REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseaux et télécommunication

Par :

BOUSTA Abdelkader et NAAB Khaled

Sur le thème

Solveur automatique d'algorithmes de graphes (flot maximal)

Soutenu publiquement le 11 / 07 / 2023 à Tiaret devant le jury composé de :

Mr DJAAFRI Laouni	MCB	Université Tiaret	Président
Mr ALEM Abdelkader	MAA	Université Tiaret	Encadreur
Mme LAAREDJ Zohra	MAA	Université Tiaret	Examinateur

2022-2023

REMERCIEMENTS

Avant tout, On remercie ALLAH le tout puissant pour nous avoir donné leCourage, la santé et la patience pour réaliser ce travail

On tient à remercier tout particulièrement notre Encadreur Mr. Alem Abdelkader pour son encadrement, sa disponibilité et ses bons conseils tout le long de ce travail.

On remercie les membres du jury pour l'honneur qu'ils nous font en participant au jugement de ce travail. Enfin, nous voulons exprimer nos remerciements à toute personne qui a contribué de près ou de loin pour la finalisation de ce travail.

Dédicace

Je dédie ce modeste travail de mes années d'étude.

A ma très chère mère qui m'a assuré un soutien inconditionnel par ses encouragements sur tous les plans, sa disponibilité, leur affection et leur grand amour aussi.

A mon cher père qui m'a toujours servi de modèle et des baux principes pour une vie idéale par sa patience, ses conseils, sa compréhension.

A mes frères et mes sœurs pour leurs appuis et leurs encouragements permanents.

A Mon binôme Khaled.

Abdelkader

Dédicace

Je dédie ce mémoire

A mes chers parents ma mère et mon père

Pour leur patience, leur amour, leur soutien et leurs encouragements.

A mes frères et sœurs

A mes amies et mes camarades surtout AHMED , KHALED

et ABDELKADER

Sans oublier tous les professeurs que ce soit du primaire, du moyen,

du secondaire ou de l'enseignement supérieur

khaled

Résume

Il nous arrive parfois de résoudre un type de problèmes plusieurs fois, jusqu'à arriver à un modèle de résolution stable, de ce fait nous refaisons les mêmes étapes et nous remarquons qu'il sera préférable d'automatiser cette résolution.

Dans la même optique, nous avons constaté que l'écriture d'un corrigé d'un certain type d'exercice peut suivre aussi un modèle et nous avons tiré avantage du langage de formatage de texte LATEX pour concevoir une application qui permet de résoudre dans un premier temps un exercice de la théorie des graphes (le flot maximal) et de produire par la suite les instructions LATEX qui permettent de produire la résolution (ou le corrigé type).

Ceci permet de libérer le correcteur de deux taches embêtantes : résoudre le problème manuellement (et donc vérifier la solution plusieurs fois) et se casser la tête pour écrire le code LATEX pour générer le corrigé type.

Mots clés : théorie des graphes, flot maximal, LaTeX, Réécriture.

Abstract

We sometimes solve a type of problem several times, until we arrive at a stable resolution model ; therefore we repeat the same steps and we notice that it will be better to automate this resolution.

In the same vein, we found that writing a correction to a certain type of exercise can also follow a model and we took advantage of the LATEX text formatting language to design an application that initially solves an exercise of the theory of graphs (the maximum flow) and then produce the LATEX instructions that allow you to produce the resolution (or the standard correction).

This frees the corrector from two annoying spots : solve the problem manually (and thus check the solution several times) and break the head to write the LATEX code to generate the standard correction.

Keywords : graph theory , maximum flow, LaTeX, Rewriting.

Sommaire

Chapitre II: Flot maximal

Chapitre III: Implémentation

CONCLUSION

LISTE DES FIGURES

Introduction générale

Introduction générale

L'automatisation des taches est devenue de plus en plus importante, vu leur nombre élevé et leurs complexités. Mais pour automatiser une tache il faut pouvoir arriver à un algorithme de résolution, à des caractéristiques communes : à un modèle.

Dans cette optique nous essayons de concevoir une application qui libère un enseignant d'une tache souvent embêtante : résolution des exercices et production de corrigé type.

Nous avons pris comme exemple, la résolution de problèmes de la théorie des graphes (le flot maximal).

Pour cette classe de problèmes, les algorithmes existent, et la solution ou le corrigé type suit implicitement un modèle, en d'autres termes, pour être cohérent et pédagogique, un enseignant ne va pas changer à chaque fois sa façon d'exprimer la solution !

En outre, nous avons tiré avantage de l'aspect programmable du langage de formatage de texte LATEX, pour produire les instructions ou le code permettant de générer la correction.

Cette application peut être généralisée sur les autres problèmes de la théorie des graphes, ou sur des problèmes d'autres matières pour lesquels la solution suit un algorithme ou un modèle.

De ce fait les longues heures de casse-tête pour vérifier la solution et surtout pour écrire le code LATEX permettant de la générer se réduisent en quelques minutes, il faut juste entrer les données du problème et lancer la résolution/génération de code.

Ceci permet à la fois de tirer avantage du langage LaTeX qui libère l'utilisateur de la mise en forme du texte (contrairement aux outils classiques de saisie !) et d'annuler les erreurs syntaxiques d'écriture du code latex, puisque le modèle est fixé au préalable et les instructions qui le produisent sont vérifiées une fois pour toutes lors de la translation.

Chapitre I

Notions de base de la Théorie de graphe

I.1 Introduction

Pour trouver des solutions optimales aux problèmes de planification, sciences sociales, architecture urbanisme, ingénierie, informatique, télécommunication, etc. Du point de vue mathématique, les graphes sont aujourd'hui liés à la combinatoire, à ce que l'on appelle les mathématiques discrètes, à la topologie, à la théorie des algorithmes, à la théorie des nœuds, etc. De nombreuses théories ont permis de développer les graphes et elles constituent à leur tour de solides armes pour résoudre des problèmes posés dans d'autres disciplines.

Les graphes permettent de trouver des solutions optimales aux problèmes de planification, sciences sociales, architecture urbanisme, ingénierie, informatique, télécommunication, etc. Du point de vue mathématique, les graphes sont aujourd'hui liés à la combinatoire, à ce que l'on appelle les mathématiques discrètes, à la topologie, à la théorie des algorithmes, à la théorie des nœuds, etc. De nombreuses théories ont permis de développer les graphes et elles constituent à leur tour de solides armes pour résoudre des problèmes posés dans d'autres disciplines.

I.2 Qu'est-ce qu'un graphe

Un graphe G est constitué d'un ensemble de points (appelés éléments, sommets ou nœuds et d'un ensemble d'arêtes ou lignes qui relient deux sommets.

Graphe G = (X, U)

X : ensemble fini des sommets notés {1, 2, ...} ou {x, y, ...}

U : ensemble fini de couples ordonnés (x, y) avec x, y ∈ X.

Les éléments de U sont appelés les arêtes du graphe. [1]

I.2.1 Graphe non orienté

Un graphe fini G = (V, E) est défini par l'ensemble fini V = {v_1, v_2,..., v_n} dont les éléments sont appelés sommets, et par l'ensemble fini E = {e_1, e_2,..., e_m} dont les éléments sont appelés arêtes. Chaque élément ei de l'ensemble E est définie par une paire non ordonnée de sommets, appelés les extrémités de ei. Si l'arête e relie les sommets a et b, on dira que ces sommets sont adjacents, ou incidents avec e_i. [2]

Exemple :

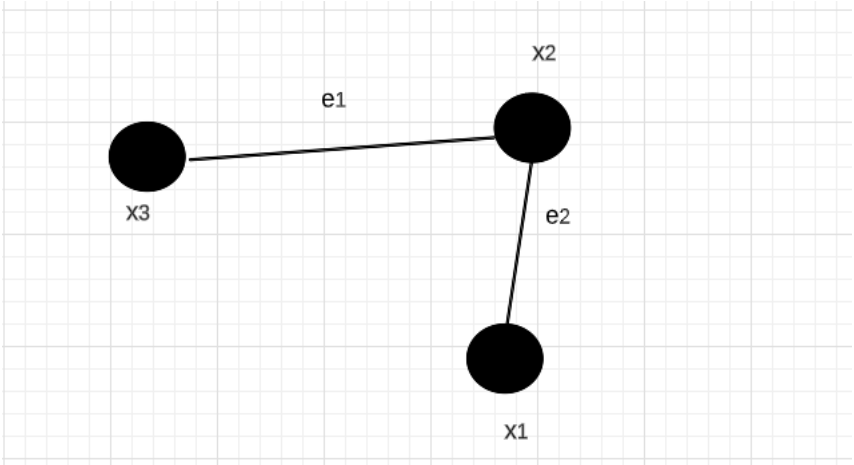


FIGURE I -1 : Graphe non orienté

I.2.2 Graphe orienté

Un graphe orienté (digraphe) est un graphe où le couple (1, 2) est représenté graphiquement par $1 \rightarrow 2$, On appelle 1 l'extrémité initiale ou origine, et 2 est l'extrémité terminale. [3]

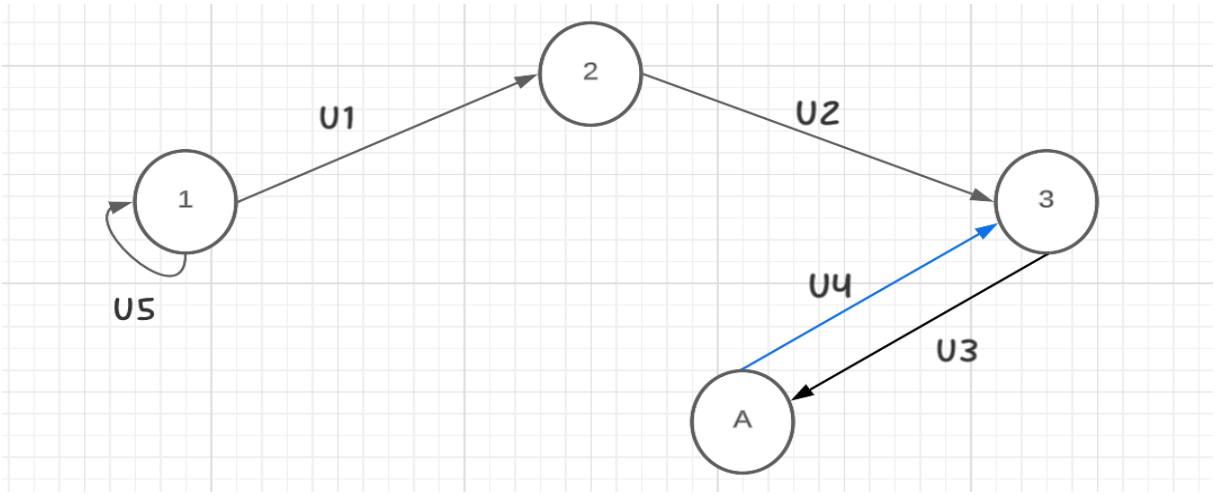


FIGURE I -2 : Graphe orienté

X = {1, 2, 3,4}

U = {u1, u2, u3, u4, u5}.

- U1 = (1, 2)
- U5 = (1, 1) est une boucle
- 2 est pour 1 un successeur

- 1 est pour 2 un prédécesseur
- Γ +(2) = {4} successeurs de 2
- Γ-1 (4) = {2,3} prédécesseurs de 4.

I.2.3 Sous-graphe

Soit G (X, U) un graphe, Un sou-graphe de G est un graphe de la forme G' (X', U') ou X' ⊂ X et U' ⊂ U sont tels que toute arête de U' a ses extrémités dans X', Notons que le fait qu'un sous-graphe est un graphe implique cette propriété que toute arête de U' a ses extrémités dans X'.[4]

I.2.4 Graphe partiel

Soit G (X, U) un graphe, on appelle graphe partiel de G, le graphe G' (X', U') tel que : X'=X, U' ⊂ U (c'est un graphe ayant le même ensemble X de sommets, mais un ensemble diffèrent des arcs U' ⊂ U).[5]

I.2.5 Degré et demi degré d'un sommet

Degré : le nombre des arêtes qui relient les sommets

- Le demi degré extérieur de x ∈ X, d +(x) est le nombre d'arcs sortant de x.
- Le demi degré inferieur de x ∈ X, d −(x) est le nombre d'arcs entrant `a x.
- Le degré de x ∈ X, d(x) = d +(x) + d −(x). [5]

I.2.6 Chaine et cycle

- **Une chaine** est une suite d'arêtes, donc de sommets adjacents.
  La longueur d'une chaine est le nombre d'arêtes qu'elle comporte. [6]

- **Chaîne simple** : C'est une chaîne ou les (arcs/arêtes) sont pris(es) une seule fois. [3]
- **Chaîne élémentaire** : C'est une chaîne dont tous les sommets du graphe sont pris une seule fois. [7]
- **Une chaîne hamiltonienne** : est une chaîne élémentaire qui passe par tous les sommets du graphe. [7]

Une chaîne eulérienne : est une chaîne qui passe par tous les arcs du graphe une et une seule fois. [3]

- **Un cycle** est une chaine dont l'origine et l'extrémité sont confondues (on pourra donc dire qu'il a une seule extrémité), à condition que toutes les arêtes soient différentes.
  Un cycle peut comporter plusieurs fois le même sommet.
  Si les arêtes ne sont pas toutes distinctes, on parlera simplement de « chaine fermé ». [2]

  - **Un cycle hamiltonien** : est un cycle élémentaire qui passe par tous les sommets du graphe. [3]
  - **Un cycle eulérien** : est une chaîne simple et fermée qui passe par tous les arcs du graphe une et une seule fois. [7]

I.2.7 Chemin et Circuit :

- **Un chemin** dans un graphe est une liste ordonnée de sommets telle que deux sommets consécutifs soient adjacents, les sommets du graphe apparaissant dans la liste sont les sommets du chemin et les arêtes du graphe dont les extrémités sont des sommets consécutifs du chemin sont les arêtes du chemin.

