Projet réalisé par:

BENLOULOU Sarra

NEKKAA Yousra

Université Clermont Auvergne

Institut d'informatique

Département d'informatique

Tp noté: Graphe : Problème de flots

Encadré par:

Limouzi Vincent & Hadhbi Youssouf

PLAN :

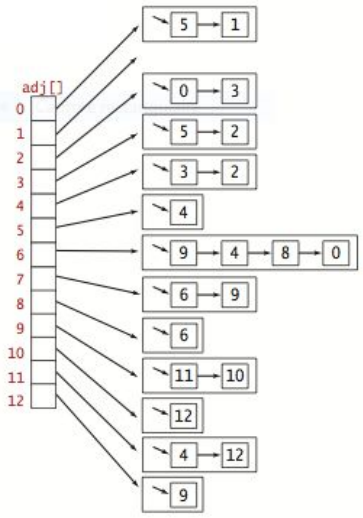
I. **Problème de flot maximum**
   A. **Représentation des Graphe Orientés Utilisé :**

II. **Modélisation du Problème de Flots en Problème de séquence de degré d'un graphe orienté**
   A. **Transformation de la liste des Degrés**
   B. **Création d'un graphe Biparti**
   C. **L'ajout d'une Source et d'un Puit**
   D. **l'attribution des valeurs à l'ensemble des arêtes**
   E. **Chercher le flot maximum grâce à l'algorithme de Edmonds Karp**
   F. **Repérer les liaisons entre les sommets sortants et entrants**
   G. **Dessiner le graphe obtenu par l'algorithme**
   H. **Autres exemples**
   I. **La Preuve que la modélisation proposée est correcte**

I. Problème de flot maximum

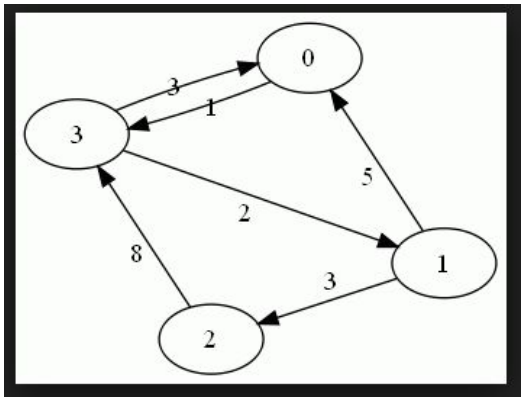
a. Représentation des Graphe Orientés Utilisé :

Au premier abord nous avons utilisé la représentation des listes d'adjacences, où nous maintenons un tableau de listes de sommets indexés par des sommets reliés par une arête à chaque sommet.



Cette représentation nous permet facilement de récupérer les voisins d'un sommet avec un coup de mémoire "faible", or comme nous travaillons avec des graphes valués cette représentation ne suffit pas pour stocker les capacités de chaque arête d'un graphe, c'est ce dont nous avons besoin pour résoudre le problème de flux maximum.
Il a fallu rajouter un autre tableau de taille n (nombre de sommets) pour stocker dedans les capacités de notre graphe.
cela veut dire que nous devons parcourir la liste d'adjacence pour récupérer les voisins puis parcourir le tableau des capacités pour récupérer le poids d'une arête.

Cette représentation devient donc inintéressante et la meilleure solution qui nous ait venu à l'esprit, est la représentation en matrice d'adjacence avec un sacrifice du coup de la mémoire, ***O(n*2)*** emplacements mémoire, mais nous pouvons accéder d'une façon optimale à un sommet et récupérer le poids d'une arête ***O(1)*** .



	0	1	2	3
0	0	0	0	1
1	5	0	3	0
2	0	0	0	8
3	3	2	0	0

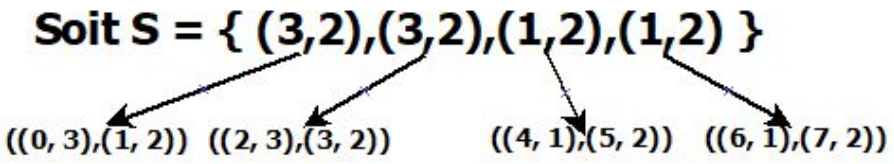
II. Modélisation du Problème de Flots en Problème de séquence de degré d'un graphe orienté

