

# Modélisation, Optimisation, Complexité et Algorithmes

KRAFT Benjamin

Session 2005-2006

# Table des matières

<b>1 Définitions sur les graphes</b>	<b>6</b>
1.1 Graphes orientés	7
1.1.1 Définition d'un graphe orienté	7
1.1.2 Exemple de graphe orienté	7
1.1.3 Définition d'un chemin	7
1.1.4 Exemple de chemin	7
1.1.5 Définition de la longueur d'un chemin	7
1.1.6 Exemple de la longueur d'un chemin	7
1.1.7 Définition d'un circuit	7
1.1.8 Exemple de circuit	7
1.1.9 Définition d'une boucle	8
1.1.10 Exemple de boucle	8
1.1.11 Définition d'un chemin élémentaire	8
1.1.12 Exemple de chemin élémentaire	8
1.1.13 Définition d'un chemin simple	8
1.1.14 Exemple de chemin simple	8
1.1.15 Définition d'un graphe complet	8
1.1.16 Exemple de graphe complet	8
1.2 Graphes non-orientés	9
1.2.1 Définition d'un graphe non-orienté	9
1.2.2 Exemple de graphe non-orienté	9
1.2.3 Définition d'une chaîne	9
1.2.4 Exemple de chaîne	9
1.2.5 Définition d'un cycle	9
1.2.6 Exemple de cycle	9
1.2.7 Définition d'un graphe complet	10
1.2.8 Exemple de graphe complet	10
1.2.9 Définition d'un graphe simple	11
1.2.10 Exemple de graphe simple	11
1.3 Graphes connexes & fortement connexes	12
1.3.1 Définition d'un graphe connexe	12
1.3.2 Exemple de graphe connexe	12
1.3.3 Définition d'un graphe fortement connexe	12
1.3.4 Exemple de graphe fortement connexe	12
1.4 Chemins et circuits Hamiltoniens	13
1.4.1 Définition d'un chemin et d'un circuit Hamiltonien	13
1.4.2 Exemple de chemin et de circuit Hamiltonien	13
1.5 Degré et demi-degré d'un sommet	14
1.5.1 Graphes orientés	14
1.5.1.1 Définition du demi-degré extérieur d'un sommet	14
1.5.1.2 Exemple de demi-degré extérieur d'un sommet	14
1.5.1.3 Définition du demi-degré intérieur d'un sommet	14
1.5.1.4 Exemple de demi-degré intérieur d'un sommet	14
1.5.1.5 Définition du degré d'un sommet	14

1.5.1.6	Exemple de degré d'un sommet . . . . .	14
1.5.2	Graphes non-orientés . . . . .	15
1.5.2.1	Définition du degré d'un sommet . . . . .	15
1.5.2.2	Exemple de degré d'un sommet . . . . .	15
1.6	Cocycle d'un graphe . . . . .	16
1.7	Application multivoque d'un graphe . . . . .	17
1.8	Représentation des graphes . . . . .	18
1.8.1	Matrice d'adjacence . . . . .	18
1.8.1.1	Graphes orientés . . . . .	18
1.8.1.2	Graphes non-orientés . . . . .	18
1.8.2	Liste d'adjacence . . . . .	19
1.8.2.1	Graphes orientés . . . . .	19
1.8.2.2	Graphes non-orientés . . . . .	19
1.8.3	Matrice d'incidence . . . . .	20
1.8.3.1	Graphes orientés . . . . .	20
1.8.3.2	Graphes non-orientés . . . . .	21
<b>2</b>	<b>Méthodes de parcours de graphe</b>	<b>22</b>
2.1	Structures de données . . . . .	23
2.1.1	Liste linéaire . . . . .	23
2.1.1.1	Définition . . . . .	23
2.1.1.2	Opérations . . . . .	23
2.1.1.3	Implémentation d'une liste . . . . .	23
2.1.2	Pile . . . . .	24
2.1.2.1	Définition . . . . .	24
2.1.2.2	Opérations . . . . .	24
2.1.3	File . . . . .	24
2.1.3.1	Définition . . . . .	24
2.1.3.2	Opérations . . . . .	24
2.2	Parcours en profondeur d'un graphe . . . . .	25
2.2.1	Principe . . . . .	25
2.2.2	Algorithme . . . . .	25
2.2.3	Vérification de l'algorithme . . . . .	26
2.2.4	Forêt courante du parcours en profondeur . . . . .	27
2.3	Parcours en largeur d'un graphe . . . . .	28
2.3.1	Méthode de parcours . . . . .	28
2.3.2	Principe . . . . .	28
2.3.3	Algorithme . . . . .	29
2.3.4	Vérification de l'algorithme . . . . .	29
<b>3</b>	<b>Représentation matricielle d'un graphe</b>	<b>31</b>
3.1	Matrice booléenne - matrice aux arcs d'un graphe . . . . .	32
3.2	Existence et dénombrement de chemins dans un graphe . . . . .	33
3.2.1	Définition . . . . .	33
3.2.2	Matrice booléenne . . . . .	33
3.2.3	Matrice aux arcs . . . . .	33
3.2.4	Chemins élémentaires - chemins hamiltoniens . . . . .	34
3.2.4.1	Définition . . . . .	34
3.3	Fermeture transitive d'un graphe . . . . .	36
3.3.1	Définition . . . . .	36
3.3.2	Exemple . . . . .	36
3.3.3	Algorithme de Warshall . . . . .	37
3.3.4	Méthode matricielle . . . . .	37

<b>4 Chemins optimaux dans un graphe</b>	<b>39</b>
4.1 Introduction . . . . .	40
4.2 Algorithme de Dijkstra . . . . .	41
4.3 Algorithme de Ford . . . . .	43
4.4 Algorithme de Floyd . . . . .	45
<b>5 Problèmes d'ordonnancement</b>	<b>47</b>
5.1 Introduction . . . . .	48
5.2 Méthode MPM . . . . .	49
5.3 Méthode PERT . . . . .	51
<b>6 Flot maximal sur un réseau de transport</b>	<b>52</b>
6.1 Définitions . . . . .	53
6.1.1 Réseau de transport . . . . .	53
6.1.2 Flot dans un réseau de transport . . . . .	53
6.2 Algorithme de Ford-Fulkerson . . . . .	55
6.2.1 Graphe d'écart . . . . .	55
6.2.2 Énoncé de l'algorithme . . . . .	55
6.2.3 Application de l'algorithme . . . . .	56
6.3 Propriétés sur les flots . . . . .	57
6.3.1 Flot complet . . . . .	57
6.3.2 Coupe . . . . .	57
6.3.3 Capacité d'une coupe . . . . .	57
6.3.4 Flot maximal . . . . .	57
6.4 Problème particulier de transport . . . . .	58
6.4.1 Énoncé du problème . . . . .	58
6.4.2 Algorithme du Stepping Stone . . . . .	59
<b>7 Arbre couvrant de coût minimum</b>	<b>63</b>
7.1 Définitions . . . . .	64
7.1.1 Arbre . . . . .	64
7.1.2 Arbre couvrant de coût minimum . . . . .	64
7.2 Algorithme de Kruskal . . . . .	65
7.3 Algorithme de Prim . . . . .	66
<b>8 Complexité des algorithmes</b>	<b>67</b>
8.1 Temps d'exécution d'un programme et complexité d'un algorithme . . . . .	68
8.1.1 Exemple 1 . . . . .	68
8.1.1.1 Algorithme . . . . .	68
8.1.1.2 But . . . . .	68
8.1.1.3 Découpage en nombre d'instructions de base de l'algorithme . . . . .	68
8.1.1.4 Somme des instructions . . . . .	69
8.1.1.5 Conclusion . . . . .	69
8.1.2 Exemple 2 . . . . .	69
8.1.2.1 Algorithme . . . . .	70
8.1.2.2 Évaluation du nombre d'étapes de l'algorithme . . . . .	70
8.1.2.3 Calcul du nombre d'étapes exécutées par l'algorithme en ne tenant compte que de l'étape élémentaire . . . . .	71
8.2 Ordre de grandeur d'un algorithme . . . . .	73
8.2.1 Reprise de l'exemple de recherche de l'indice du plus petit élément d'un tableau . .	73
8.2.2 Reprise du tri par permutation . . . . .	74
8.2.3 Echelle de comparaison de la complexité des algorithmes . . . . .	74

<b>9</b>	<b>Les réseau de Pétri</b>	<b>75</b>
9.1	Définition . . . . .	76
9.2	Utilisation d'un réseau de Pétri . . . . .	77
9.3	Propriétés des réseaux de Pétri . . . . .	78
9.3.1	Réseau de Pétri vivant . . . . .	78
9.3.2	Réseau de Pétri pseudo-vivant . . . . .	78
9.3.3	Réseau de Pétri borné . . . . .	79
9.3.4	Réseau de Pétri avec conflits . . . . .	79
9.3.4.1	Conflits structurels . . . . .	79
9.3.4.2	Conflit effectif . . . . .	80
9.3.5	Recherche de propriétés dans un réseau de Pétri . . . . .	80
9.3.5.1	Méthode du graphe des marquages . . . . .	80
9.3.5.2	Méthode de l'arborescence et graphe de couverture d'un réseau de Pétri . . . . .	80
<b>A</b>	<b>Exercices</b>	<b>83</b>
A.1	Exercice 1 . . . . .	84
A.1.1	Questions . . . . .	84
A.1.2	Réponses . . . . .	84
A.2	Exercice 2 . . . . .	86
A.2.1	Questions . . . . .	86
A.2.2	Réponses . . . . .	86
A.3	Exercice 3 . . . . .	87
A.3.1	Questions . . . . .	87
A.3.2	Réponses . . . . .	88
A.4	Exercice 4 . . . . .	89
A.4.1	Questions . . . . .	89
A.4.2	Réponses . . . . .	90
A.5	Exercice 5 . . . . .	91
A.5.1	Questions . . . . .	91
A.5.2	Réponses . . . . .	91
A.6	Exercice 7 . . . . .	92
A.6.1	Questions . . . . .	92
A.6.2	Réponses . . . . .	92
A.7	Exercice 8 . . . . .	96
A.7.1	Questions . . . . .	96
A.7.2	Réponses . . . . .	96
A.8	Exercice 9 . . . . .	98
A.8.1	Questions . . . . .	98
A.8.2	Réponses . . . . .	98
A.9	Exercice 11 . . . . .	99
A.9.1	Questions . . . . .	99
A.9.2	Réponses . . . . .	100
A.10	Exercice 12 . . . . .	102
A.10.1	Questions . . . . .	102
A.10.2	Réponses . . . . .	102
A.11	Exercice 13 . . . . .	104
A.11.1	Questions . . . . .	104
A.11.2	Réponses . . . . .	104
A.12	Exercice 14 . . . . .	105
A.12.1	Questions . . . . .	105
A.12.2	Réponses . . . . .	105
A.13	Exercice 15 . . . . .	106
A.13.1	Questions . . . . .	106
A.13.2	Réponses . . . . .	107
A.14	Exercice 17 . . . . .	108
A.14.1	Questions . . . . .	108

---

A.14.2 Réponses . . . . .	108
A.15 Exercice 17b . . . . .	111
A.15.1 Questions . . . . .	111
A.15.2 Réponses . . . . .	112
A.16 Exercice 19 . . . . .	114
A.16.1 Questions . . . . .	114
A.16.2 Réponses . . . . .	115
A.17 Exercice 18 . . . . .	117
A.17.1 Questions . . . . .	117
A.17.2 Réponses . . . . .	117
A.18 Exercice 20 . . . . .	120
A.18.1 Questions . . . . .	120
A.18.2 Réponses . . . . .	120
A.19 Exercice 21 . . . . .	121
A.19.1 Questions . . . . .	121
A.19.2 Réponses . . . . .	121

# Chapitre 1

## Définitions sur les graphes

## 1.1 Graphes orientés

### 1.1.1 Définition d'un graphe orienté

Un graphe orienté est défini par le doublet  $\langle X, U \rangle$  où :

- $X$  est l'ensemble des sommets du graphe
- $U$  est l'ensemble des arcs du graphe

### 1.1.2 Exemple de graphe orienté

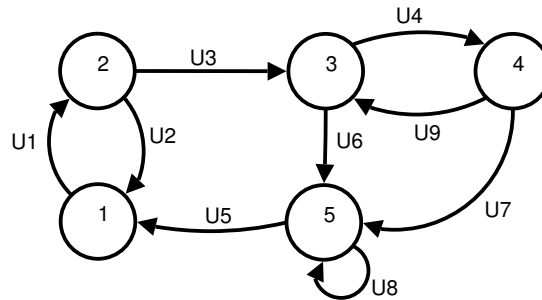


FIG. 1.1: Exemple de graphe orienté.

$$X = \{1, 2, 3, 4, 5\}$$

$$U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9\}$$

### 1.1.3 Définition d'un chemin

Un chemin est défini par une liste de sommets  $(1, 2, \dots, k)$  tel qu'il existe un arc de chaque sommet vers le suivant.

### 1.1.4 Exemple de chemin

Voir la figure 1.1 à la page 7

$$P_1 = \{1, 2, 3, 4\}$$

$$P_1 = \{u_1, u_3, u_4\}$$

### 1.1.5 Définition de la longueur d'un chemin

La longueur d'un chemin est définie par le nombre d'arcs composant le chemin.

### 1.1.6 Exemple de la longueur d'un chemin

Voir la figure 1.1 à la page 7

$$L(P_1) = 3$$

### 1.1.7 Définition d'un circuit

Un circuit est un chemin d'un sommet vers lui-même.

### 1.1.8 Exemple de circuit

Voir la figure 1.1 à la page 7

$$P_2 = \{1, 2, 3, 4, 5, 1\}$$

$$P_2 = \{u_1, u_3, u_6, u_5\}$$

### 1.1.9 Définition d'une boucle

Une boucle est un circuit de longueur 1

### 1.1.10 Exemple de boucle

Voir la figure 1.1 à la page 7

$$P_3 = \{5, 5\}$$

$$P_3 = \{u_8\}$$

### 1.1.11 Définition d'un chemin élémentaire

Un chemin élémentaire est un chemin tel qu'en le parcourant, on ne rencontre pas deux fois le même sommet.

### 1.1.12 Exemple de chemin élémentaire

Voir la figure 1.1 à la page 7

$$P_4 = \{1, 2, 3\} \quad \text{est un chemin élémentaire}$$

$$P_5 = \{2, 1, 2, 3, 4\} \quad \text{n'est pas un chemin élémentaire}$$

### 1.1.13 Définition d'un chemin simple

Un chemin simple est un chemin ne passant pas plus d'une fois par le même arc.

### 1.1.14 Exemple de chemin simple

Voir la figure 1.1 à la page 7

$$P_6 = \{2, 1, 2, 3, 4\}$$

$$P_6 = \{u_2, u_1, u_3, u_4\} \quad \text{est un chemin simple}$$

$$P_7 = \{1, 2, 1, 2, 3\}$$

$$P_7 = \{u_1, u_2, u_1, u_3\} \quad \text{n'est pas un chemin simple}$$

### 1.1.15 Définition d'un graphe complet

Un graphe est dit complet si chaque sommet possède un<sup>1</sup> arc vers tout autre sommet y compris lui-même.

### 1.1.16 Exemple de graphe complet

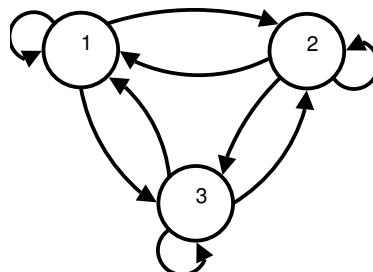


FIG. 1.2: Exemple de graphe orienté complet.

$$N \text{ sommets} \Rightarrow N^2 \text{ arcs}$$

<sup>1</sup>et un seul

## 1.2 Graphes non-orientés

### 1.2.1 Définition d'un graphe non-orienté

Un graphe non-orienté  $G$  est défini par le doublet  $\langle X, U \rangle$  où :

- $X$  est l'ensemble des sommets du graphe
- $U$  est l'ensemble des arêtes

### 1.2.2 Exemple de graphe non-orienté

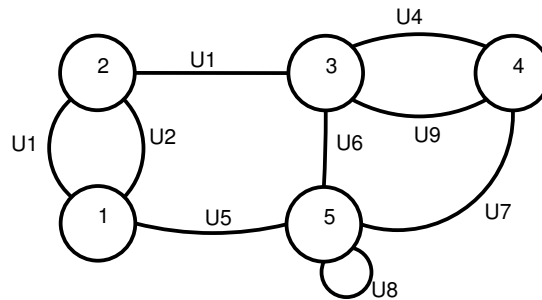


FIG. 1.3: Exemple de graphe non-orienté.

$$X = \{1, 2, 3, 4, 5\}$$

$$U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9\}$$

### 1.2.3 Définition d'une chaîne

Une chaîne est une séquence d'arêtes consécutives.

### 1.2.4 Exemple de chaîne

Voir la figure 1.3 à la page 9

$$P_8 = \{1, 2, 3\}$$

$$P_8 = \{u_1, u_3\}$$

### 1.2.5 Définition d'un cycle

Un cycle est une chaîne fermée.

### 1.2.6 Exemple de cycle

Voir la figure 1.3 à la page 9

$$P_9 = \{1, 2, 3, 5, 1\}$$

$$P_9 = \{u_1, u_3, u_6, u_5\}$$

### 1.2.7 Définition d'un graphe complet

Un graphe est dit complet si chaque sommet possède une<sup>1</sup> arête vers tout autre sommet y compris lui-même.

### 1.2.8 Exemple de graphe complet

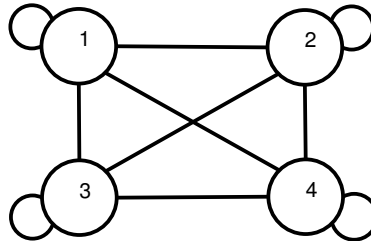


FIG. 1.4: Exemple de graphe non-orienté complet.

$$N \text{ sommets} \Rightarrow \frac{N * (N + 1)}{2} \text{ arêtes}$$

Démonstration de la formule ci-dessus :

$$S = N + (N + 1) + \dots + 1$$

$$S = 1 + 2 + \dots + (N - 1) + N$$

$$2S = \underbrace{(N + 1) + \dots + (N + 1)}_{N \text{ fois}}$$

$$2S = N * (N + 1)$$

$$S = \frac{N * (N + 1)}{2}$$

**Sommet 1** : 4 arêtes

**Sommet 2** : 3 arêtes

**Sommet 3** : 2 arêtes

**Sommet 4** : 1 arête

De façon plus générale, on peut dire :

**Sommet 1** : N arêtes

**Sommet 2** : N-1 arêtes

**Sommet ...** : ... arêtes

**Sommet N** : 1 arête

---

<sup>1</sup>et une seule

### 1.2.9 Définition d'un graphe simple

Un graphe est dit simple si<sup>1</sup> il ne comporte pas de boucle et s'il n'y a pas plus d'une arête entre deux sommets quelconques.

### 1.2.10 Exemple de graphe simple

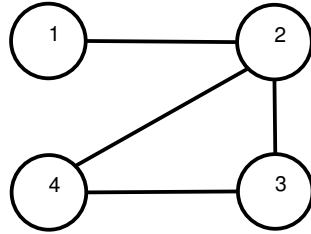


FIG. 1.5: Exemple de graphe non-orienté simple.

---

<sup>1</sup>et seulement si

### 1.3 Graphes connexes & fortement connexes

#### 1.3.1 Définition d'un graphe connexe

Un graphe non-orienté est dit connexe si pour tout couple de sommet  $(i, j)$ , il existe une chaîne joignant  $i$  et  $j$ .

#### 1.3.2 Exemple de graphe connexe

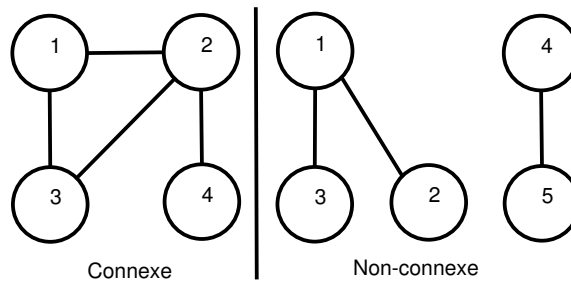


FIG. 1.6: Exemple de graphe connexe.

#### 1.3.3 Définition d'un graphe fortement connexe

Un graphe orienté est dit fortement connexe s'il existe un chemin entre tout couple de sommet.

#### 1.3.4 Exemple de graphe fortement connexe

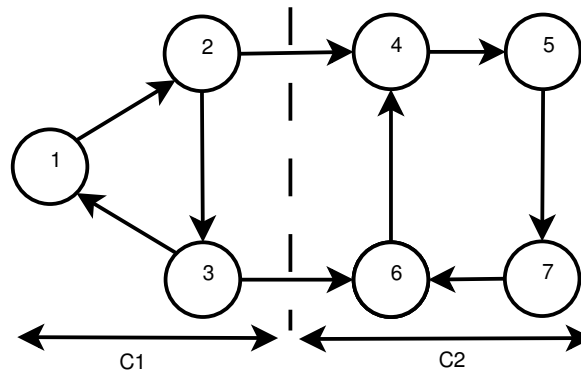
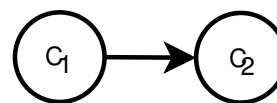


FIG. 1.7: Exemple de graphe non-fortement connexe ayant deux composantes fortement connexes

Ce graphe n'est pas connexe mais il contient deux sous-graphes fortement connexes.

$$C_1 = \{1, 2, 3\}$$

$$C_2 = \{4, 5, 6, 7\}$$



On peut en déduire un graphe réduit représentant les deux composantes fortement connexes

FIG. 1.8: Exemple de graphe réduit

## 1.4 Chemins et circuits Hamiltoniens

### 1.4.1 Définition d'un chemin et d'un circuit Hamiltonien

Un chemin Hamiltonien est un chemin passant une fois<sup>1</sup> par chaque sommet du graphe.  
Un circuit est un chemin Hamiltonien se refermant sur son sommet d'origine.

### 1.4.2 Exemple de chemin et de circuit Hamiltonien

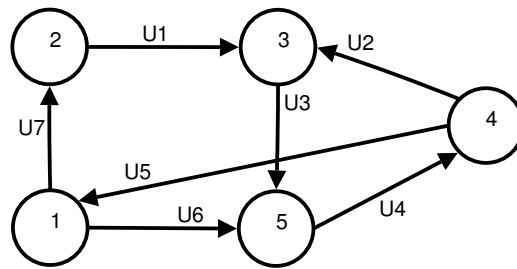


FIG. 1.9: Exemple de chemin et de circuit Hamiltonien

$H = \{1, 2, 4, 3, 5\}$  est un chemin Hamiltonien

$C = \{1, 2, 4, 3, 5, 1\}$  est un circuit Hamiltonien

---

<sup>1</sup>Et une seule fois

## 1.5 Degré et demi-degré d'un sommet

### 1.5.1 Graphes orientés

#### 1.5.1.1 Définition du demi-degré extérieur d'un sommet

On appelle le demi-degré extérieur d'un sommet  $i$  noté  $d_i^+$  le nombre d'arcs ayant pour extrémité initiale le sommet  $i$  et comme extrémité finale  $j$  avec  $i \neq j$

#### 1.5.1.2 Exemple de demi-degré extérieur d'un sommet

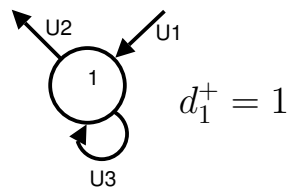


FIG. 1.10: Exemple montrant un sommet pour les degrés/demi-degrés d'un graphe orienté

#### 1.5.1.3 Définition du demi-degré intérieur d'un sommet

On appelle le demi-degré extérieur d'un sommet  $i$  noté  $d_i^-$  le nombre d'arcs ayant pour extrémité finale le sommet  $i$  et comme extrémité initiale  $j$  avec  $i \neq j$

#### 1.5.1.4 Exemple de demi-degré intérieur d'un sommet

Voir la figure 1.10 à la page 14

$$d_1^- = 1$$

#### 1.5.1.5 Définition du degré d'un sommet

On appelle le degré d'un sommet  $i$  noté  $d_i$  le nombre d'arcs ayant une<sup>1</sup> extrémité en ce sommet.

#### 1.5.1.6 Exemple de degré d'un sommet

Voir la figure 1.10 à la page 14

$$d_1 = 2$$

---

<sup>1</sup>Et une seule

## 1.5.2 Graphes non-orientés

### 1.5.2.1 Définition du degré d'un sommet

On appelle le degré d'un sommet  $i$  le nombre d'arêtes ayant une<sup>1</sup> extrémité en ce sommet.

