

*Aucun document autorisé.*

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

## Partie : Recherche Opérationnelle

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 Algorithme de Prim

1.1) En utilisant l'algorithme de Prim rappelé à la fin du document (Algorithme 1), trouver l'arbre de poids minimum dans le graphe de la Figure 1, en partant du sommet  $A$ . On donnera pour chaque étape le nom du sommet extrait de la file de priorité et les changements de valeur pour la fonction *clé*.

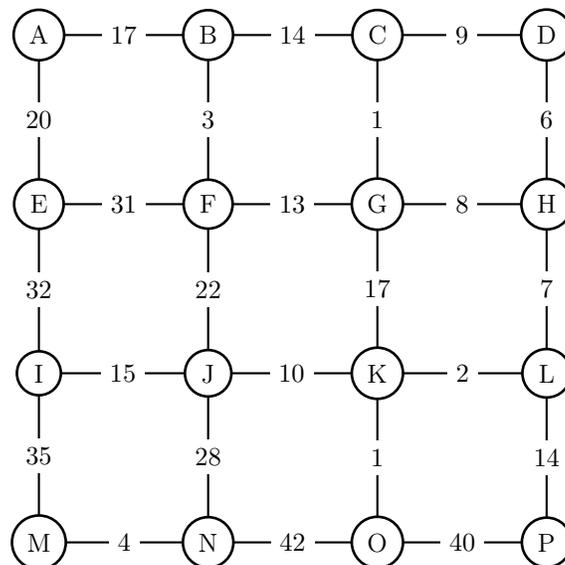


FIGURE 1 – Graphe

## 2 Plus courts chemins

Les algorithmes de Dijkstra, Bellman et Ford sont rappelés à la fin du document : Algorithmes 2, 3 et 4. Pour les trois graphes  $G1$ ,  $G2$  et  $G3$  des Figures 2, 3 et 4, calculer les plus courts chemins entre le sommet  $s$  et les autres sommets du graphe. Pour chacun des graphes, on rappellera les conditions nécessaires à l'utilisation de l'algorithme choisi.

En cas de choix possible, traiter les sommets dans l'ordre  $s, A, B, C, D, E$ . Si vous utilisez Dijkstra, donner pour chaque itération la valeur du *pivot* et les modifications de la fonction  $d$ .

Pour Bellman, donner pour chaque itération la valeur de la tête de la file  $F$  et les modifications de  $d$  et  $npred$ .

Pour Ford, pour chaque valeur de  $i$ , donner les modifications de  $d$  et de  $pere$ . Si le résultat est *FAUX*, donner le circuit négatif l'ayant engendré. Sinon, donner l'arborescence des plus courts chemins.

Dans les trois cas, dessiner l'arborescence des plus courts chemins si elles existent.