Pour modéliser le problème de séquence de degré en problème de flot maximal, on passe par différentes étapes, dans lesquelles on représente la séquence des degrés en un graphe biparti complet, en lui enlevant un couplage parfait (le **graphe biparti complet** moins son **couplage parfait**), nous appliquerons ensuite, l'algorithme de Edmond Karps pour le calcule du flot maximal à ce graphe. Les chemins augmentants obtenus représentent les arêtes du graphe qui vérifient cette séquence de degrés

a. Transformation de la liste des Degrés

Dans cette étape, nous transformons chaque sommet donné en 2 sommets, un représentant le sommet "sortant" et l'autre "entrant".

les figures suivantes expliquent les procédures suivies:

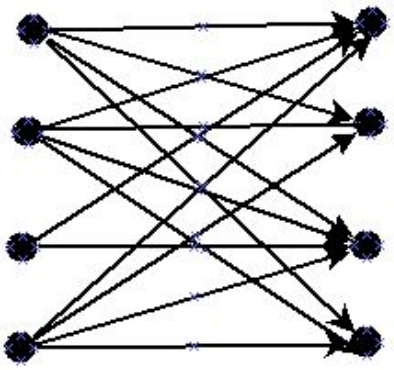


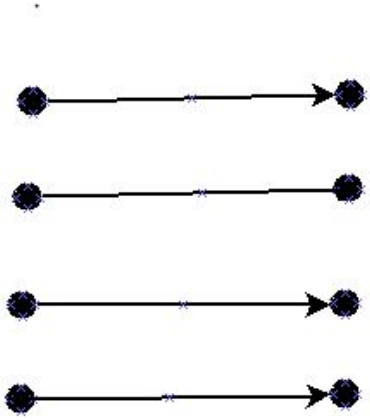
b. Création d'un graphe Biparti

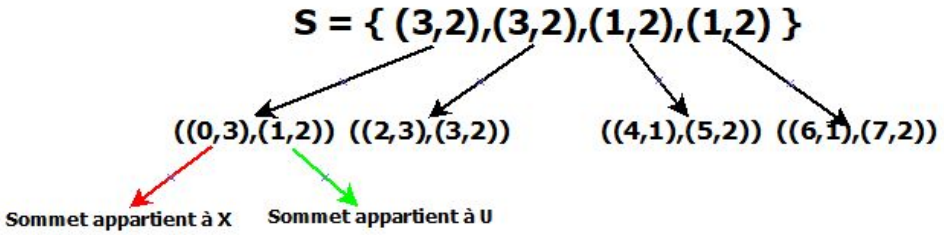
On crée un graphe **biparti complet G** "avec une bipartition V = X ∪ U" des sommets dans lequel on enlève un **couplage parfait.**

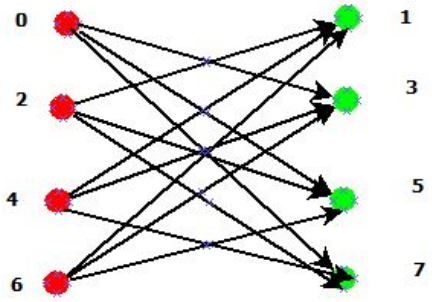
les figures suivantes expliquent les procédures suivies:

Graphe Biparti Complet **Un Couplage Parfait**





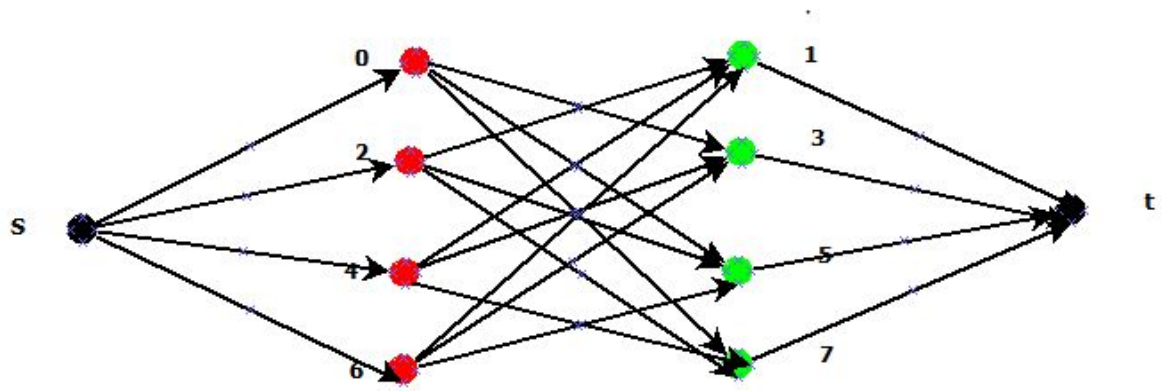




c. L'ajout d'une Source et d'un Puit

Nous ajoutons un sommet source lié à tous les sommets sortants, nous lions également tous les sommets entrants avec le Puit.