### 1.5.2.2 Exemple de degré d'un sommet

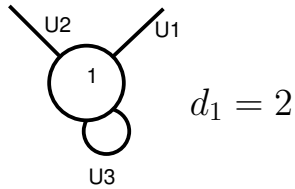


FIG. 1.11: Exemple montrant un sommet pour le degré d'un graphe non-orienté

---

<sup>1</sup>Et une seule

## 1.6 Cocycle d'un graphe

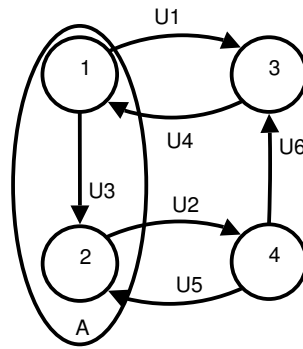


FIG. 1.12: Exemple montrant le cocycle d'un graphe

Soit :

- $G = \langle X, U \rangle$
- $X = \{1, 2, 3, 4\}$
- $U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$
- $A = \{1, 2\}$  Le cocycle de l'ensemble  $A$  est noté  $\omega(A)$

$$\omega(A) = \omega^+(A) + \omega^-(A)$$

$\omega^+(A)$  est l'ensemble d'arcs ayant leur extrémité initiale dans  $A$  et leur extrémité terminale dans  $\bar{A}$

$$\omega^+(A) = \{u_1, u_2\}$$

$\omega^-(A)$  est l'ensemble d'arcs ayant leur extrémité initiale dans  $\bar{A}$  et leur extrémité terminale dans  $A$

$$\omega^-(A) = \{u_4, u_5\}$$

$$\omega(A) = \{u_1, u_2, u_4, u_5\}$$

Remarque pour les graphes non-orientés :

$\omega(A)$  est l'ensemble des arêtes ayant une extrémité dans  $A$  et l'autre dans  $\bar{A}$

## 1.7 Application multivoque d'un graphe

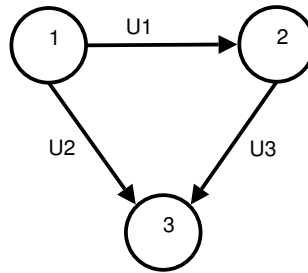


FIG. 1.13: Exemple d'application multivoque d'un graphe

Soit :

- $G = \langle X, U \rangle$
- $X = \{1, 2, 3\}$
- $U = \{u_1, u_2, u_3\}$

Sommet 1 :  $\Gamma_1$  est l'ensemble des successeurs du sommet 1

Sommet  $i$  :  $\Gamma_i$  est l'ensemble des successeurs du sommet  $i$

$\Gamma_i$  est l'application multivoque qui fait correspondre à  $X$  une partie de  $X$

$$\Gamma_i : X \rightarrow P(X)$$

$$\Gamma_1 = \{2, 3\}$$

$$\Gamma_2 = \{3\}$$

$$\Gamma_3 = \emptyset$$

On peut également définir le graphe par son application multivoque :

$$G = \langle X, \Gamma \rangle = \langle X, U \rangle$$

$\Gamma_1^{-1}$  est l'application multivoque réciproque d'un graphe<sup>1</sup>

Faire l'exercice 1 à la page 84

---

<sup>1</sup>L'ensemble des prédecesseurs du sommet  $i$

## 1.8 Représentation des graphes

### 1.8.1 Matrice d'adjacence

#### 1.8.1.1 Graphes orientés

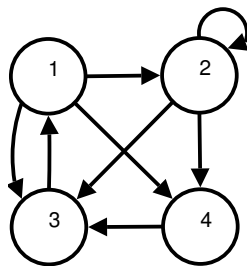
$$G = \langle X, U \rangle$$

Soit  $N$  le nombre de sommets et  $A$  la matrice d'adjacence

$$A = (a_{ij}) \text{ avec } i \in [1, N] \text{ et } j \in [1, N]$$

Pour compléter la matrice il convient de le faire de la sorte :

- $a_{ij} = 1$  si  $(i, j) \in U$  <sup>1</sup>
- $a_{ij} = 0$  si  $(i, j) \notin U$  <sup>2</sup>



	1	2	3	4
1	0	1	1	1
2	0	1	1	1
3	1	0	0	0
4	0	0	1	0

FIG. 1.14: Exemple de graphe orienté pour exemple de matrice d'adjacence

#### 1.8.1.2 Graphes non-orientés

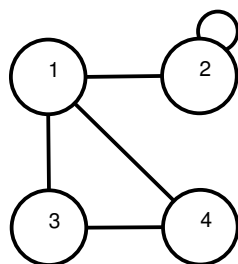
$$G = \langle X, U \rangle$$

Soit  $N$  le nombre de sommets et  $A$  la matrice d'adjacence

$$A = (a_{ij}) \text{ avec } i \in [1, N] \text{ et } j \in [1, N]$$

Pour compléter la matrice il convient de le faire de la sorte :

- $a_{ij} = 1$  et  $a_{ji} = 1$  si  $(i, j) \in U$  <sup>3</sup>
- $a_{ij} = 0$  et  $a_{ji} = 0$  si  $(i, j) \notin U$  <sup>4</sup>



	1	2	3	4
1	0	1	1	1
2	1	1	0	0
3	1	0	0	1
4	1	0	1	0

FIG. 1.15: Exemple de graphe non-orienté pour exemple de matrice d'adjacence

<sup>1</sup>Il faut mettre un 1 s'il existe un arc

<sup>2</sup>Il faut mettre un 0 s'il n'existe pas d'arc

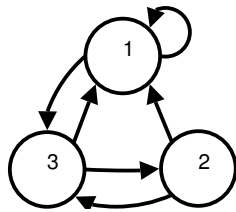
<sup>3</sup>Il faut mettre un 1 s'il existe un arc

<sup>4</sup>Il faut mettre un 0 s'il n'existe pas d'arc

### 1.8.2 Liste d'adjacence

#### 1.8.2.1 Graphes orientés

- $G = \langle X, U \rangle$
- $N$  est le nombre de sommets
- $M$  est le nombre d'arcs



N	1	2	3	4	
LP	1	3	5	7	N+1 éléments

M	1	2	3	4	5	6	7	
LS	1	3	1	3	1	2	X	M+1 éléments

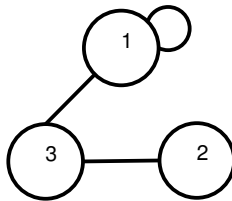
FIG. 1.16: Exemple de graphe orienté pour exemple de liste d'adjacence

Méthode :

Les successeurs du sommet  $i$  se trouvent entre les indices  $LP[i]$  et  $LP[i + 1] - 1$  du tableau  $LS$

#### 1.8.2.2 Graphes non-orientés

- $G = \langle X, U \rangle$
- $N$  est le nombre de sommets
- $M$  est le nombre d'arêtes



N	1	2	3	4	
LP	1	3	5	6	N+1 éléments

M	1	2	3	4	5	6	
LS	1	2	1	3	2	X	2M+1 éléments

FIG. 1.17: Exemple de graphe non-orienté pour exemple de liste d'adjacence

### 1.8.3 Matrice d'incidence

#### 1.8.3.1 Graphes orientés

- $G = \langle X, U \rangle$
- $N$  est le nombre de sommets
- $M$  est le nombre d'arcs

Soit  $A$  la matrice d'incidence :

$A = (a_{ij})$  avec  $i \in [1, N]$  et  $j \in [1, M]$

Pour compléter la matrice il convient de le faire de la sorte :

- $a_{ij} = 1$  si  $u_j \in \omega^+(i)$  <sup>1</sup>
- $a_{ij} = -1$  si  $u_j \in \omega^-(i)$  <sup>2</sup>
- $a_{ij} = 0$  dans les autres cas

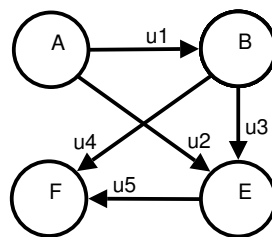


FIG. 1.18: Exemple de graphe orienté pour exemple de matrice d'incidence

Dans ce cas on voit bien que :

$$\omega^+(1) = \{u_1, u_2\}$$

$$\omega^-(1) = \emptyset$$

$$\omega^+(2) = \{u_3, u_4\}$$

$$\omega^-(2) = \{u_1\}$$

Et ainsi de suite on peut remplir la matrice suivante :

	1	2	3	4	5
1	1	1	0	0	0
2	-1	0	1	1	0
3	0	-1	-1	0	1
4	0	0	0	-1	-1

<sup>1</sup>Les arcs partant du sommet  $i$

<sup>2</sup>Les arcs arrivant au sommet  $i$

### 1.8.3.2 Graphes non-orientés

- $G = \langle X, U \rangle$
- $N$  est le nombre de sommets
- $M$  est le nombre d'arcs

Soit  $A$  la matrice d'incidence :

$$A = (a_{ij}) \text{ avec } i \in [1, N] \text{ et } j \in [1, M]$$

Pour compléter la matrice il convient de le faire de la sorte :

- $a_{ij} = 1$  si  $u_j \in \omega(i)$ <sup>1</sup>
- $a_{ij} = 0$  dans les autres cas

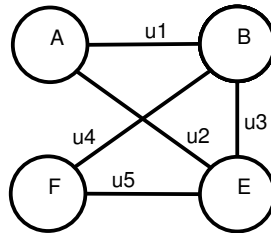


FIG. 1.19: Exemple de graphe non-orienté pour exemple de matrice d'incidence

$$\omega(1) = \{u_1, u_2\}$$

$$\omega(2) = \{u_1, u_3, u_4\}$$

$$\omega(3) = \{u_2, u_3, u_5\}$$

$$\omega(4) = \{u_4, u_5\}$$

	1	2	3	4	5
1	1	1	0	0	0
2	0	0	1	1	0
3	0	1	1	0	1
4	0	0	0	1	1

Faire les exercices 2 à la page 86 et 3 à la page 87

<sup>1</sup>Les boucles ne comptent pas

## Chapitre 2

# Méthodes de parcours de graphe

## 2.1 Structures de données

### 2.1.1 Liste linéaire

#### 2.1.1.1 Définition

Une liste linéaire est une suite finie de  $n$  éléments de même type que l'on note  $L = (e_1, \dots, e_n)$

#### 2.1.1.2 Opérations

- *Initialiser* ( $L$ )  $\rightarrow L$  <sup>1</sup>
- *Chercher* ( $x, L$ )  $\rightarrow p$  <sup>2</sup>

#### 2.1.1.3 Implémentation d'une liste

Tableau :

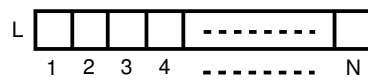


FIG. 2.1: Représentation d'une liste linéaire sous forme d'un tableau

Liste chaînée :



FIG. 2.2: Représentation d'une liste linéaire sous forme de liste chaînée

<sup>1</sup>Initialise la liste  $L$  à vide

<sup>2</sup>Cherche l'élément  $x$  dans la liste  $L$  et rend la position  $p$  de l'élément

## 2.1.2 Pile

### 2.1.2.1 Définition

Une pile est une liste linéaire où les insertions et suppressions ne se font que d'une extrémité

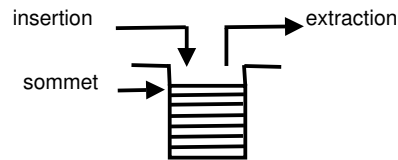


FIG. 2.3: Représentation d'une pile

### 2.1.2.2 Opérations

- $Initialiser(P) \rightarrow P$  <sup>1</sup>
- $Empiler(x, P) \rightarrow P$  <sup>2</sup>
- $Depiler(P) \rightarrow P$  <sup>3</sup>
- $Sommet(P) \rightarrow x$  <sup>4</sup>
- $PileVide(P) \rightarrow b$  <sup>5 6</sup>

## 2.1.3 File

### 2.1.3.1 Définition

Une file est une liste linéaire où les insertions se font d'une extrémité, et les extractions de l'autre

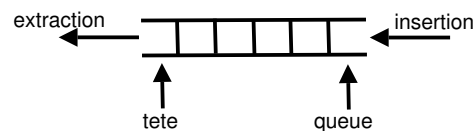


FIG. 2.4: Représentation d'une file

### 2.1.3.2 Opérations

- $Initialiser(F) \rightarrow F$  <sup>7</sup>
- $Enfiler(x, F) \rightarrow P$  <sup>8</sup>
- $Defiler(F) \rightarrow P$  <sup>9</sup>
- $Tête(F) \rightarrow x$  <sup>10</sup>
- $FileVide(F) \rightarrow b$  <sup>11 12</sup>

<sup>1</sup>Initialise la pile  $P$  à vide

<sup>2</sup>Ajoute l'élément  $x$  dans la pile  $P$

<sup>3</sup>Retire l'élément le plus récent de la pile  $P$

<sup>4</sup>Renvoie l'élément le plus récent de la pile  $P$

<sup>5</sup>Indique si la pile est vide

<sup>6</sup> $b$  est un booléen valant *vrai* si  $P$  est vide ou *faux* s'il existe au moins un élément

<sup>7</sup>Initialise la file  $F$  à vide

<sup>8</sup>Ajoute l'élément  $x$  dans la file  $F$

<sup>9</sup>Retire l'élément le plus récent de la file  $F$

<sup>10</sup>Renvoie l'élément le plus récent de la file  $F$

<sup>11</sup>Indique si la file est vide

<sup>12</sup> $b$  est un booléen valant *vrai* si  $F$  est vide ou *faux* s'il existe au moins un élément

## 2.2 Parcours en profondeur d'un graphe

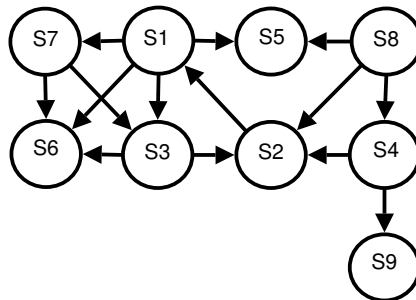


FIG. 2.5: Exemple de graphe orienté pour le parcours en profondeur

```

S1  →  S3  →  S2
      →  S6
      →  S5
      →  S7
S4  →  S9
S8

```

### 2.2.1 Principe

On part d'un sommet  $X$  du graphe et on s'en éloigne le plus possible en cheminant le long des arcs dans l'ordre lexicographique<sup>1</sup>. Si l'en rencontre un cul de sac, on revient en arrière et on repart. On ne s'arrête que lorsque tous les sommets ont été visités.

### 2.2.2 Algorithme

- $M$  est la matrice d'adjacence du graphe
- $N$  est le nombre de sommets
- $S$  est le sommet courant
- $L$  est la liste des sommets visités<sup>2</sup>

ParcoursProfondeur( $S$ )

Début

$L[S] = \text{VRAI}$

  Pour  $i$  allant de 1 à  $N$  faire

    Si  $M[S,i] \neq 0$  et  $L[i] = \text{FAUX}$  alors

      ParcoursProfondeur( $i$ )

    Fin Si

  Fin Pour

Fin

<sup>1</sup>L'ordre logique des sommets

<sup>2</sup>Chaque case contient *vrai* si le sommet à déjà été visité, *faux* sinon

```

ProgrammePrincipal
Début
    Pour i allant de 1 à N faire
        L[i] = FAUX
    Fin pour
    Pour i allant de 1 à N faire
        Si L[i] = FAUX alors
            ParcoursProfondeur(i)
        Fin Si
    Fin Pour
Fin
    
```

### 2.2.3 Vérification de l'algorithme

Matrice d'adjacence :

	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$	$S_9$
1	0	0	1	0	1	1	1	0	0
2	1	0	0	0	0	0	0	0	0
3	0	1	0	0	0	1	0	0	0
4	0	1	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0	0
8	0	0	0	1	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0

Évolution du tableau  $L$  :

L	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$	$S_9$
Initialisation	F	F	F	F	F	F	F	F	F
$PP(S_1)$	V	F	F	F	F	F	F	F	F
→ $PP(S_3)$	V	F	V	F	F	F	F	F	F
→ $PP(S_2)$	V	V	V	F	F	F	F	F	F
→ $PP(S_6)$	V	V	V	F	F	V	F	F	F
→ $PP(S_5)$	V	F	V	F	V	V	F	F	F
→ $PP(S_7)$	V	F	V	F	V	V	V	F	F
$PP(S_4)$	V	V	V	V	V	V	V	F	F
→ $PP(S_9)$	V	F	V	V	V	V	V	F	V
$PP(S_8)$	V	V	V	V	V	V	V	V	V

## 2.2.4 Forêt courante du parcours en profondeur

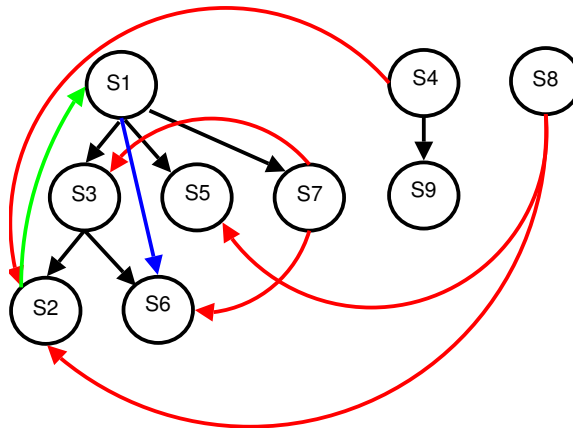


FIG. 2.6: Représentation de la forêt courante déduite du parcours en profondeur

- arcs couvrants (*noir*) : arc reliant un sommet  $x$  à un sommet  $y$  tel que  $\text{ParcoursProfondeur}(x) \rightarrow \text{ParcoursProfondeur}(y)$
- arcs arrières (*vert*) : arc reliant un sommet  $x$  à un sommet  $y$  tel que  $x$  soit un descendant de  $y$  dans l'arborescence
- arcs avant (*bleu*) : arc reliant un sommet  $x$  à un sommet  $y$  tel que  $x$  soit un ancêtre de  $y$  dans l'arborescence
- arcs croisés (*rouge*) : arc reliant deux sommets d'une ou deux arborescences

Faire les exercices 4 à la page 89 et 5 à la page 91

## 2.3 Parcours en largeur d'un graphe

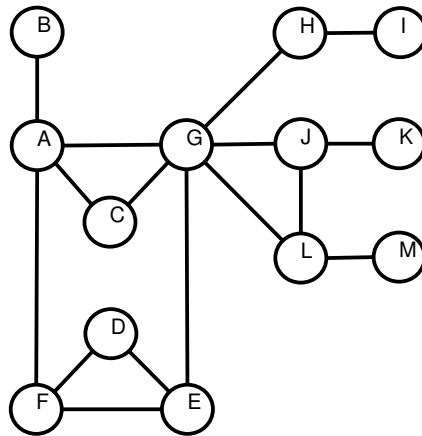


FIG. 2.7: Exemple de graphe non-orienté pour réaliser le parcours en largeur d'un graphe

### 2.3.1 Méthode de parcours

- On sélectionne un sommet  $A$  de départ
- On sélectionne les sommets  $B, C, F, G$  qui sont à une distance 1 du sommet de départ
- On sélectionne les sommets  $D, E, H, J, L$  qui sont à une distance 2 du sommet de départ
- On sélectionne les sommets  $I, K, M$  qui sont à une distance 3 du sommet de départ

### 2.3.2 Principe

- On part d'un sommet  $X$  du graphe
- On visite tous les sommets adjacents à  $X$
- On visite ensuite tous les sommets adjacents à ceux-ci et ainsi de suite.
- On ne s'arrête que lorsque tous les sommets ont été visités

### 2.3.3 Algorithme

- M est la matrice d'adjacence du graphe
- N est le nombre de sommets
- F est une File
- S est le sommet courant
- L est la liste contenant l'état de chaque sommet<sup>1</sup>

ParcoursLargeur(S)

Début

```

F = Initialiser(F)
L[S] = VRAI
F = Enfiler(S, F)
Tant que (non(FileVide(F)) faire
  T = Tête(F)
  F = Défiler(F)
  Pour i allant de 1 à N faire
    Si M[T, i] = 0 et L[i] = FAUX alors
      F = Enfiler(i, F)
      L[i] = VRAI
    Fin Si
  Fin Pour
Fin Tant que

```

Fin

ProgrammePrincipal

Début

```

Pour i allant de 1 à N faire
  Si L[i] = FAUX alors
    ParcoursLargeur(i)
  Fin Si
Fin Pour

```

Fin

### 2.3.4 Vérification de l'algorithme

Matrice d'adjacence :

	A	B	C	D	E	F	G	H	I	J	K	L	M
A	0	1	1	0	0	1	1	0	0	0	0	0	0
B	1	0	0	0	0	0	0	0	0	0	0	0	0
C	1	0	0	0	0	0	1	0	0	0	0	0	0
D	0	0	0	0	1	1	0	0	0	0	0	0	0
E	0	0	0	1	0	1	1	0	0	0	0	0	0
F	1	0	0	1	1	0	0	0	0	0	0	0	0
G	1	0	1	0	1	0	0	1	0	1	0	1	0
H	0	0	0	0	0	0	1	0	1	0	0	0	0
I	0	0	0	0	0	0	0	1	0	0	0	0	0
J	0	0	0	0	0	0	1	0	0	0	1	1	0
K	0	0	0	0	0	0	0	0	0	1	0	0	0
L	0	0	0	0	0	0	1	0	0	1	0	0	1
M	0	0	0	0	0	0	0	0	0	0	0	1	0

<sup>1</sup>Chaque case contient *vrai* si le sommet a déjà été visité, *faux* sinon

Évolution du tableau  $L$  :

L	A	B	C	D	E	F	G	H	I	J	K	L	M	Contenu de F
Init	F	F	F	F	F	F	F	F	F	F	F	F	F	$F = \{A\}$
$T = A$	V	F	F	F	F	F	F	F	F	F	F	F	F	$F = \emptyset$
$T = \emptyset$	V	V	V	F	F	V	V	F	F	F	F	F	F	$F = \{B, C, F, G\}$
$T = B$	V	V	V	F	F	V	V	F	F	F	F	F	F	$F = \{C, F, G\}$
$T = C$	V	V	V	F	F	V	V	F	F	F	F	F	F	$F = \{F, G\}$
$T = F$	V	V	V	V	V	V	V	F	F	F	F	F	F	$F = \{G, D, E\}$
$T = G$	V	V	V	V	V	V	V	V	F	V	F	V	F	$F = \{D, E, H, J, L\}$
$T = D$	V	V	V	V	V	V	V	V	F	V	F	V	F	$F = \{E, H, J, L\}$
$T = E$	V	V	V	V	V	V	V	V	F	V	F	V	F	$F = \{H, J, L\}$
$T = H$	V	V	V	V	V	V	V	V	V	V	F	V	F	$F = \{J, L, I\}$
$T = J$	V	V	V	V	V	V	V	V	V	V	V	V	V	$F = \{L, I, K, M\}$
$T = L$	V	V	V	V	V	V	V	V	V	V	V	V	V	$F = \{I, K, M\}$
$T = I$	V	V	V	V	V	V	V	V	V	V	V	V	V	$F = \{K, M\}$
$T = K$	V	V	V	V	V	V	V	V	V	V	V	V	V	$F = \{M\}$
$T = M$	V	V	V	V	V	V	V	V	V	V	V	V	V	$F = \{\emptyset\}$

Application :

Une des applications possibles serait la recherche du plus court chemin, en nombre d'arêtes entre un sommet donné et les autres sommets du graphe. Ceci est possible en modifiant légèrement l'algorithme de la fonction *ParcoursLargeur* précédemment énoncé :

ParcoursLargeur(S)

Début

    F = Initialiser(F)

    L[S] = 0

    F = Enfiler(S, F)

    Tant que (non(FileVide(F))) faire

        T = Tête(F)

        F = Défiler(F)

        Pour i allant de 1 à N faire

            Si M[T, i] = 0 et L[i] = FAUX alors

                F = Enfiler(i, F)

                L[i] = L[T] + 1

            Fin Si

        Fin Pour

    Fin Tant que

Fin

Ce qui donnerait à la fin de l'application de l'algorithme modifié le résultat suivant dans la liste  $L$  :

$L =$

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	1	2	2	1	1	2	3	2	3	2	3

## Chapitre 3

# Représentation matricielle d'un graphe

### 3.1 Matrice booléenne - matrice aux arcs d'un graphe

Soit le graphe orienté  $G = \langle X, U \rangle$  comportant  $N$  sommets :

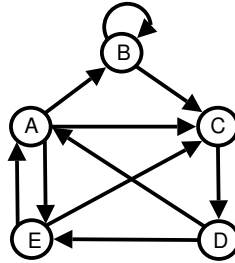


FIG. 3.1: Exemple de graphe orienté pour réaliser les matrices booléennes et aux arcs

Il peut être représenté par une matrice d'adjacence (matrice booléenne) ou par une matrice aux arcs.

Matrice d'adjacence ou matrice booléenne :

$M = (a_{ij})$  ou  $i$  et  $j$  varient de 1 à  $N$

On a pour chaque élément de la matrice :

- $a_{ij} = 1$  si  $(i, j) \in U$
- $a_{ij} = 0$  si  $(i, j) \notin U$

$$M = \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 0 & 1 \\ B & 0 & 1 & 1 & 0 & 0 \\ C & 0 & 0 & 0 & 1 & 0 \\ D & 1 & 0 & 0 & 0 & 1 \\ E & 1 & 0 & 1 & 0 & 0 \end{array}$$

Matrice aux arcs :

$A = (a_{ij})$  ou  $i$  et  $j$  varient de 1 à  $N$

On a pour chaque élément de la matrice :

- $a_{ij} = ij$  si  $(i, j) \in U$
- $a_{ij} = 0$  si  $(i, j) \notin U$

$$A = \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & AB & AC & 0 & AE \\ B & 0 & BB & BC & 0 & 0 \\ C & 0 & 0 & 0 & CD & 0 \\ D & DA & 0 & 0 & 0 & DE \\ E & EA & 0 & EC & 0 & 0 \end{array}$$

## 3.2 Existence et dénombrement de chemins dans un graphe

### 3.2.1 Définition

L'existence et le dénombrement de chemins de longueur  $k$  dans un graphe se fait en élevant à la puissance  $k$  la matrice booléenne aux arcs.

### 3.2.2 Matrice booléenne

On prend la matrice booléenne  $M$ , on l'élève successivement au carré, au cube, ..., à la puissance  $k$  pour déterminer le nombre de chemins de longueur  $2, 3, \dots, k$

#### Calcul de $M^2$

$$M^2 = M * M$$

$$M^2 = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 2 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 2 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

L'intersection de la ligne  $i$  et de la colonne  $j$  de la matrice  $M^2$  donne le nombre de chemins de longueur 2 entre les sommets  $i$  et  $j$

Plus généralement l'intersection de la ligne  $i$  et de la colonne  $j$  de la matrice  $M^k$  donne le nombre de chemins de longueur  $k$  entre les sommets  $i$  et  $j$

### 3.2.3 Matrice aux arcs

On prend la matrice aux arcs  $A$ , on l'élève successivement au carré, au cube, ..., à la puissance  $k$  pour déterminer le nombre de chemins de longueur  $2, 3, \dots, k$  ainsi que leur composition.

L'intérêt d'une telle représentation par rapport à la représentation par matrice booléenne est que l'on peut lire directement dans  $A^k$  la composition des chemins de longueur  $k$ .

#### Calcul de $A^2$

$$A^2 = M * M$$

$$A^2 = \begin{array}{|c|c|c|c|c|} \hline 0 & AB & AC & 0 & AE \\ \hline 0 & BB & BC & 0 & 0 \\ \hline 0 & 0 & 0 & CD & 0 \\ \hline DA & 0 & 0 & 0 & DE \\ \hline EA & 0 & EC & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|c|c|} \hline 0 & AB & AC & 0 & AE \\ \hline 0 & BB & BC & 0 & 0 \\ \hline 0 & 0 & 0 & CD & 0 \\ \hline DA & 0 & 0 & 0 & DE \\ \hline EA & 0 & EC & 0 & 0 \\ \hline \end{array}$$

$$A^2 = \begin{array}{|c|c|c|c|c|c|} \hline & A & B & C & D & E \\ \hline A & AEA & ABB & ABC, AEC & ACD & 0 \\ \hline B & 0 & BBB & BBC & BCD & 0 \\ \hline C & CDA & 0 & 0 & 0 & CDE \\ \hline D & DEA & DAB & DAC, DEC & 0 & DAE \\ \hline E & 0 & EAB & EAC & ECD & EAE \\ \hline \end{array}$$

La méthode pour élever la matrice aux arcs au carré, au cube, etc, est d'adopter les lois de composition suivantes pour l'addition et la multiplication :

- La multiplication revient à concaténer 2 chemins  $AB * BC \rightarrow ABC$
- L'addition est équivalente à la réunion ensembliste  $(A, B, C) + (A, E, C) \rightarrow$  ils figurent dans la même case de la matrice aux arcs  $A^2$

L'intersection de la ligne  $i$  et de la colonne  $j$  de la matrice  $A^2$  donne le nombre de chemins de longueur 2 entre les sommets  $i$  et  $j$  ainsi que leur composition.

Plus généralement, l'intersection de la ligne  $i$  et de la colonne  $j$  de la matrice  $A^k$  donne le nombre de chemins de longueur  $k$  entre les sommets  $i$  et  $j$  ainsi que leur composition.

### 3.2.4 Chemins élémentaires - chemins hamiltoniens

#### 3.2.4.1 Définition

L'existence et le dénombrement de chemins élémentaires de longueur  $k$  dans un graphe se fait en élevant à la puissance  $k$  la matrice aux arcs et en ne retenant que les chemins ne comportant pas de répétition de lettres.

Soit le graphe orienté  $G = \langle X, U \rangle$  comportant  $N$  sommets

Voir la figure 3.1 à la page 32

- Chemins élémentaires de longueur 1

Dans la matrice aux arcs  $A$ , on commence par supprimer tous les chemins qui ne sont pas élémentaires.

$(B, B) \rightarrow$  boucle sur le sommet  $B$

Ce chemin est éliminé de  $A$

On obtient la matrice aux arcs  $A'$  contenant les chemins élémentaires de longueur 1

$$A' = \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & AB & AC & 0 & AE \\ B & 0 & 0 & BC & 0 & 0 \\ C & 0 & 0 & 0 & CD & 0 \\ D & DA & 0 & 0 & 0 & DE \\ E & EA & 0 & EC & 0 & 0 \end{array}$$

- Chemins élémentaires de longueur 2

On calcule  $A'^2$  à partir de  $A'$  en ne retenant que les chemins élémentaires.

$(A, E) * (E, A) \rightarrow (A, E, A)$  est un chemin de longueur 2 passant deux fois par le sommet  $A$

Ce chemin est éliminé de  $A'^2$  puisqu'il n'est pas élémentaire.

On obtient la matrice aux arcs  $A'^2$  qui contient les chemins élémentaires de longueur 2

$$A'^2 = \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 0 & ABC, AEC & ACD & 0 \\ B & 0 & 0 & 0 & BCD & 0 \\ C & CDA & 0 & 0 & 0 & CDE \\ D & DEA & DAB & DAC, DEC & 0 & DAE \\ E & 0 & AEB & EAC & ECD & 0 \end{array}$$

- Chemins élémentaires de longueur 3

On calcule  $A'^3$  en ne retenant que les chemins élémentaires

On obtient la matrice aux arcs  $A'^3$  contenant les chemins élémentaires de longueur 3

- Chemins élémentaires de longueur 4

Dans la matrice  $A^4$  apparaissent les chemins élémentaires de longueur 4

$$A^4 = \begin{array}{c|ccccc} & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\ \hline \text{A} & 0 & 0 & 0 & 0 & \text{ABCDE} \\ \text{B} & \text{BCDEA} & 0 & 0 & 0 & \text{BCDAE} \\ \text{C} & 0 & \text{CDEAB} & 0 & 0 & 0 \\ \text{D} & 0 & 0 & \text{DEABC} & 0 & 0 \\ \text{E} & 0 & \text{ECDAB} & 0 & \text{EABCD} & 0 \end{array}$$

Ces chemins passent une fois et une seule par tous les sommets du graphe, ce sont donc des chemins Hamiltoniens. Si de plus les chemins Hamiltoniens se referment sur eux-mêmes, ce sont des circuits Hamiltoniens.