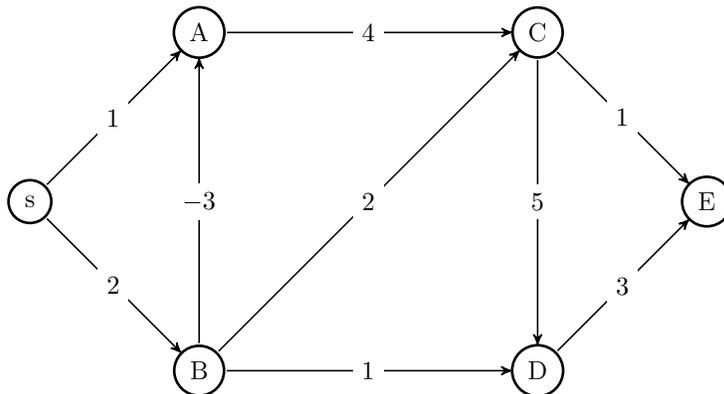


FIGURE 2 – Graphe  $G1$

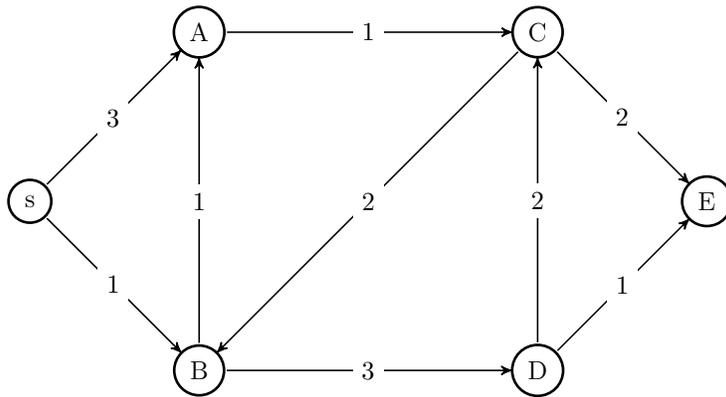


FIGURE 3 – Graphe  $G_2$

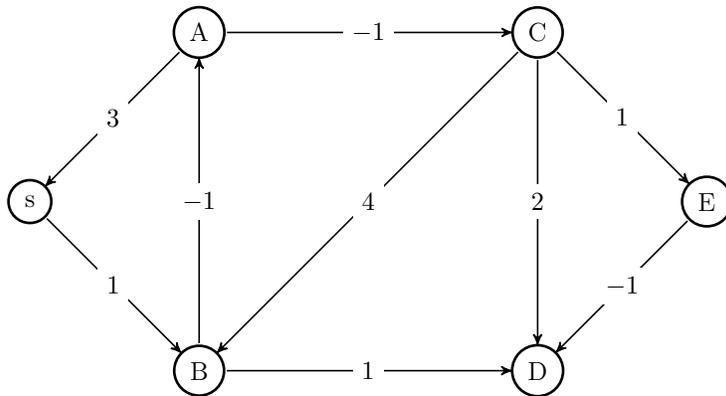


FIGURE 4 – Graphe  $G_3$

### 3 Flot Maximum

3.1) En utilisant l'algorithme de Ford-Fulkerson rappelé à la fin du document (Algorithmes 5 et 6), 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 1 et une capacité de 4. 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.

3.2) Donner la coupe minimum correspondante et redessiner le graphe avec le flot maximum.

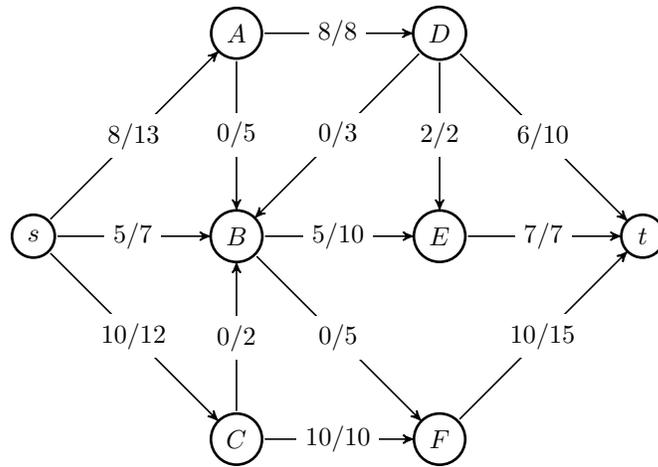


FIGURE 5 – Réseau

#### 4 Recherche d'un chemin hamiltonien dans un tournoi

Un *tournoi* est un graphe simple complet orienté. Autrement dit, si  $T$  est un tournoi et  $u$  et  $v$  deux sommets de  $T$ , alors  $(u, v) \in E(T) \Leftrightarrow (v, u) \notin E(T)$ .

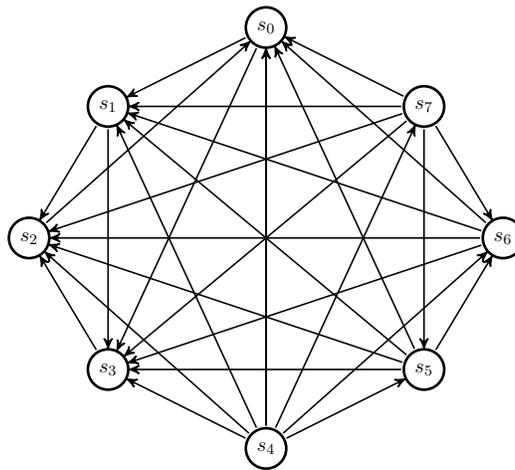


FIGURE 6 – Exemple de tournoi à 8 sommets

Un *chemin hamiltonien* est un chemin qui passe par tous les sommets du

graphe une et une seule fois. Tout tournoi possède un chemin hamiltonien. Par exemple, pour le tournoi de la Figure 6 :  $s_4, s_7, s_5, s_6, s_0, s_1, s_3, s_2$ .

4.1) Appliquer l'algorithme de parcours en profondeur rappelé à la fin du document (Algorithmes 7 et 8) au tournoi  $T$  de la Figure 6. On conviendra que dans les listes d'adjacence, les sommets sont rangés dans l'ordre lexicographique. Pour chaque sommet, donner les valeurs  $d$ ,  $f$ ,  $pere$  retournées par l'algorithme en respectant l'ordre de visite des sommets. Ordonner les sommets de  $T$  dans l'ordre inverse de leur fin de visite. Que remarquez-vous ?