-Si le chemin C écrit (x0, x1, xk),
X0 en est l'origine,
Xk l'extrémité,
K la longueur. [8]

- **Un circuit** est un chemin son origine et son extrémité sont le même sommet (chemin fermé).[8}

I.3 Différents types de graphes

I.3.1 Graphe complet

On peut dire qu'un graphe complet est un graphe simple dont tous les sommets sont adjacents, c'est-à-dire que tout couple de sommets est relié par une arête. [9]

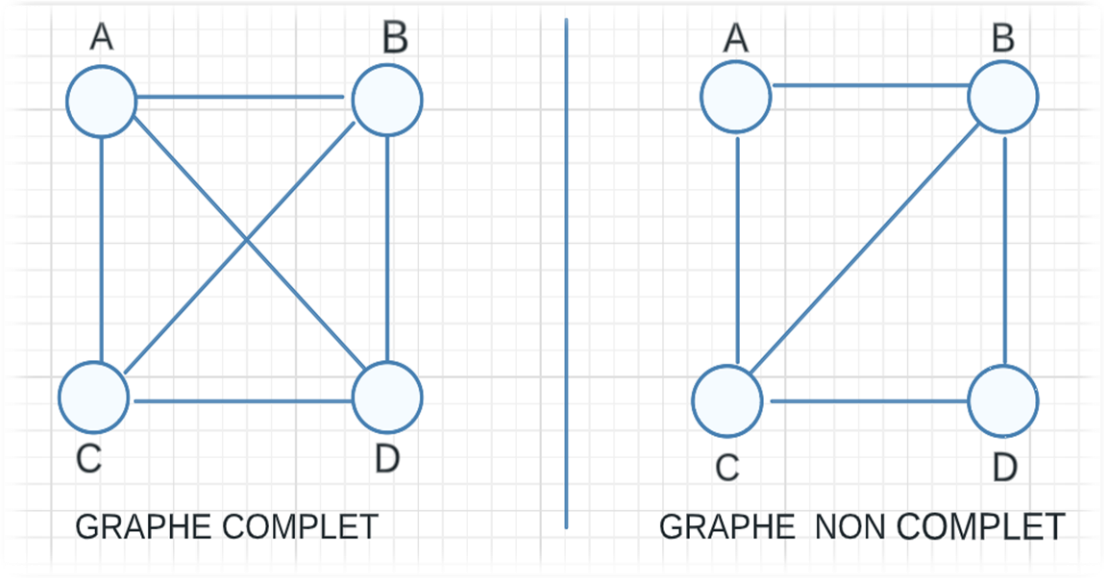


FIGURE I -3 : Graphe complet et non- complet

I.3.2 Graphe simple

Un graphe G (X, U) est dit simple si seulement si :

1. Ne contient aucune boucle.
2. $\forall$ X, y $\in$ X, $\exists$ au plus u= (x, y) $\in$ U .[9]

Exemple :

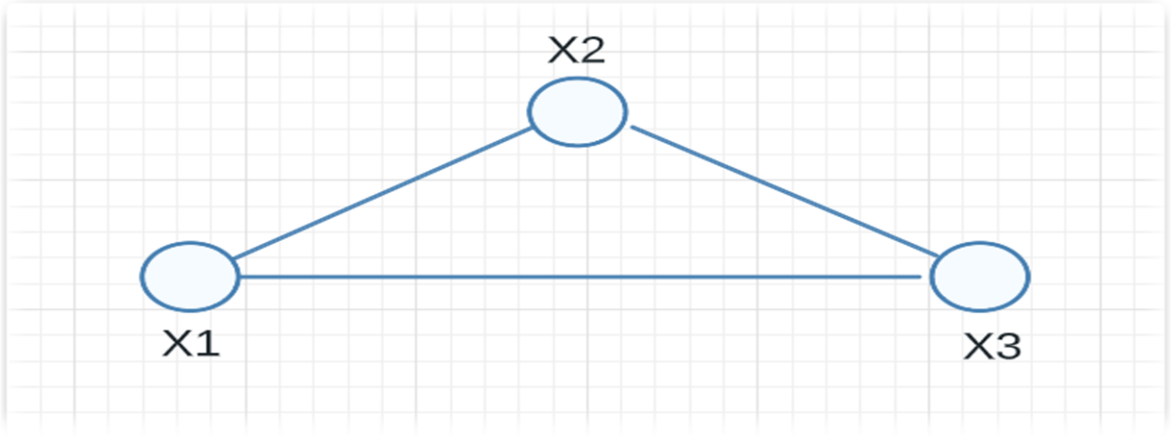


FIGURE I -4 : Graphe simple

I.4 Quelques graphes particuliers

I.4.1 Graphe biparti

Un graphe est biparti si ses sommets peuvent être divisés en deux ensembles X et Y, de sorte que toutes les arêtes du graphe relient un sommet dans X à un sommet dans Y (dans l'exemple ci-dessous, on a X = {1,3,5} et Y = {2,4}, ou vice versa). Et on note : [10]

Graphe biparti

V = {1,2,3,4,5}

E = {1,2}, {1,4}, {2,5}, {3,4}, {4,5}

Exemple :

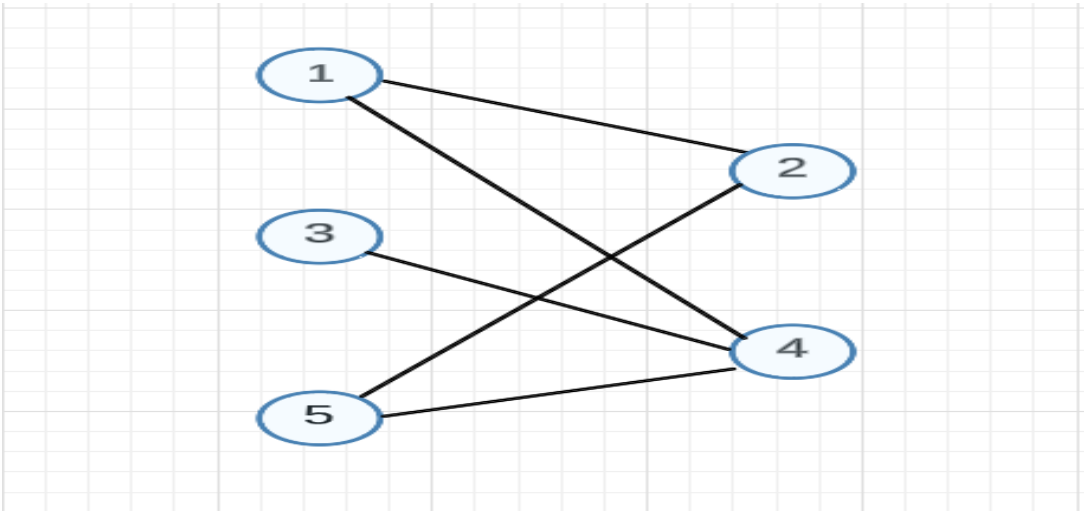


FIGURE I -5 : graphe biparti

I.4.2 Graphe planaire

On dit qu'un graphe est planaire si on arrive à le dessiner sans qu'aucune arête ne coupe une autre.[11]

Exemple :

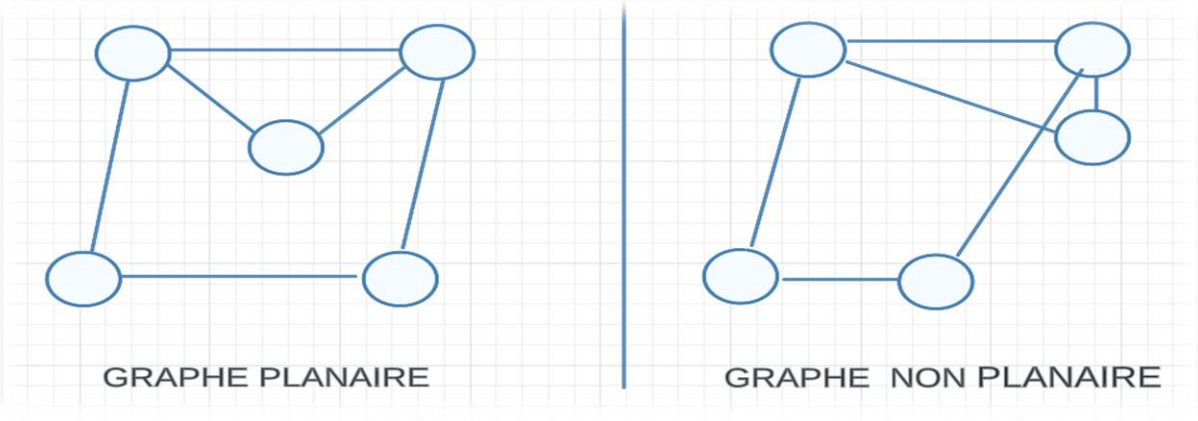


FIGURE I -6 : Graphe planaire et non planaire

I.4.3 Graphe aux arêtes

Le graphe aux arêtes de G c'est le graphe G' (X', E'), où |X'|=|E|, chaque arête e_i de G on lui associe un sommet yi de G' et deux sommets x, y de G' sont adjacents si et seulement si les arêtes qui leur correspondent dans G sont adjacentes. [12]

Exemple :

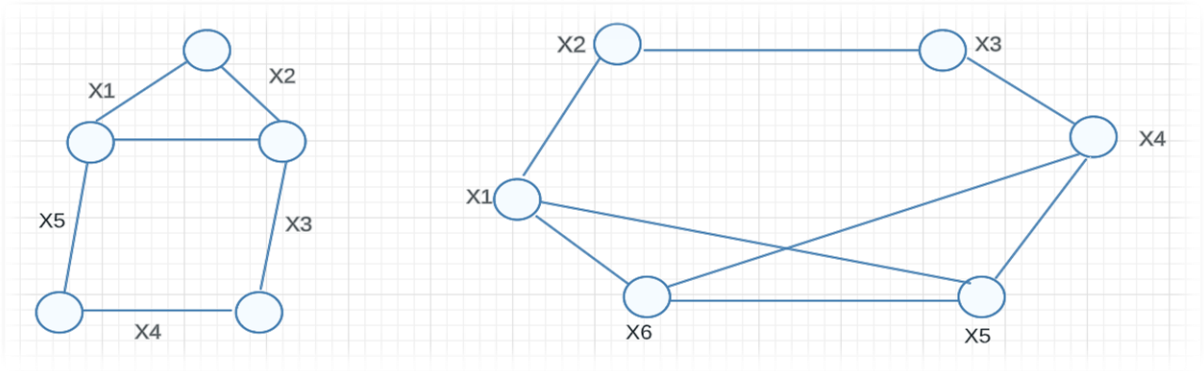


FIGURE I -7 : Graphe aux arêtes

I.5 Connexité dans les graphes

Dans un graphe G= (X, U). Elle peut être représentée par une succession de sommets et d'arcs sous la forme C= (x_1, u_1, x_2, u_2, ..., x_k), où x_i et x_i+1 sont les extrémités de l'arc u_i.

Remarque :

Une chaîne est dite élémentaire si elle ne traverse pas plus d'une fois le même sommet.

•Une chaîne est dite simple si elle ne traverse pas plus d'une fois la même arête.

Un cycle est une chaîne fermée qui relie un sommet de départ à un sommet d'arrivée en passant par une séquence de sommets intermédiaires, de sorte que les extrémités de la chaîne se confondent.

Il est important de noter que

•Un cycle est dit élémentaire s'il ne traverse pas plus d'une fois le même sommet (à l'exception de l'extrémité commune).

•Un cycle est dit simple s'il ne traverse pas plus d'une fois la même arête. [13]

Exemple :

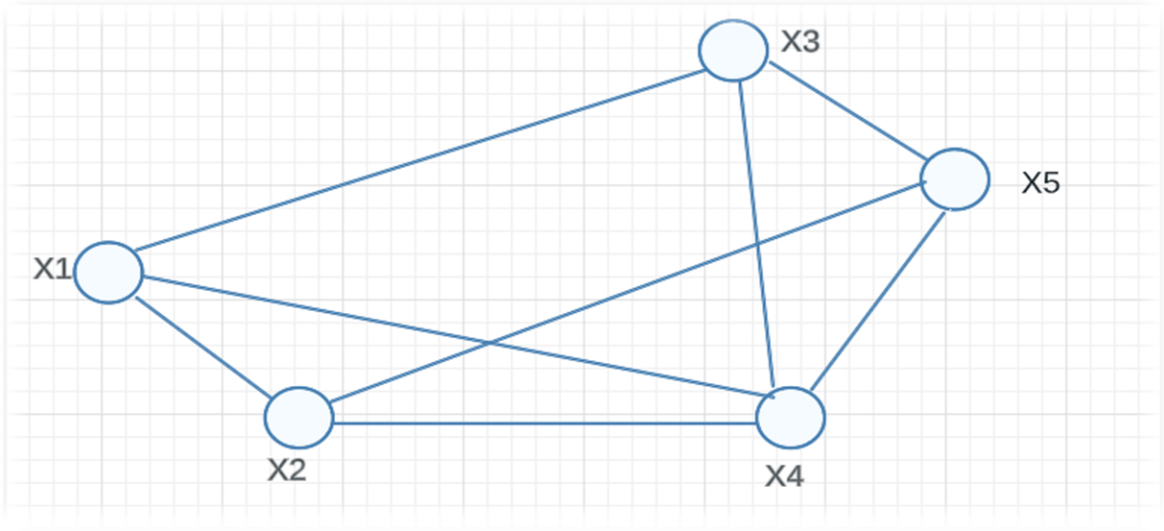


FIGURE I -8 : connexité dans les graphes

S= (x2, x1, x3, x4, x5) est une chaîne.

S= (x3, x5, x2, x1, x3) est un cycle.

I.5.1 Connexité

La connexité est une notion importante en théorie des graphes.

Soit G= (X, U) un graphe et $\mathcal{R}$ une relation sur X telle que x $\mathcal{R}$ y si et seulement s'il existe une chaine entre x et y dans G, ou si x=y.

Cette relation est une relation d'équivalence, c'est-à-dire qu'elle est réflexive, symétrique et transitive.

Les classes d'équivalence pour $\mathcal{R}$ sont appelées les composantes connexes de G.

Un graphe est connexe s'il n'a qu'une seule composante connexe, c'est-à-dire qu'il forme un seul "bloc".

En revanche, s'il a plusieurs composantes connexes, il est dit connexe.

La connexité est donc une propriété fondamentale des graphes, qui permet de déterminer leur structure et leur connectivité. [14]

Exemple :

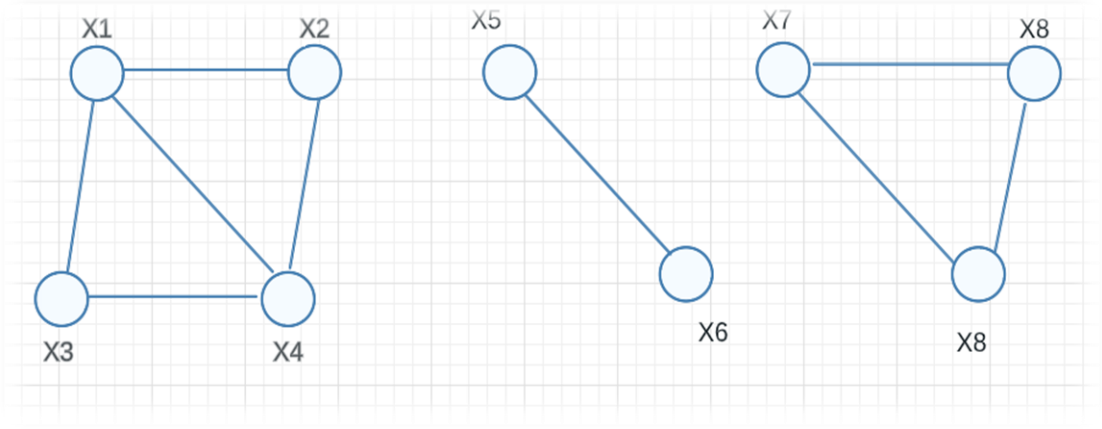


FIGURE I -9 : Graphe ayant trois connexes

I.5.2 Forte connexité

Soit G= (X, U) un graphe et soit $\mathcal{R}$ une relation sur X.

$\mathcal{R}$: x, y ϵ X, x $\mathcal{R}$ y $\Leftrightarrow$ "x=y ou bien $\exists$ un chemin orienté de x à y et $\exists$ un chemin orienté de y à x".

$\mathcal{R}$ est une relation d'équivalence :

- Elle est réflexive :

X $\mathcal{R}$ x, car il existe un chemin orienté de x à x (un chemin élémentaire qui ne visite qu'une seule fois chaque sommet).

- Elle est symétrique :

Si x $\mathcal{R}$ y, alors il existe un chemin orienté de x à y et un chemin orienté de y à x. Donc, il existe un chemin orienté de y à x et un chemin orienté de x à y, donc y $\mathcal{R}$ x.

