Institut Francophone International

MASTER I RSC PROMOTION 23

MODULE : RECHERCHE OPÉRATIONNELLE

PROBLEME D'OPTIMISATION : Affectation des tâches En utilisant L'Algorithme de Ford-Fulkerson Dans un Graphe Biparti

---

Rédigé par :
SALU PUATI Emmanuel
KIOMBA KAMBILO Eddy
TSHIBANGU MUABILA Jean

Enseignant *:*
Hoang-Thach Nguyen

Hanoi, Août 2019

Table des matières

1 INTRODUCTION

Il nous a été demandé d'implémenter une solution au problème d'affectation de tâches dans le cadre du module de la Recherche Opérationnelle, afin d'aborder la partie théorique suivie d'une partie pratique réaliser par les étudiants en vue de s'assurer que ces derniers ont bien appréhender la matière. Et le travail sera réalisé par groupe de deux à trois Étudiants se mettant ensemble afin de proposer une solution à un problème donné en se basant sur les contenus du cours.

Ce présent rapport décrit la solution proposée à un problème d'affectation de tâches à l'aide de l'algorithme de Ford-Fulkerson utilisant un BFS (Breadth First Search) dans un graphe Biparti.

2 OBJECTIF

L'objectif principal de ce travail est d'implémenter l'algorithme de Ford-Fulkerson utilisant un BFS 1 avec le langage de programmation java permettant de résoudre le problème d'affectation de tâches au sein d'une organisation souhaitant effectuer au plus une tâche par individu.

3 L'ALGORITHME DE FORD FULKERSON

L'algorithme de Ford-Fulkerson est un algorithme pour le problème du flot maximum, un problème d'optimisation classique dans le domaine de la recherche opérationnelle.

Et ce problème d'optimisation peut être représenté par un graphe comportant une entrée (à gauche) et une sortie (à droite). Le flot représente la circulation de l'entrée vers la sortie d'où l'utilisation de cet algorithme dans les problèmes de réseaux. Les applications sont multiples : problèmes informatiques, routiers, ferroviaires, etc. Il s'applique également à tous les autres problèmes de transferts comme les importations/exportations, les flux migratoires, démographiques mais aussi sur les flux plus abstraits tels que les transferts financiers. Pour les données de très grande taille, il existe plusieurs algorithmes plus performants pour résoudre le même problème connu sous le nom de problème de flot maximum.

Considérant un graphe G(X,U) orienté, valué, connexe, anti- symétrique, sans circuit. A chaque arc u on associe deux scalaires :

$$C_u \ \textit{capacité de l'arc u tel que} \ C_u \geq 0$$
$$\varphi_u \ \textit{le flot circulant sur u tel que} \ \varphi_u \leq C_u$$

A. Principe fondamental

A tout moment, la loi de Kirchhoff doit être vérifiée sur chaque sommet x de G.

$$\sum_{u \in \omega^+} \phi_u = \sum_{u \in \omega^-} \phi_u$$

Le but est d'augmenter le flot jusqu'à son maximum tout en respectant cette règle.

B. Exemple

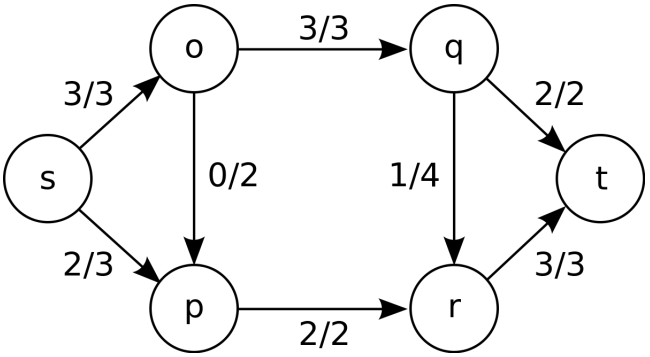

FIGURE 1 – graphe orienté.

Principe général de l'algorithme de Ford-Fulkerson :

- On part d'un flot compatible (généralement 0)
- On utilise deux fonctions alternativement : **Procédure de marquage** ainsi que la **Procédure d'augmentation du flot**.

Procédure de marquage : Le but est de trouver une chaîne améliorante. Son principe est de marquer les sommets selon deux critères : Delta (flot max que l'on peut faire parvenir au sommet), Sommet de provenance.

Procédure d'augmentation du flot : le but est d'augmenter le flot dans le graphe selon la valeur et le marquage obtenu par la procédure de marquage. Son principe est de parcourir le graphe du puit vers la source suivant les indications de provenance de la procédure de marquage.