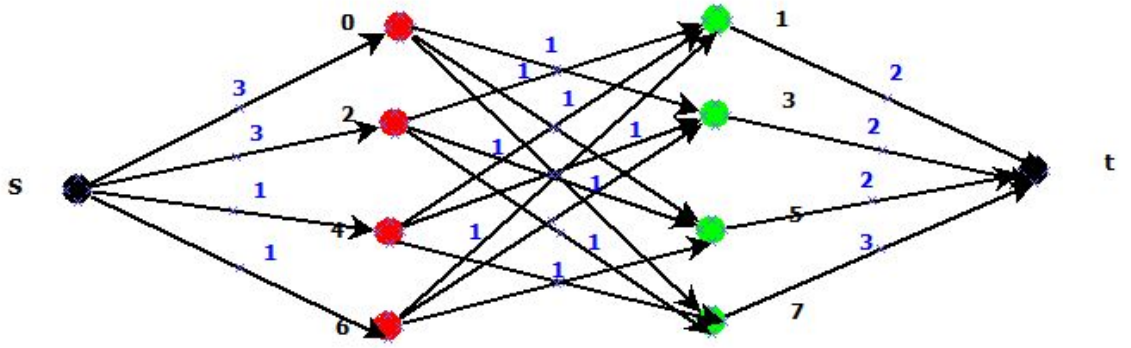
la figure suivante explique la procédure suivie