- Elle est transitive :

Si x $\mathcal{R}$ y et y $\mathcal{R}$ z, alors il existe un chemin orienté de x à y et un chemin orienté de y à z. En concaténant ces deux chemins, on obtient un chemin orienté de x à z. De même, il existe un chemin orienté de z à y et un chemin orienté de y à x. En concaténant ces deux chemins, on obtient un chemin orienté de z à x. Donc, x $\mathcal{R}$z.

Classes d'équivalence :

X ϵ X, cl(x)= {yϵ X /y $\mathcal{R}$x}.

- Une classe d'équivalence de $\mathcal{R}$ est une composante fortement connexe de G.

- Un graphe est fortement connexe s'il n'admet qu'une seule composante fortement connexe. [15]

Exemple :

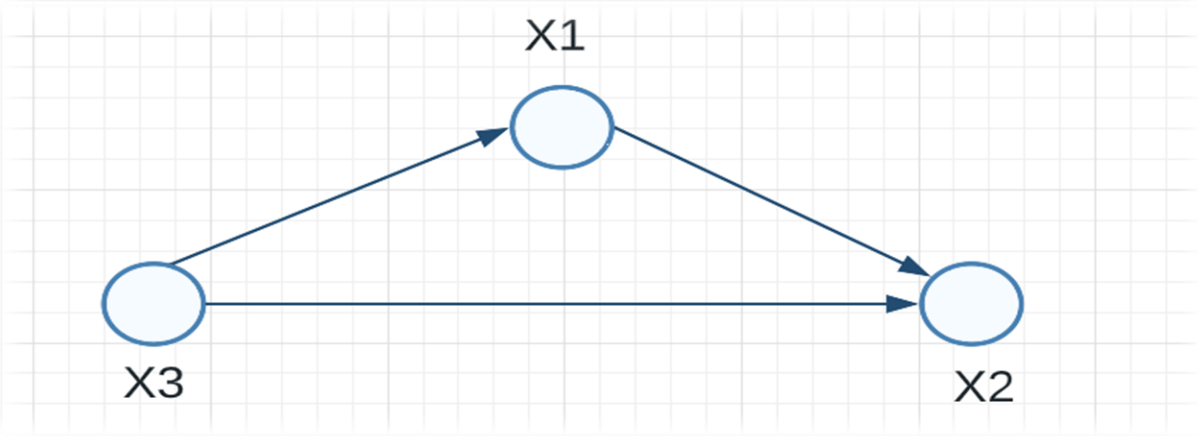


FIGURE I -10 : Graphe fortement connexe

I.5.3 Graphe réduit

On note X_r l'ensemble quotient $X/\approx$ où chaque composante fortement connexe différente est représentée par un élément dans cet ensemble.

On définit l'ensemble E_r d'arcs entre les points de X_r de la façon suivante :
$\forall (c , c') \in X_r \times X_r : (c, c') \in E_r \Leftrightarrow c=c'$ ou $\exists x \in c$ et $\exists x' \in c'$ tel que $(x , x') \in E$.
Le graphe (X_r , E_r) est appelé le graphe réduit du graphe (X,E).

Le graphe réduit d'un graphe G a donc autant de points que de composantes fortement connexe dans le graphe G.
Il y a un arc entre un point représentant une composante fortement connexe et un autre point représentant une autre composante si et seulement s'il existe au moins un arc entre un point de la première composante fortement connexe et un point de la deuxième composante fortement connexe. [16]

Exemple :

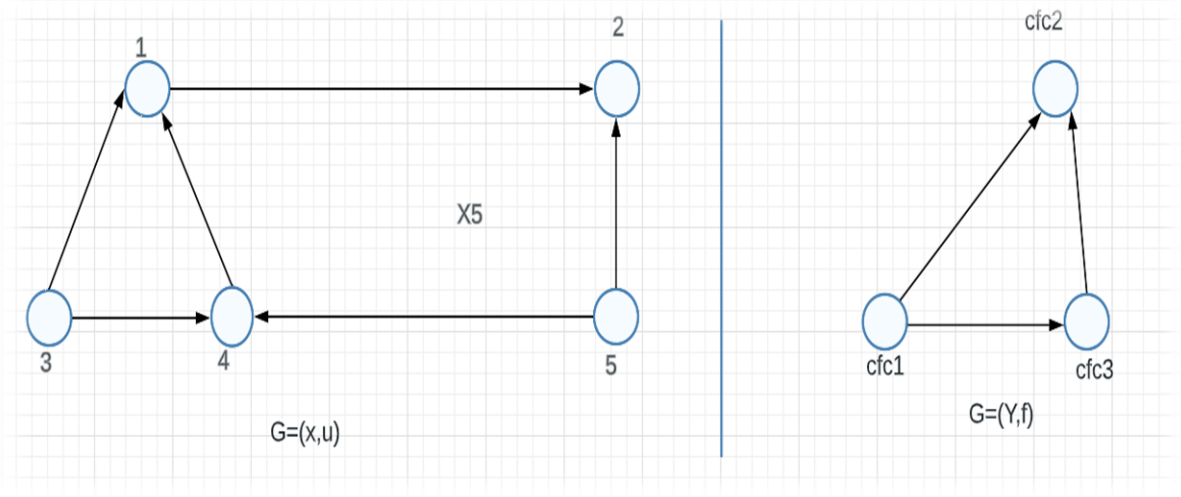


FIGURE I -11 : Graphe réduit

Cfc1= {1, 3, 4}, cfc2= {2}, cfc3= {5}.

Y= {cfc1, cfc2, cfc3}

I.6 Arbres et arborescences

I.6.1 Arbres

Les arbres et les arborescences sont des graphes particuliers très souvent utilisés en informatique pour représenter des données.
Etant donné un graphe non orienté comportant n sommets, les propriétés suivantes sont équivalentes pour caractériser un arbre :

1. G est connexe et sans cycle,
2. G est sans cycle et possède n − 1 arêtes,
3. G est connexe et admet n − 1 arêtes,
4. G est sans cycle, et en ajoutant une arête, on crée un et un seul cycle élémentaire,
5. G est connexe, et en supprimant une arête quelconque, il n'est plus connexe,
6. Il existe une chaine et une seule entre 2 sommets quelconques de G. [17]

Exemple :

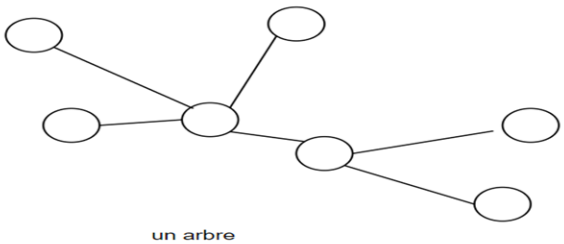


FIGURE I -12 : Arbre

I.6.2 Arbre couvrant

Un arbre d'un graphe G= (X, A) est un sous-graphe partiel de G qui est connexe et sans cycle. Si cet arbre inclut tous les sommets de G, il est appelé arbre couvrant ou arbre maximum.

Un arbre couvrant est donc un arbre qui connecte tous les sommets d'un graphe.

Une forêt est un graphe dont chaque composante connexe est un arbre. Tout graphe partiel d'un arbre est également une forêt.

Une racine d'un graphe G= (X, A) est un sommet s qui est relié par des chemins à tous les autres sommets du graphe, à l'exception de s lui-même.

Autrement dit, pour tout sommet x dans X différent de s, il existe un chemin de s à x.

Une arborescence est un graphe G= (X, A) avec n≥2 sommets qui est un arbre, où l'un des sommets est une racine. Plus précisément, une arborescence est un arbre qui a une racine désignée.

Tous les sommets de l'arborescence sont accessibles depuis la racine. Cependant, tous les arbres ne sont pas des arborescences, car un arbre peut ne pas avoir de racine désignée.

En résumé, un arbre est un sous-graphe partiel connexe et sans cycle d'un graphe G. Un arbre couvrant est un arbre qui inclut tous les sommets de G. Une forêt est un graphe dont chaque composante connexe est un arbre. Une racine d'un graphe est un sommet qui est relié à tous les autres sommets du graphe, à l'exception de lui-même. Une arborescence est un arbre qui a une racine désignée. [15]

Exemple :

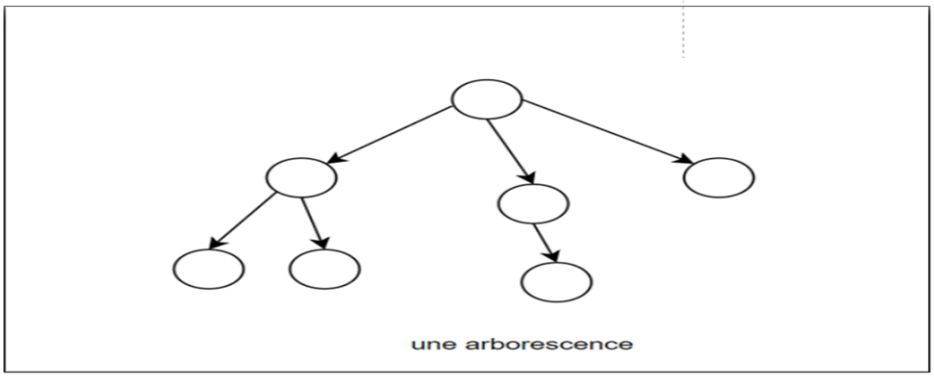


FIGURE I -13 : Arborescence

I.7 Pondération des graphes

I.7.1 Graphes à sommets pondérés

Un graphe G (X, E, p) à sommets pondérés ou values est un graphe G (X, E) où chaque sommet est associé à une valeur numérique p(x) ∈ ℝ. [15]

I.7.2 Graphes à arêtes pondérées

Un graphe G (X, E, p) à arêtes pondérées ou values est un graphe G (X, E) où chaque arête est associée à une valeur numérique p(e) ∈ ℝ. [15]

Exemple :

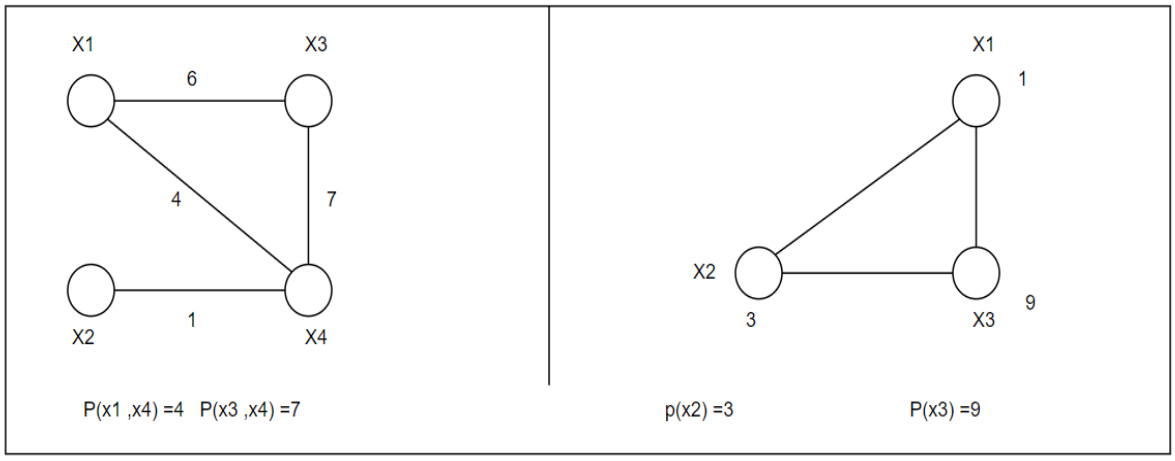


FIGURE I -14 : Arêtes pondérées

I.8 Couplages dans les graphes

Un couplage M dans un graphe G (X, E) est un sous-ensemble E'⊂ E tel que pour toute paire (ei, ej) ∈E'2, ei n'est pas adjacent à ej.

I.8.1 Couplage maximal

Un couplage maximal est un couplage M ayant la propriété que si une arête e est ajoutée, alors M ∪ {e} n'en est pas. [18]

Exemple :

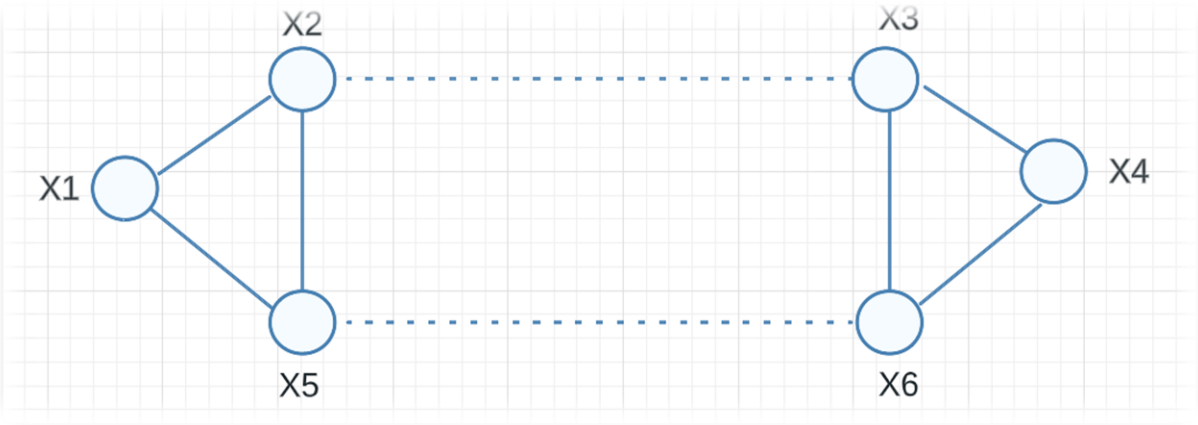


FIGURE I -15 : Couplage maximal

E'= {(x2, x3), (x5, x6)} est un couplage maximal.

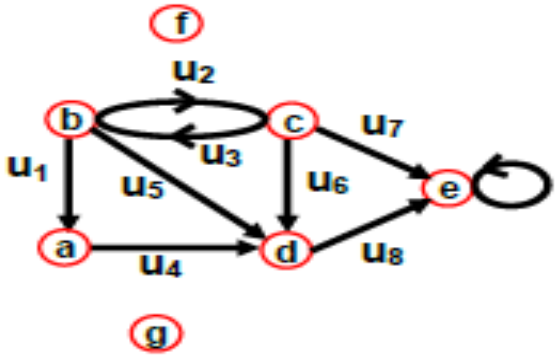
I.9 Représentation Matricielle des graphes

I.9.1 Matrice d'incidence (sommet-arc)

Soit le graphe G (X, U) suivant:

X = {a, b, c, d, e, f, g}

U = {(a, d), (b, a), (b, c), (b, d), (c, b), (c, d), (c, e), (d, e), (e, e)} [19]



	u1	u2	u3	u4	u5	u6	u7	u8	u9
a	-1	0	0	+1	0	0	0	0	0
b	+1	+1	-1	0	+1	0	0	0	0
c	0	-1	+1	0	0	+1	+1	0	0
d	0	0	0	-1	-1	-1	0	+1	0
e	0	0	0	0	0	0	-1	-1	2
f	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	0

Graphe G (X, U) **A (N, M)**

FIGURE I.16 : Représentation par Matrice d'incidence (sommet-arc) [19]

Remarques:

- Chaque colonne dans la matrice contient un seul (+1) correspond à l'extrémité initiale de l'arc, et un seul (-1) correspond à son extrémité terminale
- Le nombre des (+1) sur la ligne donne le 1/2 degré extérieur du sommet, bien que le nombre des (-1) donne le 1/2 degré intérieur du même sommet.