### 3.3 Fermeture transitive d'un graphe

#### 3.3.1 Définition

La fermeture transitive d'un graphe  $G = \langle X, U \rangle$  génère un graphe  $G' = \langle X, U' \rangle$  auquel on a rajouté des arcs reliant le sommet  $i$  au sommet  $j$  s'il existe un chemin reliant le sommet  $i$  au sommet  $j$  ou si  $i = j$ .

#### 3.3.2 Exemple

Soit le graphe orienté  $G = \langle X, U \rangle$  suivant :

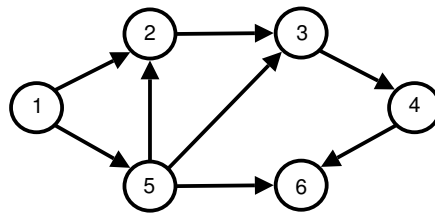


FIG. 3.2: Exemple de graphe orienté pour réaliser la fermeture transitive

Et  $M$  sa matrice booléenne :

$$M = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 1 & 1 & 0 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

La matrice booléenne  $M'$  correspondant à la fermeture transitive du graphe  $G$  est :

$$M' = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 0 & 1 & 1 & 1 & 0 & 1 \\ 3 & 0 & 0 & 1 & 1 & 0 & 1 \\ 4 & 0 & 0 & 0 & 1 & 0 & 1 \\ 5 & 0 & 1 & 1 & 1 & 1 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Cela signifie qu'à partir du sommet 1 on peut accéder aux sommets 2, 3, 4, 5 et 6 du graphe et qu'à partir du sommet 6 on ne peut accéder à aucun sommet du graphe.

Conclusion : La fermeture transitive d'un graphe permet de répondre à la question : "existe t'il un chemin de  $x$  à  $y$  pour tout couple de sommet  $(x, y)$  du graphe"

### 3.3.3 Algorithme de Warshall

Warshall est parti de l'observation suivante :  
S'il existe un moyen d'aller du sommet  $x$  vers le sommet  $y$  **ET** un moyen d'aller du sommet  $y$  vers le sommet  $z$  alors il existe un moyen d'aller du sommet  $x$  vers le sommet  $z$ .

Algorithme :

- $M$  est la matrice booléenne du graphe  $G$
- $M'$  est la matrice booléenne du graphe  $G'$
- $N$  est le nombre de sommets

```

M' = M
Pour i allant de 1 à N faire
    M'[i][i] = 1
Fin Pour
Pour k allant de 1 à N faire
    Pour i allant de 1 à N faire
        Pour j allant de 1 à N faire
            M'[i][j] = M'[i][j] OU (M'[i][k] ET M'[k][j])
        Fin Pour
    Fin Pour
Fin Pour

```

L'idée de l'algorithme est de déterminer les chemins passant par le sommet 1, le sommet 2, le sommet 3, ... le sommet  $N$ . Ce qui se fait en calculant  $M$  pour  $k = 1, k = 2, \dots, k = N^1$

### 3.3.4 Méthode matricielle

Soit le graphe orienté  $G = \langle X, U \rangle$  suivant :

Voir la figure 3.2 à la page 36

Et  $M$  sa matrice booléenne :

$$M = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 1 & 1 & 0 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

On a vu précédemment qu'on pouvait élever la matrice  $M$  au carré, au cube, ..., à la puissance  $k$  pour déterminer le nombre de chemins de longueur 2, 3, ...,  $k$ .

On se propose ici d'élever la matrice  $M$  au carré, au cube, ..., à la puissance  $k$  en adoptant des lois de composition logique pour la multiplication et l'addition.

1. loi de multiplication logique ( ET ) :  $(1 \times 1 = 1)$
2. loi d'addition logique ( OU ) :  $(1 \dot{+} 1 = 1)$  ou encore  $(1 \dot{+} 0 = 1)$

On note les matrices obtenues à partir des lois de composition  $M^{[2]}, M^{[3]}, M^{[4]}$  pour les différencier des matrices booléennes calculées à partir des lois de composition usuelles. Ces matrices nous permettent de savoir s'il existe au moins un chemin de longueur 2, 3, ...,  $k$  dans le graphe.

<sup>1</sup>La matrice obtenue pour  $k = N$  donne la fermeture transitive du graphe

Calcul de  $M^{[2]}$

$$M^{[2]} = M^{[1]} \dot{\times} M^{[1]} = M \dot{\times} M$$

$$M^{[2]} = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \dot{\times} \begin{array}{|c|c|c|c|c|c|} \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$M^{[2]} = \begin{array}{|c|c|c|c|c|c|} \hline & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ \hline 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 3 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 5 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

La présence d'un 1 à l'intersection de la ligne  $i$  et de la colonne  $j$  de  $M^{[k]}$  indique qu'il existe au moins un chemin de longueur  $k$  entre les sommets  $i$  et  $j$

De la même façon on peut calculer les matrices  $M^{[3]}$ ,  $M^{[4]}$ ,  $M^{[5]}$  et si l'on procède à la somme booléenne  $P = M \dot{+} M^{[2]} \dot{+} M^{[3]} \dot{+} M^{[4]} \dot{+} M^{[5]}$  on obtient une matrice  $P$  dans laquelle la présence d'un 1 à l'intersection de la ligne  $i$  et de la colonne  $j$  indique qu'il existe au moins un chemin de longueur inférieure ou égale à 5 entre les sommets  $i$  et  $j$ .

$$P = \begin{array}{|c|c|c|c|c|c|c|} \hline & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ \hline 2 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline 3 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 4 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 5 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

On s'aperçoit que la matrice  $P$  ne donne pas la fermeture transitive du graphe car elle n'est pas composée de 1 sur la diagonale, il faut donc rajouter à l'expression  $P = M \dot{+} M^{[2]} \dot{+} M^{[3]} \dot{+} M^{[4]} \dot{+} M^{[5]}$  la matrice identité  $I$  composée de 1 sur la diagonale et de 0 partout ailleurs. On obtient donc l'expression suivante donnant la fermeture transitive :

$$P = I \dot{+} M \dot{+} M^{[2]} \dot{+} M^{[3]} \dot{+} M^{[4]} \dot{+} M^{[5]}$$

$$P = \begin{array}{|c|c|c|c|c|c|c|} \hline & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 2 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 3 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline 4 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 5 & 0 & 1 & 1 & 1 & 1 & 1 \\ \hline 6 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Plus généralement, dans un graphe de  $N$  sommets, la somme booléenne suivante donne la fermeture transitive du graphe :

$$I \dot{+} M \dot{+} M^{[2]} \dot{+} M^{[3]} \dot{+} \dots \dot{+} M^{[N-1]} = (I \dot{+} M)^{[N-1]}$$

Faire l'exercice 7 à la page 92

## Chapitre 4

# Chemins optimaux dans un graphe

## 4.1 Introduction

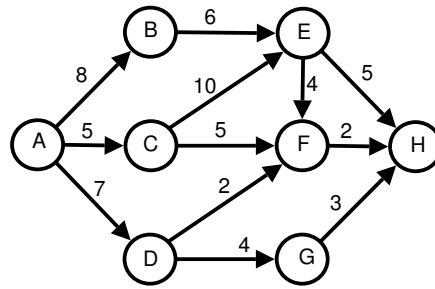


FIG. 4.1: Exemple de graphe orienté pour recherche de chemins optimaux

Le but est de rechercher le plus court chemin entre les sommets  $A$  et  $H$ .

Énumération des chemins :

$(A, B, E, H)$	$\rightarrow 19$	
$(A, B, E, F, H)$	$\rightarrow 20$	
$(A, C, E, H)$	$\rightarrow 20$	
$(A, C, E, F, H)$	$\rightarrow 21$	
$(A, C, F, H)$	$\rightarrow 12$	
$(A, D, F, H)$	$\rightarrow 11$	$\leftarrow$ Chemin de longueur minimale
$(A, D, G, H)$	$\rightarrow 19$	

Nous voyons donc que le plus court chemin est  $\mu = (A, D, F, H)$  et que  $l(\mu) = 11$

## 4.2 Algorithme de Dijkstra

Cet algorithme a pour but de réaliser la recherche des plus courts chemins entre un sommet donné et tous les autres sommets du graphe<sup>1</sup>

Algorithme :

- $G = \langle X, \Gamma \rangle$  le graphe
- $X = \{1, 2, \dots, N\}$  l'ensemble des sommets du graphe
- $N$  le nombre de sommets
- $S$  est l'ensemble des sommets traités
- $\bar{S}$  l'ensemble des sommets non traités
- $P$  la liste des prédécesseurs
- $\Pi$  la liste des distances

Hypothèse :

Le sommet de départ est le sommet 1

a) Initialisation :

- $S = \{1\}$
- $\bar{S} = X - S = \{2, 3, \dots, N\}$
- $\Pi_{(1)} = 0$
- $\Pi_{(i)} = \begin{cases} l_{1i} & \text{si } i \in \Gamma_1 \\ +\infty & \text{sinon} \end{cases}$
- $P_{(i)} = \begin{cases} 1 & \text{si } i \in \Gamma_1 \\ -1 & \text{sinon} \end{cases}$

b) Solution :

- On sélectionne le sommet  $j \in \bar{S}$  tel que  $\Pi_{(j)} = \min (\Pi_{(i)})$  avec  $i \in \bar{S}$
- $\bar{S} = \bar{S} - \{j\}$
- $S = S + \{j\}$
- Si  $\bar{S} = \emptyset$  alors Fin  
Sinon aller en (c)

c) Calcul des distances :

- Pour tout  $i \in \Gamma_j$  et  $i \in \bar{S}$  Faire
  - $\Pi_{(i)} = \min (\Pi_{(i)}, \Pi_{(j)} + l_{ji})$
  - Si  $\Pi_{(i)}$  modifié alors  $P_{(i)} = j$
- retourner en (b)

---

<sup>1</sup>Orienté ou pas, mais ayant des arcs ayant un coût uniquement supérieur à 0

Exemple :

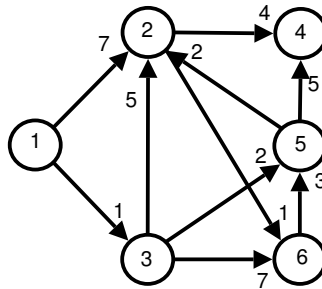


FIG. 4.2: Exemple de graphe orienté pour exemple de recherche de chemins optimaux

$S.S^1$	$S$	$\Pi_{(1)}$	$\Pi_{(2)}$	$\Pi_{(3)}$	$\Pi_{(4)}$	$\Pi_{(5)}$	$\Pi_{(6)}$	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$
1	1	0	7	1	$+\infty$	$+\infty$	$+\infty$	-1	1	1	-1	-1	-1
3	1, 3	0	6	1	$+\infty$	3	8	-1	3	1	-1	3	3
5	1, 3, 5	0	5	1	8	3	8	-1	5	1	5	3	3
2	1, 3, 5, 2	0	5	1	8	3	6	-1	5	1	5	3	2
6	1, 3, 5, 2, 6	0	5	1	8	3	6	-1	5	1	5	3	2
4	1, 3, 5, 2, 6, 4	0	5	1	8	3	6	-1	5	1	5	3	2

$$\Pi_{(6)} = 6$$

$$P_{(6)} = 2$$

$$P_{(2)} = 5$$

$$P_{(5)} = 3$$

$$P_{(3)} = 1$$

$$\Pi_{(6)} = l_{13} + l_{35} + l_{52} + l_{26}$$

$$\mu = (1, 3, 5, 2, 6)$$

$$l_{(\mu)} = \Pi_{(6)} = 6$$

Faire l'exercice 8 à la page 96

### 4.3 Algorithme de Ford

Cet algorithme permet la recherche du plus court comme du plus long chemin entre un sommet donné et les autres sommets d'un graphe<sup>1</sup>

Algorithme :

- $G = \langle X, \Gamma \rangle$  le graphe
- $X = \{1, 2, \dots, N\}$  l'ensemble des sommets du graphe
- $N$  le nombre de sommets
- $P$  la liste des prédécesseurs
- $\Pi$  la liste des distances

Hypothèse :

Le sommet de départ est le sommet 1

a) Initialisation :

- $\Pi_{(1)} = 0$
- Pour  $i$  allant de 2 à  $N$ 
  - $\Pi_{(i)} = +\infty$
- Pour  $i$  allant de 1 à  $N$ 
  - $P_{(i)} = -1$

b) Calcul :

- Pour  $i$  allant de 2 à  $N$  faire
  - $\Pi_{(i)} = \min(\Pi_{(i)}, \min(\Pi_{(j)} + l_{ij}))$  avec  $j \in \Gamma_i^{-1}$
  - Si  $\Pi_{(i)}$  modifié alors
    - $P_{(i)} = j$

c) Retour :

Recommencer l'étape (b) jusqu'à ce que les  $\Pi_{(i)}$  se stabilisent<sup>2</sup>

Exemple :

Voir la figure 4.2 à la page 42

Exemple pour  $\Pi_{(2)}$  à la première itération de l'algorithme<sup>3</sup>

$\Pi_{(2)} = \min(\Pi_{(2)}, \min(\Pi_{(1)} + l_{12}, \Pi_{(3)} + l_{32}, \Pi_{(5)} + l_{52}))$  qui donne en fait :

$\Pi_{(2)} = \min(\infty, \min(7, \infty, \infty))$

Étape	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	$\Pi_1$	$\Pi_2$	$\Pi_3$	$\Pi_4$	$\Pi_5$	$\Pi_6$
Initialisation	-1	-1	-1	-1	-1	-1	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1 <sup>ère</sup> itération	-1	1	1	2	3	2	0	<b>7</b>	1	11	3	8
2 <sup>ème</sup> itération	-1	5	1	5	3	2	0	4	1	8	3	6
3 <sup>ème</sup> itération	-1	5	1	5	3	2	0	4	1	8	3	6

<sup>1</sup>Orienté ou pas et comportant des arcs avec des coûts négatifs ou positifs

<sup>2</sup>C'est à dire que la dernière ligne doit être identique à sa ligne précédente

<sup>3</sup>En gras dans le tableau ci-dessous

Remarques :

– Pour calculer les plus longs chemins il convient de modifier l’algorithme de Ford de la façon suivante :

a) Initialisation :

- $\Pi_{(1)} = 0$
- Pour  $i$  allant de 2 à  $N$ 
  - $\Pi_{(i)} = -\infty$
- Pour  $i$  allant de 1 à  $N$ 
  - $P_{(i)} = -1$

b) Calcul :

- Pour  $i$  allant de 2 à  $N$  faire
  - $\Pi_{(i)} = \max(\Pi_{(i)}, \max(\Pi_{(j)} + l_{ij}))$  avec  $j \in \Gamma_i^{-1}$
  - Si  $\Pi_{(i)}$  modifié alors
    - $P_{(i)} = j$

c) Retour :

Recommencer l’étape (b) jusqu’à ce que les  $\Pi_{(i)}$  se stabilisent

– L’existence d’une solution repose sur l’absence de circuit absorbant<sup>1</sup> dans le graphe.  
 $u_1 : i \rightarrow k$  n’empruntant pas  $\omega$

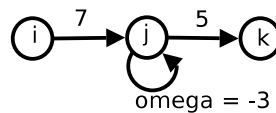


FIG. 4.3: Contre-exemple de graphe pour l’algorithme de Ford

$$l_{(u_1)} = 7 + 5 = 12$$

$u_2 : i \rightarrow k$  empruntant une fois  $\omega$

$$l_{(u_2)} = 9$$

On voit bien que  $l_{(u_2)} < l_{(u_1)}$ , ce qui provoque une boucle infinie de l’algorithme.

Faire l’exercice 9 à la page 98

---

<sup>1</sup>Circuit de longueur négative

### 4.4 Algorithme de Floyd

L'algorithme permet de recherche les plus courts où les plus longs chemins sur un graphe orienté ou non avec des longueurs d'arcs positives ou négatives entre tout couple de sommets.

Algorithme :

- $G = \langle X, U \rangle$  le graphe
- $X = \{1, 2, \dots, N\}$  l'ensemble des sommets du graphe
- $U$  l'ensemble des arcs
- $N$  le nombre de sommets
- $L$  la matrice résultat<sup>1</sup>

a) initialisation :

- $l_{ij}^{(0)} = \begin{cases} \text{longueur de l'arc } ij & \text{si } (i, j) \in U \\ +\infty & \text{sinon} \end{cases}$
- $l_{ij}^{(0)} = 0$

b) calcul :

- Pour  $k$  allant de 1 à  $N$  faire
  - $l_{ij}^k = \min(l_{ij}^{k-1}, l_{ik}^{k-1} + l_{kj}^{k-1})$

Exemple :

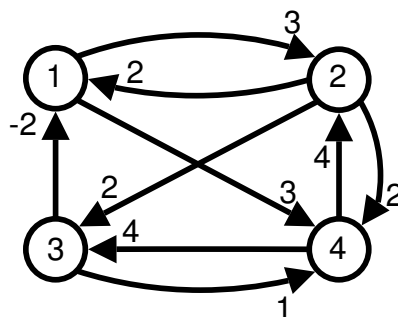


FIG. 4.4: Graphe orienté pour application de l'algorithme Floyd

$$L^{(0)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & +\infty & 3 \\ 2 & 2 & 0 & 2 & 2 \\ 3 & -2 & +\infty & 0 & 1 \\ 4 & +\infty & 4 & 4 & 0 \end{array}$$

<sup>1</sup> $l_{ij}$  étant la longueur du chemin entre le sommet  $i$  et le sommet  $j$

Pour la case 3,2 de la matrice avec  $k = 1$  :

$$l_{32}^{(1)} = \min(l_{32}^{(0)}, l_{31}^{(0)} + l_{12}^{(0)}) = \min(+\infty, -2 + 3) = 1$$

$$L^{(1)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & +\infty & 3 \\ 2 & 2 & 0 & 2 & 2 \\ 3 & -2 & 1 & 0 & 1 \\ 4 & +\infty & 4 & 4 & 0 \end{array}$$

Pour la case 1,3 de la matrice avec  $k = 2$  :

$$l_{13}^{(2)} = \min(l_{13}^{(1)}, l_{12}^{(1)} + l_{23}^{(1)}) = \min(+\infty, 3 + 2) = 5$$

$$L^{(2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 3 \\ 2 & 2 & 0 & 2 & 2 \\ 3 & -2 & +\infty & 0 & 1 \\ 4 & 6 & 4 & 4 & 0 \end{array}$$

$$L^{(3)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 3 & -2 & +\infty & 0 & 1 \\ 4 & 2 & 4 & 4 & 0 \end{array}$$

Et  $L^{(4)} = L^{(3)}$

## Chapitre 5

# Problèmes d'ordonnancement

## 5.1 Introduction

Exemple à utiliser pour l'introduction, et les méthodes d'ordonnancement MPM & PERT  
Soit un projet découpé de la façon suivante :

Désignation de la tâche $i$	Libellé	Durée $d_{ij}$	Tâches pré-alables $k \in I_i$
a	Obtention d'un permis d'exploitation	4	-
b	Etablissement d'une piste de 6km	6	a
c	Transport et installation de deux sondeuses	0.5	b
d	Création de bâtiments provisoires pour le bureau et le logement d'ouvriers	1	b
e	Goudronnage de la piste	2	b
f	Adduction d'eau	3	b
g	Campagne de sondage	8	c,d
h	Forage et équipement de trois puits	6	e,f,g
i	Transport et installation au fond du matériel d'exploitation	1	h,j
j	Construction de bureaux et de logements en dur	8	e,f,g
k	Traçage et aménagement du fond	12	h,j
i	Construction d'une laverie	8	h,j

1<sup>re</sup> étape :

- Identifier les tâches
- Contraintes des tâches
- Durée des tâches

2<sup>me</sup> étape : Résoudre un problème d'ordonnancement en calculant les paramètres

3<sup>me</sup> étape : Utiliser une méthode MPM ou PERT

## 5.2 Méthode MPM

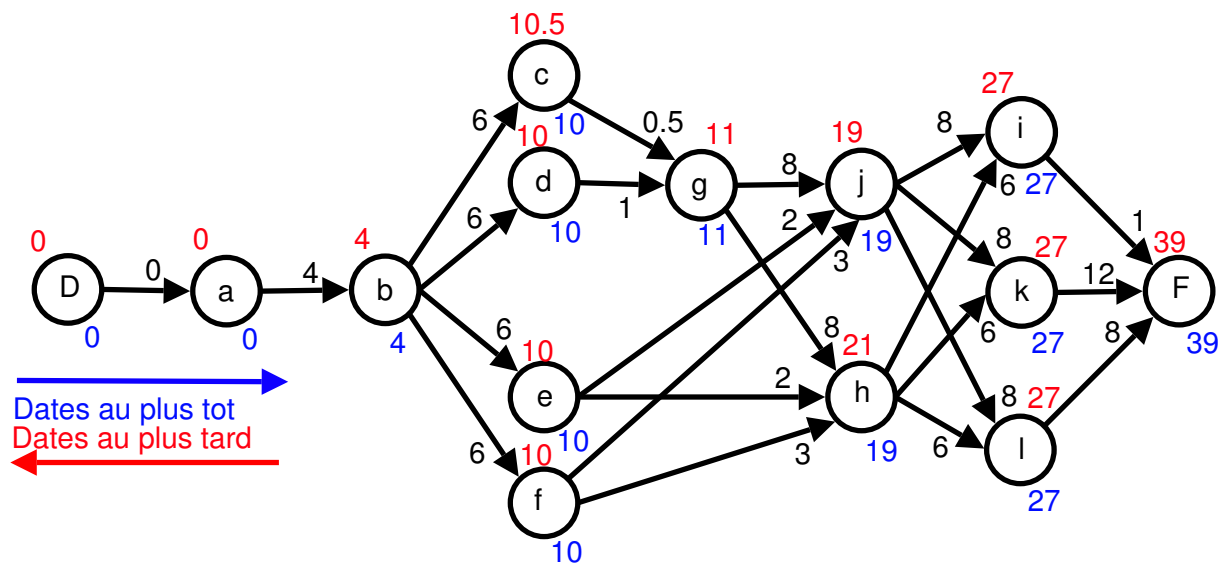


FIG. 5.1: Graphe déterminé à partir de la méthode MPM

Calcul des dates au plus tôt des différentes tâches :<sup>1,2</sup>

$$t_j = \max(t_i + d_i) \text{ avec } i \in \Gamma_j^{-1} \text{ et } d_i \text{ est la durée de la tâche } i$$

Calcul des dates au plus tard des différentes tâches :

$$t_j^* = \min(t_k^* - d_j) \text{ avec } k \in \Gamma_j$$

Chemin critique :

Le chemin critique d'un ordonnancement est formé par la succession des tâches déterminantes pour la durée totale du projet. C'est à dire les tâches pour lesquelles un retard éventuel se répercute automatiquement sur la date de réalisation du projet.

Les méthodes de détermination sont les suivantes :

- Appliquer un algorithme de recherche des plus longs chemins entre le sommet  $D$  et les autres sommets<sup>3</sup>
- Utiliser les dates au plus tôt en retrouvant la composition du plus long chemin entre le sommet  $D$  et  $F$
- En regardant la ou les dates au plus tôt sont les dates au plus tard

<sup>1</sup>Date pour laquelle les tâches précédentes sont toutes terminées

<sup>2</sup>C'est donc la date maximale à laquelle se terminent les prédecesseurs de la tâche en question

<sup>3</sup>L'algorithme de Ford

Marge des différentes tâches :

1. Marge totale d'une tâche

C'est le délai dont on peut retarder cette tâche sans affecter la date d'achèvement du projet.

$$M_j = t_j^* - t_j$$

2. Marge libre d'une tâche

C'est le délai dont on peut retarder cette tâche sans affecter la date de réalisation au plus tôt des tâches suivantes.

$$m_j = \min(t_k - t_j - d_j) \text{ avec } k \in \Gamma_j$$

3. Marge certaine d'une tâche

C'est le moment où les tâches précédentes commencent à leur date au plus tard et les tâches suivantes à leur date au plus tôt.

$$\mu = \min(t_k - \max(t_i^* + d_i) - d_j) \text{ avec } k \in \Gamma_j \text{ et } i \in \Gamma_j^{-1}$$

Faire l'exercice 12 à la page 102

### 5.3 Méthode PERT

Nous utiliserons le même énoncé que donné page 48

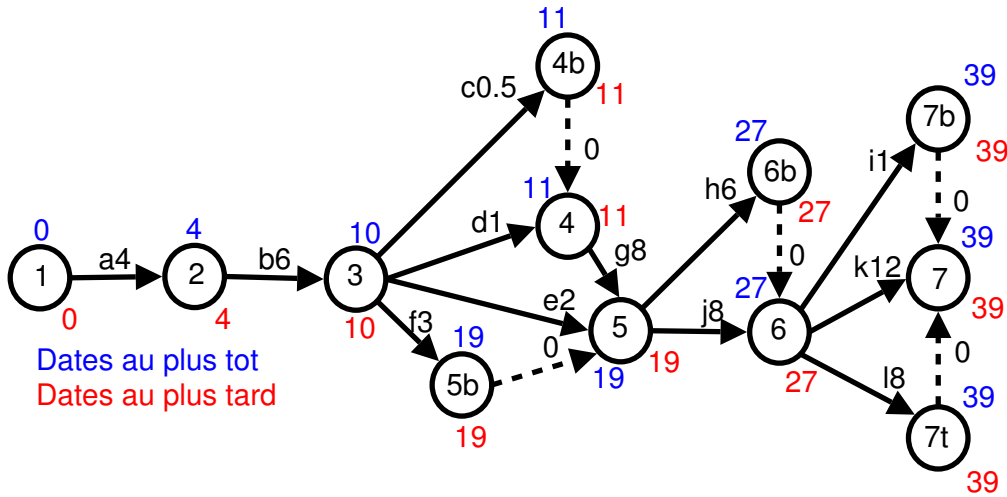


FIG. 5.2: Graphe déterminé à partir de la méthode PERT

Calcul des dates au plus tôt des différentes étapes :

$$t_j = \max(t_i + d_{ij}) \text{ avec } i \in \Gamma_j^{-1} \quad 1$$

Calcul des dates au plus tard des différentes étapes :

$$t_j^* = \min(t_k^* - d_{jk}) \text{ avec } k \in \Gamma_j \quad 2$$

Chemin critique :

1. Appliquer un algorithme permettant de déterminer le plus long chemin entre l'étape de départ et l'étape de fin<sup>3</sup>
2. Utiliser le calcul des dates au plus tôt

$$\mu = (a, b, d, g, j, k)$$

Marge des différentes tâches :

1. Marge totale d'une tâche  
 $M_j = t_j^* - t_i - d_{ij}$
2. Marge libre d'une tâche  
 $m_j = t_j - t_i - d_{ij}$
3. Marge certaine d'une tâche  
 $\mu = t_j - t_i^* - d_{ij}$

Faire l'exercice 13 à la page 104

<sup>1</sup>La date au plus tôt d'un sommet avec un ou plusieurs sommets fictifs est égale au maximum des dates au plus tôt.  
<sup>2</sup>La date au plus tard d'un sommet avec un ou plusieurs arcs fictifs avec l'arc allant du sommet réel au sommet fictif, on prends le minimum des deux dates au plus tard  
<sup>3</sup>Ford

## Chapitre 6

# Flot maximal sur un réseau de transport

## 6.1 Définitions

### 6.1.1 Réseau de transport

Un réseau de transport est un graphe  $G = \langle X, U \rangle$  dans lequel les deux conditions suivantes sont respectées :

- Il existe deux sommets uniques :

**Le sommet source** S tel que  $\Gamma_S^{-1} = \emptyset$

**Le sommet puits** T tel que  $\Gamma_T = \emptyset$

- Chaque arc doit être muni d'un nombre  $C_u \geq 0$  appelé capacité de l'arc<sup>1</sup>

*Exemple*

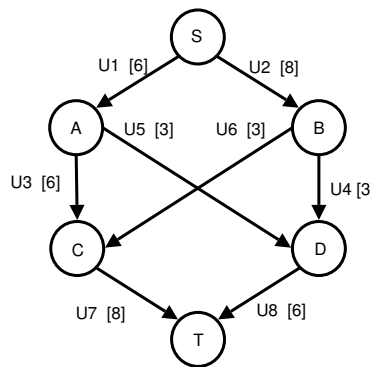


FIG. 6.1: Exemple de réseau de transport

### 6.1.2 Flot dans un réseau de transport

Soit  $G = \langle X, U \rangle$  un réseau de transport, et  $G_0 = \langle X, U^0 \rangle$  un graphe déduit de  $G$  avec un arc du sommet puits vers le sommet source.

