REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE : Des Mathématiques Et De L'informatique
DEPARTEMENT : D'INFORMATIQUE
N° :……..

DOMAINE : Mathématique et Informatique
FILIERE : Informatique
OPTION : informatique décisionnelle et optimisation

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par: BEN CHOUIKH NASREDDINE

Intitulé

TRACER DE GRAPHES DANS PLAN

Soutenu devant le jury composé de :

Dr.	Université de M'sila	Président
Dr. *moussaoui adel*	Université de M'sila	Rapporteur
Dr.	Université de M'sila	Examinateur

Année universitaire : 2019 /2020


بسم الله الرحمن الرحيم

Dédicace

Je dédie ce travail à mes parents qui m'ont toujours offert le bonheur.

Je le dédie à ma sœur, mes frères, les petits enfants de la famille, et à ma grande famille, A Tous mes amis et Mes collègues.

A la promo 2019/2020 d'informatique.

Enfin, à toutes celles et tous ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail.

Nasreddine

Remerciements

Je tiens avant tout à remercier Dieu le tout puissant de m'avoir donné la force et la volonté pour achever ce modeste travail.

Je remercie Dr.moussaoui adel encadreur d'avoir bien dirigé ce travail, avec ses judicieux conseils dont il a fait preuve durant l'élaboration de notre étude.

Je souhaite remercier toutes les personnes qui m'ont aidé d'une façon directe ou indirecte à la réalisation de ce mémoire.

Table des Matières

CHAPITRE 2 : LES GRAPHES PLANAIRES

Liste des figures

INTRODUCTION GENERALE

Introduction générale

la théorie des graphes est la discipline_mathématique_et informatique qui étudie les *graphes*, lesquels sont des modèles abstraits de dessins de réseaux reliant des objets [15]. Ces modèles sont constitués par la donnée de *sommets* (aussi appelés *nœuds* ou *points*, en référence aux polyèdres), et d'arêtes (aussi appelées *liens* ou *lignes*) entre ces sommets ; ces arêtes sont parfois non-symétriques (les graphes sont alors dits *orientés*) et sont appelés des *flèches*

En théorie des graphes, le tracé de graphes consiste à représenter des graphes dans le plan. Le tracé de graphes est utile à des applications telles que la conception de circuits VLSI, l'analyse de réseaux sociaux, la cartographie, et la bio-informatique.

L'objectif de ce mémoire est un outil et algorithme de dessin graph selon différents métriques en particulier ; un minimum de croisement d'arcs possible.

Dans le premier chapitre, nous présenterons les aspects fondamentaux de la théorie des graphes pertinents pour notre étude. Dans Le deuxième chapitre, nous présenterons les graphes planaires, On a présentera ensuite, notion complexité. Le troisième chapitre, nous présenterons les problèmes sujet étude. Quatrième chapitre On a appliqué l'algorithme pour résoudre notre problème. Nous donnerons plusieurs résultats obtenus. Enfin, on termine par une conclusion générale et quelques perspectives.

CHAPITRE 1

Les GRAPHES : CONCEPTS FONDAMENTAUX

1. Introduction

Les graphes représentent de manière simple et naturelle des relations entre les objets. Les outils mathématiques et les algorithmes mis au point en théorie des graphes permettent de résoudre une multitude de problèmes, tels que les problèmes de cheminement, d'ordonnancement, d'affectation. Ce chapitre présente les aspects fondamentaux de la théorie des graphes.

2. Définition d'un graphe et ses différentes représentations

Un graphe est une structure comportant un ensemble non vide *V* d'éléments appelés sommets et un ensemble *E* d'éléments appelés arêtes. Il existe en fait deux type de graphes : les graphes orientés et les graphes non orientes.

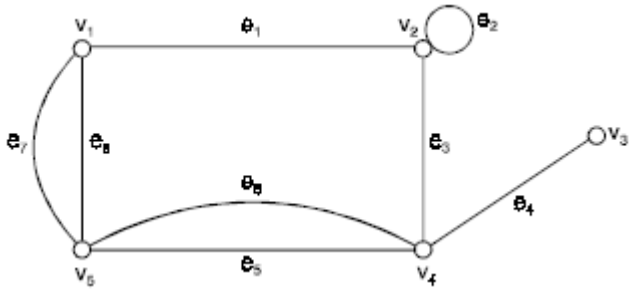


Figure10.0 : graph 5 sommet et 8 arête S'appeler graph [16]

3.1. Graphe orienté

En donnant un sens aux arêtes d'un graphe, on obtient un digraphe (ou graphe orienté). Un digraphe fini $G = (V, U)$ est défini par l'ensemble fini $V = \{v_1, v_2,..., v_n\}$ dont les éléments sont appelés sommets, et par l'ensemble fini $U = \{u_1, u_2,..., u_m\}$ dont les éléments sont appelés arcs.

Chaque élément u_i de l'ensemble U est défini par une paire ordonnée de sommets. Lorsque $u_i = (x, y)$, on dit que l'arc u va de x à y. On dit aussi que x est l'extrémité initiale et y l'extrémité terminale de u_i [1].

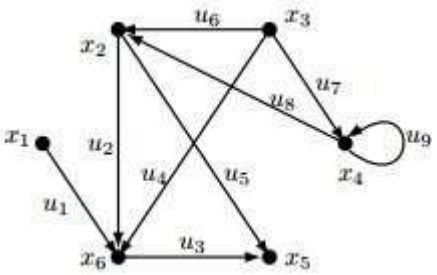


Figure 1.1 : Un graphe orienté [3].

3.2. 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 e_i de l'ensemble E est définie par une paire non ordonnée de sommets, appelés les extrémités de e_i. Si l'arête e relie les sommets a et b, on dira que ces sommets sont adjacents, ou incidents avec e_i [1].

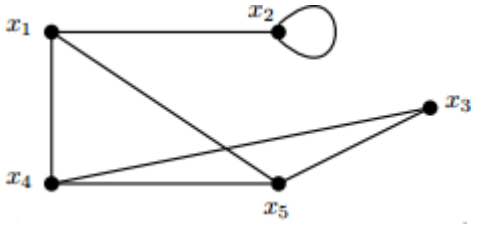


Figure 1.2 : Un graphe non orienté [3].

3.3. Terminologie

La terminologie suivante s'applique à tout graphe *G :*

- Les deux sommets *u* et *v* sont appelés extrémités de l'arête (*u, v*). s'il s'agit d'un arc, *u* et *v* désignent respectivement l'extrémité initiale et terminale de l'arc. Le sommet *u* est prédécesseur de *v* et *v* est successeur de *u*.
- Les arcs ayant même extrémités sont appelés arcs parallèles.
- Un arc de la forme (*v, v*) est une boucle.
- Un graphe simple s'il n'a ni arcs parallèles ni boucle.
- Un graphe sans arcs est un graphe vide ($G = (V, E) / E = \emptyset$).
- Un graphe sans sommets est un graphe nul ($G = (V, E) / V = \emptyset$).
- Un graphe ayant un seul sommet est un graphe travail ($G = (V, E) / Card(V) = 1$).
- Des arcs sont adjacents s'ils ont un sommet en commun à l'une des extrémités.
- Deux sommets *u* et *v* sont voisins ou adjacents s'ils sont connectes par un arc.
- Le voisinage d'un sommet désigne l'ensemble de ses sommets voisins.
- Un graphe *G* est dit complet si et seulement si $\forall\ x, y \in V$ si $(x, y)\ E \Rightarrow (y, x) \in E$. c'est-à-dire que tous les couples de sommets qu'il est possible de choisir (chacun étant pris une seule fois) sont présents dans *E* [4].

3.4. Quelques types de graphes

Il existe différents types de graphes en fonction du nombre de sommets, le nombre de arc, l'inter connectivité, et leur structure globale. Nous discuterons seulement quelques types importants de graphe :

1- Graphe simple : un graphe est dit simple s'il est sans boucle, il n'y a jamais plus d'une arête entre deux sommets quelconques.

2- Graphe multi-graphes : un multi-graphe est un graphe pour lequel il peut exister plusieurs arêtes entre deux sommets donnes.

3- Graphe régulier : un graphe G est dit k-réguliers si $\forall\ x$ sommets de G. on a $d_G(x) = k$. en d'autre termes, $\delta(G) = \Delta(G) = k$.

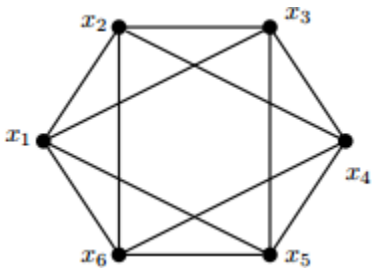


Figure 1.3 : Un graphe régulier [3].

4- Graphe complet : Un graphe complet est un graphe où chaque sommet est relié à tous les autres. Le graphe complet d'ordre n est noté *Kn.*

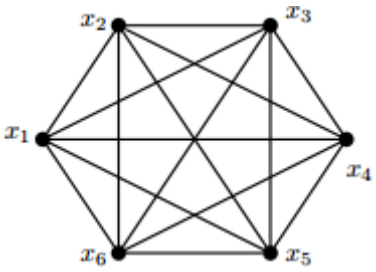


Figure 1.4 : Un graphe complet [3].

5- Graphe 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* [1].

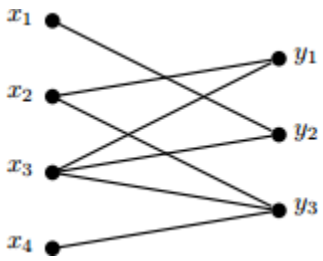


Figure 1.5 : Un graphe biparti [3].

3.5. Nombre d'un graphe

Les deux nombres d'un graphe sont l'ordre et le degré [4].

3.5.1. Ordre d'un graphe

Dans la définition ci-dessus d'un graphe, le nombre n représente l'ordre du graphe G, il s'agit du nombre de ses sommets [4].

3.5.2. Notion de degré

On appelle degré du sommet *v*, et on note *d*(*v*), le nombre d'arêtes incidentes à ce sommet. Dans un graphe simple, on peut aussi définir le degré d'un sommet comme étant le nombre de ses voisins (la taille de son voisinage).

Dans le cas d'un graphe orienté, le degré d'un sommet est calculé comme étant la somme des demi degrés extérieur et intérieur d'un sommet tel que :

- Demi-degré extérieur : $d^+(v)$ est le nombre d'arcs ayant *v* comme l'extérieur initial.
- Demi-degré intérieur : $d^-(v)$ est le nombre d'arcs ayant *v* comme l'intérieur terminal [1].

3. Représentation d'un graphe

Un certain nombre de représentations existent pour décrire un graphe. On distingue principalement la représentation par matrice d'adjacence et par liste d'adjacence.

4.1. Matrice d'adjacence

A tout graphe d'ordre *n* on associe une matrice *M* de *n* lignes et *n* colonnes dont les éléments sont notés M_{ij}. Pour un graphe non orienté $G = (V, E)$,

$$M(i,j)\begin{cases}1 \ si \ (i,j) \in E \\ 0 \ sinon\end{cases} \qquad (1.1)$$

M

Remarque

- La matrice d'adjacence *M* d'un graphe non orienté est toujours symétrique.
- La somme d'une ligne *k* = la somme d'une colonne *k* = $d_G(k)$. La boucle est comptée deux fois.

Pour un graphe orienté $G = (V, U)$, M_{ij} représente le nombre d'arcs ayant *i* comme extrémité initiale et *j* comme extrémité terminale [3].

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 1.6 : Matrice d'adjacence du graphe orienté de la figure 1.1 [3].

$$M = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Figure 1.7 : Matrice d'adjacence du graphe non orienté de la figure 1.2 [3].

4.2. Liste d'adjacence

Pour un graphe $G = (V, E)$, la liste d'adjacence associe à ce graphe permet de représenter pour chaque sommet, l'ensemble de ses successeurs. Tous les arcs émanant d'un même sommet, sont liés entre eux dans une liste. A chaque arc sont donc associes l'extrémité terminale et le pointeur au prochain sommet de la liste [3].

4. Notion de sous-graphe et graphe partiel

Soit $G = (V, U)$ un graphe orienté (ou $G = (V, E)$ un graphe non orienté). Soit $A \in V$ un sous ensemble de sommets et $V \in U$ (ou $V \in E$).

5.1. Sous graphe

Un sous graphe de G engendré par l'ensemble de sommets A est le graphe :

$G_A = (A, U_A)$ où $UA = \{u \in U/I(u) \in A$ et $T(u) \in A\}$ dans le cas orienté.

$G_A = (A, E_A)$ où $EA = \{e = \{x, y\} \in E/x \in A$ et $y \in A\}$ dans le cas non orienté [3].

5.2. Graphe partiel

Un graphe partiel de G engendré par l'ensemble d'arcs (ou d'arêtes) R est le graphe $G_R = (V, R)$.

5.3. Sous graphe partiel

Un sous graphe partiel de G engendré par l'ensemble de sommets A et l'ensemble d'arcs (ou d'arêtes) V est le graphe $G_{A, V} = (A, V_A)$. V_A est l'ensemble d'arcs (ou arêtes) qui one leurs deux extrémités dans le sous ensemble V [3].