I.9.2 Représentation matricielle des graphes valués

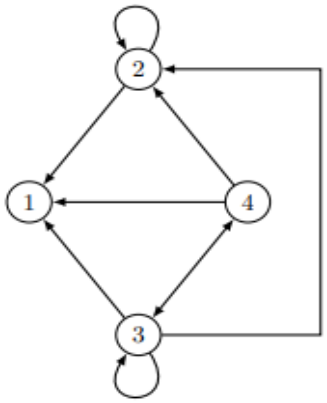
Soit le graphe G = (X, U) d'ordre N. On suppose que les sommets de S

Sont numérotés de 1 à N. La représentation par matrice d'adjacence de G consiste en une matrice booléenne M de taille N × N telle que M[i][j] = 1 si (i, j) ∈ A, et M[i][j] = 0 sinon.

Dans le cas de graphes non orientés, la matrice est symétrique par rapport à sa diagonale descendante. Dans ce cas, on peut ne mémoriser que la composante triangulaire supérieure de la matrice d'adjacence. [20]

•Exemple :

On considère la matrice d'adjacence pour le graphe suivant :



graphe G(X, U)

$$\begin{array}{c|cccc} \nearrow & 1 & 2 & 3 & 4 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$

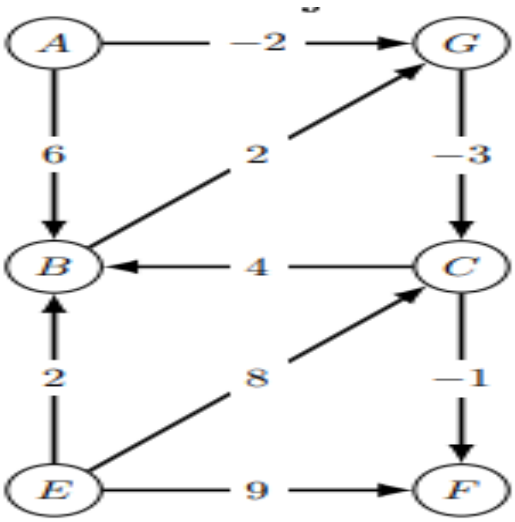
A(N, N)

FIGURE I.17 : Représentation par matrices d'adjacence (sommet-sommet)

Un graphe valué G = (X, U, c) peut être représenté par la matrice des valuations :

$$W \in (R) \text{ telle que : } W_{i,j} = \begin{cases} 0 & si\ (xi, xj) \notin U \\ c((xi, xj)) & si\ (xi, xj) \in U \end{cases}$$

Exemple : La matrice d'adjacence du graphe valué suivant est :



0	6	0	0	0	-2
0	0	0	0	0	2
0	4	0	0	-1	0
0	2	8	0	9	0
0	0	0	0	0	0
0	0	-3	0	0	0

FIGURE I.19 : Représentation des graphes valués

I.10 Conclusion

Ce chapitre comporte plusieurs définitions, nous avons essayé de simplifier les concepts de la théorie de graphes le plus possible et nous avons introduits ses bases fondamentales, y compris les définitions des graphes, des sommets, des arêtes et des différents types de graphes tels que les graphes orientés, bipartis, etc.

Nous avons exploré des concepts clés tels que les degrés des sommets, les chemins, les cycles, les graphes connexes. De plus, nous avons étudié les représentations graphiques telles que les matrices d'adjacence.

Chapitre II
Flot maximal

II.1 Introduction

Le problème du flot maximal est l'un des problèmes les plus fondamentaux en théorie des graphes et en algorithmique. Il consiste à déterminer le flot maximum qu'il est possible de faire circuler dans un réseau donné, en respectant les capacités des différentes arêtes et la conservation du flot.

Ce problème a de nombreuses applications pratiques, notamment dans les réseaux de transport, les réseaux de communication, les réseaux électriques, les réseaux de distribution d'eau, etc. Par exemple, dans un réseau de transport, le problème du flot maximal peut être utilisé pour déterminer le nombre maximal de véhicules que l'on peut faire circuler entre deux points donnés, en tenant compte des limitations de capacité des différentes routes.

II.2 Réseau de transport

II.2.1 Définition

Un Réseau de Transport est un graphe G = (X, U, C) orienté et sans boucle dans lequel chaque arc a une capacité :

Cij ≥ 0 si (i, j) ∈ X,

Cij = 0 sinon.

On distingue deux sommets particuliers : la source **s** et le puits **p**. On supposera, par commodité qu'il existe toujours un chemin permettant d'aller de **s** à **p**. [21]

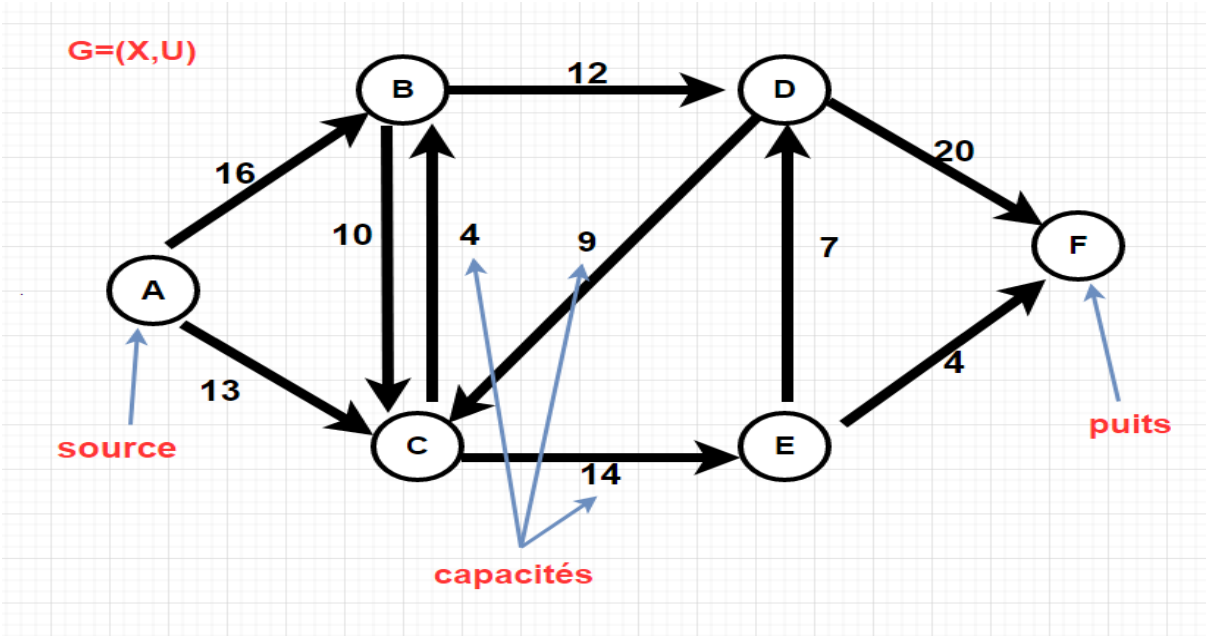


FIGURE II -1 : Exemple réseau de transport

II.2.2 Exemples de réseaux de transport

- Réseau ´électrique : haute tension, circuit....
- Réseau routier, ferroviaire : transport de marchandises,
- Réseau de fluides, gaz, hydrocarbures, [21]

II.3 Flots

II.3.1 Définition d'un flot

- ϕ_{ij} désigne le flux sur l'arc (i, j), qui correspond à la quantité de « matière » circulant sur l'arc.
- Le vecteur $\phi = (\phi_{ij})_{(i,j) \in U \cup (t,s)}$ est appelé flot.
- Les flux sur R doivent vérifier la contrainte de capacité : $0 \leq \phi_{ij} \leq c_{ij}$.
- Lorsque $\phi_{ij} = c_{ij}$, l'arc (i, j) est dit saturé
- Les flux doivent également respecter la loi de conservation (≈loi de Kirchhoff) :

$$\forall j \in X, \sum_{i \in \Gamma^-} \varphi ij = \sum_{i \in \Gamma^+} \varphi ji$$

Cette relation exprime simplement que la somme des flux entrant en un sommet est égale à la somme des flux sortant de ce sommet.

- Un flot ϕ est dit réalisable (ou admissible ou compatible) si et seulement si :

  – il respecte les contraintes de capacité sur chaque arc

  – Il respecte la loi de conservation en chaque sommet

- Un flot ϕ est dit complet si et seulement si tout chemin de s à t comporte au moins un arc saturé. [21]

II.4 Opérations sur les flots

Soient $\varphi, \varphi 1, \varphi 2$ des .flots sur G et $k \in R$.

Lemme

- $k.\varphi$ est un flot sur G.
- $\varphi 1 + \varphi 2$ est un flot sur G.
- $\varphi 1 - \varphi 2$ est un flot sur G.

II.5 Chaînes améliorantes pour un flot

II.5.1 Définition

Soit un flot (ϕ_{ij}) sur le graphe orienté G= (X, U), on appellera chaîne toute suite de sommets C= (S=x1, x2..., xp−1, x p=t) telle que pour tout j ϵ {1, , p − 1} (xj ,xj+1) appartient à U (arête directe) ou (xj+1,xj) appartient à U (arête indirecte).

La chaîne (S=x1, x2..., xp−1, x p=t) sera dite améliorante lorsque

- pour toute arête directe (i, j) reliant deux sommets consécutifs de la chaîne $\phi_{ij} < c_{ij}$

- pour toute arête indirecte (i, j) reliant deux sommets consécutifs de la chaîne $\phi_{ij} > 0$.

Théorème 1. Si un flot est maximal il n'admet pas de chaîne améliorante.

Théorème 2. Si un flot ne possède pas de chaîne améliorante il est maximal. [21]

II.6 Coupe minimum

II.6.1 Définition de la coupe

Une coupe est une séparation des sommets en deux sous-ensembles A et B , tels que sϵA , tϵB et B= X-A.

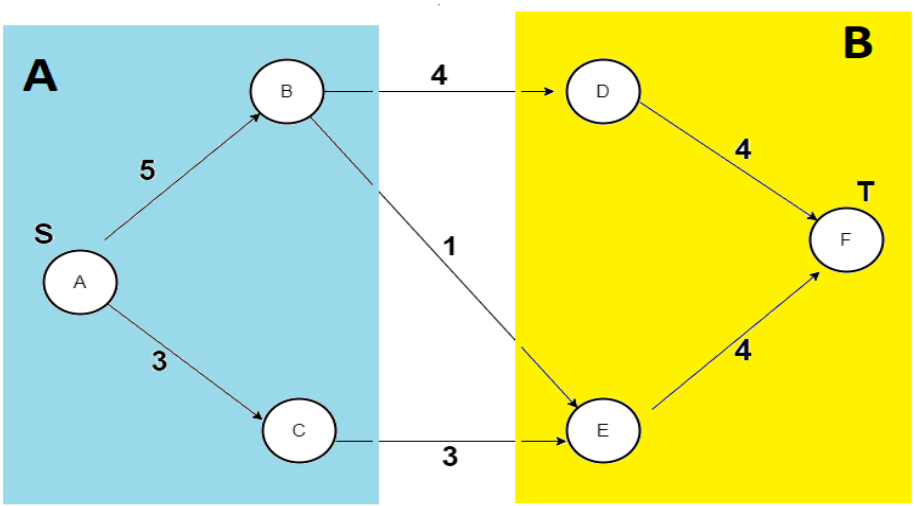


FIGURE II -2 : Exemple d'un coupe minimum

II.6.2 Capacité d'une coupe

La capacité d'une coupe (A, B) est égale à la somme des capacités des arcs allant de A vers B.

$$C(A,B) = \sum_{i \in A, j \in B} C\,(i,j)$$

La capacité de la coupe (A, B) sur la figure …. C (A, B) =4+1+3=8

II.6.3 Flot net

Etant donnée une coupe séparant les sommets en deux ensembles A et B. La somme des valeurs du flot sur les arcs allant d'A vers B moins la somme des valeurs du flot sur les arcs allant de B vers A est appelée flot net traversant la coupe. Le flot net traversant une coupe ne dépend pas de la coupe. [22]

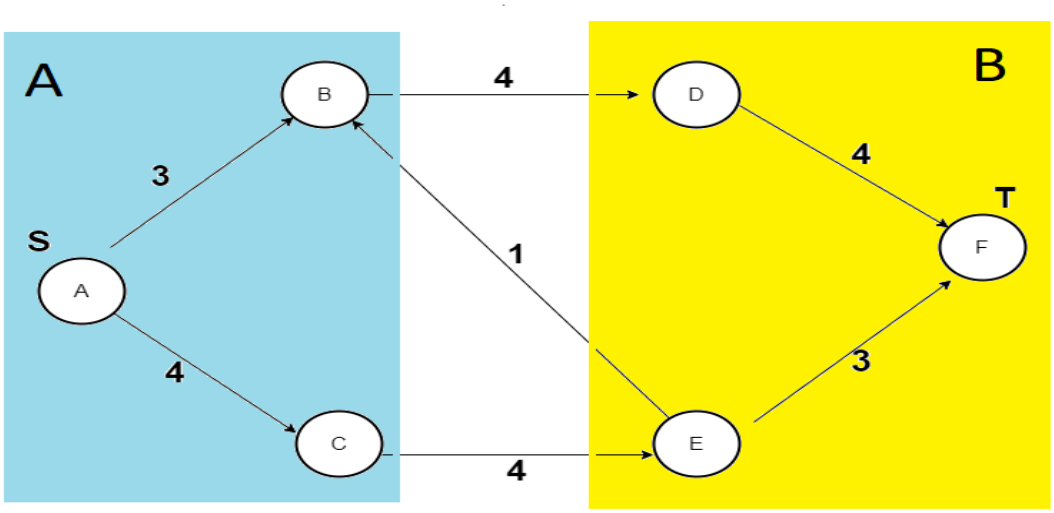


FIGURE II -3 : Exemple du flot net

II.6.4 Coupe minimale

Une coupe (A, B) est dite minimale pour une flot F si tous les arcs allant d'A vers B sont saturés et aucun arc de B vers A n'est saturé.

Pour tout flot F de valeur V(F), et pour toute coupe (A, B) de capacité C (A, B) on a :

$$V(F) \leq C\ (A, B)$$

II.6.5 Théorème coupe minimale et flot maximale

S'il existe une coupe minimale pour un flot F alors ce flot est maximum

II.7 Graphe résiduel

II.7.1 Définition

On a un réseau de transport G et un flot compatible f. Le graphe résiduel associé, R_f est le graphe qui modélise, sur chaque arc, l'écarts entre le flot et la capacité de l'arc. Il est défini de la façon suivante :

• Ses sommets sont les sommets de G.

• Il y a un arc entre u et v quand $0 < f\ (u, v) < C\ (u, v)$ ou quand $f\ (v, u) > 0$:

– Si $0 < f\ (u, v) < C\ (u, v)$, alors on met un arc $u \rightarrow v$, qui indique que l'on peut encore augmenter le flot le long de cet arc (arc augmentant).

– Si $f\ (v, u) > 0$, on met un arc $u \rightarrow v$ qui indique que l'on peut diminuer le flot entre v et u (arc diminuant).