*Exemple*

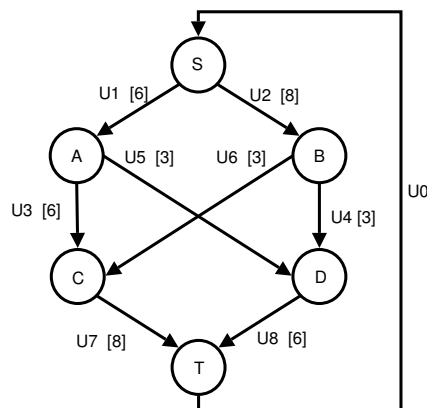


FIG. 6.2: Exemple de réseau de transport avec arc retour

<sup>1</sup>Noté entre crochets sur les graphes

$\varphi' = [\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_m]$  est un flot sur  $G^0$  si et seulement si :

1. La somme des flots sortants est égale à la somme des flots entrants<sup>1</sup>

$$\sum_{u \in \omega^+(i)} \varphi u = \sum_{u \in \omega^-(i)} \varphi u$$

2. Aux sommets  $S$  et  $T$  la somme des flots sortants sur  $S$  est égale à la somme des flots entrants sur  $T$  :

$$\sum_{u \in \omega^+(S)} \varphi u = \sum_{u \in \omega^-(T)} \varphi u \text{ avec } \varphi_u = \varphi_0 \quad ^2$$

*Exemple*

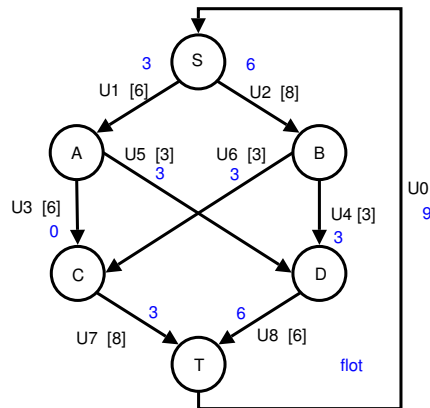


FIG. 6.3: Exemple de réseau de transport avec arc retour et flots

$\varphi' = [\varphi_0, \varphi_1, \dots, \varphi_8] = [9, 3, 6, 0, 3, 3, 3, 3, 6]$   
 $\varphi'$  est un flot sur  $G^0$  car il respecte les règles.

Faire les exercices 14 à la page 105 et 15 à la page 106

<sup>1</sup>Cette propriété doit être vérifiée pour chaque sommet de  $G^0$  exception faite des sommets  $S$  et  $T$   
<sup>2</sup> $\varphi_u$  est le flot sur l'arc retour

## 6.2 Algorithme de Ford-Fulkerson

### 6.2.1 Graphe d'écart

Soit  $G = \langle X, U \rangle$  un réseau de transport et  $\varphi = [\varphi_1, \varphi_2, \dots, \varphi_m]$  un flot sur  $G$ .

$$\bar{G}(\varphi) = \langle X, \bar{U}(\varphi) \rangle$$

L'ensemble  $\bar{U}(\varphi)$  est déterminé de la façon suivante :

- à chaque arc  $u^+ = (i, j)$  avec  $u \in U$ , on associe au plus deux arcs de  $\bar{G}(\varphi)$ 
  - $u^+ = (i, j)$  si  $\varphi_u < C_u$
  - $u^- = (j, i)$  si  $\varphi_u > 0$
- On associe une capacité à chaque arc de  $\bar{G}(\varphi)$ 
  - $C_u - \varphi_u$  si  $\varphi_u < C_u$
  - $\varphi_u$  si  $\varphi_u > 0$

Graphe d'écart déduit du réseau de transport représenté sur la figure 6.3 à la page 54 :

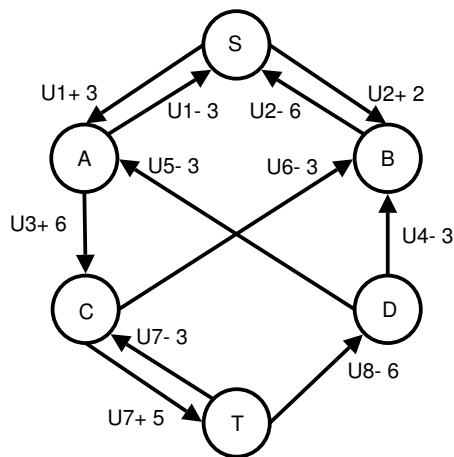


FIG. 6.4: Graphe d'écart déduit du réseau de transport représenté sur la figure 6.3

### 6.2.2 Énoncé de l'algorithme

Soit  $G = \langle X, U \rangle$  un réseau de transport

1. Initialisation

Appliquer un flot  $\varphi^0$  initial sur  $G$   $\varphi^0 = (0, 0, \dots, 0)$

Calculer le graphe d'écart associé à  $\varphi^0 \Rightarrow \bar{G}(\varphi^0)$

2. A l'itération  $k$ , soit  $\varphi^k$  le flot courant

Rechercher un chemin  $P^k$  de  $S$  à  $T$  dans le graphe  $\bar{G}(\varphi^k)$

S'il n'y a pas de chemin  $\Rightarrow$  on est arrivé à la fin (le flot  $\varphi^k$  est maximal)

Sinon aller en 3

3. Soit  $\varepsilon^k$  la capacité résiduelle du chemin  $P^k$

On va calculer  $\varphi^{k+1}$

$$\varphi_u^{k+1} = \varphi_u^k + \varepsilon^k \text{ si } u^+ \in P^k$$

$$\varphi_u^{k+1} = \varphi_u^k - \varepsilon^k \text{ si } u^- \in P^k$$

$$\varphi_0^{k+1} = \varphi_0^k + \varepsilon^k$$

Faire  $k++$  Calculer le graphe d'écart  $\bar{G}(\varphi^k)$

Retourner en 2

### 6.2.3 Application de l'algorithme

1.  $\varphi = [\varphi_1, \varphi_2, \dots, \varphi_8] = [3, \dots, 6]$  le flot initial

2.  $k = 0 \quad P^0 = (S, A, C, T)$

3. on peut écrire  $P^0 = ( \underbrace{u_1^+}_{=3}, \underbrace{u_3^+}_{=6}, \underbrace{u_7^+}_{=5} )$  <sup>12</sup>

Donc  $\varepsilon^0 = 3$

$\varphi_1^1 = 6$

$\varphi_3^1 = \varphi_3^0 + \varepsilon^0 = 3$

$\varphi_7^1 = 6$

$\varphi_0^1 = \varphi_0^0 + \varepsilon^0 = 12$

$k = 1$

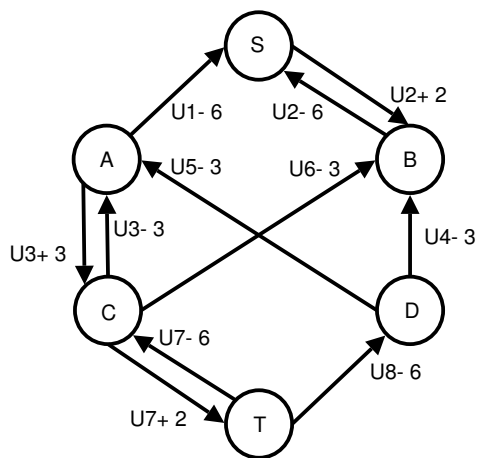


FIG. 6.5: Graphe d'écart déduit de l'application de l'algorithme de Ford-Fulkerson sur le graphe d'écart représenté sur la figure 6.4

Lorsque l'on retourne à l'étape 2 de l'algorithme, on s'aperçoit qu'il n'existe pas de chemin de  $S$  à  $T$  dans  $\bar{G}(\varphi^1)$ .

Fin de l'algorithme

$\varphi_1$  est maximal et vaut 12.

Faire l'exercice 17 à la page 108

<sup>1</sup>On prends donc le minimum des capacités sur le graphe d'écart

<sup>2</sup>C'est la valeur dont on augmente le flot

## 6.3 Propriétés sur les flots

### 6.3.1 Flot complet

Un flot est dit complet si tous les chemins de  $S$  à  $T$  comportent au moins un arc saturé ( $\varphi_u = C_u$ )

### 6.3.2 Coupe

On appelle coupe séparant  $S$  et  $T$  un ensemble d'arcs de la forme  $\omega^+(A)$  où  $A$  est un sous-ensemble de sommets<sup>1</sup> tel que  $S \in A$  et  $T \notin A$

Exemple de coupe du graphe 6.1 à la page 53 :

$$A = \{S, B\}$$

$$\omega^+(A) = \{u_1, u_4, u_6\}$$

### 6.3.3 Capacité d'une coupe

$$C(A) = \sum_{u \in \omega^+(A)} C_u$$

Où  $C(A)$  est la capacité de la coupe, et  $C_u$  est la capacité de l'arc  $u$

En reprenant l'exemple donné en 6.3.2 à la page 57 :

$$C(A) = C_{u_1} + C_{u_4} + C_{u_6} = 6 + 3 + 3 = 12$$

### 6.3.4 Flot maximal

La valeur de flot maximal de  $S$  à  $T$  est égale à la valeur minimale de toutes les coupes séparant  $S$  de  $T$ .

Faire l'exercice 17b à la page 111

---

<sup>1</sup>Un sous-ensemble contient au moins un sommet

## 6.4 Problème particulier de transport

### 6.4.1 Enoncé du problème

Soit le tableau suivant traduisant les coûts pour chaque unité transférée entre les sources (A, B, C) et les puits (1, 2, 3, 4, 5) :

	1	2	3	4	5	$a_i$
A	4	1	2	6	9	100
B	6	4	3	5	7	120
C	5	2	6	4	8	120
$c_{ij}$	40	50	70	90	90	

Où :

- $a_i$  sont les quantités disponibles
- $b_j$  sont les demandes
- $x_{ij}$  sont les quantités de l'origine  $i$  à la destination  $j$
- $n$  le nombre de destinations
- $m$  le nombre d'origines
- $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$
- $\text{Min } z = \sum_{i=1}^m \sum_{j=1}^n x_{ij} c_{ij}$
- $\sum_{i=1}^n x_{ij} = a_i$  pour  $i = 1, 2, \dots, n$
- $\sum_{i=1}^m x_{ij} = b_i$  pour  $j = 1, 2, \dots, m$

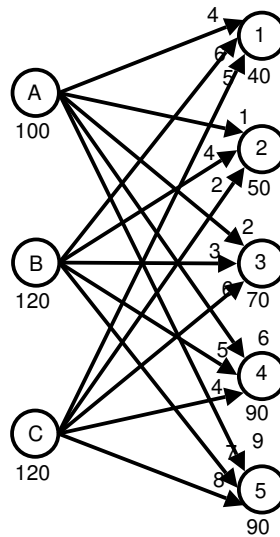


FIG. 6.6: Graphe du problème d'ordonnement particulier d'ordonnement

Le problème est que l'on souhaiterait améliorer le coût global dans ce graphe.

### 6.4.2 Algorithme du Stepping Stone

Méthode :

1. Rechercher une solution de base
2. Améliorer la solution de base

1. Recherche d'une solution de base

On assouvit les besoins des destinations en utilisant source après source.

	1	2	3	4	5	
A	40	50	10			100
B			60	60		120
C				30	90	120
	40	50	70	90	90	

Dans cette configuration  $z = 1550$ , et voici le schéma illustrant cette configuration :

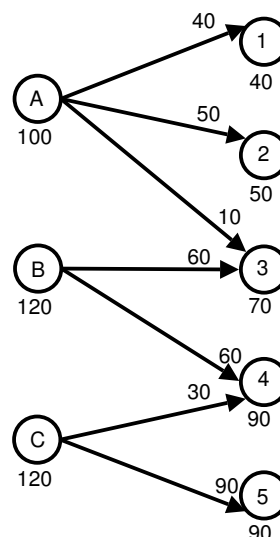


FIG. 6.7: Schéma illustrant la solution de base de l'algorithme du Stepping Stone

2. Amélioration de la solution de base

(a) calculer les coûts marginaux notés  $\delta_{ij}$  pour chaque liaison non-affectée

(b) Si tous les  $\delta_{ij}$  sont positifs ou nuls  $\Rightarrow$  Fin

Sinon, prendre le cycle de substitution  $\mu$  associé au  $\delta_{ij}$  le plus petit

– soit  $\mu^-$  l'ensembles des arcs du cycle  $\mu$  représentant les liaisons sur lesquelles la quantité transportée est diminuée.

$$q = \min [x_{ij}] \text{ avec } (i, j) \in \mu^-$$

$$x_{ij} = x_{ij} - q \text{ pour } (i, j) \in \mu^-$$

$$- \quad x_{ij} = x_{ij} + q \text{ pour } (i, j) \in \mu^+$$

(c) retour en a

Application de l'algorithme :

La méthode de détermination des coûts marginaux de l'algorithme est très graphique, il faut prendre toutes les lignes non utilisées avec la solution de base déterminée en 1, et pour chacune d'elle essayer de faire passer une unité sur celle-ci tout en préservant l'équilibre originel du graphe. A titre d'exemple, si l'on prends la liaison  $(A, 4)$ , elle n'est pas utilisée dans la solution de base, nous allons donc essayer de faire passer une unité dessus. Les modifications apportées dans le tableau sont donc :

	1	2	3	4	5	
A			-1	+1		100
B			+1	-1		120
C						120
	40	50	70	90	90	

Puis après, on détermine les coûts marginaux en additionnant un à un les coûts que les changements engendrent (voir le calcul de  $\delta_{A4}$ ).

Détermination des coûts marginaux :

- $\delta_{A4} = (+1) * 6 + (-1) * 2 + (-1) * 5 + (+1) * 3 = 2$
- $\delta_{A5} = (+1) * 9 + (-1) * 2 + (+1) * 3 + (-1) * 5 + (+1) * 4 + (-1) * 8 = 1$
- $\delta_{B1} = 1$
- $\delta_{B2} = 2$
- $\delta_{B5} = -2 \leftarrow \text{minimal}$
- $\delta_{C1} = 1$
- $\delta_{C2} = 1$
- $\delta_{C3} = 4$

On détermine maintenant le cycle de substitution de  $\delta_{B5}$  :

$$\mu = \{B5, C5, C4, B4\}$$

$$\mu^- = \{C5, B5\}$$

$$\mu^+ = \{B5, C4\}$$

$$\begin{aligned} q &= \min(x_{C5}, x_{B4}) \\ &= \min(90, 60) \\ &= 60 \end{aligned}$$

On détermine donc les modifications à effectuer au final :

$$x_{C5} = 90 - 60 = 30$$

$$x_{B4} = 60 - 60 = 0$$

$$x_{B5} = 0 + 60 = 60$$

$$x_{C4} = 30 + 60 = 90$$

On obtient donc le schéma suivant et le tableau suivant :

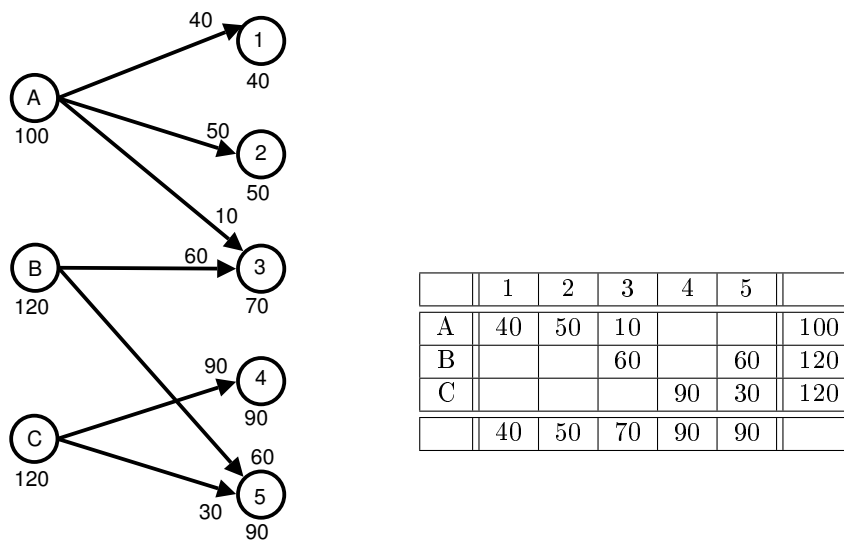


FIG. 6.8: Schéma illustrant la première optimisation de l'algorithme du Stepping Stone

On retourne maintenant à l'étape 1 de l'algorithme :

Calcul des coûts marginaux  $\delta_{ij}$  :

- $\delta_{A4} = 4$
- $\delta_{A5} = 3$
- $\delta_{B1} = 1$
- $\delta_{B2} = 2$
- $\delta_{B4} = 2$
- $\delta_{C1} = -1$
- $\delta_{C2} = -1$
- $\delta_{C3} = 2$

on détermine maintenant le cycle de substitution  $\mu$  correspondant à  $\delta_{C1}$  :

$$\mu = \{C1, A1, A3, B3, B5, C5\}$$

$$\mu^+ = \{C1, A3, B5\}$$

$$\mu^- = \{A1, B3, C5\}$$

$$\begin{aligned} q &= \min(x_{A1}, x_{B3}, x_{C5}) \\ &= \min(40, 60, 30) \\ &= 30 \end{aligned}$$

On détermine donc les modifications à effectuer au final :

$$x_{C1} = 0 + 30 = 30$$

$$x_{A3} = 10 + 30 = 40$$

$$x_{B5} = 60 + 30 = 90$$

$$x_{A1} = 40 - 30 = 10$$

$$x_{B3} = 60 - 30 = 30$$

$$x_{C5} = 30 - 30 = 0$$

On obtient donc le schéma suivant et le tableau suivant :

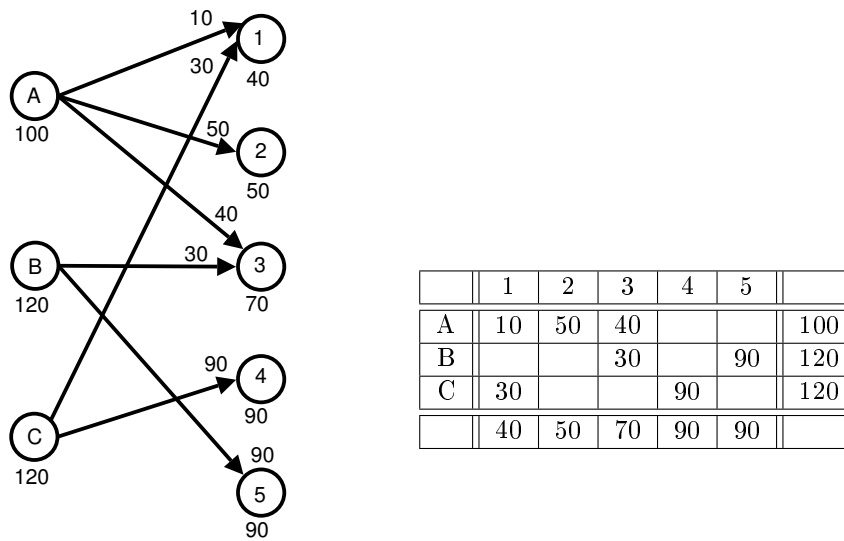


FIG. 6.9: Schéma illustrant la deuxième optimisation de l'algorithme du Stepping Stone

On retourne maintenant à l'étape 1 de l'algorithme :

- $\delta_{A4} = 3$
- $\delta_{A5} = 3$
- $\delta_{B1} = 1$
- $\delta_{B2} = 2$
- $\delta_{B4} = 1$
- $\delta_{C2} = 0$
- $\delta_{C3} = 3$
- $\delta_{C5} = 1$

Tous les  $\delta_{ij} \geq 0$ , la solution trouvée est optimale, et le coût final du transport est égale à 1400.\*

Faire l'exercice 19 à la page 114

## Chapitre 7

# Arbre couvrant de coût minimum

## 7.1 Définitions

### 7.1.1 Arbre

Un graphe  $G$  de  $N$  sommets avec  $n \geq 2$  est un arbre s'il est connexe et sans cycle.

**Exemple**

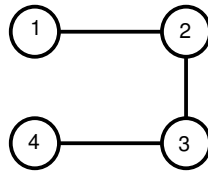


FIG. 7.1: Exemple d'arbre

### 7.1.2 Arbre couvrant de coût minimum

Soit  $G = \langle X, U \rangle$  un graphe

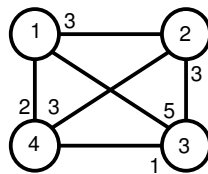
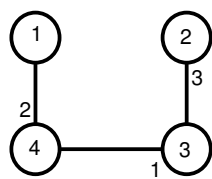


FIG. 7.2: Exemple de graphe pour recherche d'arbre couvrant

**Exemple**



$F_1 = \langle X, U_1 \rangle$  où  $W(F_1)$  est le coût de l'arbre  $F_1$ ;  $W(F_1) = 6$  ici.

FIG. 7.3: Exemple d'arbre couvrant

Soit un graphe non-orienté  $G = \langle X, U \rangle$ . Le problème de l'arbre couvrant de coût minimum est de trouver un arbre  $F^* = \langle X, U' \rangle$  de  $G$  tel que :

- $W(F^*) = \min(W(F))$
- $W(F) = \sum W(u)$  avec  $u \in U$

## 7.2 Algorithme de Kruskal

### Données

- $G = \langle X, U \rangle$  un graphe non-orienté
- $F^* = \langle X, U' \rangle$  l'arbre couvrant de coût minimal
- Une liste qui contient les arêtes du graphe  $G$  dans l'ordre croissant des poids

### Principe

$$U' = \{u_1\}^1$$

à l'étape  $k$ , c'est l'arête  $u_k \in U$  qui est lue

Si  $u_k$  ne forme pas un cycle avec les arêtes de  $U'$  alors :

$u_k$  est retenue

$$U' = U' + \{u_k\}$$

On passe à l'arête suivante

### Exemple

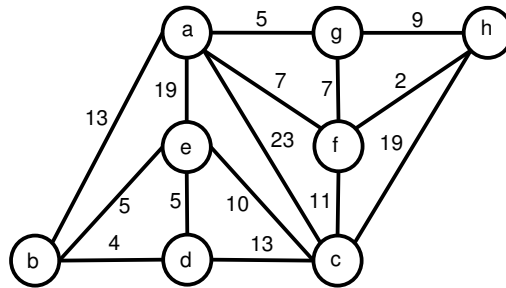


FIG. 7.4: Exemple de graphe pour application de l'algorithme de Kruskal

$u_1(h, f) = 2$	$u_6(a, f) = 7$	$u_{11}(a, b) = 13$
$u_2(b, d) = 4$	$u_7(g, f) = 7$	$u_{12}(d, c) = 13$
$u_3(a, g) = 5$	$u_8(g, h) = 9$	$u_{13}(a, e) = 19$
$u_4(b, e) = 5$	$u_9(e, c) = 10$	$u_{14}(c, h) = 19$
$u_5(d, e) = 5$	$u_{10}(c, f) = 11$	$u_{15}(a, c) = 23$

On obtient donc  $F^* = \langle X, U' \rangle$  avec  $U' = \{hf, bd, ag, be, af, ec, cf\}$

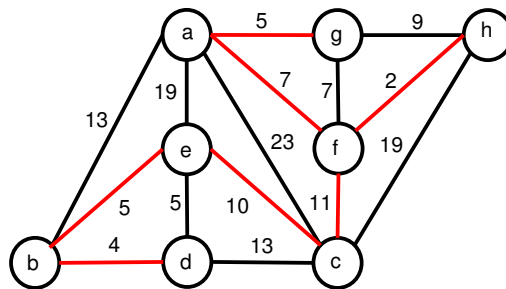


FIG. 7.5: Résultat de l'application de l'algorithme de Kruskal

$$\begin{aligned} W(F^*) &= \sum W(u) \text{ avec } u \in U' \\ &= 44 \end{aligned}$$

<sup>1</sup> $u_1$  est la première arête dans la liste des arêtes

### 7.3 Algorithme de Prim

**Données**

- $G = \langle X, U \rangle$
- $N$  le nombre de sommets
- $a$  un sommet quelconque
- $T$  la liste des sommets visités
- $\omega(T)$  est le cocycle de  $T$
- $U'$  est l'ensemble des arêtes de l'arbre

**Algorithme**

```
T=a
U'= ∅
Pour i allant de 1 à N-1 faire
    choisir une arête (x,y) de coût minimal de  $\omega(T)$ 
    U' = U' + (x,y)
    T = T + y
Fin Pour
```

Faire l'exercice 18 à la page 117

## Chapitre 8

# Complexité des algorithmes

## 8.1 Temps d'exécution d'un programme et complexité d'un algorithme

### 8.1.1 Exemple 1

On cherche à analyser le programme qui cherche l'indice du plus petit élément d'une portion d'un tableau.

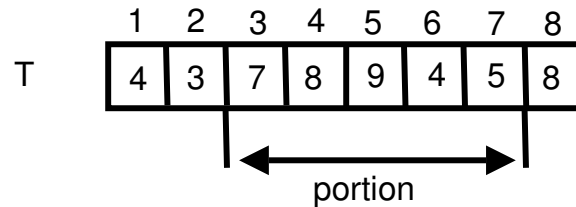


FIG. 8.1: Exemple de tableau pour analyse du temps d'exécution de l'algorithme qui retourne le plus petit indice d'un tableau

#### 8.1.1.1 Algorithme

**i** : Indice du premier élément de la portion du tableau à analyser. ( $i = 3$  dans l'exemple)

**n** : Indice du dernier élément de la portion du tableau à analyser. ( $n = 7$  dans l'exemple)

```

indice_min = i
Pour j allant de i+1 à n faire
    Si T[j] < T[indice_min] alors
        indice_min = j
    Fin Si
Fin Pour

```

#### 8.1.1.2 But

Evaluer le temps d'exécution du programme pour des données de taille  $n$ . Pour le faire, il faut compter le nombre d'instructions effectuées dans l'algorithme. Les instructions basiques sont séparées en deux parties de base :

- affectation : aff
- comparaison : comp

#### 8.1.1.3 Découpage en nombre d'instructions de base de l'algorithme

$indice\_min = i$	→ 1aff
$j = i + 1$	→ 1aff
$j > n$	→ $((n - i) + 1)$ comp
$j = j + 1$	→ $(n - i)$ aff
Si $T[j] < T[indice\_min]$	→ $(n - i)$ comp
$indice\_min = j$	→ 0 aff si $T[i]$ est minimal (le cas le plus favorable)
	→ $(n - i)$ aff si les $T[i]$ sont classés de façon décroissante (le cas le plus défavorable)
	→ $\frac{(n-i)}{2}$ aff dans le cas moyen

8.1.1.4 Somme des instructions

$$\begin{aligned}
 T(n) &= 1\text{aff} + 1\text{aff} + (n - i + 1)\text{comp} + (n - i)\text{aff} + (n - i)\text{comp} + (n - i)\text{aff} \\
 &= [2 + 2(n - i)]\text{aff} + [2(n - i) + 1]\text{comp}
 \end{aligned}$$

Si  $i = 1 \Rightarrow T(n) = 2n\text{aff} + 2(n - 1)\text{comp}$

- Le temps d'exécution est fonction<sup>1</sup>
- du nombre d'éléments à traiter dans le tableau ( $n$ )
  - du temps d'exécution d'une opération d'affectation
  - du temps d'exécution d'une opération de comparaison

Dans la pratique, il est difficile de quantifier le temps que prend un ordinateur pour faire une affectation ou une comparaison, c'est pourquoi, dans une optique de simplification, on ne fera pas de différence entre les deux, une étape<sup>2</sup> regroupe les instructions d'affectation et de comparaison. On pose  $E = \text{comp} = \text{aff}$  et obtient

$$T(n) = 2nE + (2n - 1)E = 4nE - E$$

En prenant  $E = 1$  unité de temps, on exprime  $T(n)$  en fonction de  $n$  :

$$T(n) = 4n - 1$$

8.1.1.5 Conclusion

1.  $T(n)$  nous donne une information sur le nombre d'étapes exécutées par le programme et non plus sur le temps d'exécution.
2. Le nombre d'étapes exécutées par le programme croît linéairement en fonction de la taille  $n$  des données à traiter.