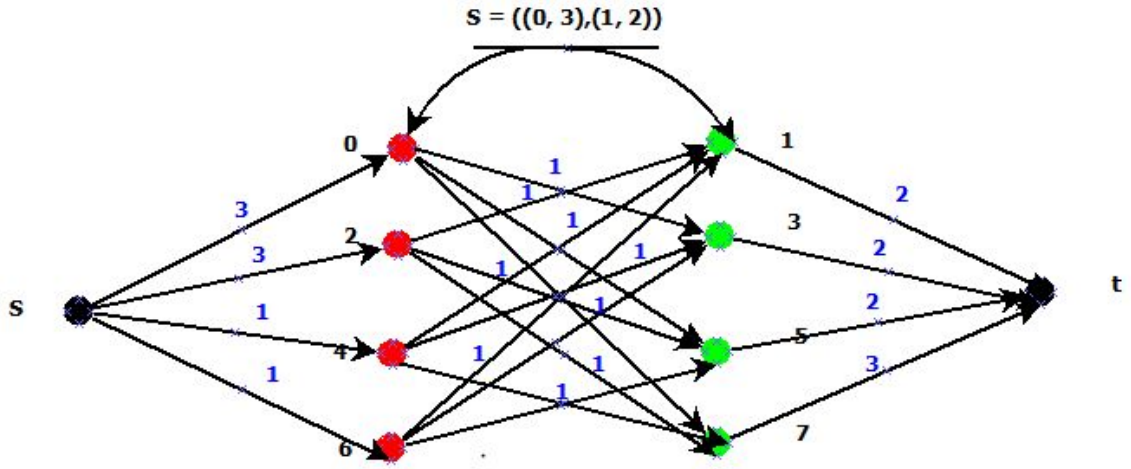


d. l'attribution des valeurs à l'ensemble des arêtes

Pour obtenir un graphe valué

les figures suivantes expliquent les procédures suivies



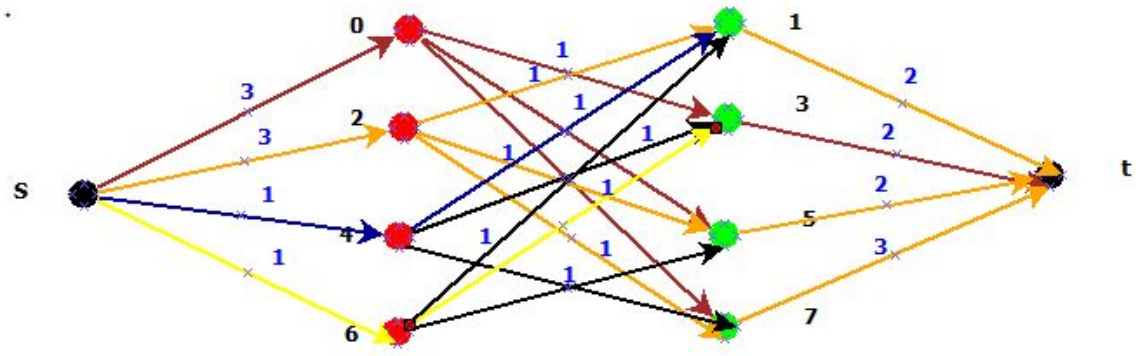


e. Chercher le flot maximum grâce à l'algorithme de Edmonds Karp

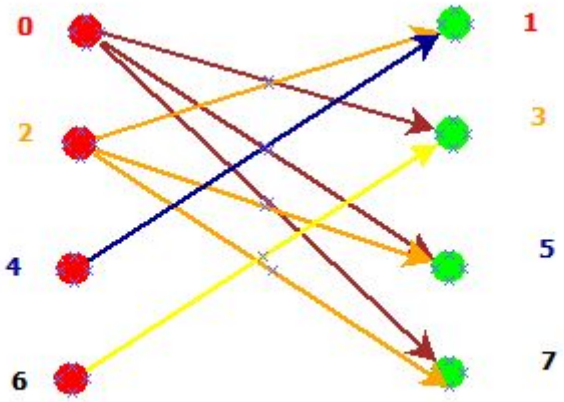
les chemins augmentants obtenus par l'algorithme dans notre exemple :

```
Chemin ====================== Flot Min de ce chemin

[('s', 0), (0, 3), (3, 't')] =====> 1
[('s', 0), (0, 5), (5, 't')] =====> 1
[('s', 0), (0, 7), (7, 't')] =====> 1
[('s', 2), (2, 1), (1, 't')] =====> 1
[('s', 2), (2, 5), (5, 't')] =====> 1
[('s', 2), (2, 7), (7, 't')] =====> 1
[('s', 4), (4, 1), (1, 't')] =====> 1
[('s', 6), (6, 3), (3, 't')] =====> 1
```

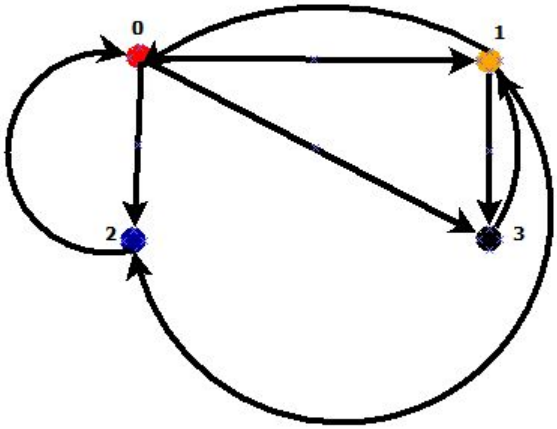


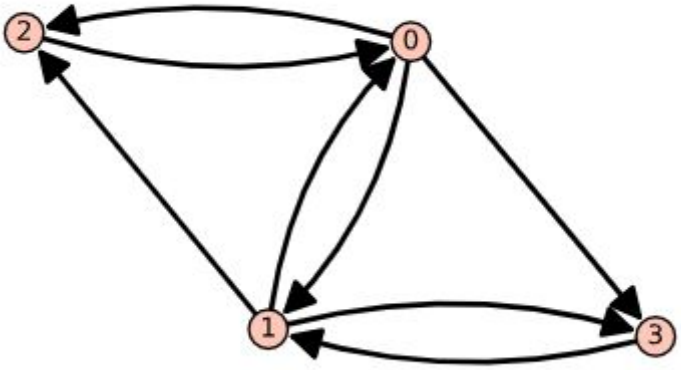
f. Repérer les liaisons entre les sommets sortants et entrants



g. Dessiner le graphe obtenu par l'algorithme :