On dit aussi qu'un arc tel que $f\ (u, v) = C\ (u, v)$ est saturé. Le graphe résiduel contient donc le graphe où on enlève les arcs saturés, et où on ajoute des arcs allant dans l'autre sens des flots strictement positifs.[22]

II.7.2 Graphe résiduel et chemin améliorant

Un chemin améliorant pour un flot f est juste un chemin de s à t dans Rf . Sa variation de flot est la valeur minimale, le long du chemin, des quantités $C(u, v) - f(u, v)$ pour les arcs augmentant et f(v, u) pour les arcs diminuants (si un arc est à la fois les deux, on ajoute les quantités).[22]

II.8 Parcours d'un graphe

II.8.1 Définition

On appelle parcours d'un graphe, tout procédé déterministe qui permet de choisir, à partir des sommets visités, le sommet suivant à visiter. Le problème de parcours consiste à déterminer un ordre sur les visites des sommets et il y a deux principales stratégies d'exploration :

– Le parcours en largeur consiste à explorer les sommets du graphe niveau par niveau, à partir d'un sommet donné.

– Le parcours en profondeur consiste, à partir d'un sommet donné, à suivre un chemin le plus loin possible, puis à faire des retours en arrière pour reprendre tous les chemins ignorés précédemment

Racine : Le sommet de départ, fixé à l'avance, dont on souhaite visiter tous les descendants est appelé racine de l'exploration. [23]

II.8.2 Propriété

Un parcours de racine r est une suite L de sommets telle que :

- r est le premier sommet de L.
- Chaque sommet apparaît une fois et une seule dans L.
- Tout sommet sauf la racine est adjacent à un sommet placé avant lui dans la liste

II.8.3 Parcours en profondeur DFS (Depth-First Search)

Principe :

Le principe du parcours en profondeur d'un graphe (orienté ou non) est celui du parcours d'un labyrinthe : on va de sommet en sommet en marquant au fur et à mesure les sommets visités. La visite se poursuit le plus loin possible tant qu'il reste des sommets accessibles non encore marqués. Quand on atteint un sommet v dont tous les voisins ont été déjà marqués alors on revient au sommet précédant v dans la visite. Autrement dit on parcoure tous les sommets d'un graphe à partir d'un sommet v donné, à suivre un chemin le plus loin possible, puis à faire des retours en arrière pour reprendre tous les chemins ignorés précédemment.

Remarque :

Dans un parcours en profondeur, on applique la règle "Dernier marqué, premier exploré".

i, e : on explore les sommets dans l'ordre inverse de celui utilisé pour les marquer. [23]

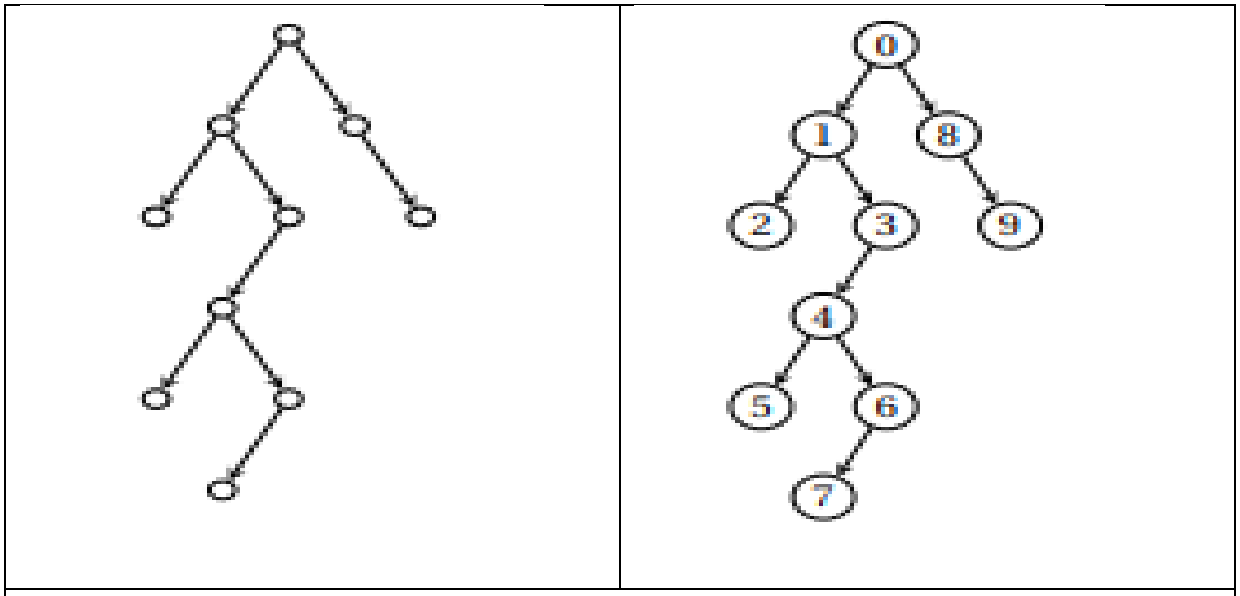


FIGURE II -4 : La méthode DFS

II.8.4 Parcours en largeur BFS (Breadth-First Search)

Principe : Le principe est de visiter tous les voisins d'un sommet avant de visiter le sommet suivant qui sera le premier voisin à avoir été visité auparavant. Autrement dit on parcoure tous les sommets d'un graphe à partir d'un sommet de départ v (racine), on commence par visiter tous les successeurs de v avant de visiter les autres. Le parcours en largeur est obtenu en gérant la liste d'attente au coloriage comme une file d'attente. Autrement dit, on enlève à chaque fois le plus vieux sommet gris dans la file d'attente, et on introduit tous les successeurs blancs de ce sommet dans la file, en les coloriant en gris. [23]

Exemple :

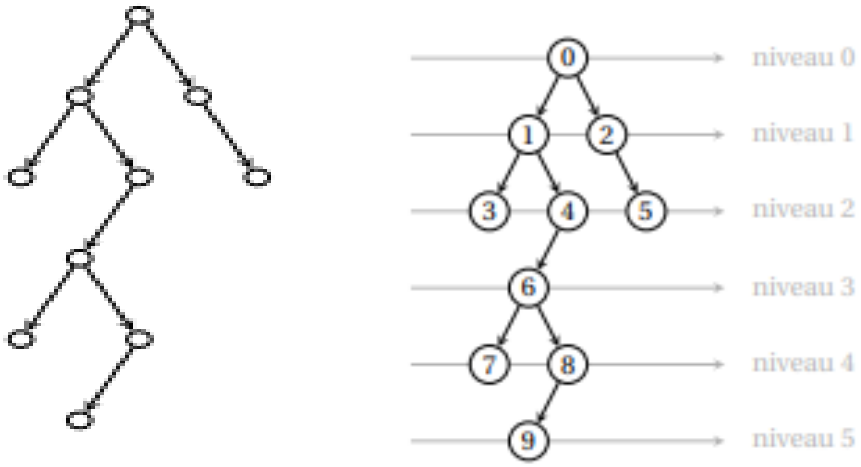


FIGURE II -5 : La méthode DFS

II.9 L'algorithme de Ford-Fulkerson

II.9.1 Définition

L'algorithme de Ford-Fulkerson est un algorithme utilisé pour résoudre le problème de flot maximal dans un graphe. Il utilise une méthode de recherche de chemin augmentant pour trouver le flot maximal dans le graphe.

L'algorithme de Ford-Fulkerson peut être implémenté de plusieurs façons.

II.9.2 Les étapes de l'algorithme de Ford-Fulkerson

1. Initialiser le flot maximal à 0.
2. Trouver un chemin augmentant, c'est-à-dire un chemin du nœud source au nœud puits dans le graphe qui a une capacité résiduelle positive pour chaque arête du chemin.
3. Trouver la capacité résiduelle minimale du chemin augmentant.
4. Augmenter le flot maximal en ajoutant la capacité résiduelle minimale au flot maximal.
5. Mettre à jour les capacités résiduelles des arêtes du chemin augmentant.
6. Répéter les étapes 2 à 5 jusqu'à ce qu'il n'y ait plus de chemin augmentant dans le graphe.[24]

II.9.3 Algorithme de base de Ford-Fulkerson

FORD-FULKERSON(G, s, t)

Pour chaque arc $(u, v) \in A[G]$

Faire $f[u, v] \leftarrow 0$

Faire $f[u, v] \leftarrow 0$

Tant que il existe un chemin p de s à t dans le réseau résiduel G_f

Faire $c_f(p) \leftarrow \min \{ c_f(u, v) : (u, v) \text{ is in } p \}$

Pour chaque arc (u, v) de p

Faire $f[u, v] \leftarrow f[u, v] + c_f(p)$

$f[v, u] \leftarrow -f[u, v]$

II.9.4 L'exécution de l'algorithme de base de Ford-Fulkerson

Exemple :

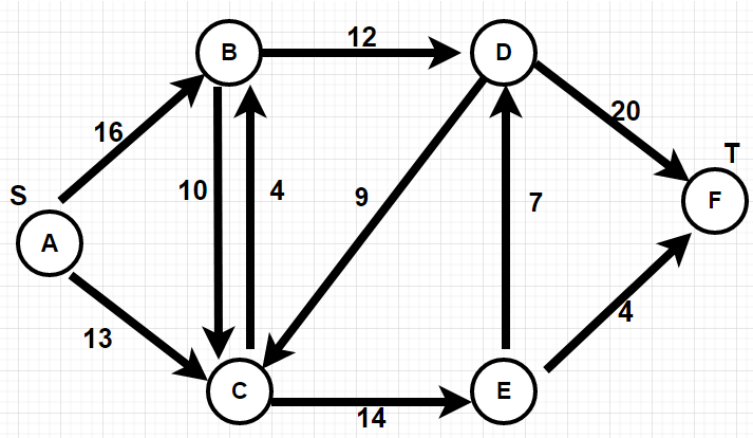


FIGURE II-**6 :** Exemple pour l'exécution de l'algorithme [25]

