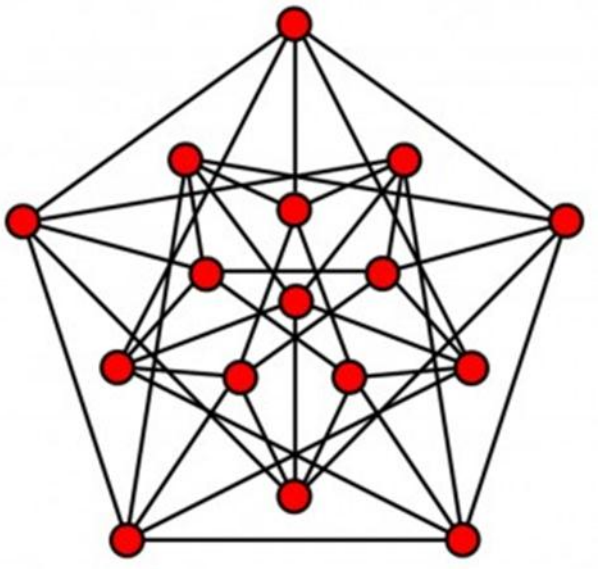
FACULTÉ DES SCIENCES DHAR EL MAHRAZ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

2018/2019

Travail Pratique 3 : Les Graphes



Enseignant :

Pr. MOHAMED MEKNASSI

Réalisé par le groupe 6:

EL BAGHDADI MOHAMED

BERRAG AYOUB

SOMMAIRE

I. Introduction

On désire dans ce TP effectuer une étude sure les graphes afin d'arriver à les implémenter sous le langage c dans un fichier. Par suite nous allons effectuer quelques opérations sous un graphe stocké dans un fichier, comme la somme des poids des arrêtes et extraire les nœuds qui composent ce graphe.

II. Les graphes

1. Définition

Un graphe est un ensemble de points nommés nœuds (parfois sommets ou cellules) reliés par des traits (segments) ou flèches nommées arêtes (ou liens ou arcs). L'ensemble des arêtes entre nœuds forme une figure similaire à un réseau. Différents types de réseaux sont étudiés suivant leur genre de forme (ou topologie) et propriétés ; les arbres sont une sous-catégorie plus simple de graphes particulièrement importante et qui est très étudiée, notamment en informatique.

2. Les types d'un graphe

Les arêtes peuvent être orientées (flèches) ou non orientées (traits). Si les arêtes sont orientées, la relation va dans un seul sens et est donc asymétrique, et le graphe lui-même est dit orienté. Sinon, si les arêtes sont non orientées, la relation va dans les deux sens et est symétrique, et le graphe est dit non orienté.

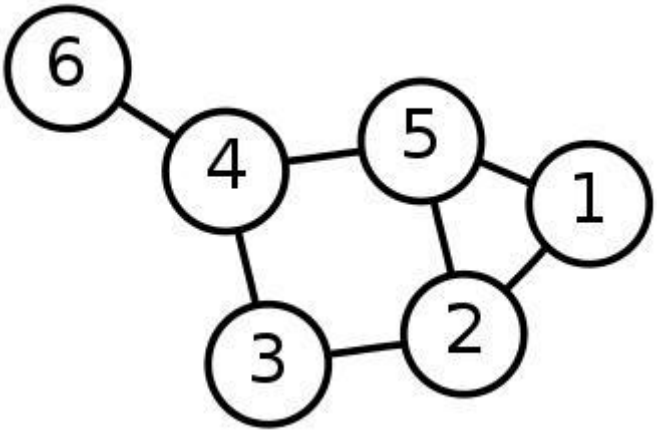


Figure 1: exemple de graphe non orienté

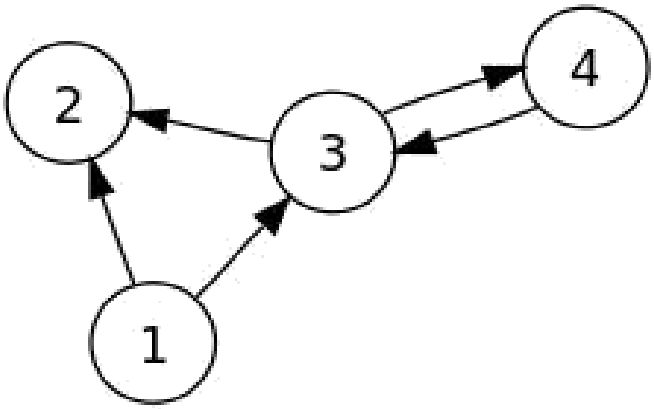


Figure 2: exemple de graphe orienté

3. Hypergraphes

Les hypergraphes généralisent la notion de graphe non orienté dans le sens où les arêtes ne relient plus un ou deux sommets, mais un nombre quelconque de sommets (compris entre un et le nombre de sommets de l'hypergraphe).