5. Cheminement dans un graphe

Plusieurs problématiques peuvent être considérées lorsque l'on s'intéresse au cheminement entre deux points d'un graphe. Cette section a pour objectif de définir les concepts de cheminement dans un graphe.

6.1. Chaines et cycles

Les concepts de chaine et cycle peuvent aussi bien s'appliquer à un graphe orienté qu'a un graphe non orienté [2].

6.1.1. Chaine

On appelle chaîne dans un graphe non orienté (ou orienté) $G = (V, E)$ (ou $G = (V, U)$), une suite alternée de sommets et d'arêtes (ou d'arcs) : $\mu = v_0, e_1, v_1 \ldots v_{k-1}, e_k, v_k$ (ou $\mu = v_0, u_1, v_1 \ldots v_{k-1}, u_k, v_k$) Tel que pour tout $1 \leq i \leq k$, v_i et v_{i+1} sont extrémités de l'arête e_i (ou de l'arc u_i). On dit que μ est une chaîne joignant les sommets v_0 et v_k de longueur k [2].

Le graphe ci-dessous $C1 = (x_1, e_2, x_4, e_5, x_3, e_6, x_5)$ est une chaine de longueur 3.

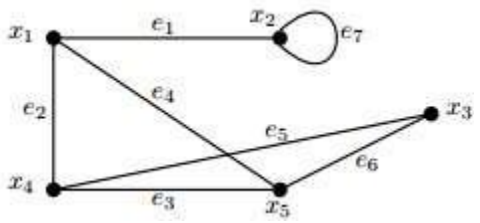


Figure 1.8 : Un graphe non orienté [3].

6.1.2. Cycle

On appelle cycle dans un graphe non orienté (ou orienté) $G = (V, E)$ (ou $G = (V, U)$), toute chaîne fermée simple : $\mu = v_0, e_1, v_1 \ldots v_{k-1}, e_k, v_k$ (ou $\mu = v_0, u_1, v_1 \ldots v_{k-1}, u_k, v_k$) Tel que k_0, et $v_0 = v_k$. On dit que μ est un cycle longueur k [2].

6.2. Chemins et circuits

Les concepts de chemin et circuit sont relatifs aux graphes orientes puisque le sens d'orientation des arcs est essentiel pour ces définitions.

6.2.1. Chemin

On appelle chemin dans un graphe orienté $G = (V, U)$, une suite alternée de sommets et d'arcs : $\gamma = v_0, u_1, v_1 \ldots v_{k-1}, u_k, v_k$ Tel que pour tout $1 \leq i \leq k$, le sommet v_i est extrémité initiale de l'arc u_i et le sommet v_{i+1} est son extrémité terminale.

Le graphe ci-dessous. On dit que γ est un chemin de v_0 vers v_k de longueur k. C2 = (v_3, u_6, v_2, u_5, v_5, u_3, v_6) est un chemin de longueur 3, C3 = (v_3, u_7, v_4, u_9, v_4, u_8, v_2, u_2, v_6) est un chemin de longueur 4, C4 = (v_1, u_1, v_6, u_2, v_2, u_6, v_3, u_7, v_4) est une chaine de longueur 4 [2].

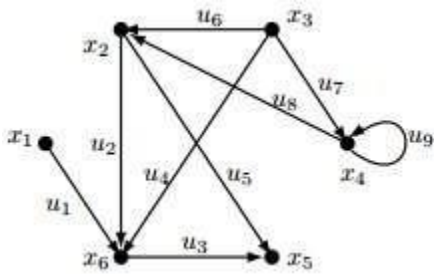


Figure 1.9 : Un graphe orienté [3].

6.2.2. Circuit

On appelle circuit dans un graphe orienté $G = (V, U)$, tout chemin fermé simple : $\gamma = v_0, u_1, v_1 \ldots v_{k-1}, u_k, v_k$ tel que $k > 0$, et $v_0 = v_k$. On dit que γ est un circuit de longueur k.

6.Notion de connexité

La connexité est une propriété essentielle des graphes surtout pour les problèmes relatifs aux réseaux. Dans ce qui suit, sont étudiées les notions de connexité simple et forte.

7.1. Connexité simple

Un graphe est dit connexe si et seulement si : [6] :

$$\forall\, i, j \in V \begin{cases} soit\ i = j\ ou \\ il\ existe\ une\ chaine\ reliant\ i\ et\ j \end{cases}$$

7.2. Connexité forte

Un graphe oriente est dit fortement connexe si seulement si :

$$\forall\, i, j \in V \{ \begin{matrix} i = j \ ou \\ il\ existe\ un\ chemin\ reliant\ i\ et\ j\ et\ un\ chemin\ reliant\ j\ et\ i \end{matrix}$$

Un graphe est fortement connexe si et seulement s'il n'a qu'une seule composante fortement connexe [6].

6. Graphes particulaires

Les graphes ont parfois des caractéristiques qui les rendent particuliers. Dans ce qui suit, nous définissons certains types de graphe particuliers, comme le graphe eulériens et hamiltonien.

8.1. Graphe Eulérien

La ville de Königsberg est construite autour de deux îles situées sur le PREGEL et reliées entre elles par un pont. Six autres ponts relient les rives de la rivière à l'une ou l'autre des deux îles, comme représentés sur le plan ci-dessous. Le problème consiste à déterminer s'il existe ou non une promenade dans les rues de Königsberg passant une et une seule fois par chaque pont de la ville [7].

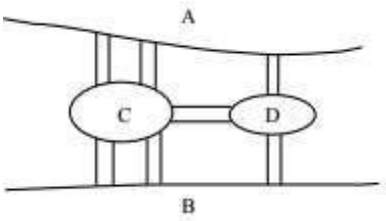


Figure 1.10 : Les ponts de Königsberg [3].

8.1.1. Chaine Eulérienne

Une chaine Eulérienne est une chaine qui passe par toutes les arêtes du graphe exactement une fois. Si la chaine est fermée, il s'agit d'un cycle Eulérien [7].

8.1.2. Chemin Eulérien

Un chemin Eulérien est un chemin qui passe par tous les arcs du graphe exactement une fois. Si le chemin est ferme, il s'agit d'un circuit Eulérien [7].

8.2. Graphe hamiltonien

Le problème hamiltonien, invente par Hamiltonien, revient à déterminer pour un ensemble de villes un itinéraire permettant de passer une et une seul fois par chaque ville et revenir au point de départ [7].

8.2.1. Chaine hamiltonienne

Une chaine hamiltonienne est une chaine qui visite chaque sommet du graphe exactement une fois. Un cycle Hamiltonien est donc une chaine Hamiltonienne fermée [7].

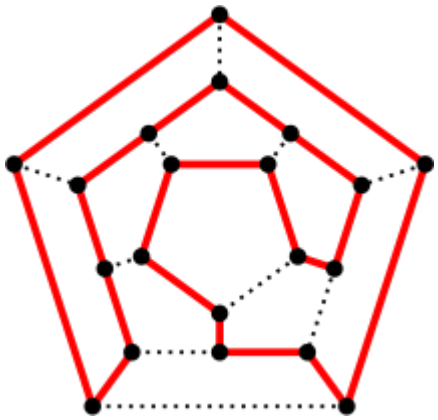


Figure 1.11 : présentation d'un graphe hamiltonien [7].

8.2.2. Chemin hamiltonien

Un chemin Hamiltonien est un chemin qui visite chaque sommet du graphe exactement une fois. Un circuit Hamiltonien est donc un chemin hamiltonien fermé [7].

7. Isomorphisme de graphes

Est une bijection entre les sommets de deux graphes qui préserve les arêtes. Ce concept est en accord avec la notion générale d'isomorphisme, une bijection qui préserve les structures.

Soit G1 = (v1, E1) et G2 = (v2, E2) deux graphes. Une fonction f: v1 $\rightarrow$ v1 est appelée isomorphisme des graphes si

(i) f est un-à-un et sur.

(ii) pour tout a, b $\in$ v1, {a, b} $\in$ E1 si et seulement si {f (a), f (b)} $\in$ E2 lorsqu'une telle fonction existe, G1 et G2 sont appelés graphes isomorphes et sont écrit comme G1 $\cong$ G2.

En d'autres termes, deux graphes G1 et G2 sont dits isomorphes l'un par rapport à l'autre s'il existe une correspondance biunivoque entre leurs sommets et entre des arêtes de sorte que la relation d'incidence est préservée. Écrit comme G1 $\cong$ G2 ou G1 = G2.

Les conditions nécessaires pour que deux graphes soient isomorphes sont

1. Les deux doivent avoir *le même nombre de sommets*
2. Les deux doivent avoir *le même nombre d'arêtes*
3. Les deux doivent avoir le *même nombre de sommets* avec le *même degré.*
4. Ils doivent avoir la même séquence de degrés et le même vecteur de cycle (c1,….., cn), où ci est le nombre de cycles de longueur i.

Exemple

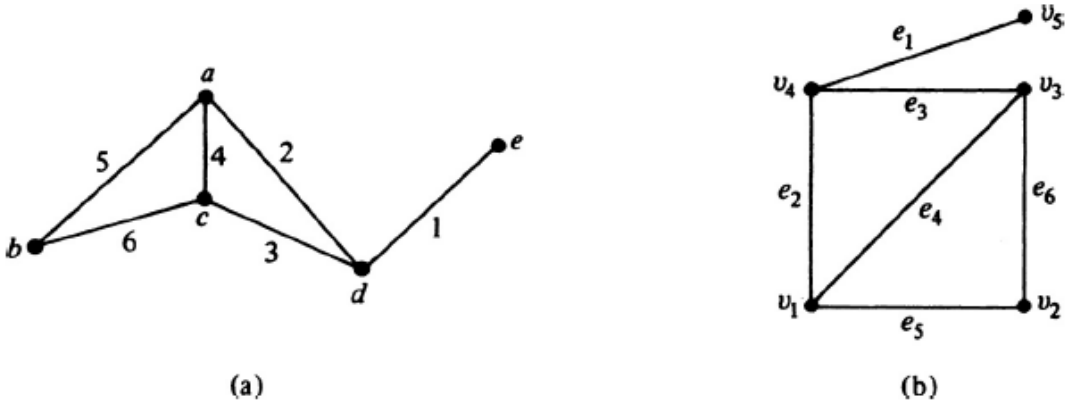


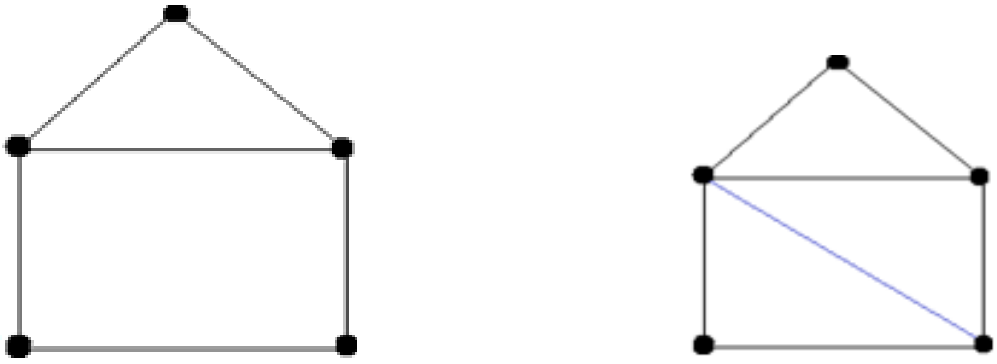
Figure .1.9 : Isomorphisme de graphes **[15]**

8. Graphe triangulé et non triangulé

Une corde est une arête qui relie deux sommets non adjacents dans un cycle. Un graphe G est *triangulé* si tout cycle dans G de longueur supérieure à 3 admet une *corde*

Exemple

Le graphe de la Figure (1) n'est pas un graphe triangulé car il existe un cycle de longueur 4 qui n'admet pas de corde. Le graphe de la Figure (2) est un graphe triangulé.



(1) (2)

Figure 1.10 : graphe triangulé

8. Conclusion

Nous avons discuté dans ce chapitre quelques concepts de la théorie des graphes. Cela devrait nous permettre de passer au prochain chapitre avec un arrière-plan acceptable sur les graphes en général.

CHAPITRE 2

LES GRAPHES PLANAIRES

1. Introduction

Dans ce chapitre on va présenter notion graphe planaire. En premier temps, on va passer en revue la définition d'un graph planaire. En deuxième temps, on va détailler ensuit présenter notion complexité NB compte.

