$ 
3: fin pour
4: répéter
5:   Marquage( $G, c, f, s, t$ )
6:   si  $t \in Y$  alors
7:      $v \leftarrow t$ 
8:      $C^+ \leftarrow \{(t, s)\}$ 
9:      $C^- \leftarrow \emptyset$ 
10:    tant que  $v \neq s$  faire
11:       $e \leftarrow A[v]$ 
12:      si  $v = T[e]$  alors
13:         $C^+ \leftarrow C^+ \cup \{e\}$ 
14:         $v \leftarrow I[e]$ 
15:      sinon
16:         $C^- \leftarrow C^- \cup \{e\}$ 
17:         $v \leftarrow T[e]$ 
18:      fin si
19:    fin tant que
20:  fin si
21:  pour tout  $e \in C^+$  faire
22:     $f(e) \leftarrow f(e) + \delta[t]$ 
23:  fin pour
24:  pour tout  $e \in C^-$  faire
25:     $f(e) \leftarrow f(e) - \delta[t]$ 
26:  fin pour
27: jusqu'à  $t \notin Y$ 
```

---

---

**Algorithme 4** Marquage( $G, c, f, s, t$ )

---

```
1:  $Y \leftarrow \{s\}$ 
2:  $\delta(s) \leftarrow +\infty$ 
3:  $Max \leftarrow \text{faux}$ 
4: tant que  $t \notin Y$  et  $Max = \text{faux}$  faire
5:   si il existe  $e = (u, v)$  avec  $u \in Y, v \notin Y, f(e) < c(e)$  alors
6:      $Y \leftarrow Y \cup \{v\}$ 
7:      $A[v] \leftarrow e$ 
8:      $\delta[v] \leftarrow \min(\delta[u], c(e) - f(e))$ 
9:   sinon
10:    si il existe  $e = (u, v)$  avec  $v \in Y, u \notin Y, f(e) > 0$  alors
11:       $Y \leftarrow Y \cup \{u\}$ 
12:       $A[u] \leftarrow e$ 
13:       $\delta[u] \leftarrow \min(\delta[v], f(e))$ 
14:    sinon
15:       $Max \leftarrow \text{vrai}$ 
16:    fin si
17:  fin si
18: fin tant que
```

---