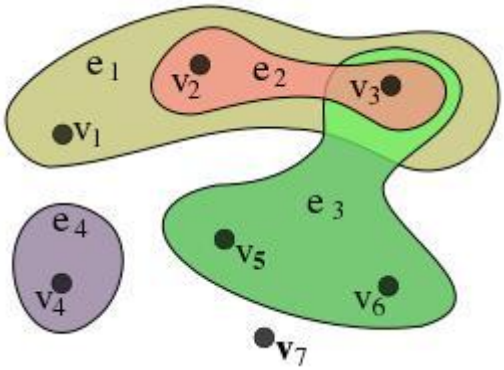


Figure 3: exemple d'hypergraphe

4. Graphe complet

Un graphe complet est un graphe simple dont tous les sommets sont adjacents, c'est-à-dire que tout couple de sommets disjoints est relié par une arête. Si le graphe est orienté, on dit qu'il est complet si chaque paire de sommets est reliée par exactement deux arcs (un dans chaque sens).

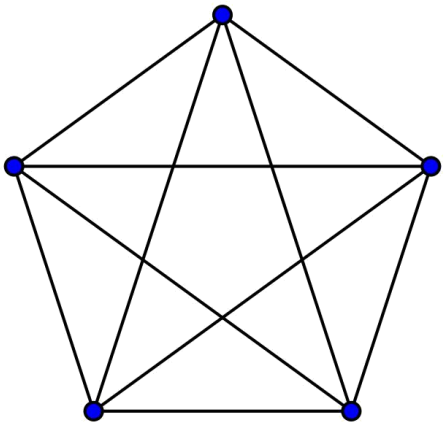


Figure 4: exemple de graphe complet

5. Domaines d'utilisation

Les graphes sont utilisés dans plusieurs contextes, on cite par exemple :

- les réseaux de télécommunications (internet, téléphonie, . . .),
- les circuits électriques,
- les hiérarchies de fichiers informatiques,
- les bases de données relationnelles,
- le stockage de données,
- le flux de contrôle dans un programme,
- le codage,
- les multiples relations entre personnes d'un même groupe,
- la séquence ARN (biologie),
- les différentes interactions dans un écosystème (écologie),

III. L'algorithme de Dijkstra

1. Définition

L'algorithme de Dijkstra sert à résoudre le problème du plus court chemin. Il permet, par exemple, de déterminer un plus court chemin pour se rendre d'une ville à une autre connaissant le réseau routier d'une région. Plus précisément, il calcule des plus courts chemins à partir d'une source dans un graphe orienté pondéré par des réels positifs. On peut aussi l'utiliser pour calculer un plus court chemin entre un sommet de départ et un sommet d'arrivée.

L'algorithme porte le nom de son inventeur, l'informaticien néerlandais Edsger Dijkstra, et a été publié en 1959.

2. Complexité

Cet algorithme est de complexité polynomiale. Plus précisément, pour n nœuds et a arcs, le temps est en ((n+a) log n),

3. L'algorithme

Entrées : $G = (S, A)$ un graphe avec une pondération positive $poids$ des arcs, s_{deb} un sommet de S

$P := \emptyset$
$d[a] := +\infty$ pour chaque sommet a
$d(s_{deb}) = 0$
Tant qu'il existe un sommet hors de P
    Choisir un sommet a hors de P de plus petite distance $d[a]$
    Mettre a dans P
    Pour chaque sommet b hors de P voisin de a
        $d[b] = \min(d[b], d[a] + poids(a, b))$
    Fin Pour
Fin Tant Que

Figure 5: algorithme de Dijikstra

IV. Insérer un graphe dans un fichier

Pour insérer un graphe dans un fichier, on utilise un code en c qui contient les fonctions suivantes :

- Ajouter_graphe () qui permet d'ajouter un nœud, une arrête et un autre nœud s'il existe.
- Somme_arrete () qui permet de calculer la somme des poids des arrêtes.
- Afficher_graphe () qui permet d'afficher les différents composantes du graphe.

Exemple :

On va utiliser le programme pour stocker le graphe ci-dessus, les résultats sont comme suit :

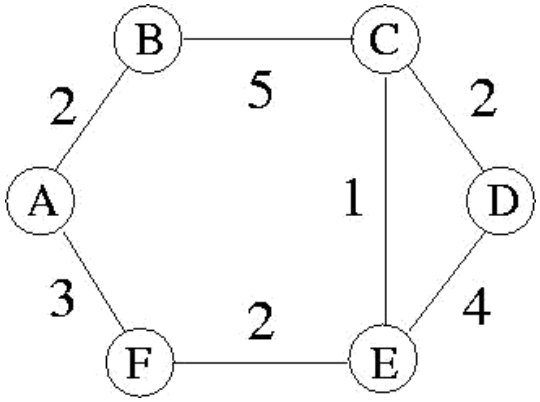


Figure 6: le graphe utilisé

```
Donner les elements  de Graphe
Noeud : A
Le noeud suivant : B
Le poid de la arrete : 2
Continue(1/0)
1
Noeud : B
Le noeud suivant : C
Le poid de la arrete : 5
Continue(1/0)
1
Noeud : C
Le noeud suivant : D
Le poid de la arrete : 2
Continue(1/0)
1
Noeud : D
Le noeud suivant : E
Le poid de la arrete : 4
Continue(1/0)
1
Noeud : E
Le noeud suivant : C
Le poid de la arrete : 1
Continue(1/0)
1
Noeud : F
Le noeud suivant : E
Le poid de la arrete : 2
Continue(1/0)
1
Noeud : A
Le noeud suivant : F
Le poid de la arrete : 3
```