2. LES GRAPHES PLANAIRES

2.1. Représentation de graphe planaire

Un graphe abstrait G peut être défini comme G = (V, E, ¥) où l'ensemble V est constitué des cinq objets nommés a, b, c, d et e, c'est-à-dire V = {a, b, c, d, e} et l'ensemble E se compose de sept objets (dont aucun n'est dans l'ensemble V) nommés 1, 2, 3, 4, 5, 6 et 7, soit

E = {1, 2, 3, 4, 5, 6, 7}

et la relation entre les deux ensembles est définie par le mappage ¥, qui consiste en une représentation combinatoire du graphe.

$$
\text{¥} = \begin{array}{l} 1 \longrightarrow (a,\ b) \\ 2 \longrightarrow (b,\ c) \\ 3 \longrightarrow (c,\ d) \\ 4 \longrightarrow (a,\ b) \\ 5 \longrightarrow (d,\ e) \\ 6 \longrightarrow (a,\ d) \\ 7 \longrightarrow (b,\ d) \end{array} \quad \longrightarrow \text{Représentation combinatoire d'un graphe}
$$

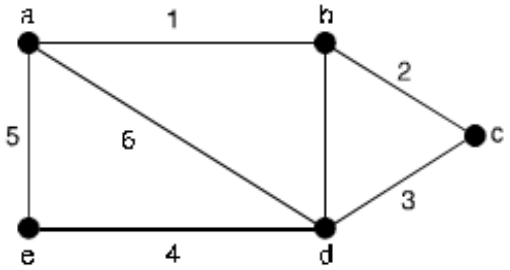


Figure2.1 : Représentation d'un graphe planaire [16]

2.2. Définition

Un graphe G est dit *plan* s'il existe une représentation géométrique de G qui peut être dessinée sur un plan de telle sorte qu'aucune de ses arêtes ne se coupe. Les points d'intersection sont appelés croisements.

Un graphe qui ne peut pas être dessiné sur un plan sans un croisement entre ses arêtes se croisant est appelé[16]

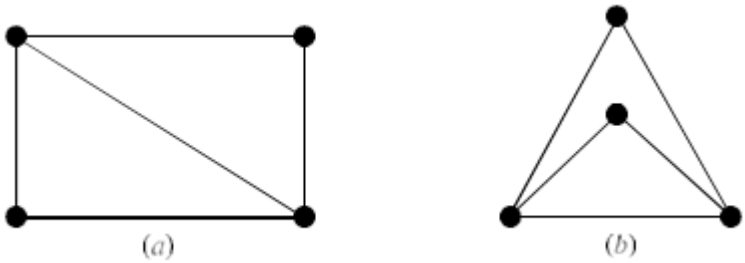


Figure 2.2 graphes planaires[16]

2.3. Théorème de Battle-Harary-Kodama

Le théorème de Battle-Harary-Kodama est un théorème mathématique de la théorie topologique des graphes et qui doit son intitulé à une publication des mathématiciens Joseph Battle, Frank Harary et Yukihiro Kodama de l'année 1962[12]. Le théorème est une réponse à une question sur la planarité de certains graphes simples et de leur graphe complémentaire et résout une conjecture de John L. Selfridge. Une démonstration plus simple a été donnée un peu plus tard, en 1963, par William Tutte [14].

Théorème : Si un graphe simple planaire G (V, U) a au moins 9 sommets, alors son graphe complémentaire n'est pas planaire ; de plus, le nombre 9 est le plus petit entier naturel avec cette propriété.

2.4. Graphes de Kuratowski

Pour cela, nous discutons de deux graphes non planaires spécifiques, qui sont d'une importance fondamentale, on les appelle les graphes de Kuratowski. Le graphe complet à 5 sommets est le premier des deux graphes de Kuratowski. Le second est un graphe régulier et connecté avec 6 sommets et 9 arêtes. [16]

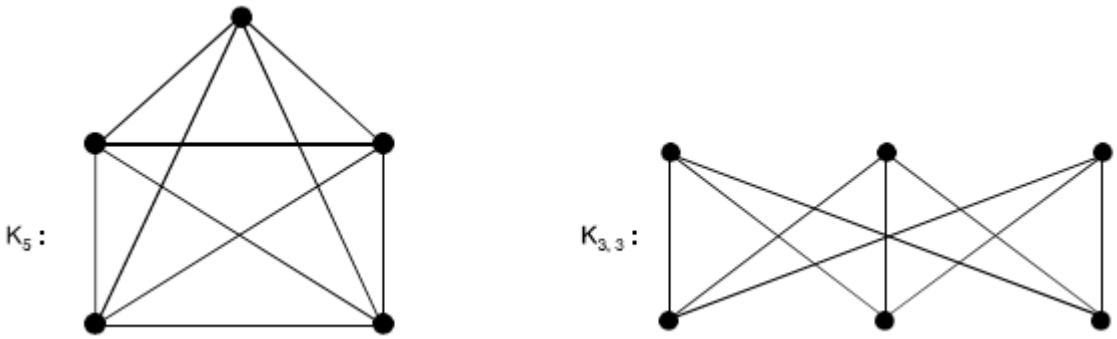


Figure2.4.les graphes de Kuratowski[16]

Observations

(i) Les deux sont des graphiques réguliers

(ii) Les deux sont des graphes non planaires

(iii) La suppression d'un sommet ou d'une arête rend le graphe planaire

Le premier graphe de Kuratowski est un graphe non plan avec le plus petit nombre de sommets et le second graphe (de Kuratowski) est un graphe non plan avec le plus petit nombre d'arêtes. Ainsi, les deux sont les graphes non planaires les plus simples.

Les premier et deuxième graphiques de Kuratowski sont représentés par K5 et K3, 3. La lettre K étant pour Kuratowski (un mathématicien polonais). [16]

2.5. Région

Une représentation plane d'un graphique divise le plan en régions (également appelées fenêtres, faces ou maillages), comme illustré dans la figure ci-dessous. Une région est caractérisée par l'ensemble des arêtes (ou l'ensemble des sommets) formant sa frontière.

Notez qu'une région n'est pas définie dans un graphe non plan ou même dans un graphe plan non incorporé dans un plan. [16]

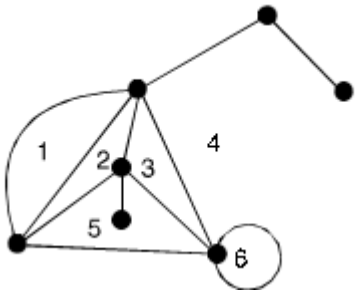


Figure 2.4 : représentation de graph planaire (nombre de région) **[16]**

2.6 GRAPHIQUES PLANAIRES MAXIMALES

Un graphe plan est planaire maximal si aucune arête ne peut être ajoutée sans perdre la planéité. Ainsi, dans tout graphe planaire maximal avec $p \leq 3$ sommets, la frontière de chaque région de G est un triangle pour ce graphe planaire maximal (ou graphes plans) sont également appelés graphe plan triangulé (ou graphe plan) [16]

2.7. une subdivision d'un graphe

Une subdivision d'un graphe est un graphe obtenu en insérant des sommets (de degré 2) dans les arêtes de G. Pour le graphe G de la figure ci-dessous, le graphe H est une subdivision de G, tandis que F n'est pas une subdivision de G.[16]

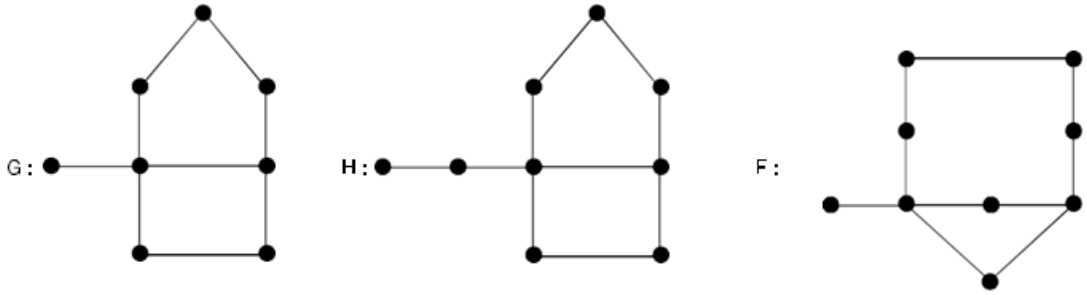


Figure 2.6 subdivision d'un graphe **[16]**

/47

2.8. ENSEMBLE DE SOMMETS INTERIEUR

Un ensemble de sommets d'un graphe plan G est appelé un ensemble de sommets intérieurs I (G) de G.SI peut être dessiné sur le plan de telle sorte que chaque sommet de I (G) ne se trouve que sur la région intérieure et I (G) contient le minimum de sommets possibles de G. Le nombre de sommets i (G) de I (G) est dit être le nombre de sommets intérieurs s'ils se trouvent dans la région intérieure de G. [16]

2.9. GRAPHIQUES PLANAIRES EXTÉRIEURES

Un graphe est appelé planaire extérieur (outerplanaire) s'il admet un plongement dans le plan tel quel tous les sommets soient situ ´es sur un cercle, et les arêtes uniquement dans le disque induit. [21]

2.9.1. Graphique planaire externe maximal :

Un graph plan extérieur G est planaire extérieur maximal si aucune arête ne peut être ajoutée sans perdre la planéité extérieure.[16]

Par example:

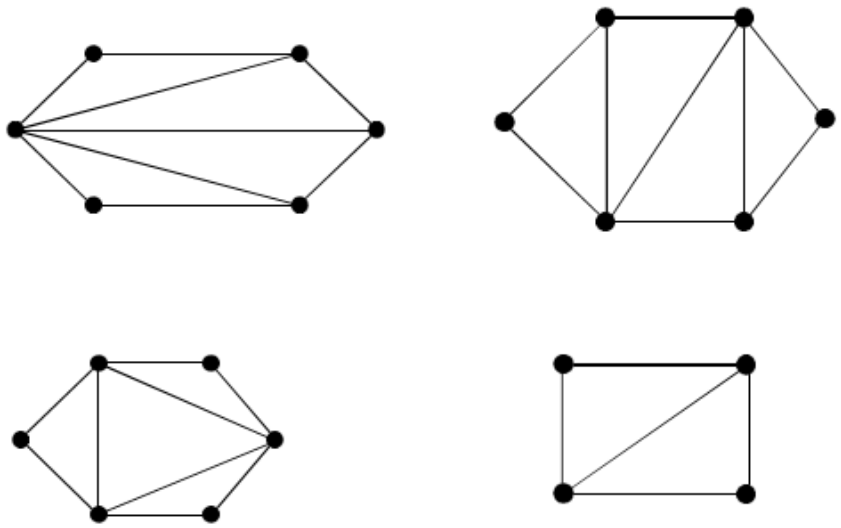


Figure2.9.1. Graphique planaire externe maximal[16]

2.9.2. Graphiques planaires minimalement non extérieurs

Un graphe plan G est dit minimalement non planaire externe si i (G) = 1

Par exemple :

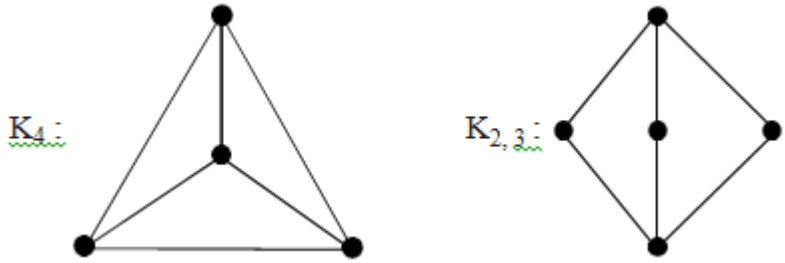


Figure2.9.2 : Graphiques planaires minimalement [16]

2.9.3 Énoncé voisin

Un énoncé voisin, donné par Dennis P. Geller, traite la question de la « planarité extérieure » : un graphe est planaire extérieur s'il admet une représentation plane dans laquelle les sommets se trouvent tous sur le bord de la face extérieure. Le théorème de Geller s'énonce comme suit :

Théorème Si un graphe simple G est planaire extérieur et a au moins 7 sommets, alors son graphe complémentaire G n'est pas planaire extérieur; de plus, le nombre 7 est le plus petit entier naturel avec cette propriété. [49]

2.10. Numéro de croisement

Le nombre de croisement C (G) d'un graphe G est le nombre minimum de croisement de ses arêtes parmi tous les dessins de G dans le plan.

Un graphe est planaire si et seulement si C (G) = 0. Puisque K4 est planaire C (K4) = 0 pour p ≤ 4. D'autre part C (K5) = 1. Aussi K3, 3 est non planaire et peut être dessiné avec un croisement. [16]

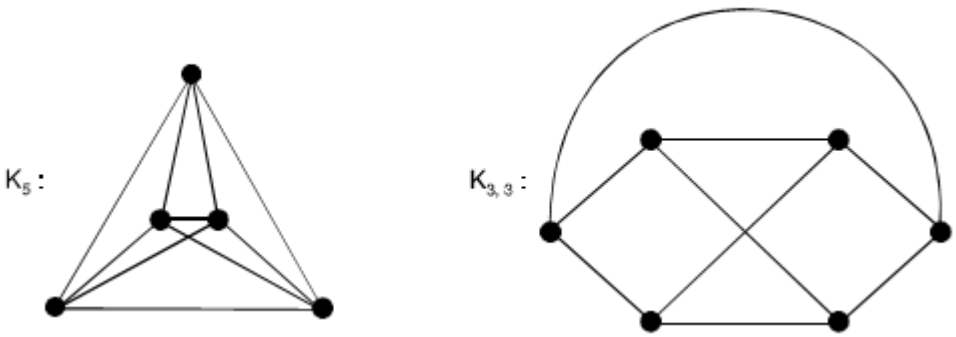


Figure2.10 : k5 : graph non planaire k3.3 : graph planaire [16]

2.11. Le graphe dual

En théorie des graphes, le graphe dual d'un graphe plongé dans une surface est défini à l'aide des composantes de son complémentaire, lesquelles sont reliées entre elles par les arêtes du graphe de départ.