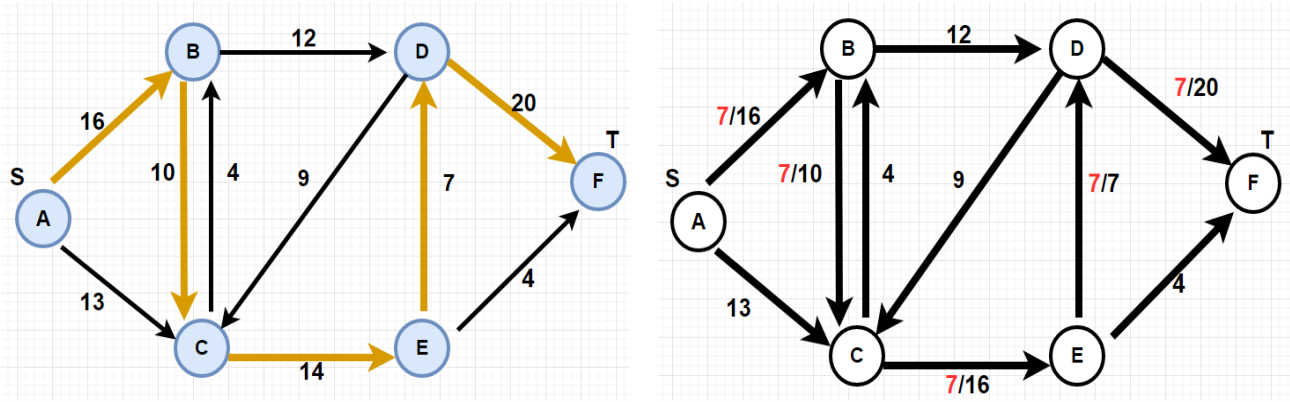


FIGURE II-7 **:** L'exécution de l'algorithme itération n° :1

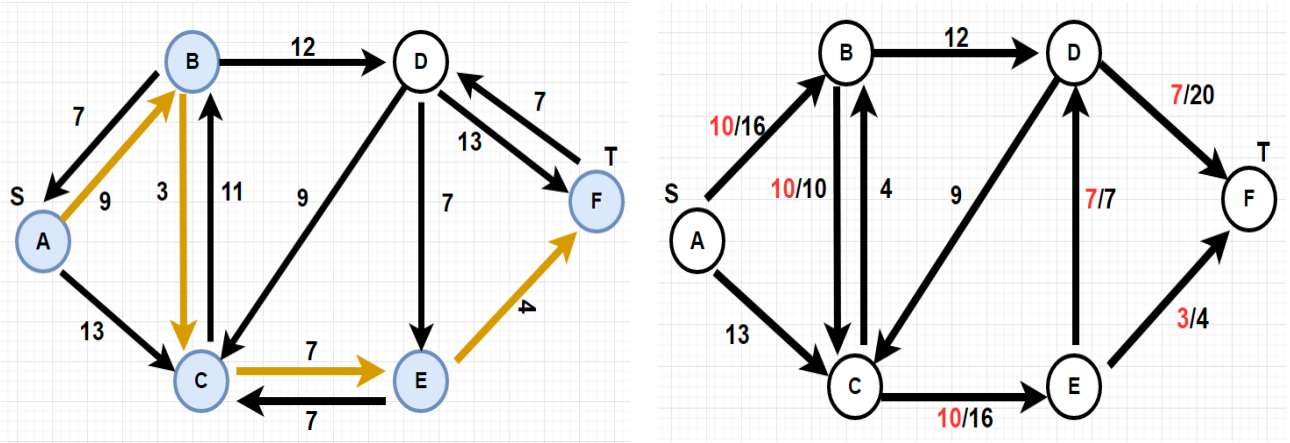


FIGURE II-8 : L'exécution de l'algorithme itération n° :2

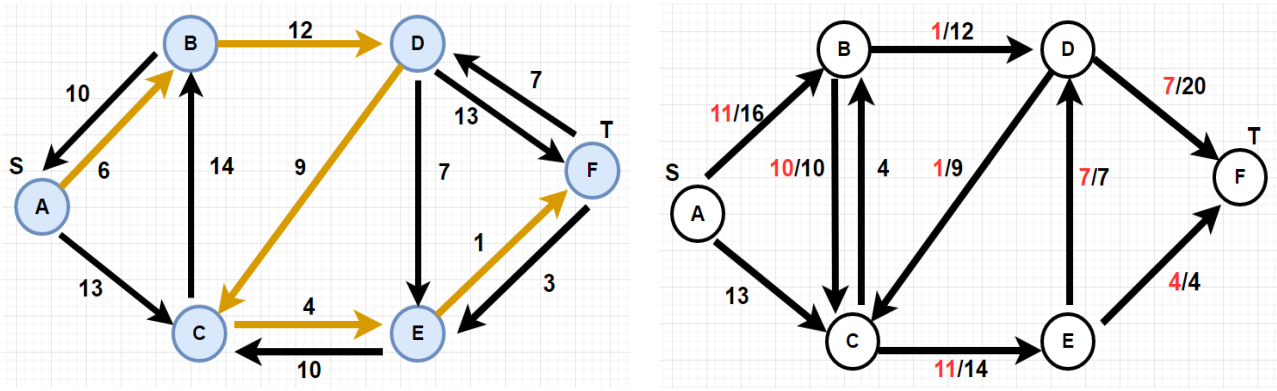


FIGURE II-**9 :** L'exécution de l'algorithme itération n° :3

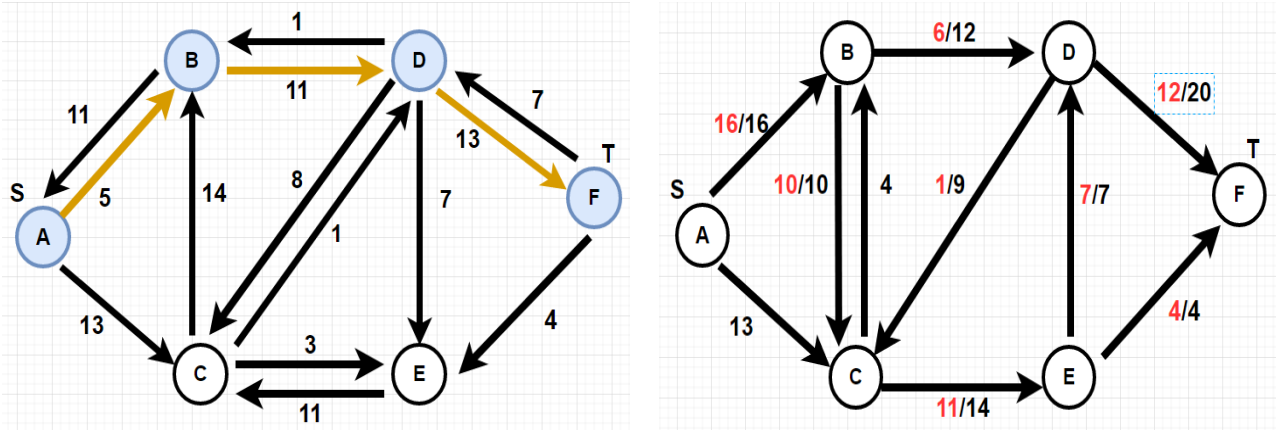


FIGURE II-10 : L'exécution de l'algorithme itération n° :4

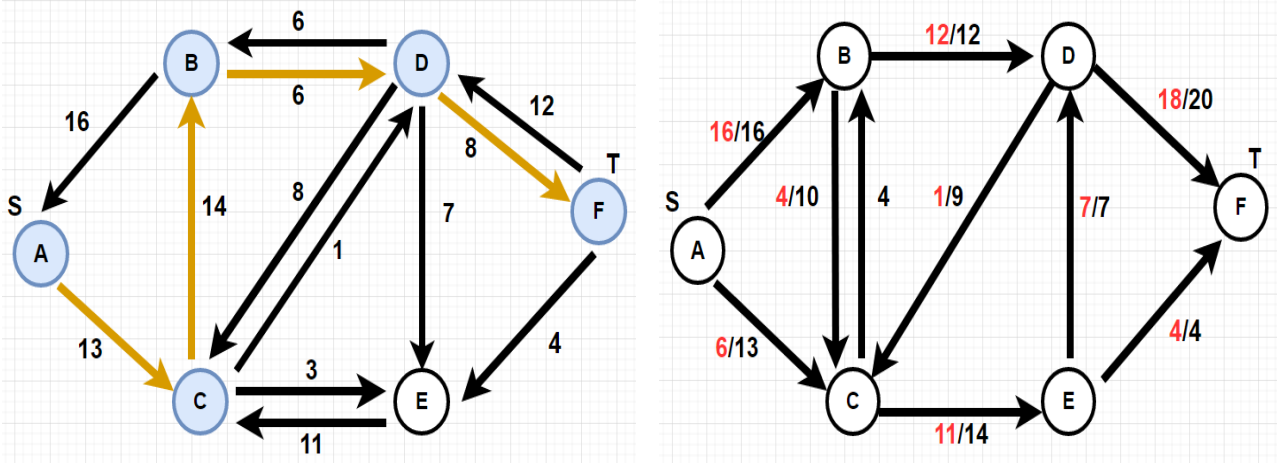


FIGURE II-11 : L'exécution de l'algorithme itération n° :5

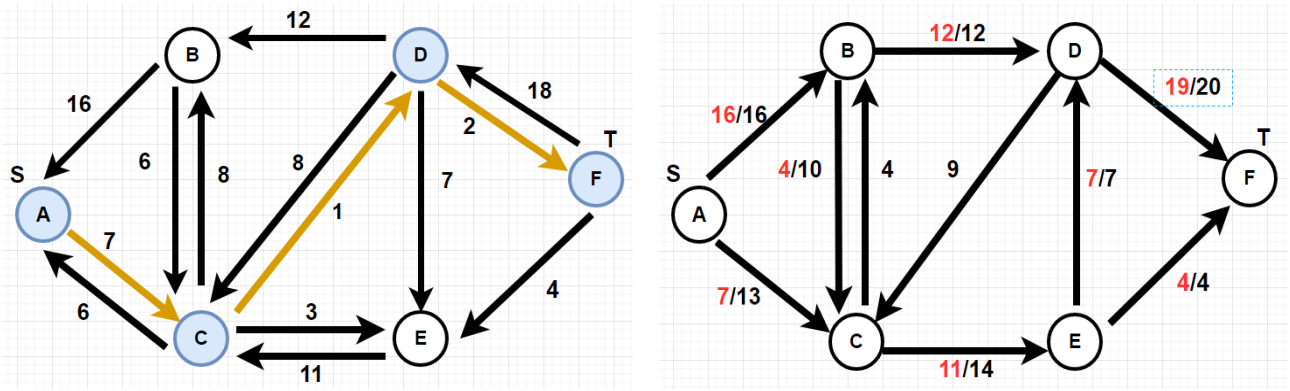
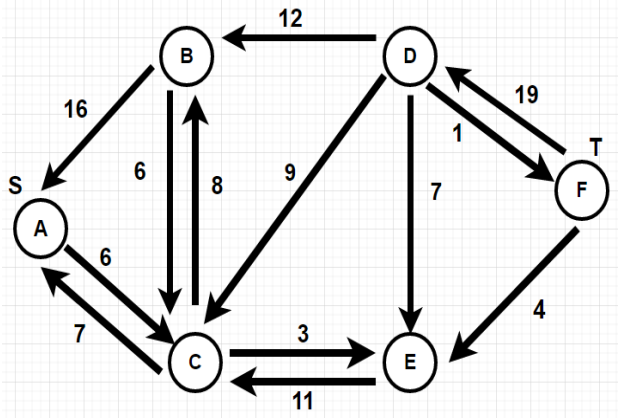


FIGURE II-12 : L'exécution de l'algorithme itération n° :6



Le réseau résiduel ne contient aucun chemin améliorant et le flot f montré en itération n° :6 est donc un flot maximum

FIGURE II-13 : L'exécution de l'algorithme itération n° :7

II.10 Conclusion

Dans ce chapitre nous avons fournis une introduction approfondie au problème de flot maximal et présenté l'algorithme de Ford-Fulkerson comme une méthode classique pour le résoudre. Nous avons donné également un aperçu des variantes et des extensions du problème, soulignant l'importance et la complexité de ce domaine de recherche en théorie des graphes et en optimisation.

Chapitre III
Implémentation

III.1 Introduction

Dans la première partie de ce chapitre, nous commençons tout d'abord par une présentation du langage de programmation choisi, et une courte présentation sur LaTeX.

Aussi, et pour des fins éducatives et pédagogiques, nous allons concevoir et implémenter l'algorithme par un langage simple à savoir Java et l'environnement éclipse.

D'autre part, l'éditeur LaTeX permet d'obtenir des documents de haute qualité sans beaucoup de connaissances en typographie ou en mise en page.

La deuxième partie de chapitre est consacrée à la présentation de la mise en œuvre de notre application.

Ensuite nous mentionnons les détails de notre application et nous terminons ce chapitre par une synthèse de nos résultats obtenus.

III.2 Outils de développement

III.2.1 Présentation du langage JAVA

III.2.1.1 Définition

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java. Le langage Java reprend en grande partie la syntaxe du langage C++, très utilisée par les informaticiens. Néanmoins, Java a été épuré des concepts les plus compliqués du C++ et à la fois les plus déroutants, tels que les pointeurs et références, ou l'héritage multiple contourné par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.).[26]

III.2.1.2 Particularité du langage java

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et Framework associés visent à guider, sinon garantir, cette portabilité des applications développées en Java Cette particularité est obtenu par :

- Des bibliothèques standard fournies pour pouvoir accéder à certains éléments de la machine hôte (le graphisme, le multithreading, la programmation réseau…) exactement de la même manière sur toutes les architectures.

- des compilateurs Java qui compilent le code source « à moitié » afin d'obtenir un bytecode (plus précisément le bytecode Java, un langage de type assembleur, proche de la machine virtuelle et spécifique à la plate-forme Java), faisant apparaître autant de fichiers avec l'extension .class que de classes développées dans le code source Toutefois, contrairement aux langages compilés traditionnels, pour lesquels le compilateur crée un fichier binaire directement exécutable par un processeur donné (c'est-à-dire un fichier binaire contenant des instructions spécifiques à un processeur).[27]

III.2.2 Swing

Swing est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package Java Fondation Classes (JFC), inclus dans J2SE. Swing constitue l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures. Swing offre la possibilité de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent, au prix de performances moindres qu'en utilisant Abstract Window Toolkit (AWT). Il utilise le principe Modèle-Vue-Contrôleur (MVC, les composants Swing jouent en fait le rôle du contrôleur au sens du MVC) et dispose de plusieurs choix d'apparence (de vue) pour chacun des composants standards.[28]

III.2.3 Éclipse

L'environnement d'éclipse fait référence à l'IDE (Integrated Development Environment) appelé Eclipse. Eclipse est une plateforme de développement open-source largement utilisée pour la programmation dans divers langages tels que Java, C++, Python, et bien d'autres.

L'IDE Eclipse fournit un ensemble d'outils et de fonctionnalités pour faciliter le développement logiciel. Il offre un éditeur de code avec des fonctionnalités d'auto complétion, de coloration syntaxique et de mise en forme du code. Il propose également des fonctionnalités de débogage avancées, telles que des points d'arrêt, l'inspection des variables et l'exécution pas à pas du code.

Eclipse est apprécié pour sa polyvalence, sa communauté active et son extensibilité. Il est utilisé par de nombreux développeurs professionnels et est également largement adopté dans le domaine de l'enseignement. L'environnement d'Eclipse facilite la création, la modification et le débogage de programmes, ce qui en fait un choix populaire pour de nombreux développeurs.[29]

III.2.4 Comment créer un fichier en java

Le langage Java offre de nombreuses possibilités dont celle d'écrire des données dans des fichiers. Il est possible d'écrire aussi bien du texte que des données binaires. Dans toutes les versions de Java, la classe PrintWriter peut être utilisée pour écrire du texte dans un fichier. Il faut spécifier lors de la création de l'objet le fichier et l'encodage utilisés pour le texte. La méthode print () écrit les données à la suite de ce qui a déjà été écrit. La méthode println () effectue la même chose mais ajoute en plus un retour à la ligne après avoir écrit le texte. Si le fichier existe déjà, son contenu sera écrasé par le texte inséré. [30]

```
String fileName = "my-file.txt";
String encoding = "UTF-8";
try{
PrintWriter writer = new PrintWriter(fileName, encoding);
writer.println("The first line");
writer.println("The second line");
writer.close();
}
```

III.3 LaTeX

III.3.1 Historique

Donald KNUTH crée simultanément METAFONT et TEX qui sont à la fois des programmes et des langages. Le premier s'occupe de la création des fontes à l'aide de commandes de description, le second s'occupe de la composition. En 1978 sort une première version de TEX mais il continuera à travailler sur TEX jusqu'en 1989.

En 1982, Leslie LAMPORT crée LATEX, une surcouche de TEX qui rend le travail de rédaction beaucoup plus simple La Philosophie de LATEX est : séparer la forme et le fond afin de mieux se concentrer sur le contenu. Les apports de LATEX concernent la gestion des fontes, de la couleur et du graphisme.

III.3.2 Définition

LaTeX est un système de composition de documents utilisé pour produire des documents de haute qualité, tels que des articles scientifiques, des rapports techniques, des livres, des thèses, etc. Il se distingue par sa capacité à créer des documents bien structurés et esthétiquement agréables, avec une mise en page professionnelle.

LaTeX se concentre sur la structure logique du document. Vous écrivez votre contenu dans un fichier texte brut avec des commandes spéciales pour indiquer la structure, le formatage et les éléments du document tels que les titres, les sections, les tableaux, les équations mathématiques, les références bibliographiques, etc. [31]

III.3.3 Installation de LaTeX

Pour faire du LATEX, il faut trois choses :

- **une distribution LATEX** : qui comporte l'ensemble des compilateurs, visualiseurs , polices, etc... nécessaire au fonctionnement de LATEX.

Installer MikTeX

http://miktex.org/download

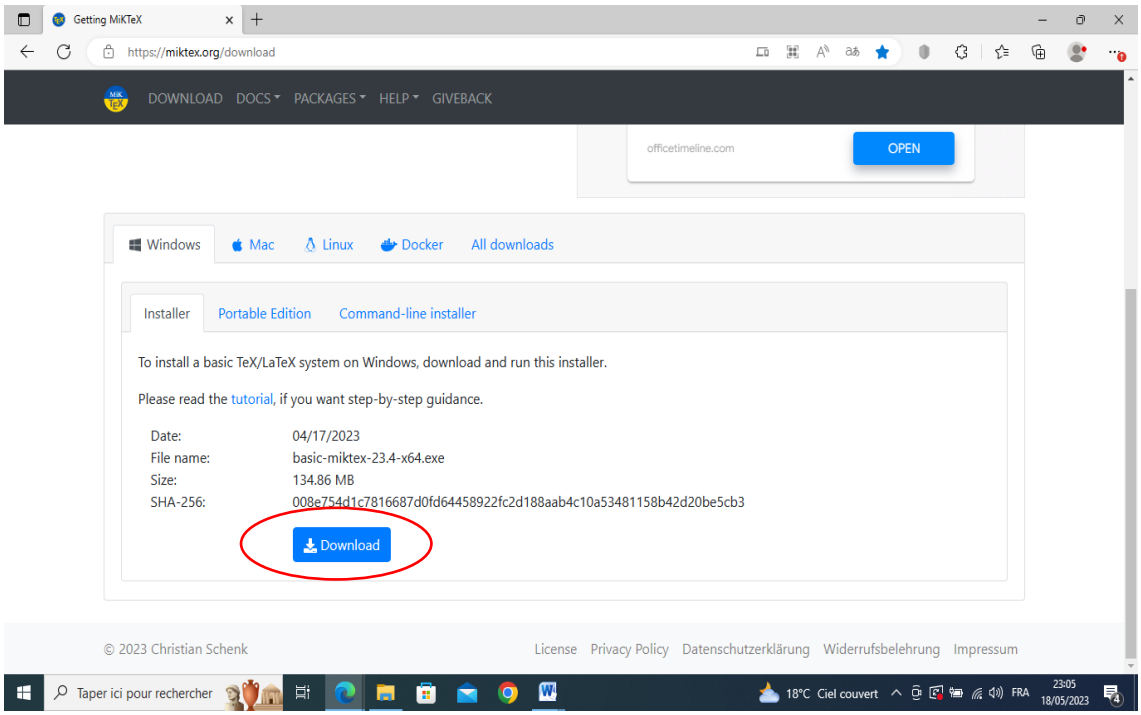


Figure III-1 : Le site officiel de miktex

- **2. un éditeur** : contenant plein de fonctions utiles (mais qui sous Linux est livré avec le programme LATEX).