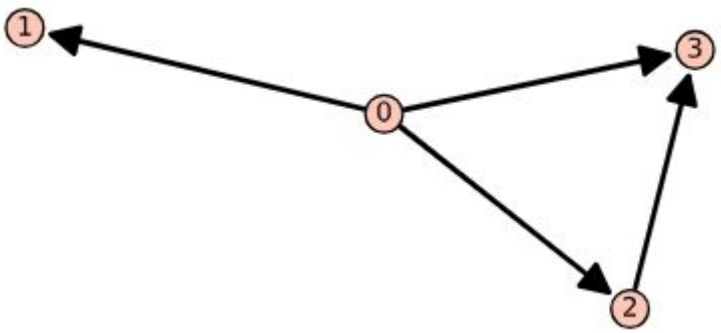
le graphe ci dessous vérifie bien la séquence des degrés donnés en entré



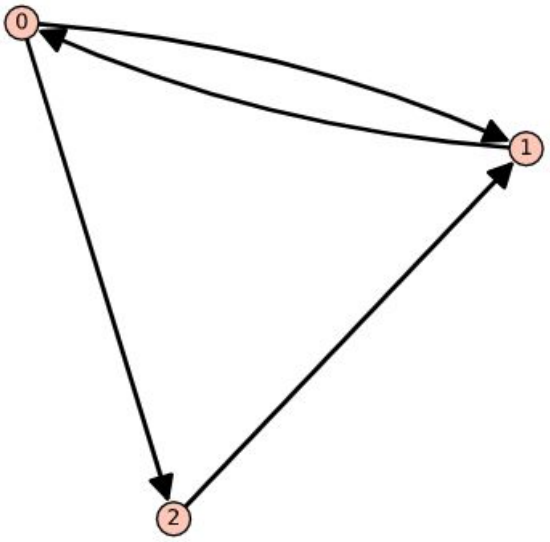


J. Autres exemples

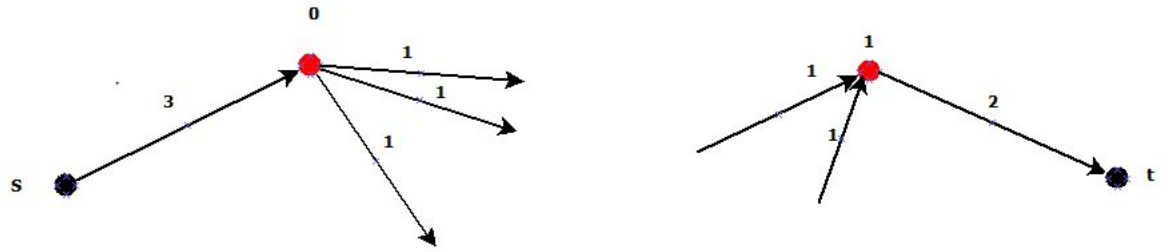
Exemple 1 : Soit la séquence des Degrés suivants : S = [[3, 0], [0, 1], [1, 1], [0, 2]]
le graphe proposé par le programme qui vérifie la liste des degrés est :



Exemple 2 : Soit la séquence des Degrés suivants: S = [[2, 1], [1, 2], [1, 1]]
le graphe proposé par le programme qui vérifie la liste des degrés est :

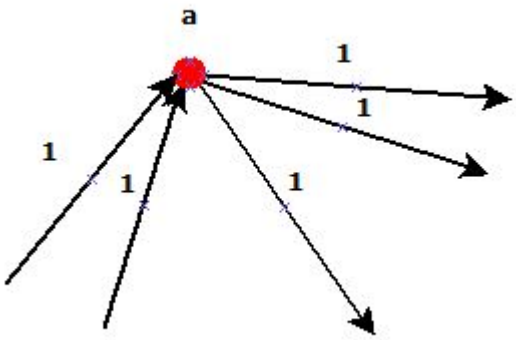


K. La Preuve que la modélisation proposée est correcte



D'après la loi de Kirchhoff – pour chaque sommet interne du réseau (sauf la source et la destination) le flot entrant est égal au sortant.
Le flot entrant dans le sommet 0 est égale à 3, le sortant est également égale à 3, par contre, les capacités des arêtes à droite de 0 sont tous à 1, le flot sortant de 0, sera divisé en 3 chemin , ces 3 chemins représentent la liaison avec 3 autres arêtes, cela représente le degré V(0)+ = 3.

Le flot sortant de 1 vers t est égale à 2, le flot entrant est aussi égale à 2 mais divisé en 2 arêtes, cela indique que ces 2 arêtes sont liées à 1, d'où v(1)- = 2



0 et 1 représentent le même sommet figurant à gauche de la page, nous avons bien V(a)+ = 3 et V(a)- = 2