Cette notion généralise celle de dualité dans les polyèdres.
Il faut noter qu'un même graphe abstrait peut avoir des graphes duaux non isomorphes en fonction du plongement choisi, même dans le cas de plongements dans le plan. [16]

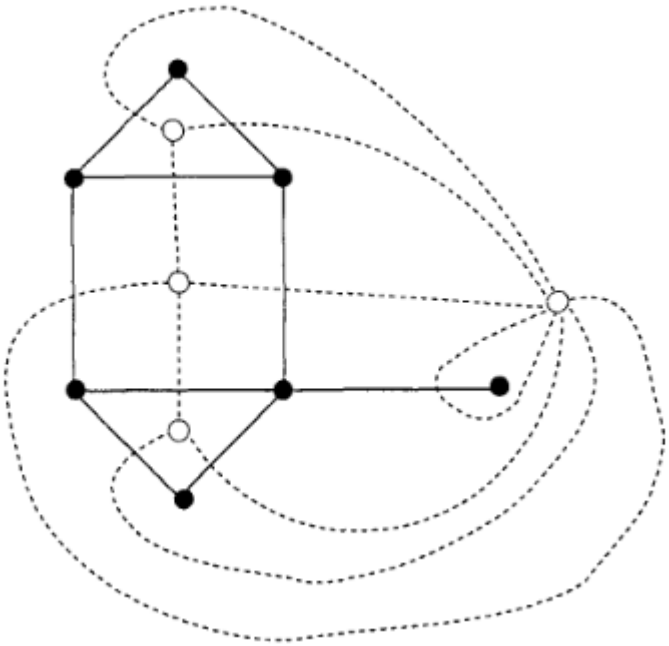


Figure2.10 : graph dual **[16]**

2.11.2. Planarité de Whitney (dual)

Critère de planarité de Whitney est une caractérisation, en théorie des matroïdes, des graphes planaires ; critère nommée d'après Hassler Whitney. Il affirme qu'un graphe G est planaire si et seulement si son matroïde graphique est également cographique (c'est-à-dire qu'il est le matroïde dual d'un autre matroïde graphique). [22]

En termes de théorie des graphes pures, ce critère énonce comme suit:
Un graphe G=(V,E) est planaire si et seulement s'il existe un autre graphe (« dual ») $G' = (V' , E')$ et une correspondance bijective entre E' et E telle qu'un sous-ensemble T de E forme un arbre couvrant de G si et seulement si les arêtes correspondantes au sous-ensemble complémentaire $E - T$ forment un arbre couvrant de G'.

2.12. La formule d'Euler

La formule d'Euler pour un graphe planaire connexe est :

$$n - a + f = 2.$$

Une méthode simple pour la démontrer est de l'établir pour un graphe sans cycle (à une face), puis par récurrence d'ajouter les arêtes qui engendrent des cycles. Cette formule permet de démontrer que tout graphe simple planaire connexe, ayant au moins trois sommets, vérifie la majoration suivante.

$$(2) \quad a \leq 3n - 6.$$

Un graphe simple est un graphe sans boucle (une boucle est une arête qui possède une origine et une extrémité confondues) et sans arête multiple (des arêtes multiples sont des arêtes ayant même origine et même extrémité). Cette formule se démontre simplement : toute face est de degré au moins égal à 3, car le graphe comporte au moins trois nœuds et est simple. On en déduit, par la formule (1), que $3f$ est inférieur ou égal à $2a$. La formule d'Euler permet de conclure.

Cette formule implique que les graphes planaires sont des graphes creux. De plus, leur arboricité est bornée par.[23]

Cette majoration est à l'origine d'une démonstration du fait que K_5 n'est pas planaire. En effet, K_5 dispose de 10 arêtes, et 5 nœuds, ce résultat est incompatible avec la majoration (2). Si, avec les hypothèses de la majoration (2), le graphe est sans triangle, on dispose alors de la majoration :

$$(3) \quad a \leq 2n - 4.$$

Nous utiliserons les notations suivantes :

G désigne un graphe planaire,

n son nombre de nœuds (sommet),

a son nombre d'arêtes (ou de liens),

f son nombre de faces.

3. COMPLEXITE DES ALGORITHMES

Un même problème peut généralement être résolu par plusieurs algorithmes, donc il faut comparer entre ces algorithmes, cette comparaison se base sur le temps de calcul et sur l'espace mémoire requis par l'algorithme. On appelle complexité en temps d'un algorithme dans le pire cas, la fonction *F(n)* qui donne une borne supérieure du nombre d'opérations élémentaires effectuées par l'algorithme lorsque la taille de l'entrée est *n.*[10]

Un algorithme dont le nombre d'opérations élémentaire nécessaires pour résoudre un exemple de taille *n* est une fonction polynomiale en *n*. Un algorithme est efficace (qu'il est <bon>) si le nombre des opérations nécessaires pour résoudre un problème est borné par une fonction polynomiale d'un paramètre caractérisant la taille de ce problème. Il faut noter qu'un problème décision, est le problème pour lequel une solution est soit « oui » soit « non », alors qu'un problème d'optimisation, est le problème pour lequel on doit chercher à déterminer une solution qui optimise un critère. A chaque problème d'optimisation on peut associer un problème de décision [10].

Réduction polynomiale

Soient Π et Π' deux problèmes de décision. Π' se réduit polynomialement a Π et on note Π' α Π, si Π' est polynomial ou s'il existe un algorithme polynomial qui construit, à partir d'une donnée *D'* de Π', une donnée D de Π telle que, la réponse pour *D'* soit oui si est seulement si la réponse pour *D* est oui [10].

3.1. La classe NP

La classe NP est la classe des problèmes de décision qui peuvent être résolus par une machine de Turing non déterministe en temps polynomial. On distingue trois sous classes : La classe P

qui contient les problèmes les plus faciles de NP, ils ont un algorithme polynomial pour les résoudre La classe des problèmes NP-complets contient les problèmes les plus difficiles de NP. Alors que la classe des problèmes ouverts englobe les problèmes de statut indéterminé [10].

3.1. Les problèmes NP-difficile

Un problème d'optimisation est dit NP-difficile sil le problème de décision associe est NP-complet [10].

2. Conclusion

Dans ce chapitre nous avons passé en revue les graphes planaires in général et ainsi leurs complexités. Ensuite, nous avons présenté. Les algorithmes trace graph dans plan.

CHAPITRE 3

PROBLEME TEST DE PLANARITE ET LALGORITHMES DESSIN GRAPHAES PLANAIRES

1. Introduction

Dans ce chapitre, nous allons présenter problème test de planarité. . Ensuite on va présenter des méthodes pour ce problème. Une grande partie Nous parlons d'algorithmes dessin graphes planaire.

2. Problème test de planarité

2.1. Présentation du problème test de planarité

Le problème qui consiste, étant donné un graphe à savoir si c'est un graphe planaire est appelé test de planarité. Plusieurs algorithmes ont été proposés pour ce problème. Les meilleurs atteignent une complexité en temps linéaire, ce qui est optimal asymptotiquement. Le premier tel algorithme date de 1974, et est dû à Robert Tarjan et John Hopcroft9. Robert Cori a décrit l'historique et les principes de cet algorithme dans un article paru dans le Bulletin de la société informatique de France [24].

le problème du test de planarité est le problème algorithmique qui consiste à tester si un graphe donné est un graphe planaire (c'est-à-dire s'il peut être dessiné dans le plan sans intersection d'arêtes). Il s'agit d'un problème bien étudié en informatique pour lequel de nombreux algorithmes pratiques ont été donnés, souvent en décrivant de nouvelles structures de données. La plupart de ces méthodes fonctionnent en temps O(n) (temps linéaire), où n est le nombre d'arêtes (ou de sommets) du graphe, ce qui est asymptotiquement optimal. La sortie d'un algorithme de test de planarité, plutôt que d'être simplement une valeur booléenne (le graphe est-il planaire, oui ou non ?), peut être un plongement du graphe dans le plan si le graphe est planaire, ou la donnée d'un obstacle à la planarité tel qu'un sous-graphe de Kuratowski s'il ne l'est pas.

2.2. Test de planarité gauche-droite

e test de planarité gauche-droite, aussi appelé critère de planarité de Fraysseix-Rosenstiehl/[25] est une caractérisation des graphes planaires basée sur les propriétés des arbres de parcours en profondeur ou arbres de Trémaux, publiée par de Fraysseix et Rosenstiehl en 1982 et 1985 [26][27], et utilisée par

eux, avec Patrice Ossona de Mendez, pour développer un algorithme de test de planarité en temps linéaire99[28]Dans une comparaison pratique de six algorithmes de test de planarité réalisée en 2003[29] , il s'agissait alors de l'un des algorithmes testés les plus rapides.

2.3. Méthodes test de planarité :

2.3.1. Méthode d'addition de chemins

La méthode maintenant classique d'*addition de chemins* de Hopcroft et Tarjan été le premier algorithme de test de planarité en temps linéaire publié en 1974[30]. Robert Cori a décrit l'historique et les principes de cet algorithme dans un article paru dans le *Bulletin de la société informatique de France*[2] où il mentionne l'activité française dans ce domaine à peu près à la même époque. Une implémentation des algorithmes de Hopcroft et Tarjan est fournie dans la *Library of Efficient Data types and Algorithms (en)* de Mehlhorn, Mutzel et Näher[31] [32] En 2012, Taylor[33] a étendu cet algorithme pour générer toutes les permutations d'ordre cyclique des arêtes pour les plongements planaires de composantes biconnexes.

2.3.2. Méthode d'addition de sommets

Les méthodes d'addition de sommets opèrent en maintenant une structure de données représentant les plongements possibles d'un sous-graphe induit du graphe donné, et en ajoutant les sommets un par un à cette structure de données. Ces méthodes ont commencé avec une méthode en O (n^2) conçue par Lempel, Even et Cederbaum en 1967[34]. Elle a été améliorée par Even et Tarjan[35], qui ont trouvé une solution en temps linéaire pour l'étape de numérotation *st* , et par Booth et Lueker[36], qui ont développé la structure de données d'arbre PQ (en). Avec ces améliorations, l'algorithme est linéaire et surpasse la méthode d'addition de chemins dans la pratique[37] . Cette méthode a également été étendue pour permettre un calcul efficace d'un plongement planaire (dessin) d'un graphe planaire[38]En 1999, Shih et Hsu ont simplifié ces méthodes en utilisant un arbre PC (une variante non enracinée des arbres PQ) et un parcours d'arbre postfixe de l'arbre de recherche en profondeur des sommets[39].

2.3.3. Méthode d'addition d'arêtes

En 2004, John Boyer et Wendy Myrvold[40] ont développé un algorithme simplifié en temps linéaire, inspiré à l'origine de la méthode de l'arbre PQ, qui se libère de l'arbre PQ et utilise des adjonctions d'arêtes de bords pour calculer un plongement planaire, s'il existe. Sinon, une subdivision de Kuratowski (de K_5 ou $K_{3,3}$) est calculée. Il s'agit de l'un des deux algorithmes les plus récents (l'autre est l'algorithme de test de planarité de de Fraysseix, de Mendez et Rosenstiehl[41]. Le test de Boyer-Myrvold a été étendu pour extraire plusieurs subdivisions de Kuratowski d'un graphe d'entrée non planaire en un temps d'exécution linéairement dépendant de la taille de sortie[42]. Le code source du test de planarité [43][44] et de l'extraction des subdivisions de Kuratowski est publique. Un autre algorithme

est donné par Bernhard Haeupler et Robert E. Tarjan[45]. D'autres algorithmes qui localisent un sous-graphe de Kuratowski en temps linéaire ont été développés par Williamson dans les années 1980[46].

2.3.4. Méthode de séquence de construction

Une méthode différente utilise une construction inductive de graphes 3-connexes pour construire de façon incrémentale des plongements planaires de chaque composante 3-connexe du graphe donné (et donc un plongement planaire du graphe lui-même)[47]. La construction commence par K_4 et est définie de telle manière que chaque graphe intermédiaire vers une composante complète est à nouveau 3-connexe. Étant donné que ces graphes ont un plongement unique (au retournement et au choix de la face externe près), le graphe suivant, s'il est toujours planaire, doit être un raffinement du graphe précédent. Cela permet de réduire le test de planéité au test, à chaque étape, si l'arête suivante ajoutée a ses deux extrémités sur la face externe du plongement courant. Bien que conceptuellement très simple (et en temps linéaire), la méthode elle-même est compliquée par la difficulté de trouver la séquence de construction.