Installer Texmaker :

http://www.xm1math.net/texmaker/index_fr.html

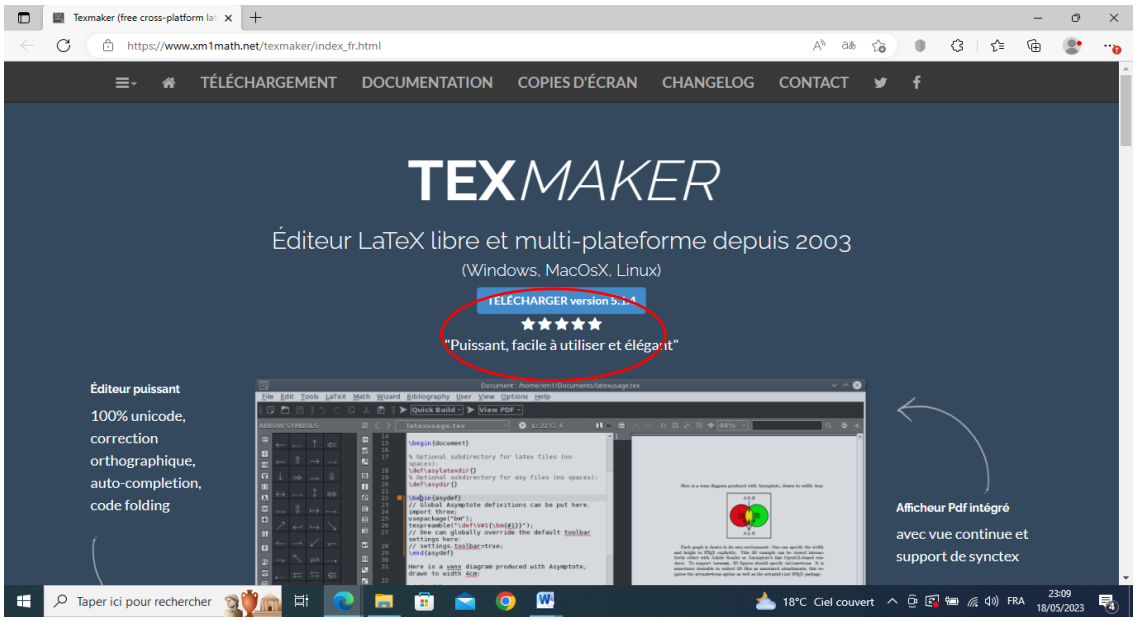


Figure III-2 : Le site officiel de texmaker

- **un visualiseur** :Postscript (type ghostview) et/ou PDF (Adobe Acrobat Reader). [2]

III.3.4 Les fichiers LATEX

LATEX est un langage de programmation, qui génère plusieurs types de fichiers. On trouve des fichiers:

.tex : Ce sont les fichiers contenant toutes les commandes que vous allez taper, i.e les fichiers sources.

.dvi : C'est le résultat de la compilation standard de vos commandes. On peut visualiser ces fichiers à l'aide du logiciel.

xdvi .ps ou .pdf : Il s'agit des fichiers destinés à la publication, après conversion depuis le .dvi .

.bib et .bbl : Ces fichiers servent à la gestion de la bibliographie.

.aux, .toc, .idx : Ces fichiers sont utilisés par LATEX pour gérer les références dans votre document. [32]

III.3.5 Différents types de documents

Il existe plusieurs classes, qui sont spécifiées par la commande :

\document class[options]{classe}.

Les différentes classes disponibles par défaut sont :

- **article** : pour des articles destinés à la publication et ne contenant que quelques pages.
- **report** : pour des documents un peu plus longs contenant plusieurs chapitres, comme des mémoires de thèse ;
- **book** : pour de véritables livres, de plusieurs centaines de pages.
- **slides** : pour faire des présentations sur transparents.

Il existe aussi d'autres classes moins répandues :

- **beamer** : pour faire des présentations utilisant la magnifique extension beamer ;
- **lettre** : pour faire des lettres.
- **memoir** : pour écrire des mémoires, par exemple de fin d'étude.

Le choix de la classe va déterminer un certain nombre de paramètres par défaut, comme par exemple les marges, mais aussi fournir des instructions supplémentaires spécifiques. [32]

III.3.6 Exemple

Code source :

```
\documentclass[12pt,a4paper,twoside]{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\begin{document}
\section{Premier essai}
Bonjour tout le monde. Je m'essaie au \LaTeX.\\
Pour l'instant, tout va bien !

\end{document}.
```

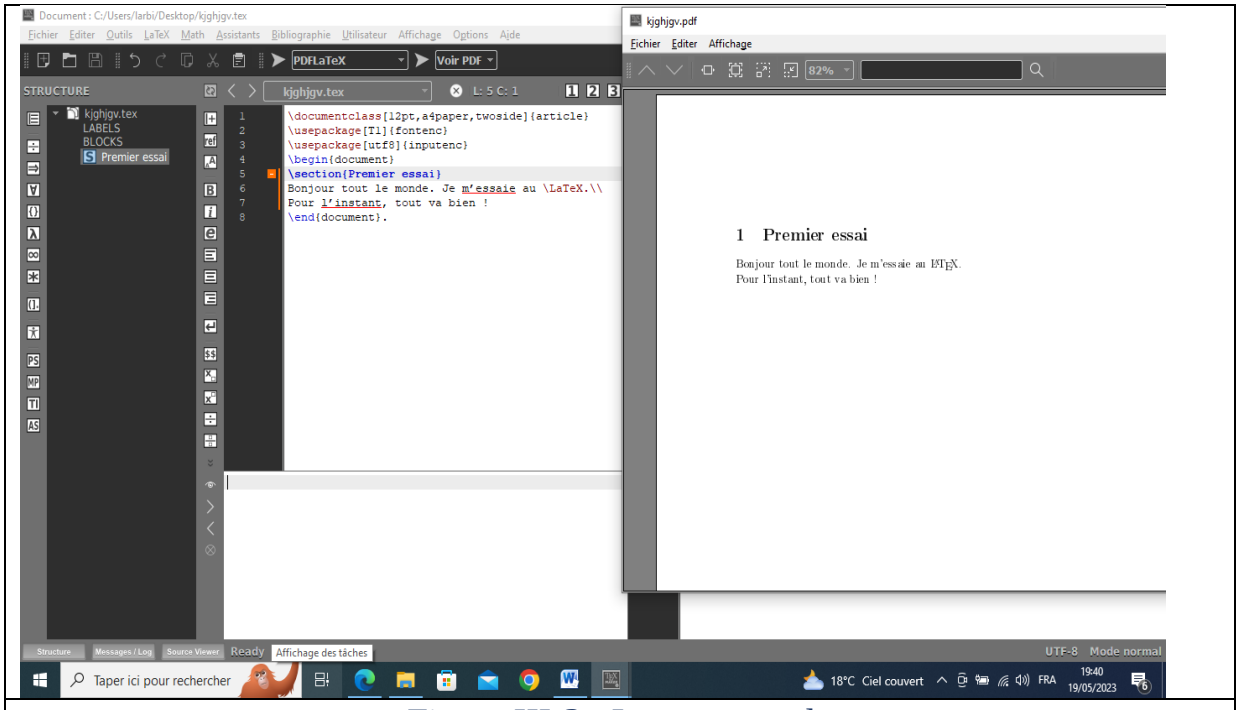


Figure III-3 : Latex exemple

III.3.1 Les environnements les plus utilisés

Différents environnements permettent de structurer un document, nous allons détailler les principaux :

- Les listes
- Les tableaux
- Les figures
- La bibliographie

Un environnement se commence toujours par begin{environnement} et se fini par end{environnement}. [33]

Les listes

Les listes se déclarent dans un environnement avec un begin et un end dont l'intitulé dépend du type de liste que vous voulez générer : enumerate, itemize, description. [34]

Exemple:

```
\begin{enumerate}
    \item item
    \item item
    \item item
\end{enumerate}
\begin{itemize}
    \item item
    \item item
    \item item
\end{itemize}
\begin{description}
```

```
\item[cas 1] item
\item[cas 2] item
\item[cas 3] item
\end{description}
```

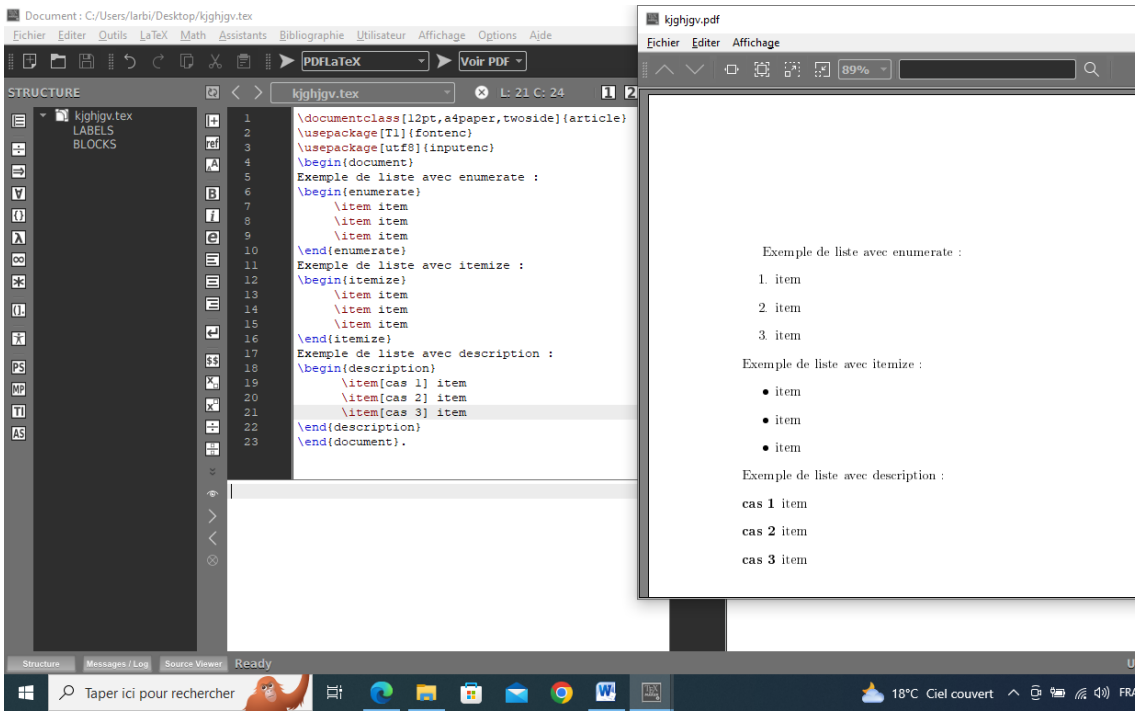


Figure III-4 :Liste latex exemple

III.3.1.1 Les tableaux

Exemple :

```
\begin{table}
\begin{tabular}{|l|cc|}
 N° inscription  & NOM & PRENOM \\
 \hline
 1451001 & nom1 & prenom1 \\
 1451002& nom2 & prenom2 \\
 1451003 & nom3 & prenom3\\
```

\end{tabular} \

caption{Ceci est un tableau présentant listes des étudiants.}\label{tab_serveur} \end{table}

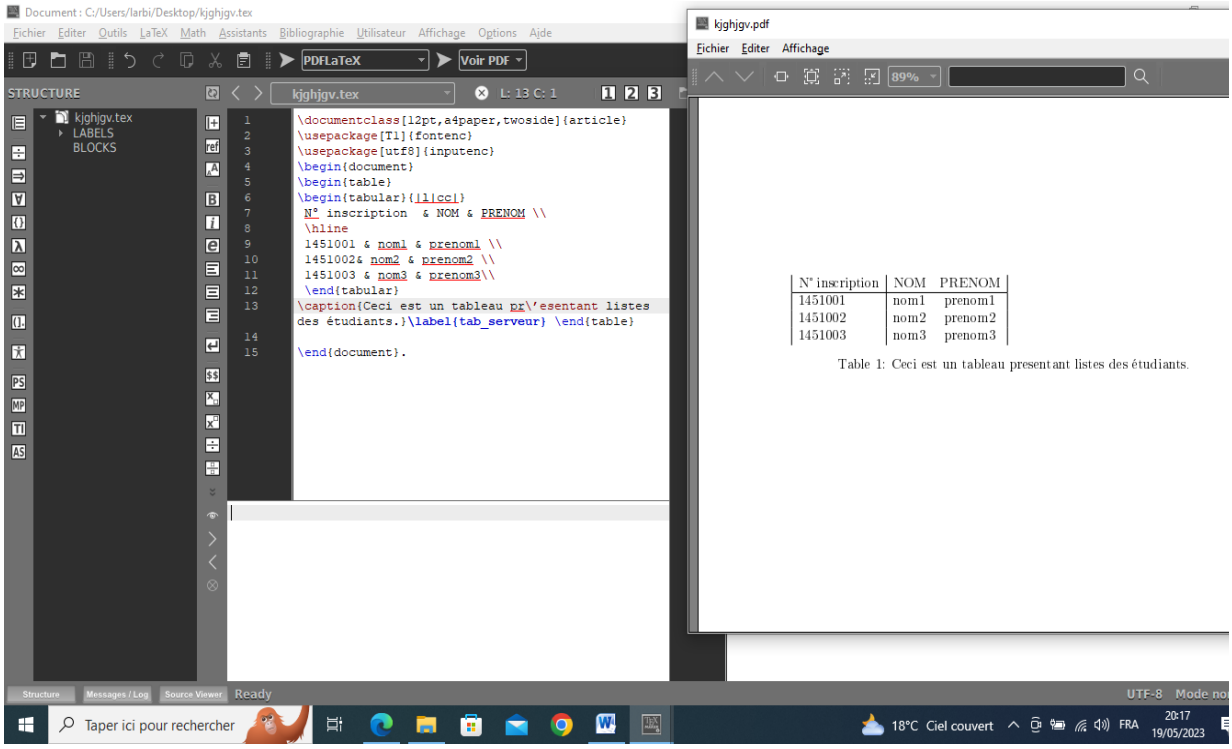


FIGURE III-5 :Table latex exemple

III.4 Présentation de l'application

III.4.1 Interface principale

Pour saisir les données du graphe, ces données sont :

- Le nombre des nœuds
- Préciser les arcs et leurs capacités.