4.2) Lorsque l'on range les sommets dans l'ordre inverse de leur fin de visite, quels sont les types d'arcs (parmi *liaison*, *retour*, *avant*, *transverse*) reliant deux sommets consécutifs ? Montrer que si les sommets sont rangés dans l'ordre inverse de leur fin de visite, alors si le sommet  $v$  suit immédiatement le sommet  $u$  dans cet ordre, l'arc entre ces deux sommets est orienté de  $u$  vers  $v$ .

4.3) Modifier l'algorithme de parcours en profondeur pour afficher les sommets dans un ordre induisant un chemin hamiltonien. Quelle est la complexité de l'algorithme obtenu ?

## Algorithmes

---

### Algorithme 1 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 les algorithmes de Dijkstra (2), Bellman (3) et Ford (4), on a :

$G$  : graphe orienté  
 $w : E(G) \rightarrow \mathbb{R}$   
 $s$  : source de  $G$

---

**Algorithme 2** 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 3** Bellman( $G, w, s$ )

---

```
1: pour tout  $v$  de  $V(G)$  faire
2:    $d[v] \leftarrow \infty$ 
3:    $pere[v] \leftarrow NIL$ 
4:    $npred[v] \leftarrow deg^-[v]$ 
5:   si  $npred(v) = 0$  alors
6:     INSERER_FILE( $F, v$ )
7:   fin si
8: fin pour
9:  $d[s] \leftarrow 0$ 
10: tant que  $F$  non vide faire
11:    $u \leftarrow TETE\_FILE(F)$ 
12:   DEFILER( $F$ )
13:   pour  $v \in Adj(u)$  faire
14:     si  $d[v] > d[u] + w(u, v)$  alors
15:        $d[v] \leftarrow d[u] + w[u, v]$ 
16:        $pere[v] \leftarrow u$ 
17:     fin si
18:      $npred(v) \leftarrow npred[v] - 1$ 
19:     si  $npred(v) = 0$  alors
20:       INSERER_FILE( $F, v$ )
21:     fin si
22:   fin pour
23: fin tant que
```

---

L'algorithme de Ford renvoie VRAI si le graphe  $G$  est sans circuit négatif, FAUX sinon.

---

**Algorithme 4** Ford( $G, w, s$ )

---

```
1: pour tout  $v$  de  $V(G)$  faire
2:    $d[v] \leftarrow \infty$ 
3:    $pere[v] \leftarrow NIL$ 
4: fin pour
5:  $d[s] \leftarrow 0$ 
6: pour  $i$  de 1 à  $n - 1$  faire
7:   pour tout arc  $e = (u, v) \in E(G)$  faire
8:     si  $d[v] > d[u] + w(u, v)$  alors
9:        $d[v] \leftarrow d[u] + w[u, v]$ 
10:       $pere[v] \leftarrow u$ 
11:    fin si
12:  fin pour
13: fin pour
14: pour tout arc  $e = (u, v) \in E(G)$  faire
15:   si  $d[v] > d[u] + w(u, v)$  alors
16:     retourner FAUX
17:   fin si
18: fin pour
19: retourner VRAI
```

---

Dans l'Algorithme 5 (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 5** 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 6** 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
```

---

---

**Algorithme 7** Parcours en profondeur PP( $G$ )

---

```
1: pour tout  $v \in V(G)$  faire  
2:    $couleur(v) \leftarrow BLANC$   
3:    $pere(v) \leftarrow NIL$   
4: fin pour  
5:  $temps \leftarrow 0$   
6: pour tout  $v \in V(G)$  faire  
7:   si  $couleur(v) = BLANC$  alors  
8:      $VisiterPP(v)$   
9:   fin si  
10: fin pour
```

---

---

**Algorithme 8** VisiterPP( $v$ )

---

```
1:  $d(v) \leftarrow temps \leftarrow temps + 1$   
2:  $couleur(v) \leftarrow GRIS$   
3: pour tout  $w \in Adj(v)$  faire  
4:   si  $couleur(w) = BLANC$  alors  
5:      $pere(w) \leftarrow v$   
6:      $VisiterPP(w)$   
7:   fin si  
8: fin pour  
9:  $couleur(v) \leftarrow NOIR$   
10:  $f(v) \leftarrow temps \leftarrow temps + 1$ 
```

---