3. L'ALGORITHMES DESSIN GRAPHAES PLANAIRES

3.1. Dessin en ligne droite

Un dessin en ligne droite d'un graphe plan est un dessin dans lequel chaque arête est dessinée sous la forme d'un segment de ligne droite sans croisement d'arêtes, Wagner Fary et Stein ont indépendamment prouvé que chaque graphe plan G a un dessin en ligne droite. Leurs preuves donnent immédiatement des algorithmes en temps polynomial pour trouver une ligne droite d'un graphe plan donné. L'aire d'un rectangle renfermant un dessin sur une grille entière obtenue par ces algorithmes n'est borné par aucun polynôme dans le nombre n de sommets de G. En fait, il est resté longtemps un problème ouvert pour obtenir un dessin de zone délimitée par un polynôme. En 1990, de Fraysseix et oJ. Et Schnyder a montré par deux méthodes différentes que chaque graphe plan de n> 3 sommets a une ligne droite dessinant sur une grille entière de taille (2n - 4) x (n - 2) et (n - 2) x (n - 2), respectivement. Les deux méthodes peuvent être implémentées sous forme d'algorithmes à temps linéaire et sont bien connues respectivement sous le nom de «méthode de décalage» et de «méthode de réalisateur».

3.2. Ordre canonique

Le premier algorithme en temps linéaire pour calculer un ordre st à partir d'un graphe biconnecté G

donné est dû à Even et Tarjan (1976, 1977). Ebert (1983) présente un algorithme légèrement plus simple qui est encore simplifié par Tarjan [1986]. Les ordres de sommets canoniques sont à l'origine introduits par de Fraysseix, Pach et Pollack [1988, 1990] pour les graphes triangulaires et généralisés par Kant [1996] aux graphes planaires triconnecté. [20]

Kant [1996] montre de manière constructive que chaque graphe planaire triconnecté a un ordre canonique et présente un algorithme en temps linéaire. Bien que plusieurs implémentations de cet algorithme soient disponibles, il est plutôt compliqué à comprendre et à implémenter, et la sortie n'est pas déterminée de manière unique. [20]

Définition : Soit G = (V, E) un graphe triangulaire et soit v1, v2 et vn les sommets sur la face externe de G dans le sens antihoraire. Une commande v1, v2, . . ., vn des sommets est un ordre canonique de (G, v1) si pour chaque k = 3 ,. . . , n - 1, il y a

$1 \leq i1 < i2 < k < j \leq n$ tel que $\{vi1, vk\}, \{vi2, vk\}, \{vk, vj\} \in E$. [20]

Pour un exemple d'ordre canonique d'un graphe triangulaire

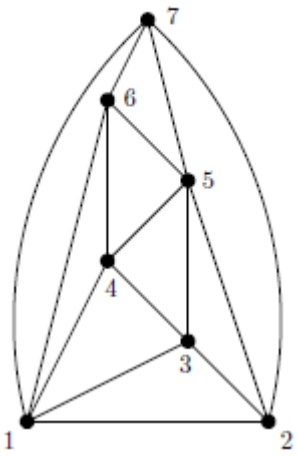


Figure3.2 : ordre canonique d'un graphe triangulaire [20]

Algorithm Canonical-Ordering(G)
begin

1 Let v_1, v_2 and v_n be the vertices appearing on the outer cycle counterclockwise in this order;
2 Set $chords(x) = 0$, $out(x) = false$, and $mark(x) = false$ for all vertices $x \in V$;
3 Set $out(v_1) = true$, $out(v_2) = true$, and $out(v_n) = true$;
4 **for** $k = n$ **down to** 3 **do**
begin
5 Choose any vertex x such that $mark(x) = false$, $out(x) = true$, $chords(x) = 0$, and $x \neq v_1, v_2$;
6 Set $v_k = x$ and $mark(x) = true$;
7 Let $C_o(G_{k-1}) = w_1, w_2, \cdots, w_t$, where $w_1 = v_1$ and $w_t = v_2$;
8 Let $w_p, w_{p+1}, \cdots, w_q$ be the neighbors of v_k which have $mark(w_i) = false$;
9 For each vertex w_i, $p < i < q$, set $out(w_i) = true$, and update the variable *chords* for w_i and its neighbors.
end
end.

[17]

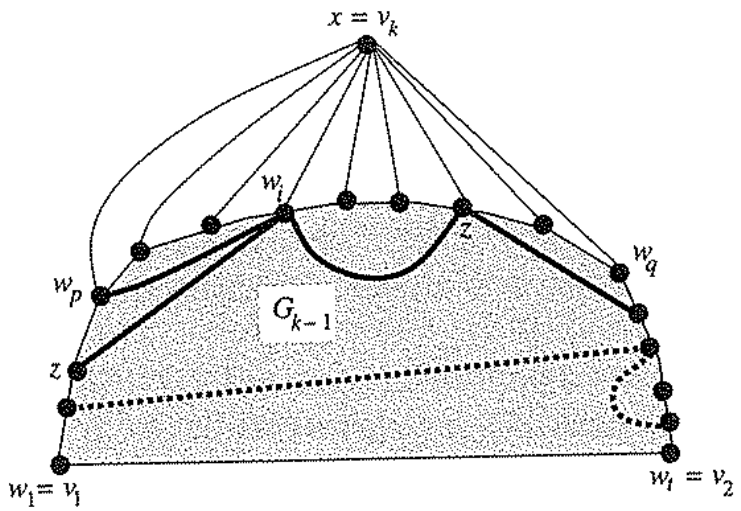


Figure 3.3 : Gk[17]

3.3. Dessins convexes

Une autre façon de dessiner des graphiques plans consiste à les dessiner avec des faces convexes,

c'est-à-dire un dessin en ligne droite plane de sorte que toutes les limites de face internes soient des polygones convexes. Ce problème d'obtention de dessins convexes a d'abord été étudié plus en détail par Tutte .Tutte a également donné une méthode simple pour trouver un dessin convexe. Ici, la face externe est n'importe quel polygone convexe prescrit et la position P (v) = (x (v); y (v)) de chaque sommet v est donnée par :

$$x(v) = \frac{1}{deg(v)} \sum_{(v,w)\in E} x(w) \qquad y(v) = \frac{1}{deg(v)} \sum_{(v,w)\in E} y(w)$$

En utilisant l'élimination gaussienne, les coordonnées peuvent être trouvées par un algorithme simple, travaillant en temps O (n3) et nécessitant un espace O (n2). En utilisant un schéma d'élimination de matrice clairsemé plus sophistiqué qui repose sur le théorème du séparateur planaire, cela conduit à un algorithme O (n n), ne nécessitant qu'un espace O (n log n) . Thomassen [109] a caractérisé la classe des graphes plans qui admettent un dessin convexe. Nous ne donnons pas ici la caractérisation complète, mais elle peut être décrite comme la classe des graphes planaires biconnectés, où \ presque "toutes les paires de séparation font partie de la face externe. Sur la base de cette caractérisation, Chiba et al. Présenter un O (n) algorithme récursif temporel, qui peut être décrit comme suit: supposons qu'une face externe F du graphe planaire biconnecté G a été choisie, supposons que tous les sommets v ϵ F sont placés, et que les sommets de degré deux sont éliminés, tout en reliant leurs voisins. (Le sommet de degré 2 peut plus tard être placé sur le segment de droite joignant les deux sommets adjacents.) La partie restante de l'algorithme est la suivante: [18]

ConvexDraw **(G)**; { *assume n<= 4, otherwise G is drawn as a triangle* }

Let v be a vertex, which is a corner point of the outerface;

Let $G' = G - \{v\}$;

Let $B_1;::: ; B_p$ *be the blocks of G' ;*

Let for each B_i, v_i *and* v_{i+1} *be two cutvertices of* B_i, *with* $(v; v_i); (v; v_{i+1}) \epsilon E$;

Place the vertices on the outerface of every B_i *on a convex area*

Inside triangle $v; v_i; v_{i+1}$ *such that*

The vertices adjacent to v are corner points of a convex polygon;

The other vertices are on straight-line segments of this polygon;

For each block B_i *do ConvexDraw* (B_i) *rof;*

End ConvexDraw[18]

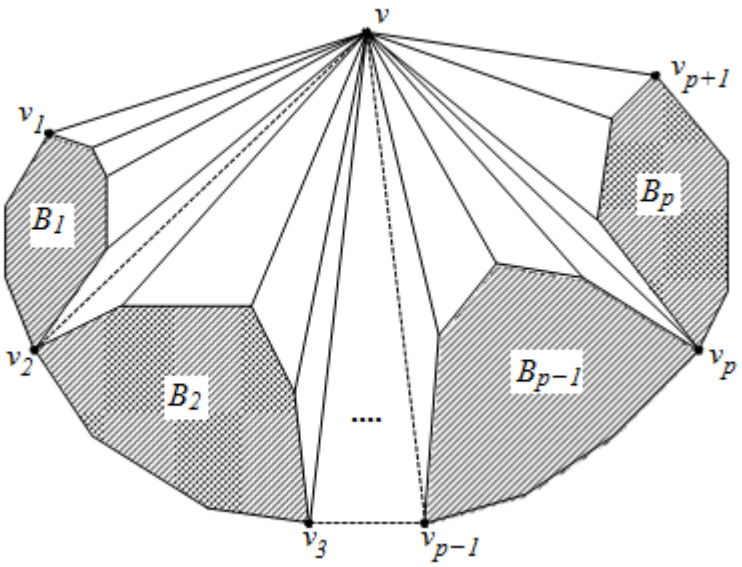


Figure 3.3: dessin de graphes plans convexes[18]

4. Conclusion

Nous avons vu dans ce chapitre le problème test planarité et d'algorithmes dessin graphes planaire.

Dans le chapitre suivant nous présenter application que test planarité

CHAPITRE 4

REALISATION ET EXPERIMENTATIONS

1. Introduction

L'implémentation d'un logiciel vient après un enchainement de plusieurs étapes dans le processus de développement, son but principal est de réaliser un produit capable de résoudre les problèmes posés on utilisant des outils et des algorithmes.

Dans ce dernier chapitre, nous présentons les différents résultats de simulations que nous avons réalisés.

2. Environnement de développement

Pour la réalisation de ce travail, nous avons eu recours aux environnements suivants :

2.1. Environnement matériel

Pour développer l'application, nous avons utilisé comme environnement matériel :

Système

Processeur :	Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz 2.50 GHz
Mémoire installée (RAM) :	4,00 Go
Type du système :	Système d'exploitation 64 bits, processeur x64
Stylet et fonction tactile :	La fonctionnalité de saisie tactile ou avec un stylet n'est pas disponible sur cet écran.

2.2. Environnement logiciel

- Windows 8.1 comme système d'exploitation
- Le Microsoft Visual studio 2012 comme Environnement de développement facile a utilisé.
- Langage C#
- comme langage de programmation.

2.2.1. Le Microsoft Visual Studio 2012

Figure 4.1 : Visuel studio 2012.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications Web ASP.NET, des services Web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C# et Visual C++ utilisent tous le même environnement de développement intégré (IDE), qui permet le partage d'outils et facilite la création de solutions à plusieurs langages. Par ailleurs, ces langages utilisent les fonctionnalités du .NET Framework, qui fournit un accès à des technologies clés simplifiant le développement d'applications Web ASP et de services Web XML [40].

Le langage de programmation C#

C# est un langage de programmation orientée objet, fortement typé, dérivé de C et de C++, ressemblant au langage Java. Il est utilisé pour développer des applications web, ainsi que des applications de bureau, des services web, des commandes, des widgets ou des bibliothèques de classes. En C#, une application est un lot de classes où une des classes comporte une méthode Main, comme cela se fait en Java. [13]

C# est destiné à développer sur la plateforme .NET, une pile technologique créée par Microsoft pour succéder à COM.