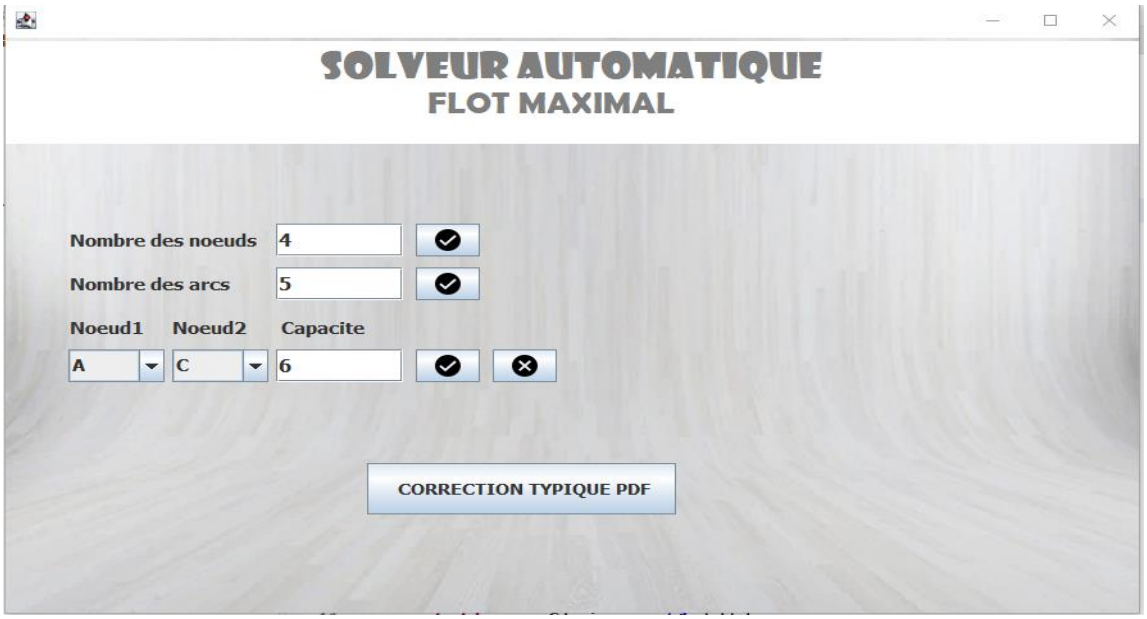
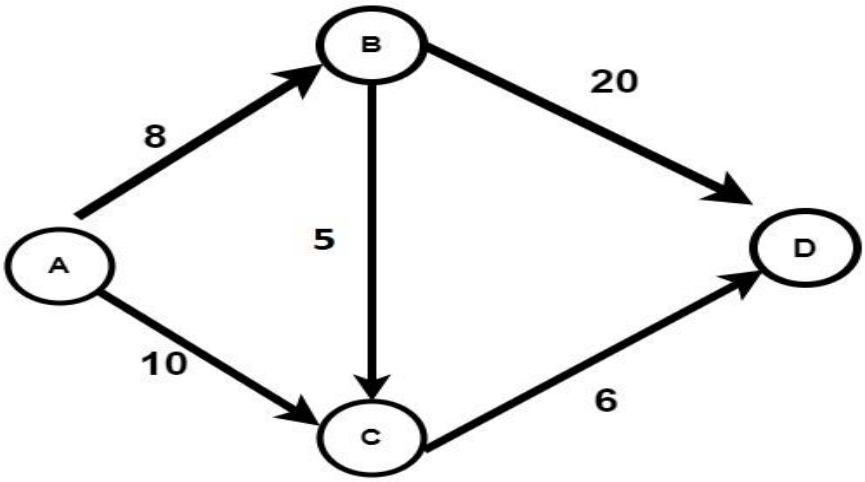


FIGURE III-6 :Interface principale

III.4.2 Résultat

Nous allons expliquer, à l'aide de l'exemple suivant :



Graphe G=(X,U.c)

Code latex :

```
\documentclass[10pt,a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
 \usepackage{geometry}
```

```
 \geometry{hmargin=1 cm,vmargin=1.5cm}
\begin{document}
\begin{center}
\textbf{CORRECTION TYPIQUE}
\end{center}
\textbf{solution de l'exercice n°:1 (06 points)} \newline

\begin{tabular}{|c|c|c|c|c|c|}
\hline
U & $C_u$ & $\varphi_u$  \\
\hline
$ \overrightarrow{\rm(A,B)}$&8&0&+5&+8&8&8\\
 \hline
$ \overrightarrow{\rm(A,C)}$&10&0&0&0&+1&+6\\
 \hline
$ \overrightarrow{\rm(B,C)}$&5&0&+5&5&5&-0\\
 \hline
$ \overrightarrow{\rm(B,D)}$&20&0&0&+3&3&+8\\
 \hline
$ \overrightarrow{\rm(C,D)}$&6&0&+5&5&+6&6\\
 \hline
\end{tabular}
\newline
\newline  marq1: A,$ \overrightarrow{\rm B,}$$
\overrightarrow{\rm C,}$$ \overrightarrow{\rm D,}$\newline
$\epsilon$ =Min$\{$8-0,5-0,6-0,$\}$=5
\newline  marq2: A,$ \overrightarrow{\rm B,}$$
\overrightarrow{\rm D,}$\newline   $\epsilon$ =Min$\{$8-5,20-
0,$\}$=3
\newline  marq3: A,$ \overrightarrow{\rm C,}$$
\overrightarrow{\rm D,}$\newline   $\epsilon$ =Min$\{$10-0,6-
5,$\}$=1
\newline  marq4: A,$ \overrightarrow{\rm C,}$$
\overleftarrow{\rm B,}$$ \overrightarrow{\rm D,}$\newline
$\epsilon$ =Min$\{$10-1,0--5,20-3,$\}$=5
\newline marq5: A$\implies$FIN, CM=$\{$A$\}$
\newline flot maximale=14
•

\end{document}
```

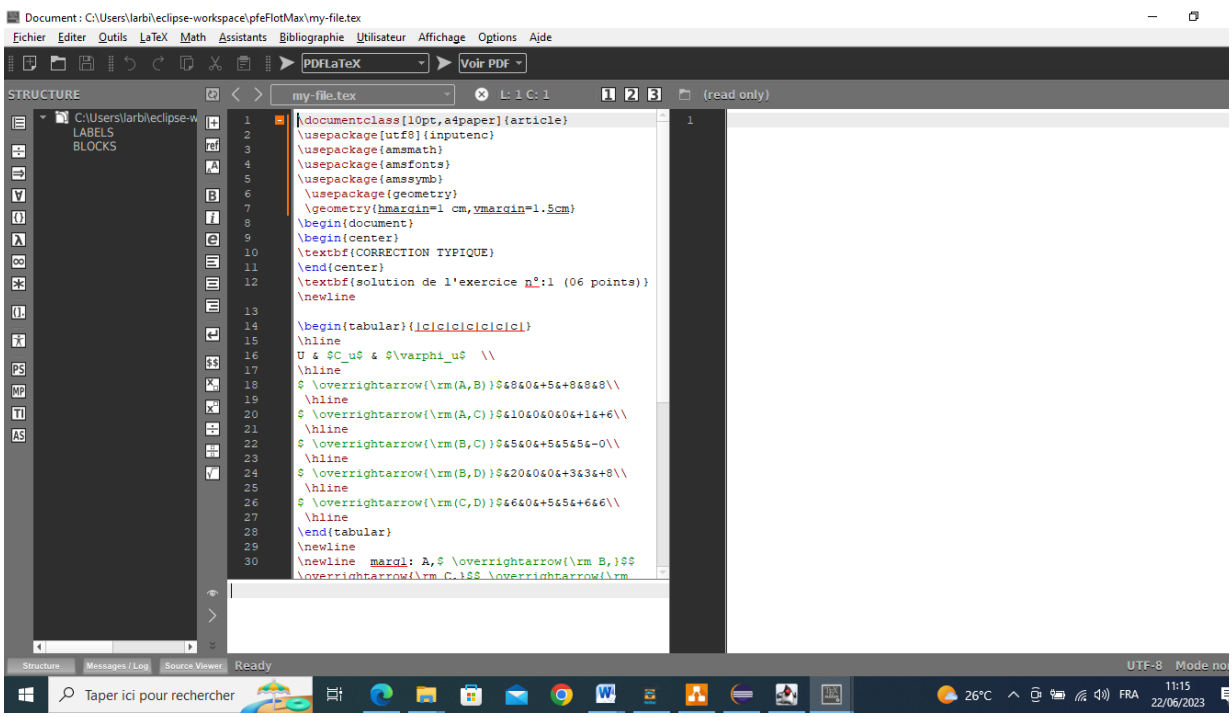


FIGURE III-7 : Code latex

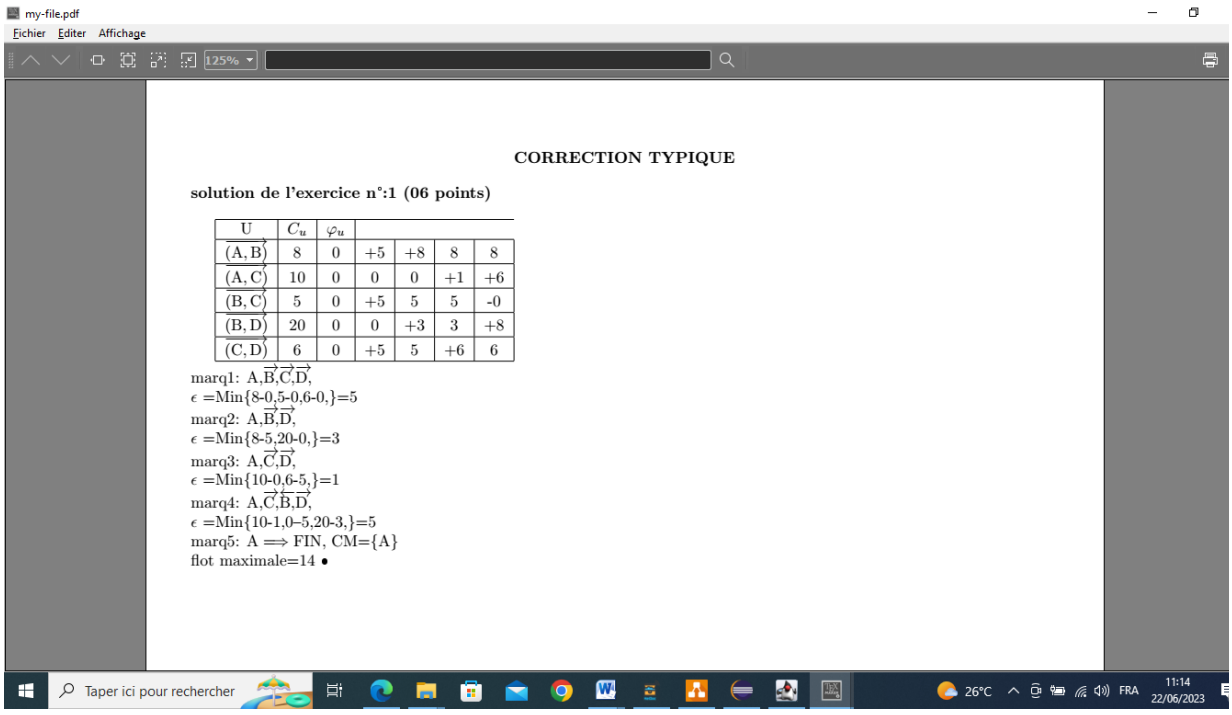


FIGURE III-8 : Résultat final (correction typique en pdf)

III.5 Conclusion

Latex est un système de composition de documents largement utilisé pour la création de documents techniques et scientifiques de haute qualité, il permet de produire des documents bien structurés et formatés. Latex se distingue par sa facilité à gérer des équations mathématiques complexes et des tableaux de contenu. Dans cette partie nous avons créé une application java pour résoudre le problème de flot maximal et afficher la solution sous forme PDF bien fait grâce à latex.

Conclusion Générale

Conclusion générale

Nous pensons que l'objectif fixé au début est dans une certaine mesure atteint. Nous avons réalisé une application qui reçoit en entrée les paramètres d'un graphe orienté, c'est-à dire les nœuds, les arcs et les leurs poids respectifs et qui produit en sortie un fichier PDF contenant la solution typique du problème du flot maximal.

Ceci permet de faciliter la tâche de l'enseignant correcteur ou pour un étudiant de vérifier sa solution initiale.

Donc l'intérêt de notre travail est clair, c'est de réduire considérablement la correction d'un exercice sous une forme professionnelle et lisible produite par le langage LATEX sans taper une seule commande LATEX !

Les perspectives ouvertes suite à ce travail sont entre autres :

- La généralisation de l'application vers d'autres problèmes de la théorie des graphes,

- La généralisation vers d'autres domaines de résolution.

- Performer l'interface homme machine.

[1] Kheireddine yamina Quelques applications réelles du problème de flot .Master en : Informatique Décisionnel et Optimisation Université Mohamed Boudiaf M'sila 2019 /2020.

[2] Didier Müller, Introduction à la théorie des graphes, CAHIER NO 6, COMMISSION ROMANDE DE MATHÉMATIQUE, France, Décembre 2011.

[4] Kadri Saïd. Les fondements de la théorie des graphes. Chp1 : concepts de bases. Université Mohammed Boudiaf. Msila,2017 /2018.

[6] https://perso.liris.cnrs.fr/samba-ndojh.ndiaye/fichiers/App_Graphes.pdf7- Francette Bories- Longuet Jorge Ramirez Alfonsin. Graphes et combinatoire ellipses France 2015.

[8] Christian Roux. Initiation à la Théorie des graphes ellipses France 2011.

[9] Jean-Claude Fournier. Théorie des graphes et applications avec exercices et Problèmes. Hermes Science Publications Paris 2011.

[10] D.MULER, Introduction à la théorie des graphes, 2008

[12] Kheireddine yamina Quelques applications réelles du problème de flot .Master en : Informatique Décisionnel et Optimisation Université Mohamed Boudiaf M'sila 2019 /2020

[13] Bretto.A, A. Faisant, F. Hennecart. Éléments de théorie des graphes. 15/05/2012.

[15] Hadj Arab Hanen Problème du lot dans les réseaux de transport et Application sur un réseau hydraulique de la ville T.O

[17] Christine Solnon Théorie des graphes et optimisation dans les graphes

[18] Johanne Cohen LRI-CNRS, Université Paris-Sud, Université Paris-Saclay, France 10/09/2018.

[21] Marie-Christine Costa TRANSFERT, RESEAUX, GRAPHES Flots et coupes ENSTA Paris-Tech 2014 78 pages

[22] Mr ABDELMALEK Hichem , Mr BELGACEM Feiçal Connexité et Flots dans un réseau Master en recherche opérationnelle Université Abderrahmane Mira Bejaia.2018-2019.

[30] Dekki mohamed amine et cherak mohamed amine Proposition et implémentation d'un solveur d'algorithmede graphes Master en Réseaux et télécommunication université tiaret 2020-2021.

[23] lélia blin THEORIE DES GRAPHES flots maximum université de paris

[34] M. Bailly-Bechet Introduction au LATEX Université Claude Bernard Lyon 1 France

Sites internet

[3] https://www.imo.universite-paris-s aclay.fr/~damien.thomine/fr/archives/Enseignement1617/GeomPCSO/GraphesLexique.pdf

[5] www.graphes.fr/Cours%20Graphes.pdf

[14] https://www.math.univ-toulouse.fr/~msablik/Cours/Graphes/GrapheNotes.pdf

[19] https://labsticc.univ-brest.fr/~lemarch/FR/Cours/optimGraphes.pdf

[11] https://www.math.univ-toulouse.fr/~msablik/Cours/Graphes/GrapheNotes.pdf.

[16] http://ressources.unit.eu/cours/EnsROtice/module_de_base_voo7/co/notionsdeconnexites.html#:~:text=D%C3%A9finition%20%3A%20Graphe%20r%C3%A9duit,x%2Cx')%E2%88%88E.

[20] http://ressources.unit.eu/cours/EnsROtice/module_de_base_voo7/co/represmatricielle.html

[24] https://nicolas.thiery.name/Enseignement/M1-ISD-AlgorithmiqueAvancee/Devoirs/3-ReseauxFlots/02-FordFulkersonCours.html25-

https:// fr.wikibooks.org/wiki/LaTeX/Les classes

[25] Chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.editions.polytechnique.fr/files/pdf/EXT_1641_8.pdf

[26] https://www.techno-science.net/glossaire-definition/Java-langage.html

[27]https://edu-html.ac-versailles.fr/lyc-rabelais-meudon/YannPierreOctaveLangages/page_java.html

[28] https://fr.wikipedia.org/wiki/Swing_(Java)

[29] https://fr.wikipedia.org/wiki/Eclipse_(projet)

[31] https://fr.wikipedia.org/wiki/LaTeX

[33] https://perso.math.univ-toulouse.fr/mleroy/files/2012/03/M1_LaTeX_TPinit.pdf

[32] https://pbil.univ-lyon1.fr/members/mbailly/Comm_Scientifique/M1/cours_latex.pdf.