4 ALGORITHME DE BFS

L'algorithme de parcours en largeur (ou BFS, pour Breadth First Search en anglais), est un algorithme qui permet le parcours d'un graphe ou d'un arbre de la manière suivante : on commence par explorer un nœud source, puis ses successeurs, puis les successeurs non explorés des successeurs, etc. L'algorithme de parcours en largeur permet de calculer les distances de tous les nœuds depuis un nœud source dans un graphe non pondéré (orienté ou non orienté). Il peut aussi servir à déterminer si un graphe non orienté est connexe.

A. Caractéristiques de l'algorithme BFS

L'algorithme de recherche en largeur d'abord parcourt les sommets par "niveaux" dans l'arbre formé par le graphe. Le but est donc ici d'explorer tous les sommets suivants (enfants directs) d'un sommet donné, alors que DFS explore les sommets successeurs (du haut vers le bas, branche par branche). Nous n'avons donc pas de backtracking pour BFS, car nous ne devons jamais considérer le sommet précédent.

B. principe de fonctionnement

Cet algorithme diffère de l'algorithme de parcours en profondeur par le fait que, à partir d'un nœud source S, il liste d'abord les voisins de S pour ensuite les explorer un par un. Ce mode de fonctionnement utilise donc une file dans laquelle il prend le premier sommet et place en dernier ses voisins non encore explorés.

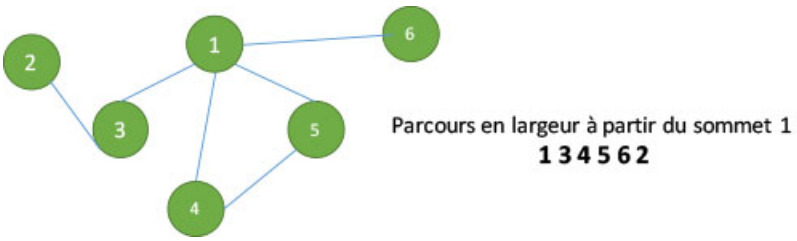

FIGURE 2 – Parcours d'un graphe en largeur

Les nœuds déjà visités sont marqués afin d'éviter qu'un même nœud soit exploré plusieurs fois. Dans le cas particulier d'un arbre, le marquage n'est pas nécessaire.

Étapes de l'algorithme :

- Mettre le nœud source dans la file.
- Retirer le nœud du début de la file pour le traiter.
- Mettre tous les voisins non explorés dans la file (à la fin).
- Si la file n'est pas vide reprendre à l'étape 2.

L'utilisation d'une pile au lieu d'une file transforme l'algorithme du parcours en largeur en l'algorithme de parcours en profondeur.

C. Exemple

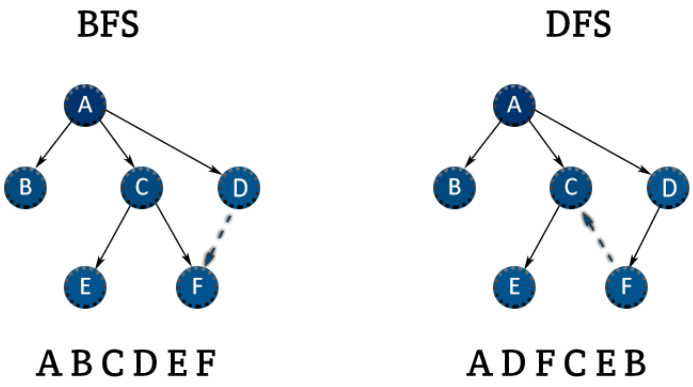

FIGURE 3 – Un exemple de graphe pour les algorithmes DFS et BFS.

nous voyons ici que l'Algorithme de parcours en largeur explore dans l'ordre les sommets A, B, C, D, E, F, contrairement à l'algorithme de parcours en profondeur qui cherche dans cet ordre : A, D, F, C, E, B.

5 IMPLÉMENTATION

A. Modèle

— **Graphe Biparti** : Au démarrage du programme, celui-ci prend paramètre un fichier représentant les liaisons possible entre les éléments de deux parties principales d'un graphe biparti. En théorie de graphe, un graphe est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles et elle que chaque arête ait une extrémité dans et l'autre dans.

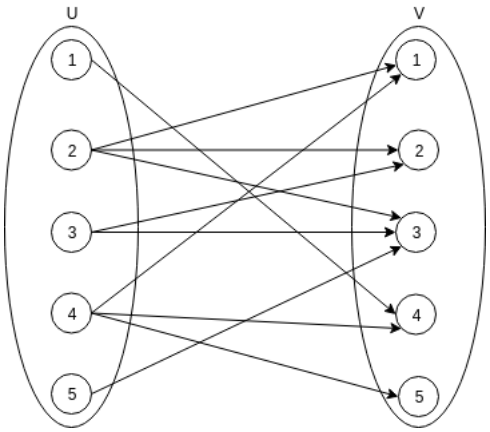

FIGURE 4 – graphe biparti.