Figure 7: insertion du graphe

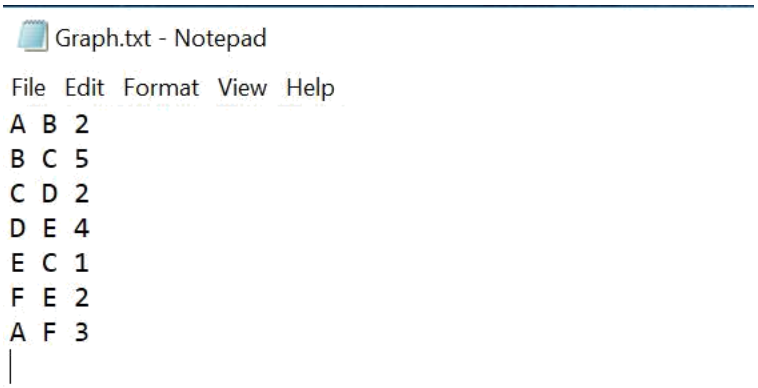


Figure 8: le graphe stocké dans le fichier

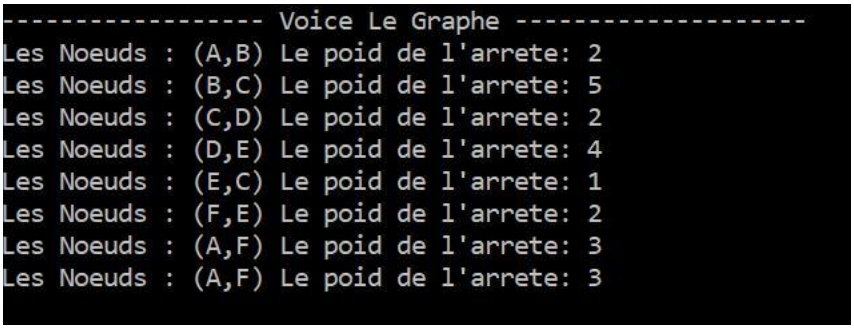


Figure 9: affichage du graphe

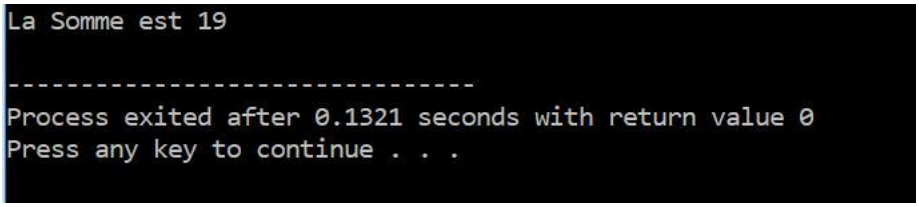


Figure 10: la Somme des poids des arrêtes

V. Annexe

```
#include<stdio.h>
#include<string.h>

typedef struct Noeud{
        char ned[1];
        int poid;
        char ned_suivant[1];
};
FILE *fd;
int nbr;

void ajouter_graphe(){
        int i=1;
        struct Noeud n;
        fd=fopen("Graph.txt","a+");
        printf("Donner les elements de Graphe \n");
        while (i!=0){
                printf("Noeud : ");
                scanf("%s",&n.ned);
                printf("Le noeud suivant : ");
                scanf("%s",&n.ned_suivant);
                printf("Le poid de la arrete : ");
                scanf("%d",&n.poid);
                fprintf(fd,"%s %s %d\n",n.ned,n.ned_suivant,n.poid);
                printf("Continue(1/0)\n");
                scanf("%d",&i);
                nbr++;
        }
        fclose(fd);
}
void affichage_noeud(){
        struct Noeud n;
        fd=fopen("Graph.txt","r");
        printf("------------------ Voice Le Graphe ----------------------\n");
        do{
                fscanf(fd,"%s %s %d",&n.ned,&n.ned_suivant,&n.poid);
                printf("Les Noeuds : (%s,%s) Le poid de l'arrete:
        %d\n",n.ned,n.ned_suivant,n.poid); }while(!feof(fd));
}
void somme_arrete(){
        struct Noeud n;
        int nbr=0;
        fd=fopen("Graph.txt","r");
        printf("------------------ La somme des arretes  ----------------------\n");
        while(!feof(fd)){
```

```
                fscanf(fd,"%s %s
                %d",&n.ned,&n.ned_suivant,&n.poid); nbr+=n.poid;
        }
        printf("La Somme est %d\n",nbr);
        }


int main(){
        int i;
        printf("1-Creation du graphe\n");
        printf("2-Affichage du graphe\n");
        printf("3-Somme des Poids des arretes\n");
        printf("Choisir L'operation\n");
        scanf("%d",&i);
        switch(i){
                case 1:ajouter_graphe();break;
                case 2:affichage_noeud();break;
                case 3:somme_arrete();break;
        }
}
```