8.1.2 Exemple 2

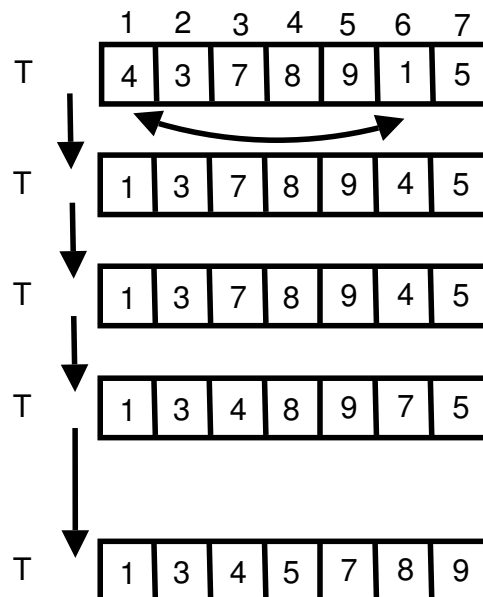


FIG. 8.2: Exemple de déroulement de l'algorithme du tri par sélection

<sup>1</sup>Dans le cas où  $i = 1$

<sup>2</sup>Une étape vaut une unité de temps

### 8.1.2.1 Algorithme

**T** : tableau

**n** : nombre d'éléments du tableau

```

Pour i allant de 1 à n-1 faire
  indice_min = i
  Pour j allant de i+1 à n faire
    Si T[j] < T[indice_min] alors
      indice_min = j
  Fin Si
Fin Pour
Tmp = T[i]
T[i] = T[indice_min]
T[indice_min] = Tmp
Fin Pour

```

### 8.1.2.2 Evaluation du nombre d'étapes de l'algorithme

On évalue le nombre d'étapes de l'algorithme ( $E = \text{aff} = \text{comp}$ ) :

-  $i = 1 \rightarrow 1E$

- 1<sup>ère</sup> itération : ( $i = 1$ )

$$\begin{aligned}
 i &> n - 1 \rightarrow 1E \\
 T(n) &= [2 + 2(n - i)]E + [2(n - i) + 1]E \\
 &= [2 + 2(n - 1)]E + [2(n - 1) + 1]E \quad (\text{si } i = 1) \\
 &= 4(n - 1)E + 3E
 \end{aligned}$$

$$\left. \begin{aligned}
 Tmp &= T[i] \\
 T[i] &= T[indice\_min] \\
 T[indice\_min] &= Tmp \\
 i &= i + 1
 \end{aligned} \right\} 4E$$

On a donc pour la première itération :  $T_1(n) = 4(n - 1)E + 8E$

- 2<sup>ème</sup> itération : ( $i = 2$ )

$$\begin{aligned}
 i &> n - 1 \rightarrow 1E \\
 T(n) &= [2 + 2(n - i)]E + [2(n - i) + 1]E \\
 &= [2 + 2(n - 2)]E + [2(n - 2) + 1]E \quad (\text{si } i = 2) \\
 &= 4(n - 2)E + 3E
 \end{aligned}$$

$$\left. \begin{aligned}
 Tmp &= T[i] \\
 T[i] &= T[indice\_min] \\
 T[indice\_min] &= Tmp \\
 i &= i + 1
 \end{aligned} \right\} 4E$$

On a donc pour la deuxième itération :  $T_2(n) = 4(n - 2)E + 8E$

-  $(n - 1)$ <sup>ème</sup> itération : ( $i = n - 1$ )

$$\begin{aligned}
 i &> n - 1 \rightarrow 1E \\
 T(n) &= [2 + 2(n - i)]E + [2(n - i) + 1]E \\
 &= [2 + 2(1)]E + [2(1) + 1]E \quad (\text{si } i = n - 1) \\
 &= 4(1)E + 3E
 \end{aligned}$$

$$\left. \begin{aligned}
 Tmp &= T[i] \\
 T[i] &= T[indice\_min] \\
 T[indice\_min] &= Tmp \\
 i &= i + 1
 \end{aligned} \right\} 4E$$

On a donc pour la  $(n - 1)^{eme}$  itération :  $T_{(n-1)}(n) = 4(1)E + 8E$

- et enfin  $i > n - 1 \rightarrow 1E$

Au final il y a au total :

$$\begin{aligned}
 T(n) &= 1E + 4(n-1)E + 8E \\
 &\quad + 4(n-2)E + 8E \\
 &\quad \vdots \\
 &\quad + 4(1)E + 8E \\
 &\quad + 1E \\
 &= 2E + S(n) \\
 S(n) &= 4(n-1)E + 8E + 4(n-2)E + 8E + \dots + 4(1)E + 8E \\
 + S(n) &= 4(1)E + 8E + 4(2)E + 8E + \dots + 4(n-1)E + 8E \\
 2S(n) &= \underbrace{(4nE + 16E) + (4nE + 16E) + \dots + (4nE + 16E)}_{(n-1)\text{fois}} \\
 &= (n-1)(4nE + 16E) \\
 S(n) &= \frac{4E(n-1)(n+4)}{2} \\
 S(n) &= 2E(n-1)(n+4)
 \end{aligned}$$

Si on néglige  $2E$  on peut dire que  $T(n) = S(n)$

$$\begin{aligned}
 T(n) &= 2E(n-1)(n+4) \\
 &= 2E(n^2 + 3n - 4) \\
 &= E(2n^2 + 3n - 8)
 \end{aligned}$$

Si  $E = 1$  unité de temps alors :

$$T(n) = 2n^2 + 6n - 8$$

Et donc si  $n$  est grand,  $T(n) = 2n^2$

On peut dire que :

- Le calcul détaillé est long et fastidieux. Un ordre de grandeur du nombre d'étapes exécutées par l'algorithme suffit.
- Pour cela on sélectionne une étape élémentaire<sup>1</sup> de l'algorithme pour obtenir un résultat d'un ordre de grandeur similaire à celui obtenu via le calcul détaillé, dans l'exemple précédent. L'étape élémentaire est  $T[j] < T[indice\_min]$  (c'est l'étape qui exécutée le plus souvent dans l'algorithme).

### 8.1.2.3 Calcul du nombre d'étapes exécutées par l'algorithme en ne tenant compte que de l'étape élémentaire

- 1<sup>ere</sup> itération : ( $i = 1$ )

$$T_1(n) = \begin{matrix} j = 2 \text{ à } n \\ (n-1)E \end{matrix}$$

- 2<sup>eme</sup> itération : ( $i = 2$ )

$$T_2(n) = \begin{matrix} j = 3 \text{ à } n \\ (n-2)E \end{matrix}$$

-  $(n - 1)^{eme}$  itération : ( $i = n - 1$ )

$$T_{n-1}(n) = \begin{matrix} j = n \text{ à } n \\ 1E \end{matrix}$$

---

<sup>1</sup>Instruction qui est exécutée le plus souvent dans le programme

Nombre total d'exécution de l'opération élémentaire :

$$\begin{aligned}
 T(n) &= T_1(n) + T_2(n) + \dots + T_{n-1}(n) \\
 &= (n-1)E + (n-2)E + \dots + 1E \\
 + &= \underline{1E + 2E + \dots + (n-1)E} \\
 2T(n) &= \underbrace{nE + nE + \dots + nE}_{(n-1) \text{ fois}} \\
 T(n) &= \frac{n(n-1)E}{2} = E\left(\frac{n^2}{2} - \frac{n}{2}\right)
 \end{aligned}$$

Si  $E = 1$  unité de temps,  $T(n) = \frac{n^2}{2} - \frac{n}{2}$

Si  $n$  est grand, on a  $T(n) = \frac{n^2}{2}$

### Définition de la complexité d'un algorithme

Soit  $A$  un algorithme et  $n$  une donnée d'entrée de  $A$ . On appelle complexité de  $A$  pour  $n$  le nombre de fois que l'opération élémentaire est exécutée lors de l'exécution de  $A$  sur  $n$ .

## 8.2 Ordre de grandeur d'un algorithme

L'ordre de grandeur d'un algorithme permet de quantifier sa complexité afin de pouvoir le comparer avec d'autres algorithmes permettant de résoudre un même problème lorsque la taille des données à traiter devient grande. Un ordre de grandeur sur la complexité de l'algorithme suffit à déterminer son efficacité.

On utilise la notation en  $O$  pour exprimer l'ordre de grandeur :

On dit que  $T(n) = O(f(n))$  s'il existe un entier  $n_0$  et une constante  $c > 0$  tel que pour tout entier  $n \geq n_0$  nous avons  $T(n) = c * f(n)$ .

### 8.2.1 Reprise de l'exemple de recherche de l'indice du plus petit élément d'un tableau

On a déterminé plus haut (8.1.1 à la page 68) que :

$T(n) = 4n - 1$  avec un décompte précis

$T(n) = n - 1$  avec le décompte de l'opération élémentaire  $T[j] < T[indice_{min}]$

On peut dire alors que  $T(n)$  est en  $O(n) \rightarrow T(n) = O(n)$

Si  $c = 2$  et  $n_0 = 1$  :

$n_0 - 1 \leq 2n$  pour  $n \leq n_0$

$f(n) = n$  donc  $T(n) = O(n)$

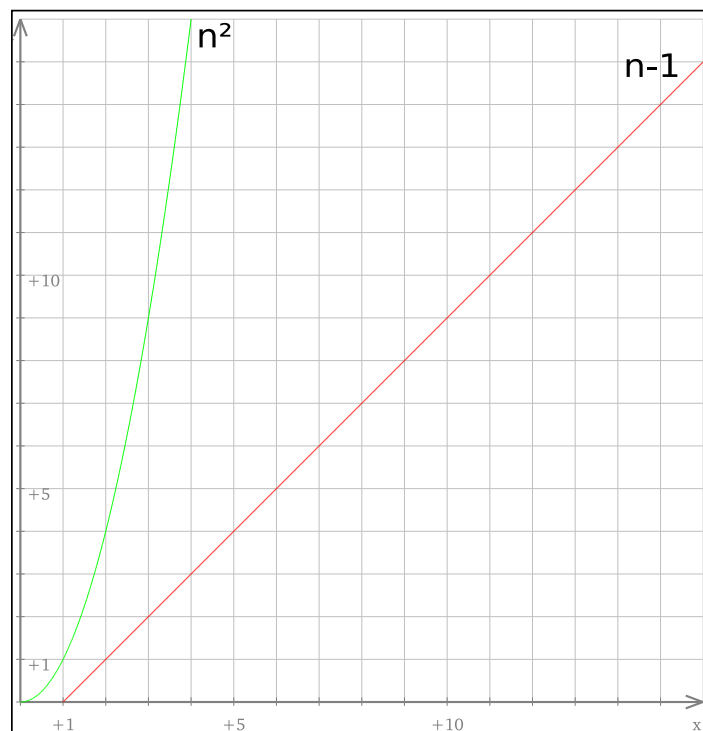


FIG. 8.3: Exemple de déroulement de l'algorithme du tri par sélection

On peut donc dire que  $T(n) = O(n^2)$

### 8.2.2 Reprise du tri par permutation

$$T(n) = \frac{n^2}{2} - \frac{n}{2}$$

$T(n) = O(n^2)$  on peut dire que  $T(n)$  est  $O(n^2)$

Si  $c = 1$  et  $n_0 = 1$

$$\frac{n^2}{2} - \frac{n}{2} \leq 2n^2 \text{ pour } n \geq n_0 \geq 1$$

### 8.2.3 Echelle de comparaison de la complexité des algorithmes

Valeur de $f(n)$	Complexité
1	constante
$\log(n)$	logarithmique
$n$	Linéaire
$n^2$	Quadratique
$n^k$	Polinomiale <sup>1</sup>
$2^n$	Exponentielle

Temps mis pour n éléments				
Algorithme	Complexité	100	10000	100000
Tri par sélection	$n^2$	1,75ms	17s	17min
Tri rapide	$n * \log(n)$	0,5ms	100ms	1,2s
Algorithme exponentiel	$2^n$		$5 * 10^{2969}$ jours	$\infty$

<sup>1</sup>Quand  $k \geq 3$

## Chapitre 9

# Les réseaux de Pétri

## 9.1 Définition

Un réseau de Pétri est un graphe représentant les relations entre trois ensembles d'éléments :

1. places
2. transitions
3. arcs

**exemple**

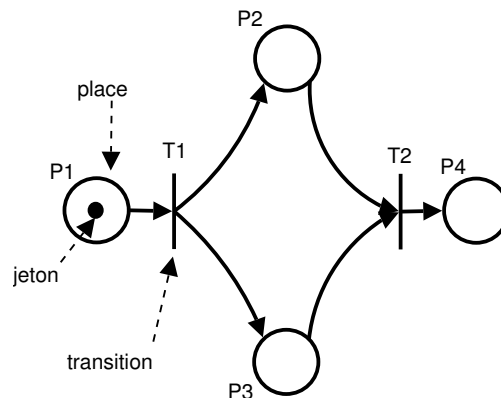


FIG. 9.1: Exemple de réseau de Pétri

Les réseaux de Pétri sont utilisés pour modéliser la dynamique d'un système. Le marquage dans le réseau de Pétri se fait au moyen d'un jeton (correspondant à un nombre entier que l'on va attribuer à chaque place).

On représente communément un marquage par un vecteur colonne  $M$  :

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Une transition est validée ou franchissable si toutes ses places d'entrée sont marquées (toutes les marques contiennent au moins un jeton). Dans l'exemple,  $T_1$  est franchissable alors que  $T_2$  ne l'est pas. Pour franchir une transition on utilise la règle suivante :

- on retire un jeton de chacune des places d'entrée
- on ajoute un jeton dans chacune des des places de sortie

Si on franchit  $T_1$  on obtient donc le marquage  $M_1$  suivant :

$$M_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

## 9.2 Utilisation d'un réseau de Pétri

Les réseaux de Pétri sont souvent utilisés dans des systèmes informatiques (surtout quand cela utilise du parallélisme, synchronisation de processus, ressources partagées) ou dans les systèmes automatiques.

### Exemple de modélisation de réseau de Pétri

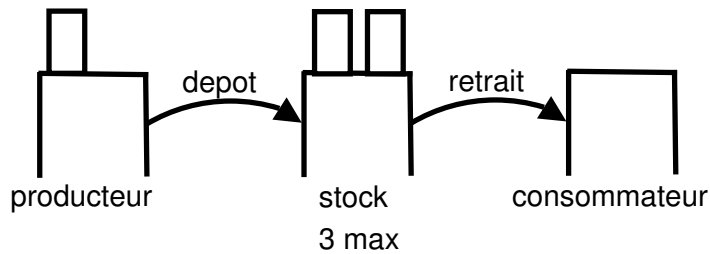


FIG. 9.2: Exemple à modéliser en réseau de Pétri

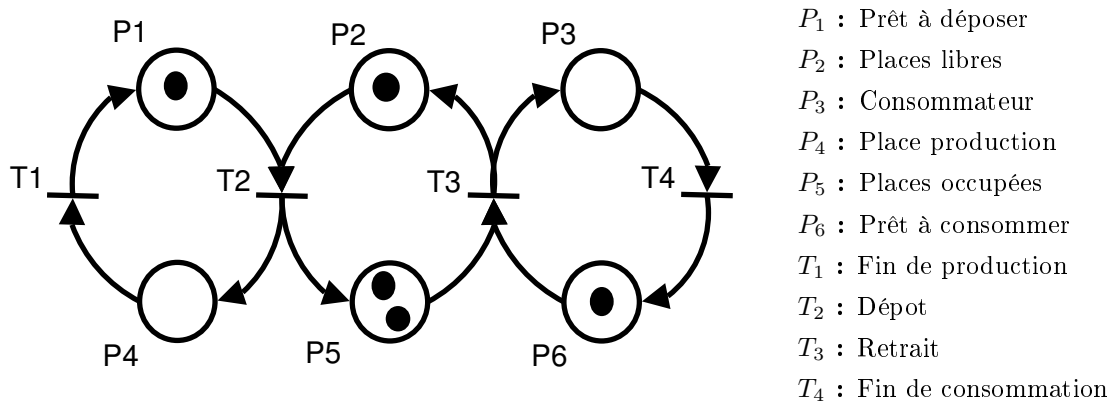


FIG. 9.3: Exemple de modélisation de réseau de Pétri

### 9.3 Propriétés des réseaux de Pétri

Vérifier les propriétés d'un réseau de Pétri permet de voir s'ils comportent des erreurs de modélisation.

#### 9.3.1 Réseau de Pétri vivant

Un réseau de Pétri vivant est tel qu'à partir du marquage initial et de tout marquage consécutif, toute transition du réseau puisse être franchie.

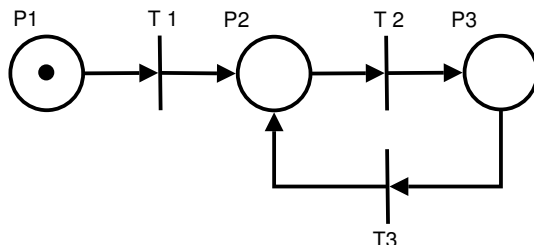


FIG. 9.4: Exemple de réseau de Pétri non-vivant

$T_1$  n'est plus franchie au bout d'un certain temps alors le réseau n'est pas vivant.

#### 9.3.2 Réseau de Pétri pseudo-vivant

Un réseau de Pétri dit est pseudo-vivant si pour chacun des marquages accessibles, il existe au moins une transition pouvant être franchie.

Si l'on prends l'exemple sur la figure 9.4 (page 78), on peut dire qu'il est pseudo-vivant car  $T_2$  et  $T_3$  sont franchies.

Autre exemple :

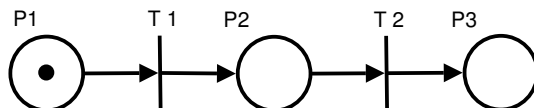


FIG. 9.5: Exemple de réseau de Pétri non-pseudo-vivant

Lorsque  $P_3$  est marquée, on arrive à une situation de blocage, car aucune transition ne peut être franchie.

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{T_1} M_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{T_2} M_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

### 9.3.3 Réseau de Pétri borné

Un réseau de Pétri est borné si, pour tout marquage consécutif du marquage initial, le nombre de marques contenues dans chaque place du réseau est inférieur au nombre  $k$ .

Exemple non-borné :

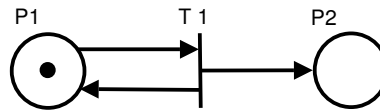


FIG. 9.6: Exemple de réseau de Pétri non-borné

La valeur de  $P_2$  n'arrête pas d'augmenter.

Exemple borné :

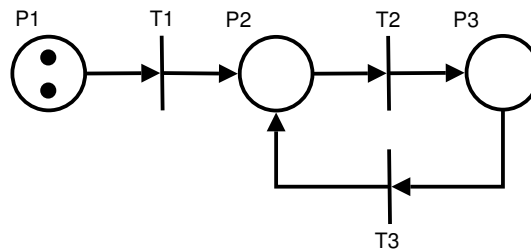


FIG. 9.7: Exemple de réseau de Pétri borné

### 9.3.4 Réseau de Pétri avec conflits

#### 9.3.4.1 Conflits structurels

Un conflit structurel correspond à un ensemble d'au moins deux transitions  $T_1$  et  $T_2$  qui ont en commun  $k = \langle P_i, |T_1, \dots, T_2| \rangle$

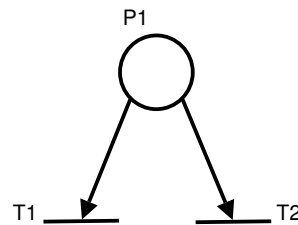


FIG. 9.8: Exemple de conflit structurel dans un réseau de Pétri borné

### 9.3.4.2 Conflit effectif

Un conflit effectif est l'existence d'un conflit structurel  $k$  et d'un marquage  $M$  tel que le nombre de marques dans  $P_i$  est inférieur au nombre de transitions de  $P_i$  qui sont validées par  $M$  noté  $k^e = \langle P_i, |T_1, \dots, T_2|, M \rangle$

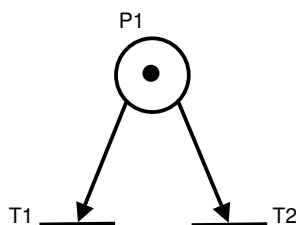


FIG. 9.9: Exemple de conflit effectif dans un réseau de Pétri borné

## 9.3.5 Recherche de propriétés dans un réseau de Pétri

### 9.3.5.1 Méthode du graphe des marquages

On reprend l'exemple vu sur la figure 9.4 à la page 78, le graphe des marquages déduit de ce graphe est le suivant :

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{T_1} M_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \xrightleftharpoons[T_2]{T_3} M_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Propriétés déduites :

- Il n'est pas vivant car  $T_1$  n'est pas franchissable qu'une fois
- Il est pseudo-vivant car pour tous les marquages accessibles, on a au moins une transition qui est franchie
- Il est borné car le nombre de marques dans chaque place est inférieur ou égal à  $k$

### 9.3.5.2 Méthode de l'arborescence et graphe de couverture d'un réseau de Pétri

Méthode de construction de l'arborescence de couverture d'un réseau de Pétri

1. - A partir d'un marquage initial  $M_0$  qui est la racine de l'arborescence, on indique toutes les transitions validées et les marquages successeurs correspondants.
  - Si un des marquages couvre strictement  $M_0$ , on met  $\omega$  pour chacune des composantes supérieures aux composantes correspondantes de  $M_0$ .
2. Pour chaque nouveau marquage  $M_i$  de l'arborescence, on fait soit le 2.a ou le 2.b
  - (a) S'il existe sur le chemin de  $M_0$  à  $M_i$ <sup>1</sup> un marquage  $M_j = M_i$  alors  $M_i$  n'a pas de successeur.
  - (b) Sinon, on prolonge l'arborescence en ajoutant tous les successeurs de  $M_i$ . Pour chaque successeur  $M_k$  de  $M_i$  faire :
    - Une composante  $\omega$  de  $M_i$  reste une composante  $\omega$  de  $M_k$
    - S'il existe un marquage  $M_j$  sur le chemin  $M_0$  à  $M_k$  tel que  $M_k$  couvre strictement  $M_j$ , alors on met  $\omega$  pour chacune des composantes supérieures aux composantes de  $M_j$ .

<sup>1</sup>Ce dernier exclu

Remarques :

1. On dit que le marquage  $M_1$  couvre le marquage  $M_2$  si le marquage de chaque place  $P_i$  dans  $M_1$  est supérieur ou égal à son marquage dans  $M_2$ .  
 $M_1 \geq M_2 \Leftrightarrow M_1(P_i) \geq M_2(P_i)$  pour toute place  $P_i$
2. On dit que le marquage  $M_1$  est supérieur au marquage  $M_2$ <sup>1</sup> si  $M_1$  couvre  $M_2$  avec au moins une place  $P_i$  telle que  $M_1(P_i) > M_2(P_i)$   
 $M_1 > M_2 \Leftrightarrow M_1 \geq M_2$  et il y a au moins une place  $P_i$  telle que  $M_1(P_i) > M_2(P_i)$

Exemple :

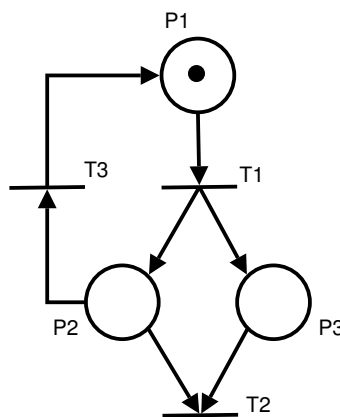


FIG. 9.10: Exemple de réseau de Pétri pour application de la méthode de construction de l'arbre de couverture

**Etape 1 :**  $M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{T_1} M_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$   
 $M_1$  ne couvre pas strictement le marquage  $M_0$

**Etape 2 :** Etape 2.2 pour  $M_1$

$$M_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{array}{l} \xrightarrow{T_2} M_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ \searrow_{T_3} \\ M_3 = \begin{bmatrix} 1 \\ 0 \\ \omega \end{bmatrix} \end{array}$$

$M_2$  n'est ni supérieur à  $M_0$  ni à  $M_1$   
 $M_3$  est supérieur à  $M_0$  mais pas à  $M_1$

**Etape 2.b pour  $M_2$  :** Aucune transition n'est validée, pas de nouveau marquage.

**Etape 2.b pour  $M_3$  :**  $M_3 = \begin{bmatrix} 1 \\ 0 \\ \omega \end{bmatrix} \xrightarrow{T_1} M_4 = \begin{bmatrix} 0 \\ 1 \\ \omega \end{bmatrix}$  Composantes de  $M_4$  :

- La composante  $\omega$  de  $M_3$  reste égale à  $\omega$  dans  $M_4$
- Il n'y a pas de marquage  $M_j$  sur le chemin de  $M_0$  à  $M_4$  tel que  $M_4 > M_j$

<sup>1</sup>On couvre strictement le marquage  $M_2$

**Etape 2 :** Etape 2.b pour  $M_4$

$$M_4 = \begin{bmatrix} 0 \\ 1 \\ \omega \end{bmatrix} \xrightarrow{T_2} M_5 = \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix}$$

$$\searrow_{T_3}$$

$$M_6 = \begin{bmatrix} 1 \\ 0 \\ \omega \end{bmatrix}$$

$M_5$  n'a pas de successeur et est une situation de blocage.

$M_3 = M_6$  donc  $M_6$  n'a pas de successeur

On obtient donc au final cet arbre de couverture :

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{T_1} M_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{T_2} M_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\searrow_{T_3}$$

$$M_3 = \begin{bmatrix} 1 \\ 0 \\ \omega \end{bmatrix} \xrightarrow{T_1} M_4 = \begin{bmatrix} 0 \\ 1 \\ \omega \end{bmatrix} \xrightarrow{T_6} M_5 = \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix}$$

Déductions :

- Les places  $P_1$  et  $P_2$  sont bornées mais la place  $P_3$  n'est pas bornée  $\rightarrow$  le réseau n'est pas borné
- Il y a une infinité de blocages liés au marquage  $M_5 \rightarrow$  le réseau n'est ni vivant ni pseudo-vivant.

Faire l'exercice 21 à la page 121

Annexe A

Exercices

### A.1 Exercice 1

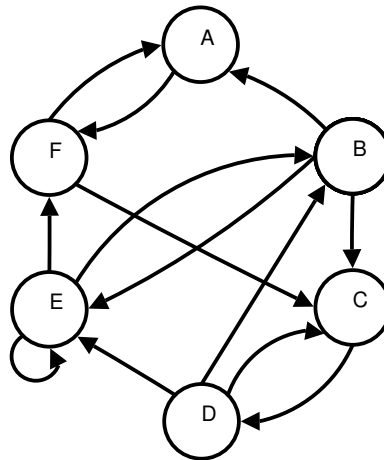


FIG. A.1: Exercice 1

#### A.1.1 Questions

1. Énumérer  $\Gamma_A, \Gamma_A^{-1}, \dots, \Gamma_F, \Gamma_F^{-1}$
2. Donner les demi-degrés intérieurs et extérieurs de chaque sommet
3. Donner un exemple de chemin simple mais non-élémentaire
4. Existe-t'il un circuit Hamiltonien ?
5. Tracer le graphe simple non-orienté  $G'$  déduit de  $G$
6.  $G'$  est-il connexe?  $G$  est-il fortement connexe?

#### A.1.2 Réponses

1. Application multivoque :

Sommet	$\Gamma_{Sommet}$	$\Gamma_{Sommet}^{-1}$
A	F	B, F
B	A, C, E	D, E
C	D	B, D, F
D	B, C, E	C
E	B, E, F	B, D, E
F	A, C	A, E

2. Demi-degrés intérieurs et extérieurs du graphe

Demi-degré de	A	B	C	D	E	F
$d^+$	1	3	1	3	2	2
$d^-$	2	2	3	1	2	2

3.  $P = \{B, C, D, B, A\}$
4.  $P = \{A, F, C, D, E, B, A\}$

5. Le graphe non-orienté  $G'$  déduit de  $G$  :

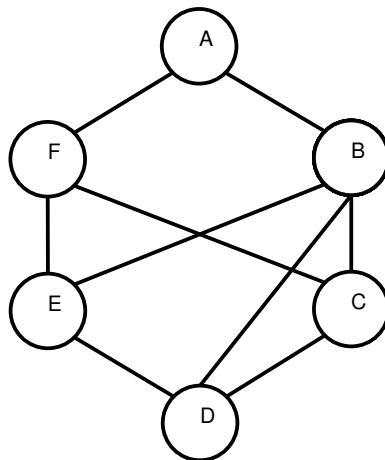


FIG. A.2: Réponse à la question 5 de l'exercice 1

6. Le fait qu'il existe un chemin Hamiltonien implique que le graphe orienté  $G$  soit fortement connexe et que le graphe non-orienté  $G'$  soit fortement connexe

## A.2 Exercice 2

### A.2.1 Questions

Soit le graphe  $G = \langle X, U \rangle$  de 6 sommets et 10 arcs. On associe à chaque arc un coût.