Étant donné que ; les nœuds du vecteur U sont considérés comme les successeurs du nœud source S et le nœud puit et le successeur des nœuds du vecteur V, cela n'est pas nécessaire d'être représenté. Le programme considère que, les nombres de nœuds de deux parties sont égaux. La première ligne du fichier donné en paramètre indique le nombre de nœuds d'un vecteur. Le reste de lignes, représente la correspondance de chaque nœud du vecteur U vers un ou plusieurs nœuds du vecteur V.

— Les classes : Noeud.java, Couplage.java, Graphe.java

Ces trois classes sont constituées de différentes méthodes pour la mise en fonctionne de quelques fonctionnalités.

B. Résultat

```
1 5
2 4
3 2 1 3
4 2 3
5 1 5 4
6 3
```

FIGURE 5 – tâches souhaitée par personne.

La figure 5 montre les contenus du fichier data.txt qui liste les tâches souhaité par personne. La première ligne contient un chiffre représentant le nombre tâches qui bien sûr doit être égal au nombre de personne. Dans le cas présent 5 ; soit 5 personne 5 tâches.

Les lignes suivantes contiennent les tâches de préférence de chaque personne ; soit la personne souhaite effectué la tâche 4, la deuxième personne souhaite la tâche 2, 1 et 3 ; etc.

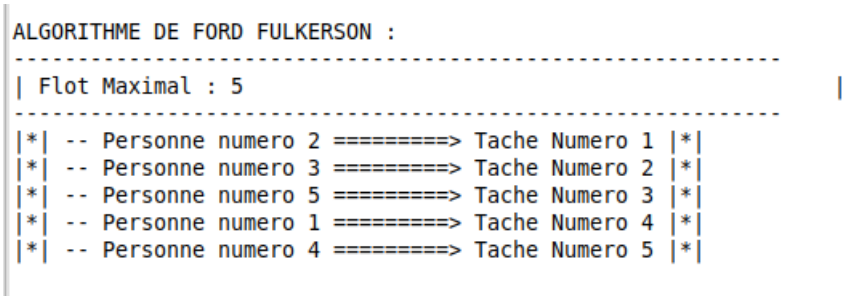

FIGURE 6 – résultat obtenu.

La figure 6, illustre le résultat obtenu avec les données à la figure 5. Remarquons que chaque personne du groupe à obtenu une de ses tâches souhaité. L'objectif de l'algorithme de Ford-Fulkerson étant d'obtenir le flot maximum ; nous pouvons dire que le problème d'affectation de tâches est résolu avec le flot max qui égal à 5.

— Chemin BFS et son inverse

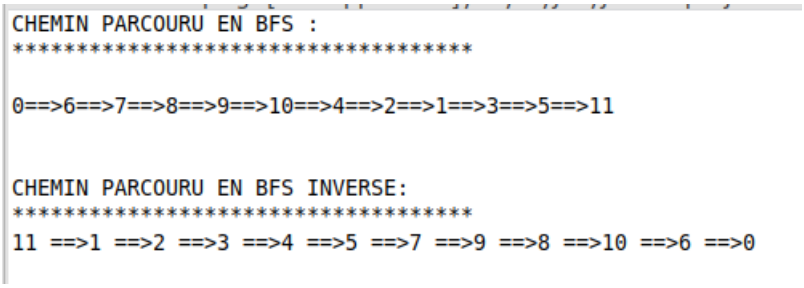

FIGURE 7 – Chemin BFS et son inverse.

La figure 7, le chemin BFS de la source 0 au puit 11.

6 CONCLUSION

En guise de conclusion, nous pouvons dire que nous avons atteint notre objectif que nous nous sommes fixé à savoir la l'implémentation de l'algorithme de Ford-Fulkerson utilisant le BFS pour résoudre le problème d'affectation de tâche au sein d'un groupe de personnes.

Tout au début nous avons commencé par une brève introduction de notre travail en parlant du contexte dans lequel il fait cela à était suivi de l'objectif visé. Dans la deuxième partie et troisième partie nous avons parlé de deux type d'algorithme auxquels nous avons fait recours à savoir l'algorithme de Ford- Fulkeson et l'algorithme de Breadth First Search (BFS) et en fin nous avons montré les résultats obtenus et le modele du présent travail dans la partie implémentation.

Références

Notes de cours (Recherche Opérationnelle), Master1-IFI/P23-RSC, Hoang-Thach Nguyen, 2019
https ://fr.wikipedia.org/wiki/Algorithme_de_Ford-Fulkerson (le 23 Aout 2019)
https ://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur (le 23 Aout 2019)
https ://fr.wikipedia.org/wiki/Algorithme_de_Ford-Fulkerson (le 23 Aout 2019)