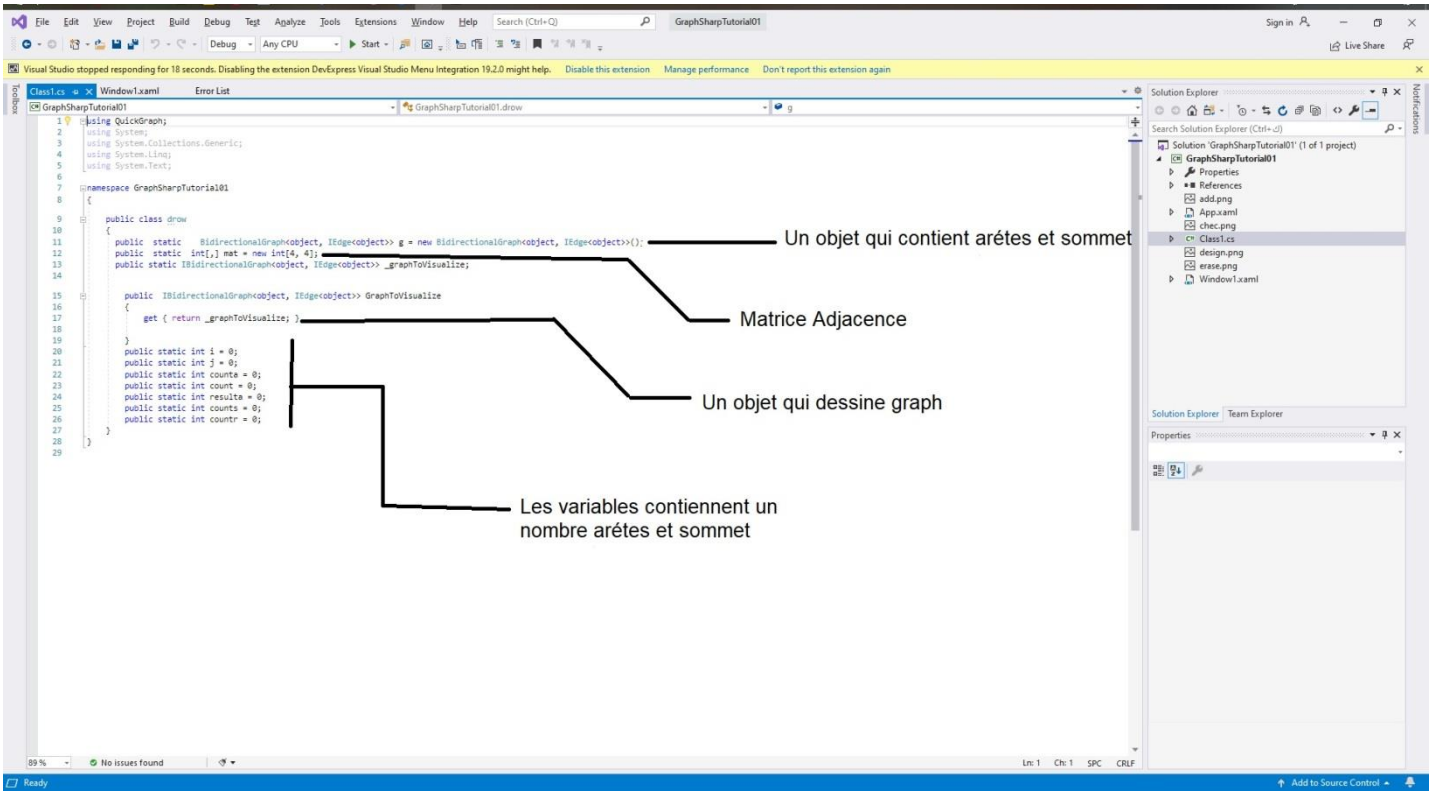
3. PARAMETRES D'APPLICATION

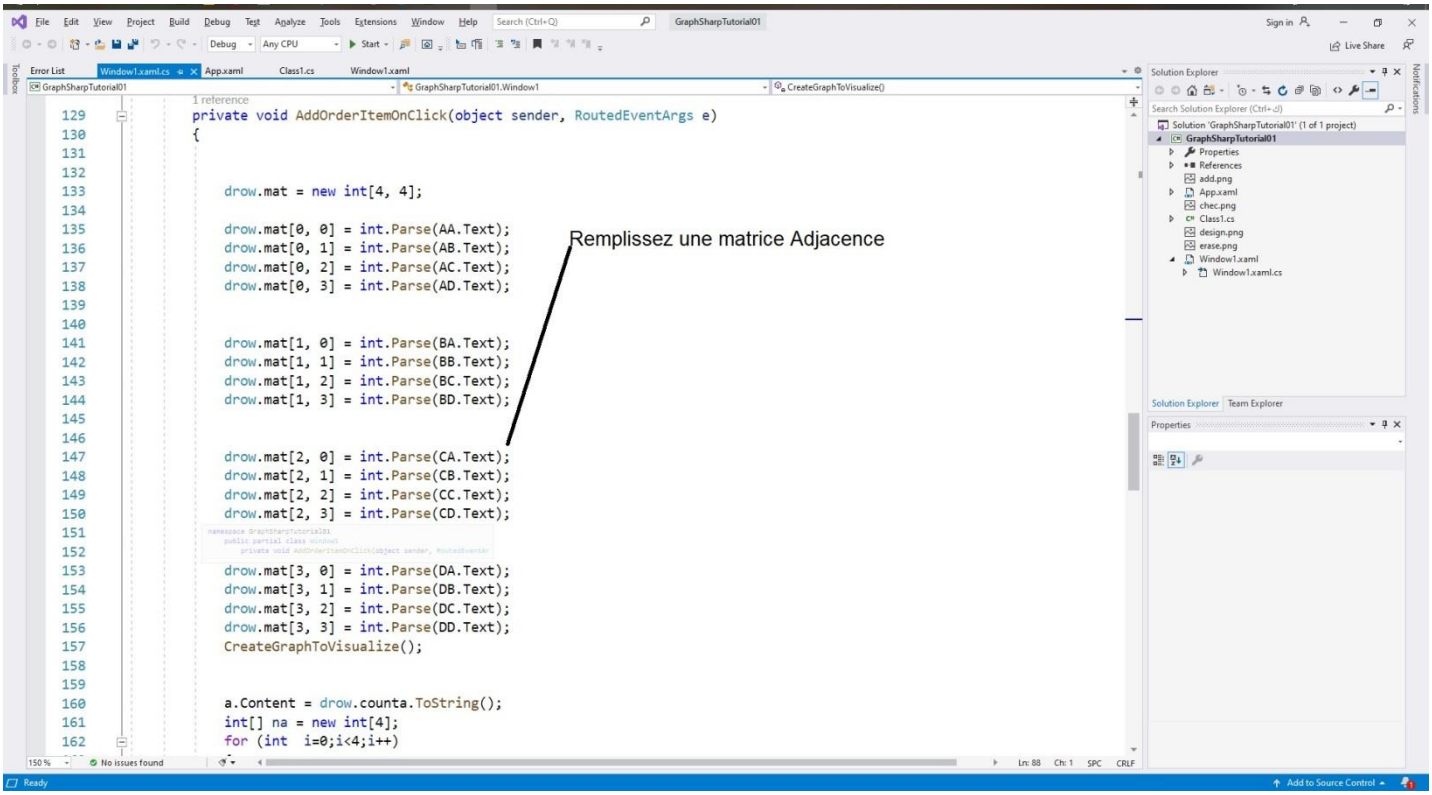
Les différents paramètres d'application, Nous l'utilisons comme éléments suivant :

3.1. Représentation d'un graphe

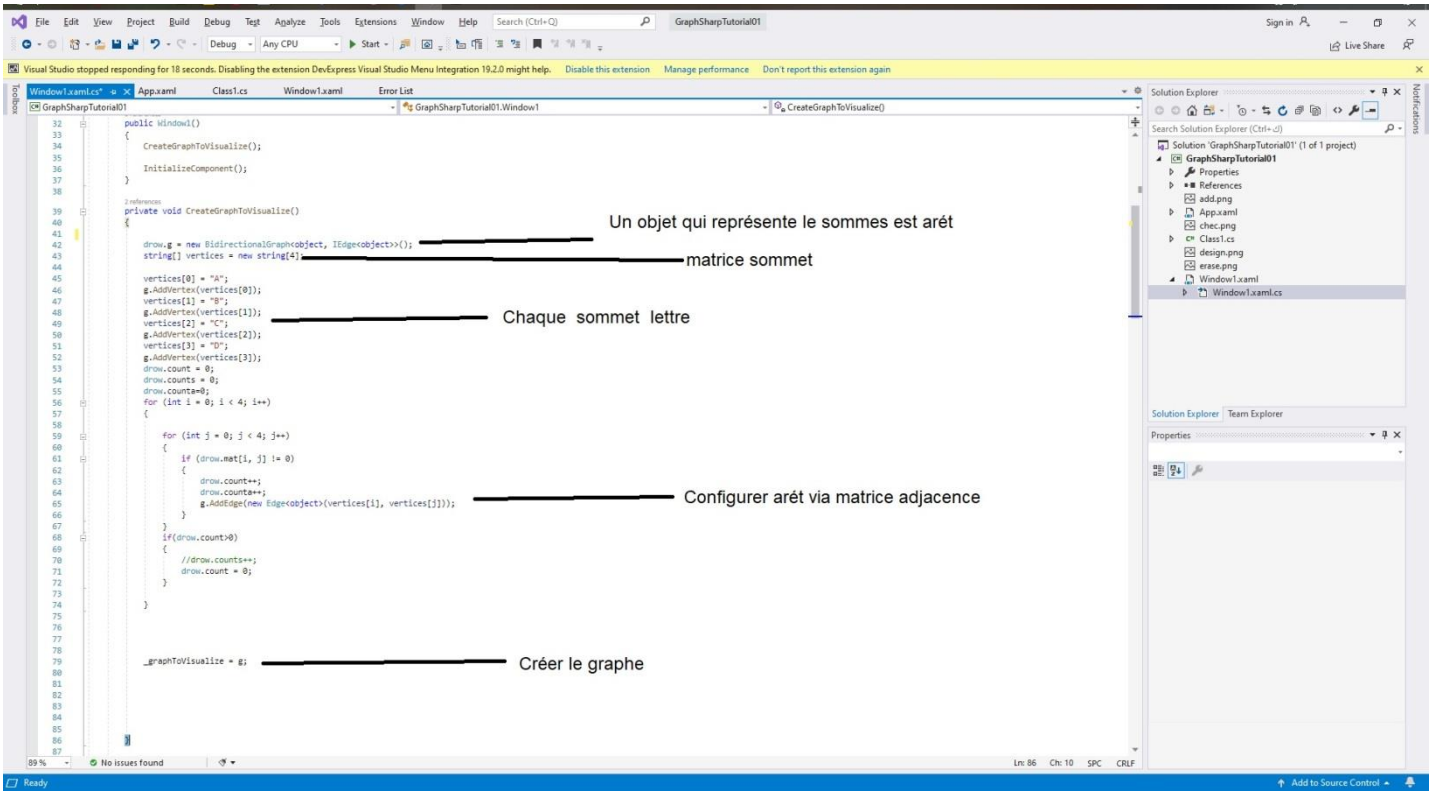
Dans cette partie Nous expliquons le fonctionnement de l'application en matrice adjacence

Et comment dessiner graphes :


Un objet qui contient arétes et sommet
Matrice Adjacence
Un objet qui dessine graph
Les variables contiennent un nombre arétes et sommet

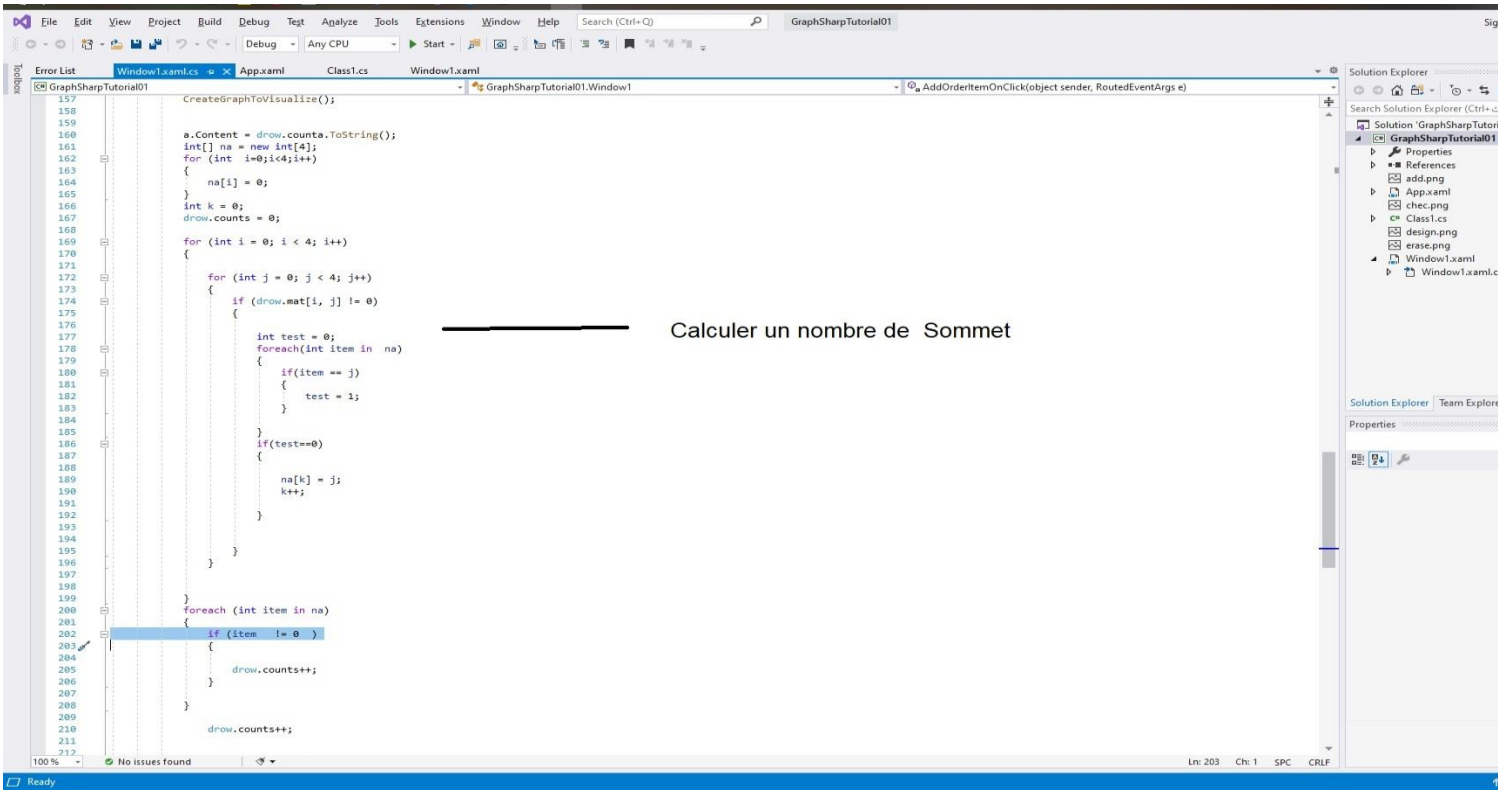



```
1 reference
private void AddOrderItemOnClick(object sender, RoutedEventArgs e)
{


    drow.mat = new int[4, 4];

    drow.mat[0, 0] = int.Parse(AA.Text);
    drow.mat[0, 1] = int.Parse(AB.Text);
    drow.mat[0, 2] = int.Parse(AC.Text);
    drow.mat[0, 3] = int.Parse(AD.Text);


    drow.mat[1, 0] = int.Parse(BA.Text);
    drow.mat[1, 1] = int.Parse(BB.Text);
    drow.mat[1, 2] = int.Parse(BC.Text);
    drow.mat[1, 3] = int.Parse(BD.Text);


    drow.mat[2, 0] = int.Parse(CA.Text);
    drow.mat[2, 1] = int.Parse(CB.Text);
    drow.mat[2, 2] = int.Parse(CC.Text);
    drow.mat[2, 3] = int.Parse(CD.Text);

    drow.mat[3, 0] = int.Parse(DA.Text);
    drow.mat[3, 1] = int.Parse(DB.Text);
    drow.mat[3, 2] = int.Parse(DC.Text);
    drow.mat[3, 3] = int.Parse(DD.Text);
    CreateGraphToVisualize();


    a.Content = drow.counta.ToString();
    int[] na = new int[4];
    for (int  i=0;i<4;i++)
```