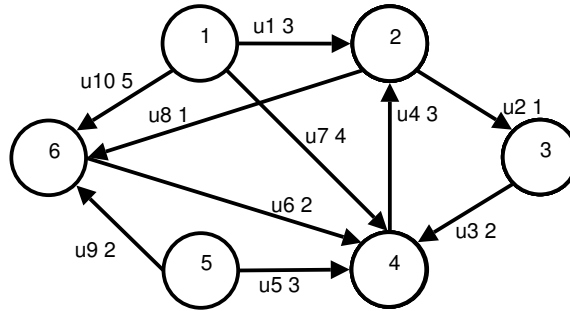


FIG. A.3: Exercice 2

1. Représenter le graphe  $G$  par une matrice d'adjacence donnant le coût de chaque arc
2. Représenter le graphe  $G$  par une liste d'adjacence donnant le coût des arcs
3. Représenter le graphe  $G$  par une matrice d'incidence donnant le coût des arcs

### A.2.2 Réponses

1.

	1	2	3	4	5	6
1	0	3	0	4	0	5
2	0	0	1	0	0	1
3	0	0	0	2	0	0
4	0	3	0	0	0	0
5	0	0	0	3	0	2
6	0	0	0	2	0	0

2.

LP : 

1	4	6	7	8	10	11
---	---	---	---	---	----	----

LC & LS : 

Coûts :	3	4	5	1	1	2	3	3	2	2	
Suivants :	2	4	6	3	6	4	2	4	6	4	X

3.

	1	2	3	4	5	6	7	8	9	10
1	3	0	0	0	0	0	4	0	0	5
2	-3	1	0	-3	0	0	0	1	0	0
3	0	-1	2	0	0	0	0	0	0	0
4	0	0	-2	3	-3	-2	-4	0	0	0
5	0	0	0	0	3	0	0	0	2	0
6	0	0	0	0	0	2	0	-1	-2	-5

## A.3 Exercice 3

### A.3.1 Questions

Soit un circuit de train électrique ayant le schéma suivant :

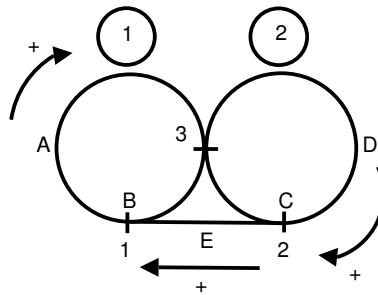


FIG. A.4: Exercice 3

Le train peut se déplacer soit en marche avant (+) soit en marche arrière (-). Le circuit a été découpé en 5 tronçons : A, B, C, D et E.

Les aiguillages permettent les passages suivants<sup>1</sup> :

- Passage de A vers B ou E (sens -) et de E vers A (sens +) et de B vers A (sens +)
- Passage de D vers C ou E (sens +) et de E vers D (sens -) et de C vers D (sens -)
- Passage de A vers B ou C (sens + et sens -), de B vers A ou D (sens - et sens +), de C vers A ou D (sens - et sens +) et de D vers B ou C (sens + et sens -)

1. Dessiner le graphe décrivant les déplacements du train sur ce circuit, en associant à chaque tronçon  $X$  deux sommets  $X^+$  et  $X^-$  suivant le sens de déplacement du train sur ce tronçon.
2. Montrer que si l'on maintient constant le sens de marche du train, ce train n'emprunte plus jamais le tronçon  $E$  au bout d'un certain temps.
  - (a) Montrer que le graphe possède quatre composantes fortement connexes  $\{C_1, C_2, C_3, C_4\}$
  - (b) Faire des observations sur la topologie du graphe réduit pour conclure que le train n'emprunte plus le tronçon  $E$  au bout d'un certain temps

<sup>1</sup>Après réalisation de l'exercice, il est préférable de n'utiliser que le schéma ci-dessus pour répondre aux questions à venir plutôt qu'aux trois points ci-dessous qui ne sont pas des modèles de clarté

### A.3.2 Réponses

1. Schéma :

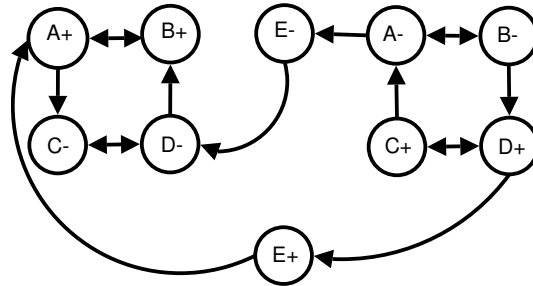


FIG. A.5: Réponse à la question 1 de l'exercice 3

2. (a) Les composantes fortement connexes sont les suivantes :
- $C_1 = \{A^+, B^+, C^-, D^-\}$
  - $C_2 = \{A^-, B^-, C^+, D^+\}$
  - $C_3 = \{E^+\}$
  - $C_4 = \{E^-\}$
- (b) Voici le graphe déduit des observations :

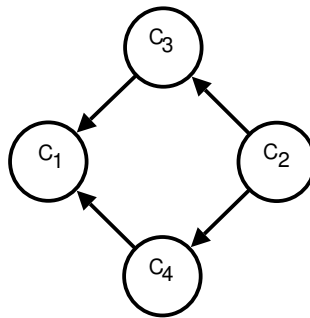


FIG. A.6: Réponse à la question 2.b de l'exercice 3

3.  $C_3$  et  $C_4$  sont des étapes transitives<sup>1</sup>. En maintenant le sens constant un certain temps, le train n'emprunte plus le tronçon E.

<sup>1</sup>états par lesquels on ne peut passer qu'une fois au plus

## A.4 Exercice 4

### A.4.1 Questions

Soit le graphe  $G = \langle X, U \rangle$  suivant :

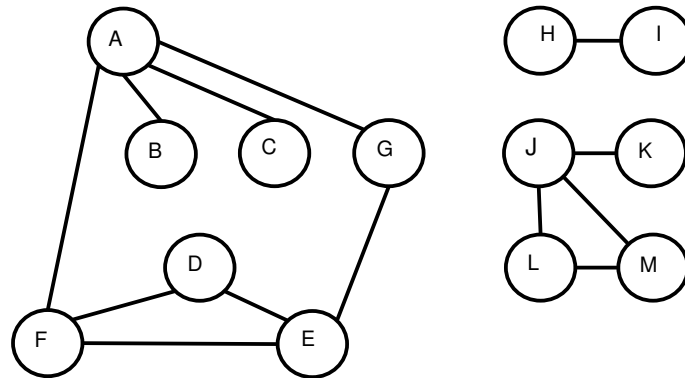


FIG. A.7: Exercice 4

On rappelle qu'un graphe non-orienté est connexe si tous les sommets du graphe à partir d'un sommet donné. On propose un algorithme de recherche des composantes connexes qui se déroule en trois phases :

1. effectuer un parcours en profondeur à partir d'un sommet quelconque du graphe  
 ⇒ l'ensemble des sommets rencontrés à l'issue de ce parcours forme une composante connexe
2. si tous les sommets du graphe ont été rencontrés  
 ⇒ fin
3. sinon, on retourne en 1 en partant d'un sommet qui n'a pas encore été exploré

Traduire cet algorithme en pseudo-code.

### A.4.2 Réponses

$M$  : Matrice d'adjacence du graphe

$N$  : Nombre de sommets

$S$  : Sommet courant

$L$  : Liste contenant le numéro de composante connexe<sup>1</sup>

$k$  : Compteur incrémentiel

ParcoursProfondeurConnexe( $S, k$ )

Début

$L[S] = k$

    Pour  $i$  allant de 1 à  $N$  faire

        Si  $M[S, i] \neq 0$  et  $L[i] = 0$  alors

            ParcoursProfondeurConnexe( $i, k$ )

        Fin Si

    Fin Pour

Fin

ProgrammePrincipal

Début

    Pour  $i$  allant de 1 à  $N$  faire

$L[i] = 0$

    Fin pour

    Pour  $i$  allant de 1 à  $N$  faire

        Si  $L[i] = 0$  alors

            ParcoursProfondeurConnexe( $i, k$ )

        Fin Si

    Fin Pour

Fin

#### Vérification

	A	B	C	D	E	F	G	H	I	J	K	L	M
Init.	0	0	0	0	0	0	0	0	0	0	0	0	0
(A, 1)	1	0	0	0	0	0	0	0	0	0	0	0	0
→ (B, 1)	1	1	0	0	0	0	0	0	0	0	0	0	0
→ (C, 1)	1	1	1	0	0	0	0	0	0	0	0	0	0
→ (F, 1)	1	1	1	0	0	1	0	0	0	0	0	0	0
→ (D, 1)	1	1	1	1	0	1	0	0	0	0	0	0	0
→ (E, 1)	1	1	1	1	1	1	0	0	0	0	0	0	0
→ (G, 1)	1	1	1	1	1	1	1	0	0	0	0	0	0
(H, 2)	1	1	1	1	1	1	1	2	0	0	0	0	0
→ (I, 2)	1	1	1	1	1	1	1	2	2	0	0	0	0
(J, 3)	1	1	1	1	1	1	1	2	2	3	0	0	0
→ (K, 3)	1	1	1	1	1	1	1	2	2	3	3	0	0
→ (L, 3)	1	1	1	1	1	1	1	2	2	3	3	3	0
→ (M, 3)	1	1	1	1	1	1	1	2	2	3	3	3	3

<sup>1</sup>Contient 0 si le sommet n'a pas encore été visité

## A.5 Exercice 5

### A.5.1 Questions

Soit le graphe  $G = \langle X, U \rangle$  de 6 sommets et 10 arcs. On associe à chaque arc un coût.

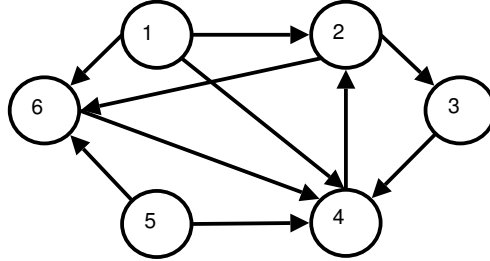


FIG. A.8: Exercice 5

1. Appliquer l'algorithme de parcours en profondeur en partant du sommet 1
2. Par rapport à la forêt d'exploration obtenue, préciser :
  - les arcs couvrants
  - les arcs avant
  - les arcs arrières
  - les arcs croisés

### A.5.2 Réponses

1.
 
$$\begin{array}{ccccccc}
 S_1 & \rightarrow & S_2 & \rightarrow & S_3 & \rightarrow & S_4 \\
 & & & & & \rightarrow & S_6 \\
 & & & & S_5 & & 
 \end{array}$$
2. *noir* : arcs couvrants, *vert* : arcs arrières, *bleu* : arcs avant, *rouge* : arcs croisés

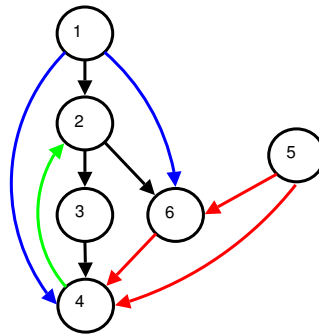


FIG. A.9: Réponse à la question 2 de l'exercice 5

## A.6 Exercice 7

### A.6.1 Questions

Soit la matrice  $M$  suivante :

$$M = \begin{array}{c|ccccc} & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\ \hline \text{A} & 0 & 1 & 0 & 1 & 0 \\ \text{B} & 0 & 0 & 1 & 0 & 0 \\ \text{C} & 0 & 0 & 0 & 0 & 1 \\ \text{D} & 0 & 0 & 1 & 0 & 1 \\ \text{E} & 0 & 0 & 0 & 0 & 0 \end{array}$$

1. Tracer le graphe associé à cette matrice
2. Calculer  $M^2$ ,  $M^3$ ,  $M^4$
3. Calculer  $A^2$ ,  $A^3$ ,  $A^4$ , existe-t'il un chemin Hamiltonien ?
4. Calculer  $M^{[2]}$ ,  $M^{[3]}$ ,  $M^{[4]}$
5. Calculer  $P = I + M + M^{[2]} + M^{[3]} + M^{[4]}$
6. Appliquer l'algorithme de WARSHALL

### A.6.2 Réponses

1.

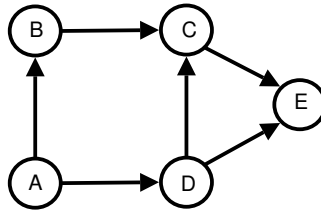


FIG. A.10: Réponse à la question 1 de l'exercice 7

2.

$$- M^2 = \begin{array}{c|ccccc} & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\ \hline \text{A} & 0 & 0 & 2 & 0 & 1 \\ \text{B} & 0 & 0 & 0 & 0 & 1 \\ \text{C} & 0 & 0 & 0 & 0 & 0 \\ \text{D} & 0 & 0 & 0 & 0 & 1 \\ \text{E} & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$- M^3 = \begin{array}{c|ccccc} & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\ \hline \text{A} & 0 & 0 & 0 & 0 & 2 \\ \text{B} & 0 & 0 & 0 & 0 & 0 \\ \text{C} & 0 & 0 & 0 & 0 & 0 \\ \text{D} & 0 & 0 & 0 & 0 & 0 \\ \text{E} & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$- M^4 = \begin{array}{c|ccccc} & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\ \hline \text{A} & 0 & 0 & 0 & 0 & 0 \\ \text{B} & 0 & 0 & 0 & 0 & 0 \\ \text{C} & 0 & 0 & 0 & 0 & 0 \\ \text{D} & 0 & 0 & 0 & 0 & 0 \\ \text{E} & 0 & 0 & 0 & 0 & 0 \end{array}$$

3.

	A	B	C		D	E	
$- A^2 =$	A	0	0	ABC, ADC		0	ADE
	B	0	0	0		0	BCE
	C	0	0	0		0	0
	D	0	0	0		0	DCE
	E	0	0	0		0	0
	A	B	C	D	E		
$- A^3 =$	A	0	0	0	0	ABCE, ADCE	
	B	0	0	0	0	0	
	C	0	0	0	0	0	
	D	0	0	0	0	0	
	E	0	0	0	0	0	
	A	B	C	D	E		
$- A^4 =$	A	0	0	0	0	0	
	B	0	0	0	0	0	
	C	0	0	0	0	0	
	D	0	0	0	0	0	
	E	0	0	0	0	0	

L'absence de valeur dans la matrice  $A^4$  indique qu'il n'existe pas de chemin Hamiltonien dans le graphe.

4.

	A	B	C	D	E	
$- M^{[2]} =$	A	0	0	1	0	1
	B	0	0	0	0	1
	C	0	0	0	0	0
	D	0	0	0	0	1
	E	0	0	0	0	0
	A	B	C	D	E	
$- M^{[3]} =$	A	0	0	0	0	1
	B	0	0	0	0	0
	C	0	0	0	0	0
	D	0	0	0	0	0
	E	0	0	0	0	0
	A	B	C	D	E	
$- M^{[4]} =$	A	0	0	0	0	0
	B	0	0	0	0	0
	C	0	0	0	0	0
	D	0	0	0	0	0
	E	0	0	0	0	0

5.

	A	B	C	D	E	
$P =$	A	1	1	1	1	1
	B	0	1	1	0	1
	C	0	0	1	0	1
	D	0	0	1	1	1
	E	0	0	0	0	1

6. Déroulement de l'algorithme :

$k = A$

	A	B	C	D	E
A	1	1	0	1	0
B	0	1	1	0	0
C	0	0	1	0	1
D	0	0	1	1	1
E	0	0	0	0	1

$M'[A][A] = M'[A][A]$  OU (  $M'[A][A]$  ET  $M'[A][A]$  )  
 $M'[A][B] = M'[A][B]$  OU (  $M'[A][A]$  ET  $M'[A][B]$  )  
 $M'[A][C] = M'[A][C]$  OU (  $M'[A][A]$  ET  $M'[A][C]$  )  
 $M'[A][D] = M'[A][D]$  OU (  $M'[A][A]$  ET  $M'[A][D]$  )  
 $M'[A][E] = M'[A][E]$  OU (  $M'[A][A]$  ET  $M'[A][E]$  )  
 $M'[B][A] = M'[B][A]$  OU (  $M'[B][A]$  ET  $M'[A][A]$  )  
 $M'[B][B] = M'[B][B]$  OU (  $M'[B][A]$  ET  $M'[A][B]$  )  
 $M'[B][C] = M'[B][C]$  OU (  $M'[B][A]$  ET  $M'[A][C]$  )  
 $M'[B][D] = M'[B][D]$  OU (  $M'[B][A]$  ET  $M'[A][D]$  )  
 $M'[B][E] = M'[B][E]$  OU (  $M'[B][A]$  ET  $M'[A][E]$  )  
 $M'[C][A] = M'[C][A]$  OU (  $M'[C][A]$  ET  $M'[A][A]$  )  
 $M'[C][B] = M'[C][B]$  OU (  $M'[C][A]$  ET  $M'[A][B]$  )  
 $M'[C][C] = M'[C][C]$  OU (  $M'[C][A]$  ET  $M'[A][C]$  )  
 $M'[C][D] = M'[C][D]$  OU (  $M'[C][A]$  ET  $M'[A][D]$  )  
 $M'[C][E] = M'[C][E]$  OU (  $M'[C][A]$  ET  $M'[A][E]$  )  
 $M'[D][A] = M'[D][A]$  OU (  $M'[D][A]$  ET  $M'[A][A]$  )  
 $M'[D][B] = M'[D][B]$  OU (  $M'[D][A]$  ET  $M'[A][B]$  )  
 $M'[D][C] = M'[D][C]$  OU (  $M'[D][A]$  ET  $M'[A][C]$  )  
 $M'[D][D] = M'[D][D]$  OU (  $M'[D][A]$  ET  $M'[A][D]$  )  
 $M'[D][E] = M'[D][E]$  OU (  $M'[D][A]$  ET  $M'[A][E]$  )  
 $M'[E][A] = M'[E][A]$  OU (  $M'[E][A]$  ET  $M'[A][A]$  )  
 $M'[E][B] = M'[E][B]$  OU (  $M'[E][A]$  ET  $M'[A][B]$  )  
 $M'[E][C] = M'[E][C]$  OU (  $M'[E][A]$  ET  $M'[A][C]$  )  
 $M'[E][D] = M'[E][D]$  OU (  $M'[E][A]$  ET  $M'[A][D]$  )  
 $M'[E][E] = M'[E][E]$  OU (  $M'[E][A]$  ET  $M'[A][E]$  )

$k = B$

	A	B	C	D	E
A	1	1	1	1	0
B	0	1	1	0	0
C	0	0	1	0	1
D	0	0	1	1	1
E	0	0	0	0	1

$M'[A][A] = M'[A][A]$  OU (  $M'[A][B]$  ET  $M'[B][A]$  )  
 $M'[A][B] = M'[A][B]$  OU (  $M'[A][B]$  ET  $M'[B][B]$  )  
 $M'[A][C] = M'[A][C]$  OU (  $M'[A][B]$  ET  $M'[B][C]$  )  
 $M'[A][D] = M'[A][D]$  OU (  $M'[A][B]$  ET  $M'[B][D]$  )  
 $M'[A][E] = M'[A][E]$  OU (  $M'[A][B]$  ET  $M'[B][E]$  )  
 $M'[B][A] = M'[B][A]$  OU (  $M'[B][B]$  ET  $M'[B][A]$  )  
 $M'[B][B] = M'[B][B]$  OU (  $M'[B][B]$  ET  $M'[B][B]$  )  
 $M'[B][C] = M'[B][C]$  OU (  $M'[B][B]$  ET  $M'[B][C]$  )  
 $M'[B][D] = M'[B][D]$  OU (  $M'[B][B]$  ET  $M'[B][D]$  )  
 $M'[B][E] = M'[B][E]$  OU (  $M'[B][B]$  ET  $M'[B][E]$  )  
 $M'[C][A] = M'[C][A]$  OU (  $M'[C][B]$  ET  $M'[B][A]$  )  
 $M'[C][B] = M'[C][B]$  OU (  $M'[C][B]$  ET  $M'[B][B]$  )  
 $M'[C][C] = M'[C][C]$  OU (  $M'[C][B]$  ET  $M'[B][C]$  )  
 $M'[C][D] = M'[C][D]$  OU (  $M'[C][B]$  ET  $M'[B][D]$  )  
 $M'[C][E] = M'[C][E]$  OU (  $M'[C][B]$  ET  $M'[B][E]$  )  
 $M'[D][A] = M'[D][A]$  OU (  $M'[D][B]$  ET  $M'[B][A]$  )  
 $M'[D][B] = M'[D][B]$  OU (  $M'[D][B]$  ET  $M'[B][B]$  )  
 $M'[D][C] = M'[D][C]$  OU (  $M'[D][B]$  ET  $M'[B][C]$  )  
 $M'[D][D] = M'[D][D]$  OU (  $M'[D][B]$  ET  $M'[B][D]$  )  
 $M'[D][E] = M'[D][E]$  OU (  $M'[D][B]$  ET  $M'[B][E]$  )  
 $M'[E][A] = M'[E][A]$  OU (  $M'[E][B]$  ET  $M'[B][A]$  )  
 $M'[E][B] = M'[E][B]$  OU (  $M'[E][B]$  ET  $M'[B][B]$  )  
 $M'[E][C] = M'[E][C]$  OU (  $M'[E][B]$  ET  $M'[B][C]$  )  
 $M'[E][D] = M'[E][D]$  OU (  $M'[E][B]$  ET  $M'[B][D]$  )  
 $M'[E][E] = M'[E][E]$  OU (  $M'[E][B]$  ET  $M'[B][E]$  )

$k = C$ 

	A	B	C	D	E
A	1	1	1	1	1
B	0	1	1	0	1
C	0	0	1	0	1
D	0	0	1	1	1
E	0	0	0	0	1

$M^2[A][A] = M^1[A][A] \text{ OU } ( M^1[A][C] \text{ ET } M^1[C][A] )$   
 $M^2[A][B] = M^1[A][B] \text{ OU } ( M^1[A][C] \text{ ET } M^1[C][B] )$   
 $M^2[A][C] = M^1[A][C] \text{ OU } ( M^1[A][C] \text{ ET } M^1[C][C] )$   
 $M^2[A][D] = M^1[A][D] \text{ OU } ( M^1[A][C] \text{ ET } M^1[C][D] )$   
 $M^2[A][E] = M^1[A][E] \text{ OU } ( M^1[A][C] \text{ ET } M^1[C][E] )$   
 $M^2[B][A] = M^1[B][A] \text{ OU } ( M^1[B][C] \text{ ET } M^1[C][A] )$   
 $M^2[B][B] = M^1[B][B] \text{ OU } ( M^1[B][C] \text{ ET } M^1[C][B] )$   
 $M^2[B][C] = M^1[B][C] \text{ OU } ( M^1[B][C] \text{ ET } M^1[C][C] )$   
 $M^2[B][D] = M^1[B][D] \text{ OU } ( M^1[B][C] \text{ ET } M^1[C][D] )$   
 $M^2[B][E] = M^1[B][E] \text{ OU } ( M^1[B][C] \text{ ET } M^1[C][E] )$   
 $M^2[C][A] = M^1[C][A] \text{ OU } ( M^1[C][C] \text{ ET } M^1[C][A] )$   
 $M^2[C][B] = M^1[C][B] \text{ OU } ( M^1[C][C] \text{ ET } M^1[C][B] )$   
 $M^2[C][C] = M^1[C][C] \text{ OU } ( M^1[C][C] \text{ ET } M^1[C][C] )$   
 $M^2[C][D] = M^1[C][D] \text{ OU } ( M^1[C][C] \text{ ET } M^1[C][D] )$   
 $M^2[C][E] = M^1[C][E] \text{ OU } ( M^1[C][C] \text{ ET } M^1[C][E] )$   
 $M^2[D][A] = M^1[D][A] \text{ OU } ( M^1[D][C] \text{ ET } M^1[C][A] )$   
 $M^2[D][B] = M^1[D][B] \text{ OU } ( M^1[D][C] \text{ ET } M^1[C][B] )$   
 $M^2[D][C] = M^1[D][C] \text{ OU } ( M^1[D][C] \text{ ET } M^1[C][C] )$   
 $M^2[D][D] = M^1[D][D] \text{ OU } ( M^1[D][C] \text{ ET } M^1[C][D] )$   
 $M^2[D][E] = M^1[D][E] \text{ OU } ( M^1[D][C] \text{ ET } M^1[C][E] )$   
 $M^2[E][A] = M^1[E][A] \text{ OU } ( M^1[E][C] \text{ ET } M^1[C][A] )$   
 $M^2[E][B] = M^1[E][B] \text{ OU } ( M^1[E][C] \text{ ET } M^1[C][B] )$   
 $M^2[E][C] = M^1[E][C] \text{ OU } ( M^1[E][C] \text{ ET } M^1[C][C] )$   
 $M^2[E][D] = M^1[E][D] \text{ OU } ( M^1[E][C] \text{ ET } M^1[C][D] )$   
 $M^2[E][E] = M^1[E][E] \text{ OU } ( M^1[E][C] \text{ ET } M^1[C][E] )$

On voit que la fermeture transitive est déjà obtenue maintenant, mais l'algorithme continue tout de même.

## A.7 Exercice 8

### A.7.1 Questions

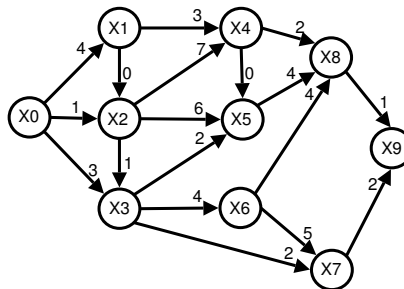


FIG. A.11: Exercice 8

1. Appliquer l'algorithme de Dijkstra au graphe ci-dessus à partir du sommet  $X_0$
2. Déterminer la composition du chemin de valeur minimale allant de  $X_0$  à  $X_9$
3. Montrer à l'aide d'un contre exemple que cet algorithme n'est pas applicable lors que certaines valuations sont strictement négatives
4. Modifier l'algorithme pour la recherche d'un chemin de valeur minimale de  $X_0$  vers un sommet donné  $X_k$

### A.7.2 Réponses

1.

SS	$\Pi_x$										$P_x$									
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
0	0	4	1	3	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	-1	0	0	0	-1	-1	-1	-1	-1	-1
2	0	4	1	2	8	7	$\infty$	$\infty$	$\infty$	$\infty$	-1	0	0	2	2	2	-1	-1	-1	-1
3	0	4	1	2	8	4	5	4	$\infty$	$\infty$	-1	0	0	2	2	3	3	3	-1	-1
1	0	4	1	2	7	4	5	4	$\infty$	$\infty$	-1	0	0	2	1	3	3	3	-1	-1
5	0	4	1	2	7	4	5	4	8	$\infty$	-1	0	0	2	1	3	3	3	5	-1
7	0	4	1	2	7	4	5	4	8	6	-1	0	0	2	1	3	3	3	5	7
6	0	4	1	2	7	4	5	4	8	6	-1	0	0	2	1	3	3	3	5	7
9	0	4	1	2	7	4	5	4	8	6	-1	0	0	2	1	3	3	3	5	7
4	0	4	1	2	7	4	5	4	8	6	-1	0	0	2	1	3	3	3	5	7
8	0	4	1	2	7	4	5	4	8	6	-1	0	0	2	1	3	3	3	5	7

2.  $\Pi_{(9)} = 6$

$$P_{(9)} = 7$$

$$P_{(7)} = 3$$

$$P_{(3)} = 2$$

$$P_{(2)} = 0$$

$$\Pi_{(9)} = l_{02} + l_{23} + l_{37} + l_{79}$$

$$\mu = (0, 2, 3, 7, 9)$$

$$l_\mu = \Pi_{(9)} = 6$$

3.

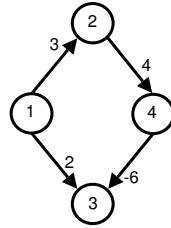


FIG. A.12: Réponse à la question 3 de l'exercice 8

Les valeurs réelles sont les suivantes :

- $1 \rightarrow 2 = 3$
- $1 \rightarrow 4 = 7$
- $1 \rightarrow 3 = 1$

Et si on applique l'algorithme de Dijkstra

	$\Pi_1$	$\Pi_2$	$\Pi_3$	$\Pi_4$
1	0	3	2	$\infty$
3	0	3	2	$\infty$
2	0	3	2	7
4	0	3	2	7

On voit bien que sur la dernière ligne la distance entre le sommet 4 et le sommet 2 n'est pas calculée car le sommet 2 a déjà été sélectionné.