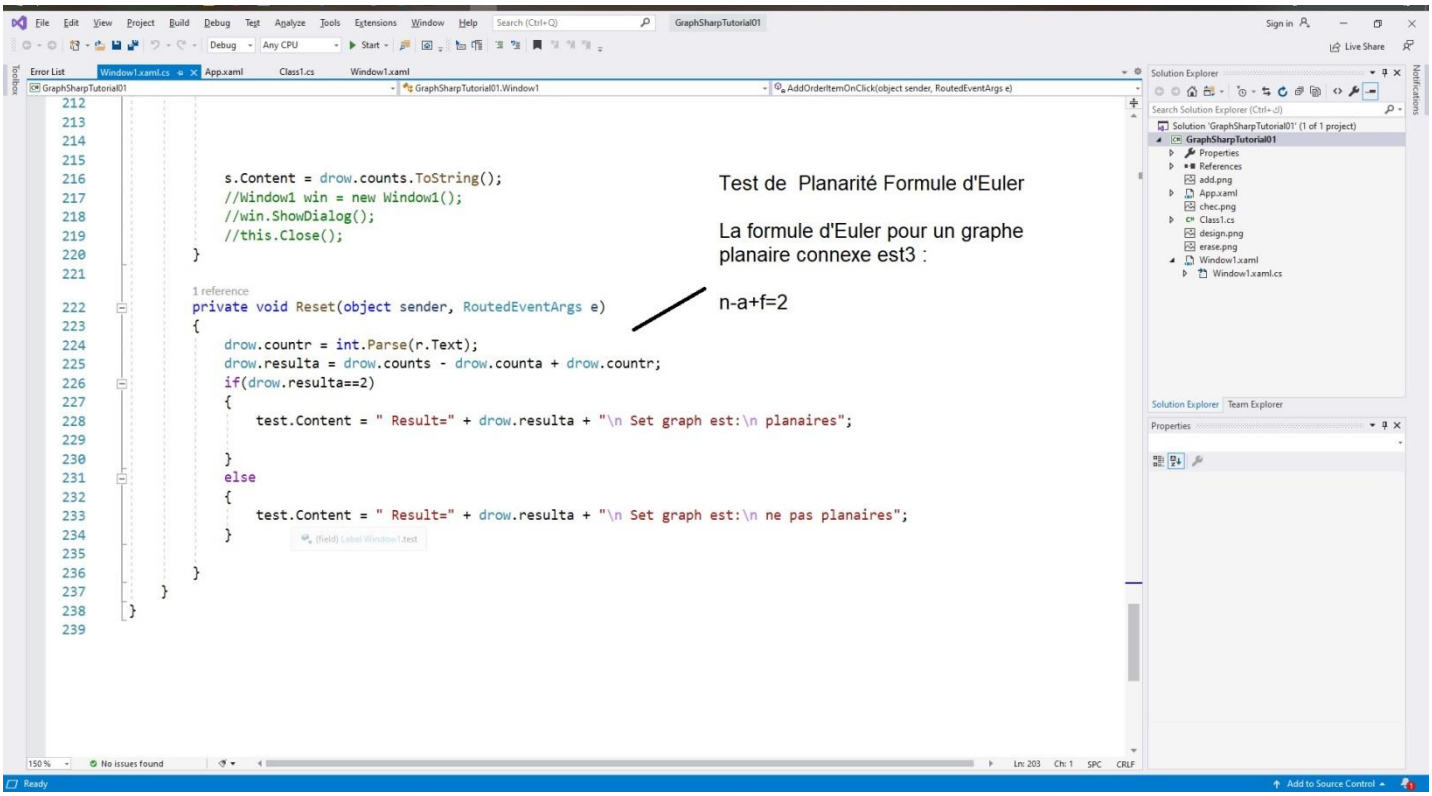
3.2. Calculer Nombre des arêtes et sommet :

Dans cette partie fonction que Calculer Nombre des arêtes et sommet :



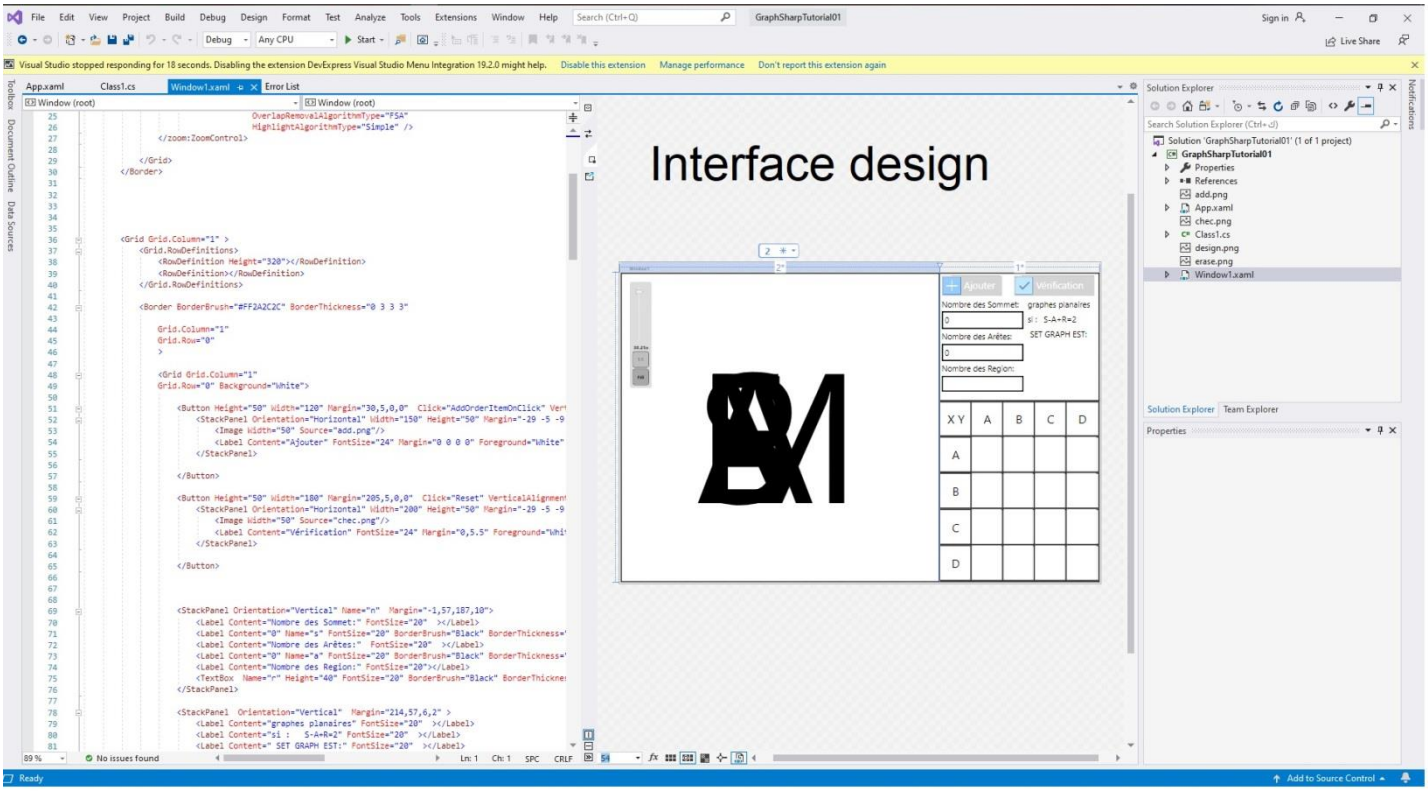
3.3. Test de planarité (formule Euler)

Dans cette partie Nous présentons fonction Utilisé théorème formule Euler (n-a+f=2) :



3.4. Interface de désigne

Dans cette partie Nous présentons Interface de cette application :



.

4. Résultats obtenus

Dans cette partie, nous allons représenter un ensemble des résultats obtenus par les tests sur notre problème.

4.1. Premier exemple

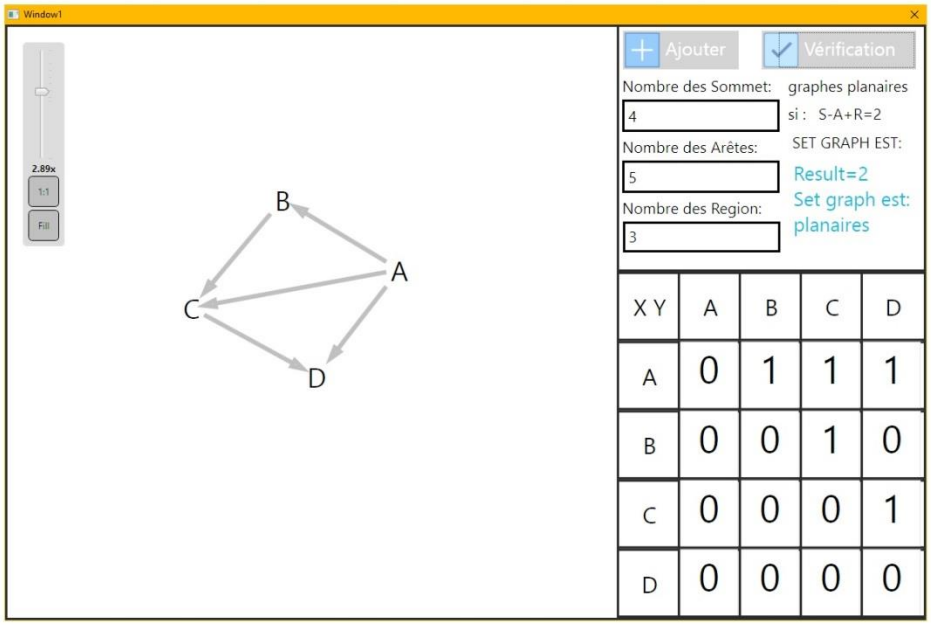
Dans le premier exemple, nous considérons matrice adjacence suivant :

M=

0	1	1	1
0	0	1	0
0	0	0	1
0	0	0	0

4.1.1. Tests et résultats

Nous avons testé l'application par matrice adjacence M, Et nous obtenons les résultats suivants :



Résultats :

s=Nombre de sommet =4

a=Nombre d'arête =5

f=Nombre de région (Observation)=3

Formule Euler :

s-a+f= 2➔ **graphe planaire**

4-5+3= 2 ➔ **graphe M est graph planaire**

4.2. Deuxième exemple

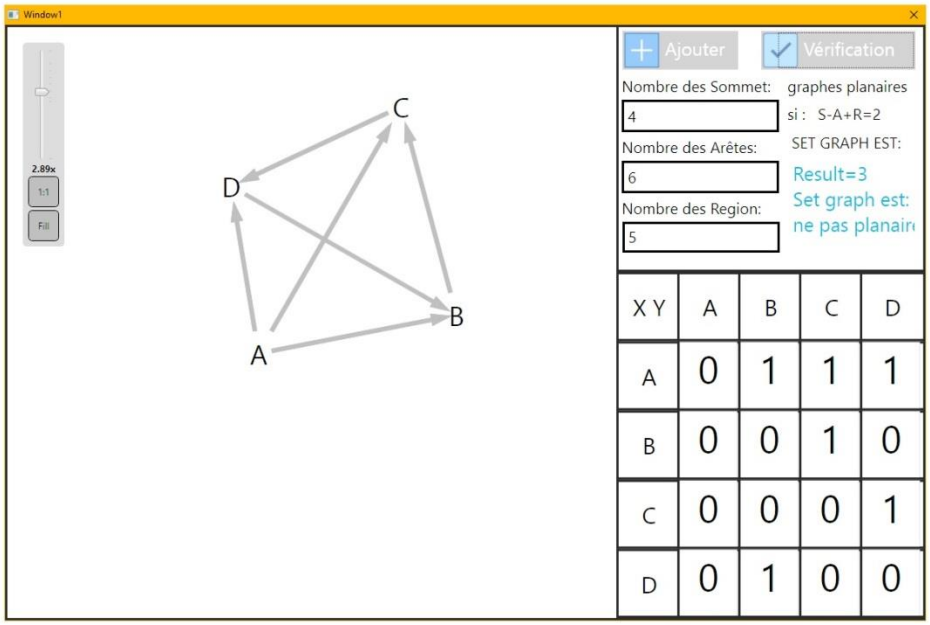
Dans le deuxième exemple, nous considérons matrice adjacence N suivant :

N=

0	1	1	1
0	0	1	0
0	0	0	1
0	1	0	0

4.2.1. Tests et résultats

Nous avons testé l'application par matrice adjacence M, Et nous obtenons les résultats suivants :



X Y	A	B	C	D
A	0	1	1	1
B	0	0	1	0
C	0	0	0	1
D	0	1	0	0

Résultats :

s=Nombre de sommet =4

a=Nombre d'arête =6

f=Nombre de région (Observation)=5

Formule Euler :

s-a+f= 2➔ **graphe planaire**

4-6+5= 3 ➔ **graphe N est graph non planaire**

5. Conclusion

Dans ce chapitre, en premier temps nous avons présenté les environnements matériel et logiciel et les données utilisés, ensuite de test les résultats obtenues sur des exemples de graphes, et en fin nous avons présenté et étudier.