4. À remplacer dans l'algorithme :

**b) Solution**

- On sélectionne le sommet  $j \in \bar{S}$  tel que  $\Pi_{(j)} = \min(\Pi_{(i)})$  avec  $i \in \bar{S}$
- $\bar{S} = \bar{S} - \{j\}$
- $S = S + \{j\}$
- Si  $X_k \in S$  alors Fin  
Sinon aller en (c)

## A.8 Exercice 9

### A.8.1 Questions

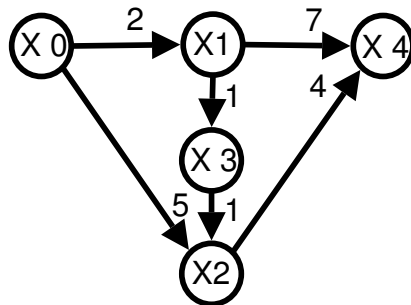


FIG. A.13: Exercice 9

1. Appliquer l'algorithme de Ford au graphe ci-dessus à partir du sommet  $X_0$
2. Déterminer la composition du chemin de valeur minimale allant de  $X_0$  à  $X_4$
3. Le nombre d'itérations dépend-il de la numérotation des sommets du graphe

### A.8.2 Réponses

1.

Étape	$P_0$	$P_1$	$P_2$	$P_3$	$P_4$	$\Pi_0$	$\Pi_1$	$\Pi_2$	$\Pi_3$	$\Pi_4$
Initialisation	-1	-1	-1	-1	-1	0	$\infty$	$\infty$	$\infty$	$\infty$
1 <sup>ère</sup>	-1	0	0	1	1	0	2	5	3	9
2 <sup>ème</sup>	-1	0	3	1	2	0	2	4	3	8
3 <sup>ème</sup>	-1	0	3	1	2	0	2	4	3	8

2.  $X_0 \rightarrow X_4$

$$P_{(4)} = 2$$

$$P_{(2)} = 3$$

$$P_{(3)} = 1$$

$$P_{(1)} = 0$$

$$P = l_{01} + l_{13} + l_{32} + l_{24} = 8$$

3. Oui, le nombre d'itérations change si l'ordre des sommets change. Par exemple si on inverse  $X_2$  et  $X_3$  dans le graphe précédent,  $X_2$  sera déterminé avant  $X_3$  car  $X_1$  sera déterminé auparavant dans le déroulement de l'algorithme.

Étape	$P_0$	$P_1$	$P_2$	$P_3$	$P_4$	$\Pi_0$	$\Pi_1$	$\Pi_2$	$\Pi_3$	$\Pi_4$
Initialisation	-1	-1	-1	-1	-1	0	$\infty$	$\infty$	$\infty$	$\infty$
1 <sup>ère</sup>	-1	0	1	2	3	0	2	3	4	8
2 <sup>ème</sup>	-1	0	1	2	3	0	2	3	4	8

## A.9 Exercice 11

### A.9.1 Questions

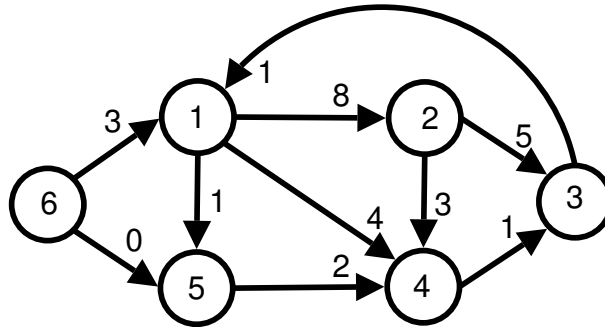


FIG. A.14: Exercice 11

Dans certains problèmes de communications on a besoin de connaître le point (les points) le moins éloigné de tous les autres points d'un réseau. C'est le **centre de gravité** du graphe. D'autre part, on appelle **excentricité** d'un sommet  $x$ , dans un graphe orienté  $G = \langle X, U \rangle$  et valué par les coûts positifs, la quantité :

$\text{exc}(x) = \max(\text{ppdistance}(y, x))$  //plus petite distance  
avec  $y \in X - \{x\}$

C'est la distance maximale nécessaire pour atteindre  $x$  depuis chaque sommet.

On appelle **centre de gravité** du graphe  $G$ , un sommet d'excentricité minimale. C'est donc un sommet tel que :

$\text{exc}(\text{centre\_gravite}) = \min(\text{exc}(x))$   
avec  $x \in X$

1. Trouve le centre de gravité du graphe
2. Proposer un algorithme permettant de trouver le centre de gravité d'un graphe

## A.9.2 Réponses

1.

$$L^{(0)} =$$

	1	2	3	4	5	6
1	0	8	$\infty$	4	1	$\infty$
2	$\infty$	0	5	3	$\infty$	$\infty$
3	1	$\infty$	0	$\infty$	$\infty$	$\infty$
4	$\infty$	$\infty$	1	0	$\infty$	$\infty$
5	$\infty$	$\infty$	$\infty$	2	0	$\infty$
6	3	$\infty$	$\infty$	$\infty$	0	0

$$L^{(1)} =$$

	1	2	3	4	5	6
1	0	8	$\infty$	4	1	$\infty$
2	$\infty$	0	5	3	$\infty$	$\infty$
3	1	9	0	$\infty$	$\infty$	$\infty$
4	$\infty$	$\infty$	1	0	$\infty$	$\infty$
5	$\infty$	$\infty$	$\infty$	2	0	$\infty$
6	3	$\infty$	$\infty$	$\infty$	0	0

$$L^{(2)} =$$

	1	2	3	4	5	6
1	0	8	13	4	1	$\infty$
2	$\infty$	0	5	3	$\infty$	$\infty$
3	1	9	0	$\infty$	$\infty$	$\infty$
4	$\infty$	$\infty$	1	0	$\infty$	$\infty$
5	$\infty$	$\infty$	$\infty$	2	0	$\infty$
6	3	$\infty$	16	$\infty$	0	0

$$L^{(3)} =$$

	1	2	3	4	5	6
1	0	8	13	4	1	$\infty$
2	6	0	5	3	7	$\infty$
3	1	9	0	$\infty$	$\infty$	$\infty$
4	2	10	1	0	3	$\infty$
5	$\infty$	$\infty$	$\infty$	2	0	$\infty$
6	3	$\infty$	16	$\infty$	0	0

$$L^{(4)} =$$

	1	2	3	4	5	6
1	0	8	5	4	1	$\infty$
2	5	0	4	3	6	$\infty$
3	1	9	0	$\infty$	$\infty$	$\infty$
4	2	10	1	0	3	$\infty$
5	4	12	3	2	0	$\infty$
6	3	$\infty$	8	$\infty$	0	0

$$L^{(5)} =$$

	1	2	3	4	5	6
1	0	8	4	3	1	$\infty$
2	5	0	4	3	6	$\infty$
3	1	9	0	4	$\infty$	$\infty$
4	2	10	1	0	3	$\infty$
5	4	12	3	2	0	$\infty$
6	3	$\infty$	8	$\infty$	0	0

$$L^{(6)} = L^{(5)}$$

Détermination du/des centres d'excentricité :

$$exc(c) = \max(ppdistance(y, x)) \text{ avec } y \in X - \{x\}$$

$$exc(1) = \max(ppdistance(y, 1)) \text{ avec } y \in \{2, 3, 4, 5, 6\}$$

$$= \max(5, 1, 2, 4, 3)$$

$$= 5 \text{ à partir du sommet 2}$$

$$exc(2) = 12 \text{ à partir du sommet 5}$$

$$exc(3) = 4 \text{ à partir du sommet 1}$$

$$exc(4) = 4 \text{ à partir du sommet 3}$$

$$exc(5) = 6 \text{ à partir du sommet 2}$$

$$exc(6) = +\infty \text{ à partir du sommet 1}$$

} centres de gravités car  $\min(exc(x))$  avec  $x \in X$

## 2. Algorithme

Soit :

- L la matrice résultat
- N le nombre de sommets
- EXC la liste des excentricités de chaque sommet initialisée à 0

Debut

L = FLOYD() //Retourne la matrice résultant de l'algorithme de Floyd

Pour j allant de 1 à N faire

  Pour i allant de 1 à N faire

    Si L[i, j] > EXC[j] alors

      EXC[j] = L[i, j]

    Fin Si

  Fin Pour

Fin Pour

centre\_gravité = indice\_min(EXC)

retourner( centre\_gravité)

Fin

## A.10 Exercice 12

### A.10.1 Questions

On doit exécuter sept tâches a, b, c, d, e, f, g soumises aux contraintes de succession rapportées dans le tableau ci-dessous :

Tâches	Durée	Contraintes
a	6	
b	3	
c	6	
d	2	b achevée
e	4	b achevée
f	3	d et a achevées
g	1	f, e, c achevées

1. Dessiner le graphe potentiel-tâches associé (Méthode MPM de B. ROY)
2. Après avoir remarqué que ce graphe ne comporte pas de circuit, former le tableau des prédecesseurs et calculer, à l'aide de ce tableau, les dates au plus tôt de début des tâches.
3. Déterminer le chemin critique ; est-il unique ?
4. Calculer, en formant le tableau des successeurs, les dates au plus tard des tâches lorsque la durée de l'ordonnancement est optimale.
5. Calculer les marges totales, libres et certaines des tâches.
6. Dessiner un diagramme de GANTT de l'ordonnancement « au plus tôt » (on dit aussi « calé à gauche »), c'est à dire dans lequel toute tâche est commencée à sa date de début au plus tôt.

### A.10.2 Réponses

- 1.

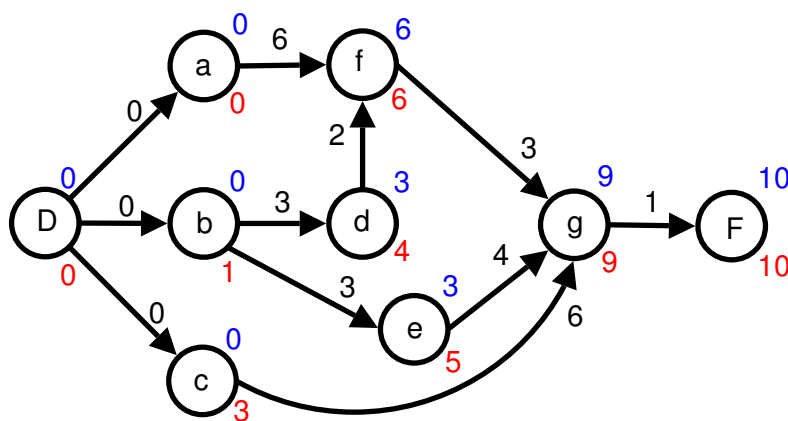


FIG. A.15: Réponse à la question 1 de l'exercice 12

2.

x	D	a	b	c	d	e	f	g	F
$\Gamma_x^{-1}$	0	D	D	D	b	b	d, a	f, e, c	g
Date au plus tôt	0	0	0	0	3	3	6	9	10

3. le chemin critique est composé des sommets suivants :  $D, a, f, g, F$

4.

x	D	a	b	c	d	e	f	g	F
$\Gamma_x$	a,b,c	f	d,e	g	f	g	g	F	0

5. Marges :

j	a	b	c	d	e	f	g
$\Gamma_j$	0	1	3	1	2	0	0
$m_j$	0	0	3	1	2	0	0
$\mu_j$	0	0	3	0	1	0	0

6. Diagramme de Gantt :

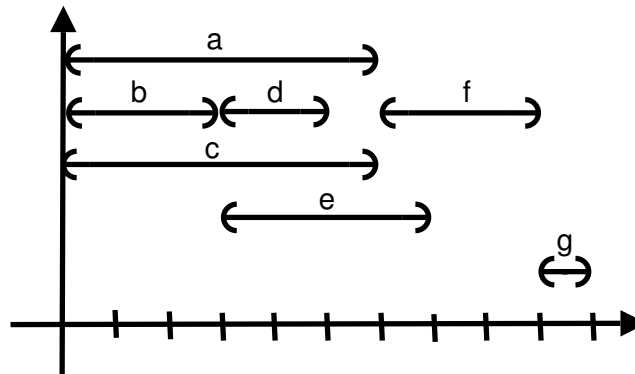


FIG. A.16: Réponse à la question 6 de l'exercice 12

## A.11 Exercice 13

### A.11.1 Questions

Reprendre l'énoncé de l'exercice 12 à la page 102

1. Dessiner un graphe PERT en associant à chaque tâche  $i$  des événements début et fin de tâche et en introduisant des événements début et fin de l'ordonnancement.
2. Calculer les dates au plus tôt des événements du graphe. Quel est le chemin critique? Quelles sont les dates au plus tard? Calculer les marges totales, libres et certaines des tâches.

### A.11.2 Réponses

1.

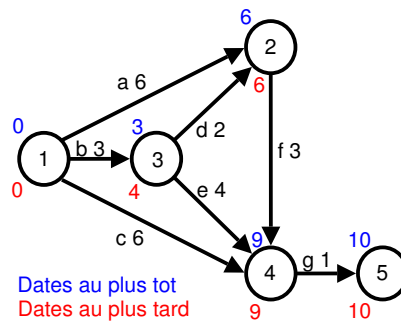


FIG. A.17: Réponse à la question 1 de l'exercice 13

2.

	a	b	c	d	e	f	g
$M_{ij}$	0	1	3	1	2	0	0
$m_{ij}$	0	0	3	1	2	0	0
$\mu_{ij}$	0	0	3	0	1	0	0

## A.12 Exercice 14

### A.12.1 Questions

Un projet peut être décomposé en sept tâches élémentaires. Dans le tableau ci-dessous, on a indiqué pour chaque tâche quelle est sa durée et quelles sont les conditions de son démarrage.

Tâche	Conditions de démarrage	Durée en jours
a	début du projet	3
b	début du projet	6
c	début du projet	5
d	début du projet	6
e	a et b terminées	4
f	b, c et d terminées	7
g	c terminé	6

1. Tracer le graphe PERT associé à ce problème
2. Calculer les dates au plus tôt et au plus tard des événements
3. Calculer les marges totales de toutes les tâches. Quelles sont les tâches critiques?
4. On suppose que, lors de l'exécution du projet, la tâche b a été retardée d'une journée alors que la tâche d a été retardée de 2 jours. Quel est l'effet de ces 2 retards sur le déroulement du projet?

### A.12.2 Réponses

- 1.
- 2.

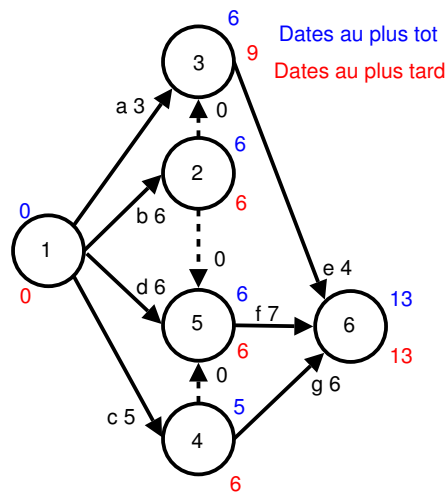


FIG. A.18: Réponse aux questions 1 et 2 de l'exercice 14

- 3.

	a	b	c	d	e	f	g
$M_{ij}$	6	0	1	0	3	0	2

Il existe dans ce graphe deux chemins critiques  $u_1 = (b, f)$  et  $u_2 = (d, f)$  avec  $l(u_1) = l(u_2) = 13$

4. L'étape 2 sera retardée de 2 jours, ainsi que l'étape 5 et la date 6

## A.13 Exercice 15

### A.13.1 Questions

Un graphe de  $n = 8$  sommets (indices par  $i$ ) et  $m = 13$  arcs (indices par  $j$ ) est décrit par les tableaux ci dessous :

i	1	2	3	4	5	6	7	8					
$d_i^+$	3	2	1	2	0	2	2	1					
j	1	2	3	4	5	6	7	8	9	10	11	12	13
$ext(j)$	3	5	8	3	6	8	2	7	3	8	1	3	5

On rappelle que  $d_i^+$  désigne le nombre d'arcs ayant le sommet  $x_i$  comme extrémité initiale, c'est à dire le nombre de successeurs de  $x_i$ . Le tableau  $ext(j)$  liste dans l'ordre lexicographique les extrémités terminales des arcs issus de  $x_1$  (c'est à dire les successeurs de  $x_1$ ) puis celles des arcs issus de  $x_2$  et ainsi de suite. Ainsi  $x_1$  a 3 successeurs qui sont  $x_3, x_5$  et  $x_8$ .

1. Quel est l'intérêt informatique de cette représentation d'un graphe?
2. Construire le graphe à grande échelle, après avoir déterminé son entrée et sa sortie. On disposera les sommets de telle sorte que les arcs ne s'intersectent pas.
3. En fait, dans un contexte d'ordonnancement, tout arc  $j$  représente une tâche et est valué par une durée  $D(j)$  donnée dans le tableau ci-dessous : tout sommet correspond à évènement (étape). Le graphe obtenu est le graphe PERT d'un projet comportant 13 tâches (sans tâches fictives). Donner l'indice du sommet "débuté" et celui du sommet "fin" de ce projet.

	A	B	C	D	E	F	G	H	I	J	K	L	M
j	1	2	3	4	5	6	7	8	9	10	11	12	13
$D(j)$	4	2	1	3	3	2	1	5	1	3	2	2	1

Retracer le graphe en représentant chaque tâche par une lettre et valuer l'arc correspondant par sa durée

4. Calculer les dates au plus tôt et au plus tard des étapes.
5. Donner le chemin critique
6. Calculer les marges totales et libres des tâches

**A.13.2 Réponses**

1. L'intérêt est de pouvoir utiliser le premier tableau comme un pointeur sur le second. Le fait d'utiliser des listes est moins gourmand en espace mémoire qu'une matrice par exemple.
- 2.
- 3.
- 4.

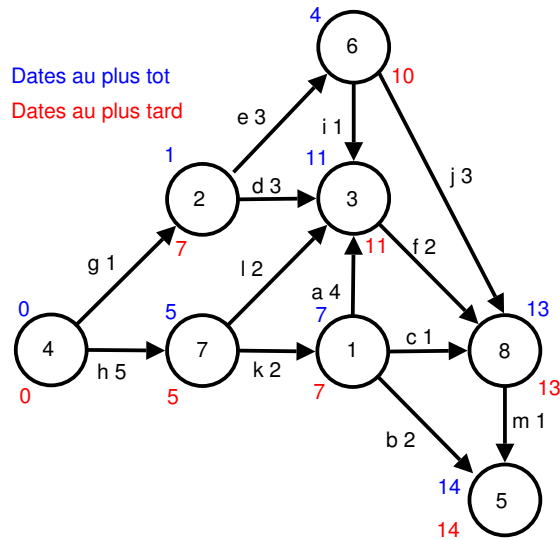


FIG. A.19: Réponse aux questions 2, 3 et 4 de l'exercice 15

5. Le chemin critique est le suivant : *H, K, A, F, M*
- 6.

	A	B	C	D	E	F	G	H	I	J	K	L	M
$M_j$	0	5	5	7	6	0	6	0	6	6	0	4	0
$m_j$	0	5	5	7	0	0	0	0	6	6	0	4	0

## A.14 Exercice 17

### A.14.1 Questions

Avant d'établir un projet de construction d'autoroute on désire étudier la capacité du réseau autoroutier, représenté par le graphe ci-dessous, reliant la ville E à la ville S.

Pour cela, on a évalué le nombre maximal de véhicules que chaque route peut écouler par heure, compte tenu des ralentissements aux traversées des villes et villages, des arrêts aux feux etc... Ces évaluations sont indiquées en centaines de véhicules par heure sur les arcs du graphe. Les temps de parcours entre villes sont tels que les automobilistes n'emprunteront que les chemins représentés par le graphe.

Quel est le débit horaire total maximal de véhicules susceptible de s'écouler entre les villes E et S ?

En utilisant l'algorithme de Ford-Fulkerson, augmenter le flot entre les sommets E et S jusqu'à ce qu'il devienne maximal. (La valeur initiale du flot : valeurs indiquées à côté des capacités)

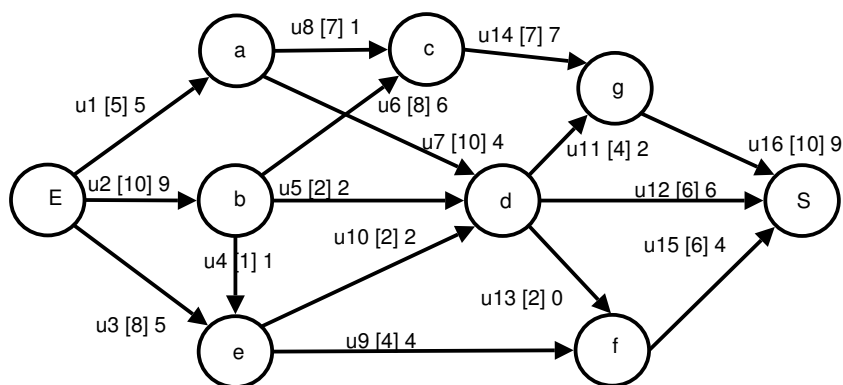


FIG. A.20: Exercice 17

### A.14.2 Réponses

Application de l'algorithme de Ford-Fulkerson :

On applique un flot de retour  $\varphi_0$  sur le graphe

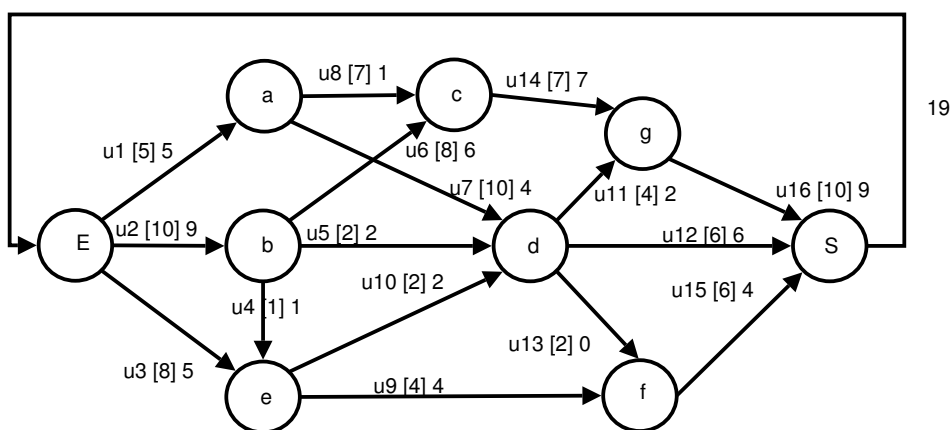


FIG. A.21: Application d'un flot de retour sur le graphe A.20

1.  $\overline{G}(\varphi^0)$  :

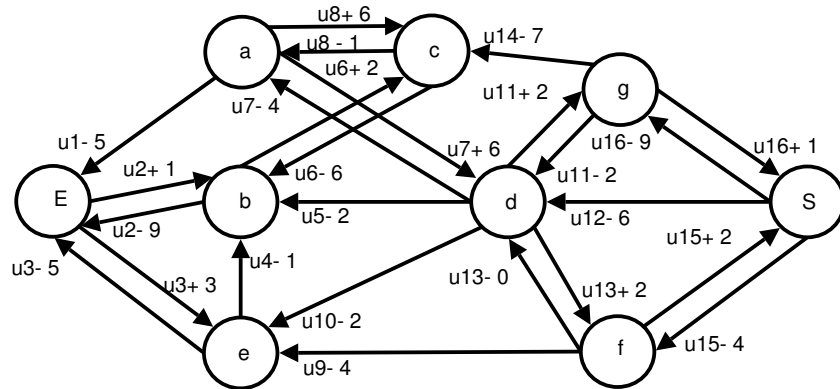


FIG. A.22: Graphe d'écart déduit du graphe A.20

2.  $k = 0$

$$P^0 = (E, b, c, a, d, f) = (\underbrace{u_2^+}_{1}, \underbrace{u_6^+}_{2}, \underbrace{u_8^-}_{1}, \underbrace{u_7^+}_{6}, \underbrace{u_{13}^+}_{2}, \underbrace{u_{15}^+}_{2})$$

3.

- $\varepsilon^0 = 1$
- $\varphi'_{u_2} = 10$
- $\varphi'_{u_6} = 7$
- $\varphi'_{u_8} = 0$
- $\varphi'_{u_7} = 5$
- $\varphi'_{u_{13}} = 1$
- $\varphi'_{u_{15}} = 5$
- $\varphi'_{u_0} = 20$

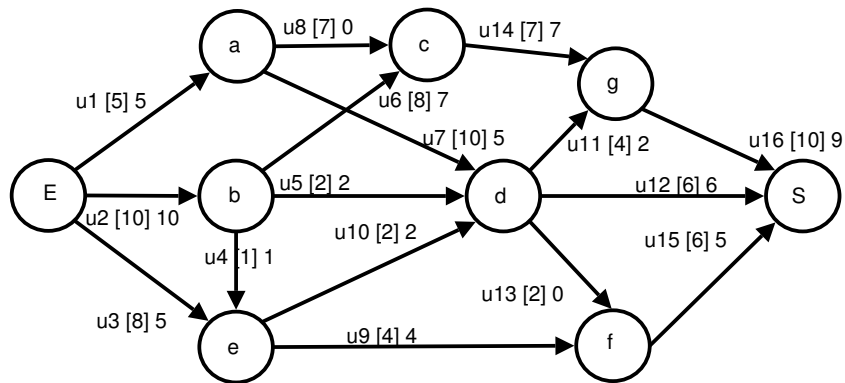


FIG. A.23: Graphe déduit de l'augmentation du flot sur le graphe A.22

1.  $\overline{G}(\varphi^1)$  :

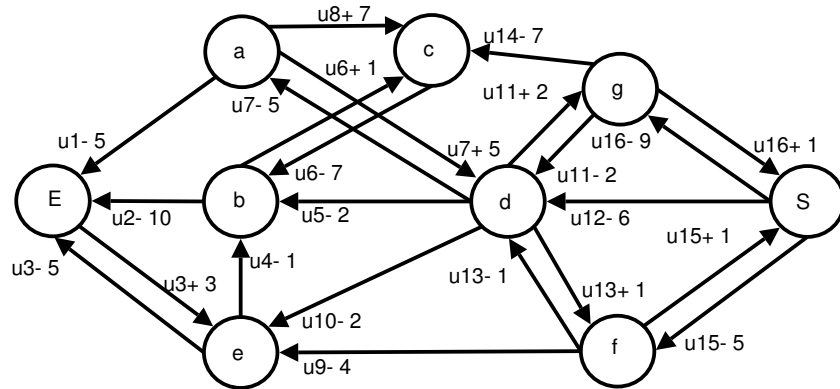


FIG. A.24: Graphe d'écart déduit du graphe A.22

Il n'existe pas de chemin entre  $E$  et  $S$ , l'algorithme s'arrête,  $\varphi_0$  est maximal et vaut 20.

## A.15 Exercice 17b

### A.15.1 Questions

Soit le graphe  $G = \langle X, U \rangle$  de  $n = 5$  sommets et  $m = 7$  arcs, valué par des capacités et défini par les tableaux suivants :

- Le tableau NARC désigne le demi-degré extérieur de chacun des sommets du graphe.  $NARC(I)$  indique le nombre d'arcs ayant le sommet  $I$  comme extrémité initiale.

$I$	1	2	3	4	5
$NARC(I)$	1	2	2	0	2

- Le tableau SUCC liste d'abord les successeurs (s'ils existent) du sommet 1, puis du sommet 2 et ainsi de suite.  $SUCC(J)$  désigne l'indice du sommet qui est l'extrémité terminale de l'arc d'indice  $J$  : On a numéroté les arcs en commençant par ceux issus du sommet 1, puis du sommet 2, etc ...