CONCLUSION GENERALE

CONCLUSION GENERALE

Le besoin d'utiliser dessin graphes planaire. Afin de dessiner un graphe clair, beau et organisé il est utilisé dans les graphes plusieurs nombre de croisement et pas claire et désorganisé. Est utilisé dans de nombreux domaines dans l'analyse de réseaux sociaux et la cartographie, bio-informatique.

Dans ce travail, nous avons étudié un problème dessin un graph dans plan. Au cours de ce mémoire, nous avons commencé par la définition générale sur la théorie des graphes. En deuxième temps, nous avons passé en revue les graphes planaire on générale et la définition de la notion de complexité des algorithmes. Ensuite nous avons défini problème test de planarité et algorithmes dessin graphes planaire. Enfin, nous avons étudié l'effet de l'ensemble des paramètres d'application sur les résultats de ce problème. Cette application Non utilisé dans tous les graphes. L'objectif est de trouver un outil et algorithme de dessin graph selon différents métriques en particulier ; un minimum de croisement d'arcs possible. .

Nous avons choisi le langage C# qui est très forte et rapides. Nous l'avons mis en œuvre dans différents graphes. Nous avons utilisé la matrice d'adjacence et la matrice de cout pour obtenir la bande passante.

Comme perspective à ce travail, nous voudrions essayer de travailler avec Améliorer et utiliser l'application Aide à dessiner graphes planaire Pour utiliser de réseaux sociaux Et d'autres domaines du sujet.

BIBLIOGRAPHIE

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

[2] [A. Bretto, A. Faisant, F. Hennecart. Eléments de théorie des graphes, Editeur] }SPRINGER, collection Iris, Année 2012.]

[3] site web :[http://perso.usthb.dz/~mboukala/CoursTG.pdf, consulté le 25/02/2017,15:45.]

[4] [Dr. Chahinez BACHTARZI, Théorie des graphes, Support de cours du module TG,]]Université CONSTANTINE 2, Année 2014/2015.]

[5] [Site web: http://www.w3ii.com/fr/graph_theory/graph_theory_introduction.html, consulte le 18/02/2017, 17.02.]

[6] [M. Gondran, M. Minoux, Graphes et algorithmes. Collection EDF R&D.4eme édition revu et augmentée, ISBN : 978-2-7430-1035-5. 2009.]

[7] Site web :[http://www.mathraining.be/chapters/42?type=10 , consulte le 18/02/2017, 15.00.]

[10][carfier J. et Chertienne P.1988. Problèmes d'ordonnancement. Modélisation/ complexité/ algorithme.1988.]

[12][Joseph Battle, Frank Harary et Yukihiro Kodama, « Every planar graph with nine points has a]nonplanar complement », Bull. Amer. Math. Soc. (N.S.), vol. 68, 1962, p. 569–577]

[13][Balagurusamy, Programming In C#, Tata McGraw-Hill Education - 2008, (ISBN 9780070667570)]

[14] [William T. Tutte, « The non-biplanar character of the complete 9-graph », Canadian Mathematical Bulletin, vol. 6, 1963, p. 319–319]

[15][narsigh deo millican chair Graph Theory with Applications to Engineering and Computer Science UNIVERSITY OF CENTRAL FLORIDA 1974 PAGE 33]

[16] [C. VASUDEV Graph Theory with Applications Indian universities page 2. 27 ,109,137,111.112.113.114]

[17][takao nishizeki md . saidur rahman Planar Graph Drawing academia sinica ,Taiwan page 48 .49]

[18][gossen kant Algorithms for drawing planar graphs 3januari 1967 en Nederland page 115 ,117]

[20] [combinatorial concept and algorithms for drawing planar graphs dr ulrik brands 10 juli 2012 univ Konstanz page 37 39]

[21][F.Havet. A. Perrut Universit´e Claude Bernard Lyon 1 Juillet 2008 Coloration des graphes planaires]

[22][Hassler Whitney, « Non-separable and planar graphs », Transactions of the American Mathematical Society, vol. 34, no 2, 1932, p. 339–362 (DOI 10.1090/S0002-9947-1932-1501641-2).]

[23]
(en) [Wilfried Imrich et Sandi Klavžar (sl), Product Graphs : Structure and recognition, John Wiley & Sons, coll. « Wiley-Interscience Series in Discrete Mathematics and Optimization », 2000.]

[24] Robert Cori, « L'algorithme de test de planarité de R. E. Tarjan », 1024– Bulletin de la société informatique de France, no 4, octobre 2014, p. 55-65

[25][Christopher Auer, Andreas Gleißner, Kathrin Hanauer et Sebastian Vetter, « Testing planarity by switching trains », Graph Drawing: 20th International Symposium, GD 2012, [26]Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers, Berlin, Springer, vol. 7704, 2013, p. 557–558 (DOI 10.1007/978-3-642-36763-2_51).]
[27][Hubert de Fraysseix et Pierre Rosenstiehl, « A depth-first-search characterization of planarity », Graph Theory (Cambridge, 1981), North-Holland, Amsterdam-New York, 1982,
[28]//p. 75–80 (Math Reviews 671906Hubert de Fraysseix et Pierre Rosenstiehl, « A characterization of planar graphs by Trémaux orders », Combinatorica, vol. 5, no 2, 1985, p. 127–135 (DOI 10.1007/BF02579375, Math Reviews 815578).
[29][Hubert de Fraysseix, Patrice Ossona de Mendez et Pierre Rosenstiehl, « Trémaux trees and planarity », International Journal of Foundations of Computer Science, vol. 17, no 5, 2006, p. 1017–1029 (DOI 10.1142/S0129054106004248, Math Reviews 2270949, arXiv math.CO/0610935).]
[30] [Hubert de Fraysseix et Patrice Ossona de Mendez, « Trémaux trees and planarity », European Journal of Combinatorics, vol. 33, no 3, 2012, p. 279–293]

[31][/John Hopcroft et Robert E. Tarjan, « Efficient planarity testing », Journal of the Association for Computing Machinery, vol. 21, no 4, 1974, p. 549–568 (DOI 10.1145/321850.321852.]
[32][Robert Cori, « L'algorithme de test de planarité de R. E. Tarjan », 1024– Bulletin de la société informatique de France, no 4, octobre 2014, p. 55-65 (ISSN 2270-1419, lire en ligne [archive], consulté le 29 juin 2020).]

[33][Kurt Mehlhorn et Petra Mutzel, « On the Embedding Phase of the Hopcroft and Tarjan Planarity Testing Algorithm », Algorithmica, vol. 16, no 2, 1996, p. 233–242 (DOI 10.1007/bf01940648)]
[34][Kurt Mehlhorn et Stefan Näher, « LEDA: A library of efficient data types and algorithms », Communications of the ACM, vol. 38, no 1, 1995, p. 96–102 (DOI 10.1145/204865.204889)]
[35][(Martyn G. Taylor, Planarity Testing by Path Addition, University of Kent, 2012 (lire en ligne [archive]) — Thèse de Ph. D.]
[36][Abraham Lempel, Shimon Even et I. Cederbaum, « An algorithm for planarity testing of graphs », dans Pierre Rosenstiehl, Theory of Graphs, New York, Gordon and Breach, 1967, p. 215–232.]
[37][Shimon Even et Robert E. Tarjan, « Computing an st-numbering », Theoretical Computer Science, vol. 2, no 3, 1976, p. 339–344 (DOI 10.1016/0304-3975(76)90086-4).]
[38][Kellogg S. Booth et George S. Lueker, « Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms », J. Comput. Syst. Sci., vol. 13, no 3, 1976, p. 335-379.]
[39][Boyer et Myrvold 2004, p. 243: “Its implementation in LEDA is slower than LEDA implementations of many other O(n)-time planarity algorithms.”]
[40][Norishige Chiba, Takao Nishizeki, Shigenobu Abe et Takao Ozawa, « A linear algorithm for embedding planar graphs using PQ–trees », Journal of Computer and System Sciences, vol. 30, no 1, 1985, p. 54–76 (DOI 10.1016/0022-0000(85)90004-2).]
[41][Wei-Kuan Shih Shih et Wen-Lian Hsu, « A new planarity test », Theoretical Computer Science, vol. 223, nos 1–2, 1999, p. 179–191 (DOI 10.1016/S0304-3975(98)00120-0).]
[42][John M. Boyer et Wendy J. Myrvold, « On the cutting edge: simplified O(n) planarity by edge addition », Journal of Graph Algorithms and Applications, vol. 8, no 3, 2004, p. 241–273 (DOI 10.7155/jgaa.00091, lire en ligne [archive].
[43][Hubert de Fraysseix de Fraysseix, Patrice Ossona de Mendez et Pierre Rosenstiehl, « Trémaux Trees and Planarity », International Journal of Foundations of Computer Science, vol. 17, no 5, 2006, p. 1017–1030 (DOI 10.1142/S0129054106004248, arXiv math/0610935).]
[44][Markus Chimani, Petra Mutzel et Jens M. Schmidt, « Efficient extraction of multiple Kuratowski subdivisions », Symposium on Graph Drawing (GD'07), Sydney, Australia, Springer-Verlag, 2008, p. 159–170.]
[45][Bernhard Haeupler et Robert E. Tarjan, « Planarity Algorithms via PQ-Trees (Extended Abstract) », Electronic Notes in Discrete Mathematics, vol. 31, 2008, p. 143–149 (DOI 10.1016/j.endm.2008.06.029)]
[46][S. G. Williamson, « Depth First Search and Kuratowski Subgraphs », Journal of the ACM, vol. 31, no 4, 1984, p. 681–693 (DOI 10.1145/1634.322451).]

[47][Jens M. Schmidt, « The Mondshein Sequence », Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP'14), 2014, p. 967–978 (ISBN 978-3-662-43947-0, DOI 10.1007/978-3-662-43948-7_80).]

[48] [(la) Leonhard Euler - Solutio problematis ad geometriam situs pertinentis [archive], Commentarii academiae scientiarum Petropolitanae 8, 1741, pages 128-140.]
[49][Frank Harary, Graph Theory, Addison-Wesley, 1969 (SUDOC 014230593]

Résumé

Le travail de ce mémoire concerne l'étude du problème du Tracer graph dans plan qui a de nombreuses algorithmes importantes et diverses dans différents domaines. Il est un problème de type NP-complet. Nous avons étudié deux algorithmes Dans ce problème.

Nous avons appliqué un logiciel pour Représentation d'un graphe (Matrice d'adjacence) et Test de planarité Dans ce graphe, Nous utilisons formule Euler .L'efficacité de cet logiciel a été testée sur quelques exemples.

Mots clés : NP-complet, test de planarité, bio-informatique, Dessins convexe, Ordre canonique

---

Abstract

The work of this dissertation concerns the study of the problem of the Tracer graph in plane which has many important and diverse algorithms in different fields. It is an NP-complete type problem. We have studied two algorithms in this problem.

We applied application for Representation of a graph (Adjacency matrix) and Planarity test in this graph; we use Euler formula the effectiveness of this application has been tested on a few examples.

Keywords: NP-complete, planarity test, bioinformatics, convex draw, Canonical order.

ملخص

يتعلق عمل هذه مذكرة بدراسة مشكلة الرسم البياني التتبعي في المستوى الذي يحتوي على العديد من الخوارزميات الهامة والمتنوعة في مختلف المجالات لقد د رسنا خوارزميتين في هذه المشكلة.

لقد استخدمنا برنامج لتمثيل الرسم البياني (مصفوفة الجوار) واختبار الاستواء في هذا الرسم البياني ؛ نستخدم صيغة أويلر ، وقد تم اختبار فعالية هذا التطبيق في بعض الأمثلة .

الكلمات المفتاحية

اختبار السوية ، المعلوماتية الحيوية ، الرسم المحدب ، الترتيب الكنسي. كامل -NP.