$J$	1	2	3	4	5	6	7
$SUCC(J)$	4	1	4	2	5	1	2

- Le tableau CAPA désigne la capacité de l'arc d'indice  $J$

$J$	1	2	3	4	5	6	7
$CAPA(J)$	1	4	7	3	5	2	2

- Le tableau FLOT indique le flot transporté sur l'arc d'indice  $J$

$J$	1	2	3	4	5	6	7
$FLOT(J)$	1	1	4	3	2	0	2

1. Tracer le graphe  $G$  à grande échelle. On doit trouver pour chaque arc son numéro, sa capacité entre crochet et le flot transporté. Le tout devant apparaître très clairement et très lisiblement.
2. Donner la valeur du flot défini dans le tableau FLOT.
3. Ce flot est-il complet ?
4. Déterminer toutes les coupes du réseau de transport et donner leur capacité. Donner la coupe de capacité minimale. Le flot est-il optimal ?
5. Sinon l'optimiser en utilisant l'algorithme de Ford-Fulkerson.

**A.15.2 Réponses**

1.

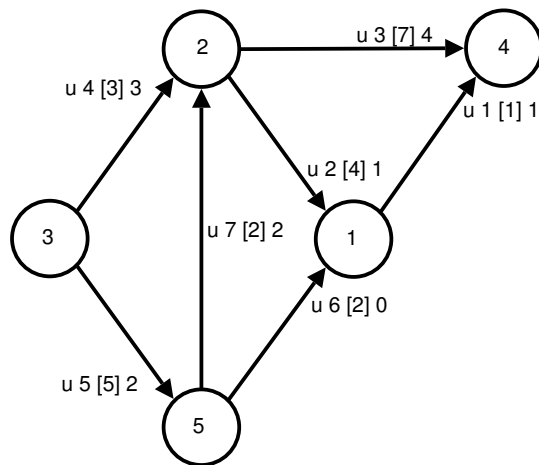


FIG. A.25: Réponse à la question 1 de l'exercice 17b

2. Le flot vaut 5.

3. Oui, le flot est complet car il existe au minimum un arc saturé.

4.

$$C(3) = 8$$

$$C(3, 2) = 16$$

$$C(3, 5) = 7$$

$$C(3, 2, 1) = 13$$

$$C(3, 5, 1) = 6 \leftarrow \text{Valeur minimale, comme la capacité est inférieure au flot, le flot n'est pas maximal}$$

$$C(3, 1) = 9$$

$$C(3, 5, 2, 1) = 8$$

5. Application de l'algorithme de Ford-Fulkerson :

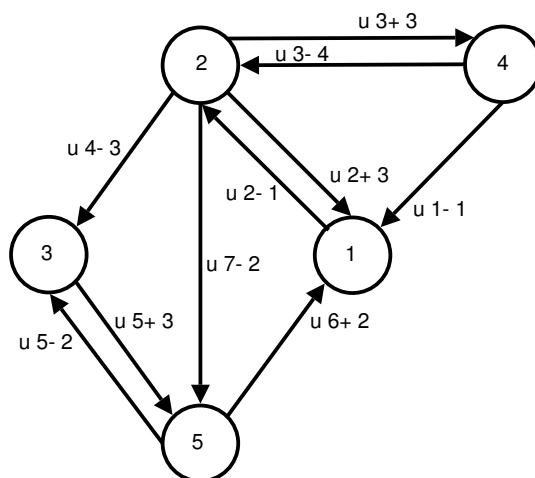


FIG. A.26: Réponse à la question 5 de l'exercice 17b - première étape de l'application de l'algorithme de Ford-Fulkerson

$$P^0 = (3, 5, 1, 2, 4) = (u_{35}^+, u_{51}^+, \underline{u_{12}^-}, u_{24}^+)$$

$$\varepsilon^0 = 1$$

$$\varphi_{35}^1 = 3$$

$$\varphi_{51}^1 = 1$$

$$\varphi_{12}^1 = 0$$

$$\varphi_{24}^1 = 5$$

$$\varphi_0^1 = 6$$

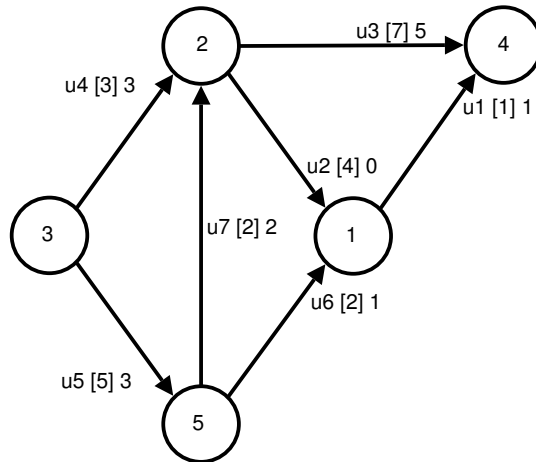


FIG. A.27: Réponse à la question 5 de l'exercice 17b - deuxième étape de l'application de l'algorithme de Ford-Fulkerson

On obtient donc le graphe d'écart suivant :

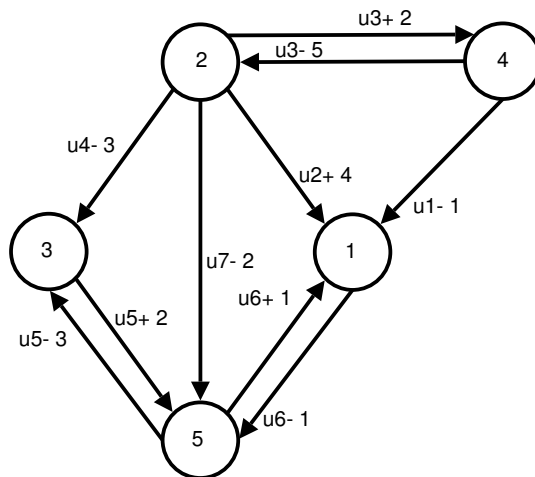


FIG. A.28: Réponse à la question 5 de l'exercice 17b - graphe d'écart final déduit de l'application de l'algorithme de Ford-Fulkerson

Il n'existe plus de chemin de 3 à 4, l'algorithme s'arrête. Le flot est optimal et vaut 6.

## A.16 Exercice 19

### A.16.1 Questions

Une société de distribution doit effectuer le transport d'un produit de trois centres de production  $A$ ,  $B$ ,  $C$  vers trois points de vente  $X$ ,  $Y$ ,  $Z$ . Le schéma ci-dessous donne la localisation de ces centres et la configuration du réseau de communication constitué d'une voie ferrée et de routes.  $N$  est un simple point intermédiaire.

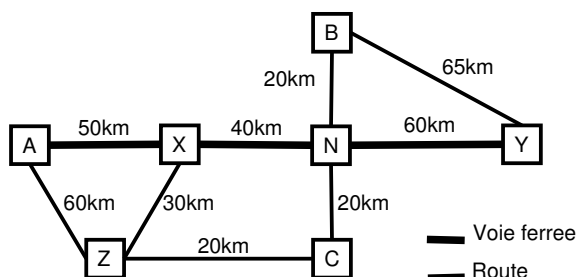


FIG. A.29: Énoncé de l'exercice 19

On suppose que :

- le transport d'une tonne de produit par voie ferrée coûte 2 unités monétaires/km
- le transport d'une tonne de produit par route coûte 3 unités monétaires/km
- le transfert d'une tonne de produit de la voie ferrée vers la route et inversement coûte 20 unités monétaires

1. Vérifier la valeur, couple par couple (centre de production, point de vente) du chemin de moindre coût, par comparaison des divers chemins possibles et indiquer les points ou centres par lesquels passe ce chemin de moindre coût.

	X	Y	Z
A	100	300	180
B	160	195	180
C	150	200	60

Comment pourrait-on procéder de façon systématique si le réseau était de plus grande taille? (introduire un graphe valué - le tracer pour l'exemple ci-dessus - et proposer une méthode de résolution adaptée.

2. Les productions des trois centres  $A$ ,  $B$ ,  $C$  sur la période étudiée sont :
  - $A$  : 300 tonnes
  - $B$  : 400 tonnes
  - $C$  : 300 tonnes

Les demandes des trois points de vente  $X$ ,  $Y$ ,  $Z$ , pendant le même temps, sont respectivement :

- 270 tonnes
- 310 tonnes
- 420 tonnes

Comment organiser le transport pour que les demandes de  $X$ ,  $Y$ ,  $Z$  soient exactement satisfaites et que le coût de ce transport soit le plus petit possible? On utilisera un algorithme approprié sur un réseau de transport dans lequel tout chemin de moindre coût obtenu à la question précédente sera figuré par un arc : (centre de production, point de vente) valué convenablement.

## A.16.2 Réponses

1. Il faudrait modéliser le réseau de transport, puis appliquer l'algorithme de Ford pour trouver les plus courts chemins des centres de production vers les points de vente.

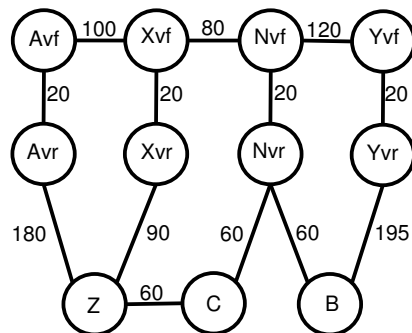


FIG. A.30: Réponse à la question 1 de l'exercice 19

2. On cherche une solution de base au problème :

	X	Y	Z	
A	270	30		300
B		280	120	400
C			300	300
	270	310	420	

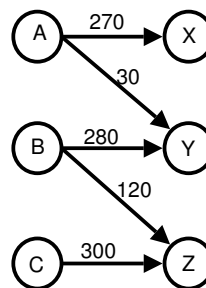


FIG. A.31: Solution de base de l'algorithme du stepping stone dans la question 2 de l'exercice 19

$\delta_{AZ} = -105 \leftarrow$  on sélectionne AZ car il est le plus petit coût marginal.

$$\delta_{BX} = 165$$

$$\delta_{CX} = 275$$

$$\delta_{CY} = 120$$

$$\mu = \{AZ, BZ, BY, AY\}$$

$$\mu^- = \{BZ, AY\}$$

$$\mu^+ = \{AZ, BY\}$$

$$q = \min[x_{BZ}, x_{AY}] = \min(120, 30) = 30$$

$$x_{BZ} = 120 - 30 = 90$$

$$x_{AY} = 30 - 30 = 0$$

$$x_{AZ} = 0 + 30 = 30$$

$$x_{BY} = 280 + 30 = 310$$

On obtient donc la solution suivante :

	X	Y	Z	
A	270		30	300
B		310	90	400
C			300	300
	270	310	420	

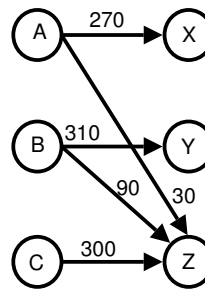


FIG. A.32: Résultat de la première itération de l'algorithme du stepping stone dans la question 2 de l'exercice 19

On recalcule les coûts marginaux :

$$\delta = AY = 105$$

$$\delta = BX = 60$$

$$\delta = CX = 170$$

$$\delta = CY = 125$$

Tous les coûts marginaux sont supérieurs à 0, l'algorithme s'arrête.  $z = 127050$

## A.17 Exercice 18

### A.17.1 Questions

Une banque désire installer au moindre coût un réseau de transmissions de données entre son agence centrale située dans le quartier de la bourse à Paris et sept de ses succursales.

Il s'agit d'un réseau arborescent composé de lignes privées point à point à 2400 bauds avec des possibilités de concentrateurs. Le coût de construction d'une ligne entre deux agences est donné par le tableau suivant (en unités monétaires) :

	B	O	E	R	$S^t$ L	L	N	C
Bourse	-							
Opéra	5	-						
Etoile	18	17	-					
République	9	11	27	-				
St-Lazare	13	7	23	20	-			
Louvre	7	12	15	15	15	-		
Neuilly	38	38	20	40	40	35	-	
Châtelet	22	15	25	25	30	10	45	-

(Ces coûts ont été déterminés en fonction des distances entre les différentes agences et du chiffre d'affaires de chaque succursale).

1. Quel problème classique reconnaissez-vous ?
2. Déterminez la solution optimale du problème.

### A.17.2 Réponses

1. On reconnaît un problème de recherche d'arbre couvrant de coût minimal
2. Application de l'algorithme de Prim<sup>1</sup> :

$$T = \{B\}$$

$$U' = \emptyset$$

(a)

$$U' = \{(B, O)\}$$

$$T = T + \{O\} = \{B, O\}$$

	B	O	E	R	$S^t$ L	L	N	C
B	-							
O	<del>5</del>	-						
E	18	17	-					
R	9	11	27	-				
StL	13	7	23	20	-			
L	7	12	15	15	15	-		
N	38	38	20	40	40	35	-	
C	22	15	25	25	30	10	45	-

<sup>1</sup>On va barrer, au fur et à mesure que l'on ajoute des sommets dans la liste  $U'$ , des coûts dans le tableau correspondant aux liens existants entre tous les sommets de la liste  $U'$

(b)

$$U' = \{(B, O), (O, S^t L)\}$$

$$T = \{B, O, S^t L\}$$

	B	O	E	R	$S^t L$	L	N	C
B	-							
O	<del>7</del>	-						
E	18	17	-					
R	9	11	27	-				
$S^t$	<del>13</del>	<del>7</del>	23	20	-			
L	7	12	15	15	15	-		
N	38	38	20	40	40	35	-	
C	22	15	25	25	30	10	45	-

(c)

$$U' = \{(B, O), (O, S^t L), (B, L)\}$$

$$T = \{B, O, S^t L, L\}$$

	B	O	E	R	$S^t L$	L	N	C
B	-							
O	<del>7</del>	-						
E	18	17	-					
R	9	11	27	-				
$S^t$	<del>13</del>	<del>7</del>	23	20	-			
L	<del>7</del>	<del>12</del>	15	15	<del>15</del>	-		
N	38	38	20	40	40	35	-	
C	22	15	25	25	30	10	45	-

(d)

$$U' = \{(B, O), (O, S^t L), (B, L), (B, R)\}$$

$$T = \{B, O, S^t L, L, R\}$$

	B	O	E	R	$S^t L$	L	N	C
B	-							
O	<del>7</del>	-						
E	18	17	-					
R	<del>9</del>	<del>11</del>	27	-				
$S^t$	<del>13</del>	<del>7</del>	23	<del>20</del>	-			
L	<del>7</del>	<del>12</del>	15	<del>15</del>	<del>15</del>	-		
N	38	38	20	40	40	35	-	
C	22	15	25	25	30	10	45	-

(e)

$$U' = \{(B, O), (O, S^t L), (B, L), (B, R), (L, C)\}$$

$$T = \{B, O, S^t L, L, R, C\}$$

	B	O	E	R	$S^t L$	L	N	C
B	-							
O	<del>7</del>	-						
E	18	17	-					
R	<del>9</del>	<del>11</del>	27	-				
$S^t$	<del>13</del>	<del>7</del>	23	<del>20</del>	-			
L	<del>7</del>	<del>12</del>	15	<del>15</del>	<del>15</del>	-		
N	38	38	20	40	40	35	-	
C	<del>22</del>	<del>15</del>	25	<del>25</del>	<del>30</del>	<del>10</del>	45	-

(f)

$$U' = \{(B, O), (O, S^t L), (B, L), (B, R), (L, C), (L, E)\}$$

$$T = \{B, O, S^t L, L, R, C, E\}$$

	B	O	E	R	$S^t L$	L	N	C
B	-							
O	<del>7</del>	-						
E	<del>18</del>	<del>17</del>	-					
R	<del>9</del>	<del>11</del>	<del>27</del>	-				
$S^t$	<del>13</del>	<del>7</del>	<del>23</del>	<del>20</del>	-			
L	<del>7</del>	<del>12</del>	<del>15</del>	<del>15</del>	<del>15</del>	-		
N	38	38	20	40	40	35	-	
C	<del>22</del>	<del>15</del>	<del>25</del>	<del>25</del>	<del>30</del>	<del>10</del>	45	-

(g)

$$U' = \{(B, O), (O, S^t L), (B, L), (B, R), (L, C), (L, E), (E, N)\}$$

$$T = \{B, O, S^t L, L, R, C, E, N\}$$

	B	O	E	R	$S^t L$	L	N	C
B	-							
O	<del>5</del>	-						
E	<del>18</del>	<del>17</del>	-					
R	<del>9</del>	<del>11</del>	<del>27</del>	-				
$S^t$	<del>13</del>	<del>7</del>	<del>23</del>	<del>20</del>	-			
L	<del>7</del>	<del>12</del>	<del>15</del>	<del>15</del>	<del>15</del>	-		
N	<del>38</del>	<del>38</del>	<del>20</del>	<del>40</del>	<del>40</del>	<del>35</del>	-	
C	<del>22</del>	<del>15</del>	<del>25</del>	<del>25</del>	<del>30</del>	<del>10</del>	<del>45</del>	-

On obtient donc finalement un arbre couvrant de coût égal à 73 de la forme :

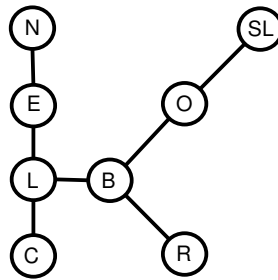


FIG. A.33: Arbre couvrant déduit de l'algorithme de Prim

## A.18 Exercice 20

### A.18.1 Questions

Le principe du calcul du produit de deux matrices<sup>1</sup> est le suivant :

$$c_{ij} = \sum_{k=1}^N a_{ik} * b_{jn} \text{ pour } i \text{ et } j \in [1, N]$$

1. Ecrire l'algorithme de multiplication des matrices
2. Évaluer la complexité de l'algorithme

### A.18.2 Réponses

```

1. Pour a allant de 1 à N faire      →      nE
    Pour i allant de 1 à N faire     →      (n * n)E
        Pour j allant de 1 à N faire →      (n * n * n)E
            Cai = Cai + (Aaj * Bij)  →      (n * n * n)E
        Fin pour                      →      (n * n * n)E
    Fin pour                          →      (n * n)E
Fin pour                             →      nE

```

$$\begin{aligned}
 2. \quad T(n) &= T_1(n) + T_2(n) + \dots + T_n(n) \\
 &= \underbrace{n^2E + n^2E + \dots + n^2E}_{N \text{ fois}} \\
 &= n * n^2 \\
 &= n^3
 \end{aligned}$$

Si  $E = 1UT$ , alors  $T(n) = n^3$  et  $T(n) = O(n^3)$

---

<sup>1</sup>On a comme hypothèse que les matrices soient toutes deux de taille  $N * N$

### A.19 Exercice 21

#### A.19.1 Questions

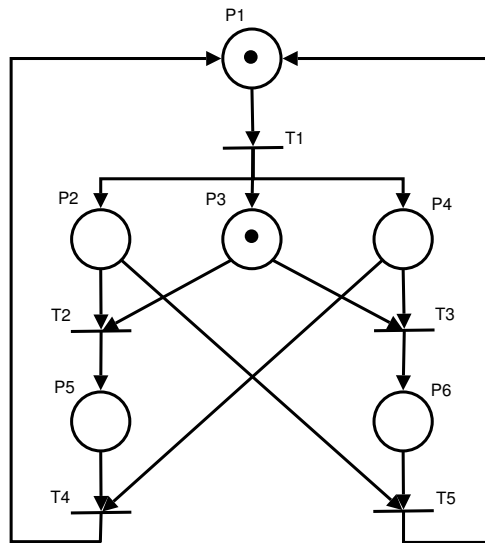


FIG. A.34: Réseau de Pétri à Analyser

1. Construire le graphe des marquages du réseau de Pétri
2. Le réseau est-il borné? Sans blocage?

#### A.19.2 Réponses

1. Arbre de couverture :

$$\begin{array}{ccccccc}
 M_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \xrightarrow{T_1} & M_1 = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \end{bmatrix} & \xrightarrow{T_2} & M_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} & \xrightarrow{T_4} & M_0 \\
 & & & \downarrow & & \searrow_{T_3} & \\
 & & & & & & M_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \\
 & & & \downarrow & & & \\
 & & & & & \nearrow & \\
 & & & & M_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} & \xrightarrow{T_5} & M_0
 \end{array}$$

2. Oui le graphe est borné ( $k = 2$ ) et il n'est pas sans blocage ( $M_4$  est un marquage de blocage)

# Table des figures

1.1	Exemple de graphe orienté. . . . .	7
1.2	Exemple de graphe orienté complet. . . . .	8
1.3	Exemple de graphe non-orienté. . . . .	9
1.4	Exemple de graphe non-orienté complet. . . . .	10
1.5	Exemple de graphe non-orienté simple. . . . .	11
1.6	Exemple de graphe connexe. . . . .	12
1.7	Exemple de graphe non-fortement connexe ayant deux composantes fortement connexes . . . . .	12
1.8	Exemple de graphe réduit . . . . .	12
1.9	Exemple de chemin et de circuit Hamiltonien . . . . .	13
1.10	Exemple montrant un sommet pour les degrés/demi-degrés d'un graphe orienté . . . . .	14
1.11	Exemple montrant un sommet pour le degré d'un graphe non-orienté . . . . .	15
1.12	Exemple montrant le cocycle d'un graphe . . . . .	16
1.13	Exemple d'application multivoque d'un graphe . . . . .	17
1.14	Exemple de graphe orienté pour exemple de matrice d'adjacence . . . . .	18
1.15	Exemple de graphe non-orienté pour exemple de matrice d'adjacence . . . . .	18
1.16	Exemple de graphe orienté pour exemple de liste d'adjacence . . . . .	19
1.17	Exemple de graphe non-orienté pour exemple de liste d'adjacence . . . . .	19
1.18	Exemple de graphe orienté pour exemple de matrice d'incidence . . . . .	20
1.19	Exemple de graphe non-orienté pour exemple de matrice d'incidence . . . . .	21
2.1	Représentation d'une liste linéaire sous forme d'un tableau . . . . .	23
2.2	Représentation d'une liste linéaire sous forme de liste chaînée . . . . .	23
2.3	Représentation d'une pile . . . . .	24
2.4	Représentation d'une file . . . . .	24
2.5	Exemple de graphe orienté pour le parcours en profondeur . . . . .	25
2.6	Représentation de la forêt courante déduite du parcours en profondeur . . . . .	27
2.7	Exemple de graphe non-orienté pour réaliser le parcours en largeur d'un graphe . . . . .	28
3.1	Exemple de graphe orienté pour réaliser les matrices booléennes et aux arcs . . . . .	32
3.2	Exemple de graphe orienté pour réaliser la fermeture transitive . . . . .	36
4.1	Exemple de graphe orienté pour recherche de chemins optimaux . . . . .	40
4.2	Exemple de graphe orienté pour exemple de recherche de chemins optimaux . . . . .	42
4.3	Contre-exemple de graphe pour l'algorithme de Ford . . . . .	44
4.4	Graphe orienté pour application de l'algorithme Floyd . . . . .	45
5.1	Graphe déterminé à partir de la méthode MPM . . . . .	49
5.2	Graphe déterminé à partir de la méthode PERT . . . . .	51
6.1	Exemple de réseau de transport . . . . .	53
6.2	Exemple de réseau de transport avec arc retour . . . . .	53
6.3	Exemple de réseau de transport avec arc retour et flots . . . . .	54
6.4	Graphe d'écart déduit du réseau de transport représenté sur la figure 6.3 . . . . .	55
6.5	Graphe d'écart déduit de l'application de l'algorithme de Ford-Fulkerson sur le graphe d'écart représenté sur la figure 6.4 . . . . .	56

6.6	Graphe du problème d'ordonnancement particulier d'ordonnancement . . . . .	58
6.7	Schéma illustrant la solution de base de l'algorithme du Stepping Stone . . . . .	59
6.8	Schéma illustrant la première optimisation de l'algorithme du Stepping Stone . . . . .	61
6.9	Schéma illustrant la deuxième optimisation de l'algorithme du Stepping Stone . . . . .	62
7.1	Exemple d'arbre . . . . .	64
7.2	Exemple de graphe pour recherche d'arbre couvrant . . . . .	64
7.3	Exemple d'arbre couvrant . . . . .	64
7.4	Exemple de graphe pour application de l'algorithme de Kruskal . . . . .	65
7.5	Résultat de l'application de l'algorithme de Kruskal . . . . .	65
8.1	Exemple de tableau pour analyse du temps d'exécution de l'algorithme qui retourne le plus petit indice d'un tableau . . . . .	68
8.2	Exemple de déroulement de l'algorithme du tri par sélection . . . . .	69
8.3	Exemple de déroulement de l'algorithme du tri par sélection . . . . .	73
9.1	Exemple de réseau de Pétri . . . . .	76
9.2	Exemple à modéliser en réseau de Pétri . . . . .	77
9.3	Exemple de modélisation de réseau de Pétri . . . . .	77
9.4	Exemple de réseau de Pétri non-vivant . . . . .	78
9.5	Exemple de réseau de Pétri non-pseudo-vivant . . . . .	78
9.6	Exemple de réseau de Pétri non-borné . . . . .	79
9.7	Exemple de réseau de Pétri borné . . . . .	79
9.8	Exemple de conflit structurel dans un réseau de Pétri borné . . . . .	79
9.9	Exemple de conflit effectif dans un réseau de Pétri borné . . . . .	80
9.10	Exemple de réseau de Pétri pour application de la méthode de construction de l'arbre de couverture . . . . .	81
A.1	Exercice 1 . . . . .	84
A.2	Réponse à la question 5 de l'exercice 1 . . . . .	85
A.3	Exercice 2 . . . . .	86
A.4	Exercice 3 . . . . .	87
A.5	Réponse à la question 1 de l'exercice 3 . . . . .	88
A.6	Réponse à la question 2.b de l'exercice 3 . . . . .	88
A.7	Exercice 4 . . . . .	89
A.8	Exercice 5 . . . . .	91
A.9	Réponse à la question 2 de l'exercice 5 . . . . .	91
A.10	Réponse à la question 1 de l'exercice 7 . . . . .	92
A.11	Exercice 8 . . . . .	96
A.12	Réponse à la question 3 de l'exercice 8 . . . . .	97
A.13	Exercice 9 . . . . .	98
A.14	Exercice 11 . . . . .	99
A.15	Réponse à la question 1 de l'exercice 12 . . . . .	102
A.16	Réponse à la question 6 de l'exercice 12 . . . . .	103
A.17	Réponse à la question 1 de l'exercice 13 . . . . .	104
A.18	Réponse aux questions 1 et 2 de l'exercice 14 . . . . .	105
A.19	Réponse aux questions 2, 3 et 4 de l'exercice 15 . . . . .	107
A.20	Exercice 17 . . . . .	108
A.21	Application d'un flot de retour sur le graphe A.20 . . . . .	108
A.22	Graphe d'écart déduit du graphe A.20 . . . . .	109
A.23	Graphe déduit de l'augmentation du flot sur le graphe A.22 . . . . .	109
A.24	Graphe d'écart déduit du graphe A.22 . . . . .	110
A.25	Réponse à la question 1 de l'exercice 17b . . . . .	112
A.26	Réponse à la question 5 de l'exercice 17b - première étape de l'application de l'algorithme de Ford-Fulkerson . . . . .	112
A.27	Réponse à la question 5 de l'exercice 17b - deuxième étape de l'application de l'algorithme de Ford-Fulkerson . . . . .	113

---

A.28 Réponse à la question 5 de l'exercice 17b - graphe d'écart final déduit de l'application de l'algorithme de Ford-Fulkerson . . . . .	113
A.29 Énoncé de l'exercice 19 . . . . .	114
A.30 Réponse à la question 1 de l'exercice 19 . . . . .	115
A.31 Solution de base de l'algorithme du stepping stone dans la question 2 de l'exercice 19 . . . . .	115
A.32 Résultat de la première itération de l'algorithme du stepping stone dans la question 2 de l'exercice 19 . . . . .	116
A.33 Arbre couvrant déduit de l'algorithme de Prim . . . . .	119
A.34 Réseau de Pétri à Analyser . . . . .	121

# Bibliographie

- [Des] Desbazeille. *Exercices et problèmes de Recherche Opérationnelle*. Dunod.
- [Faua] Faure. *Guide de la Recherche Opérationnelle : Les Applications*, volume 2. Masson.
- [Faub] Faure. *Guide de la Recherche Opérationnelle : Les Fondements*, volume 1. Masson.
- [Fauc] Faure. *Précis de Recherche Opérationnelle*. Dunod.
- [Gon] Gondrand. *Graphes et algorithmes*. Eyrolles.
- [Rosa] Roseaux. *Exercices et problèmes résolus de Recherche Opérationnelle : Les graphes, leurs usages et leurs algorithmes*, volume 1. Masson.
- [Rosb] Roseaux. *Exercices et problèmes résolus de Recherche Opérationnelle : Phénomènes aléatoires en Recherche opérationnelle*, volume 2. Masson.