



INSTITUT
FRANCOPHONE
INTERNATIONAL



VNU
ĐẠI HỌC QUỐC GIA HÀ NỘI
Vietnam National University, Hanoi

RAPPORT TRAVAUX PRATIQUES RECHERCHE OPÉRATIONNELLE

MASTER SYSTÈMES INTELLIGENTS ET
MULTIMÉDIA

AFFECTATION DES TÂCHES

Étudiants

KAMBU MBUANGI fabien

KANA NGUIMFACK Kévin

Superviseur

Dr. Hoang-Thach Nguyen

1. Introduction

Le 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 dans un graphe Biparti. Nous avons implémenté ladite solution tels qu'il nous a été recommandé dans le cadre de notre cours de Recherche Opérationnelle. L'entrée du programme est un fichier texte dont la première ligne N représente le nombre de personnes et de tâches et le reste de lignes représentent les tâches de préférences d'une personne. Il s'agit des numéros de tâches séparés par des espace.

2. Objectif

L'objectif du présent travail est l'implémentation de l'algorithme de Ford-Fulkerson utilisant un BFS avec un langage de programmation java permettant de résoudre le problème d'affectation de tâches au sein d'une organisation afin que le nombre de personnes satisfaisantes soit maximal.

3. 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.

4. 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, la 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.

5. Implémentation

A. Modèle

Graphe Biparti : au démarrage du programme, celui-ci prend en paramètre un fichier représentant les liaisons possibles 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 arrête ait une extrémité dans et l'autre dans.

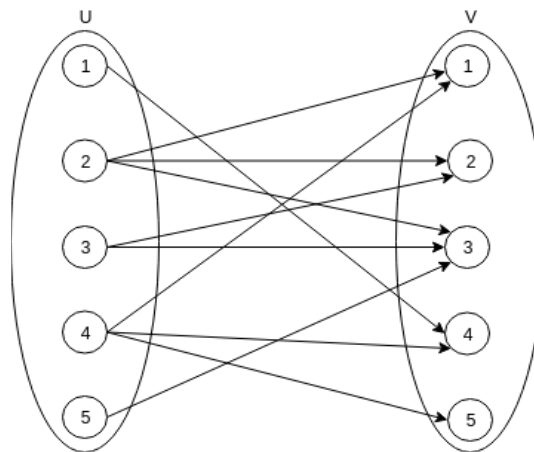


Figure 1 : graphe Biparti.

Etant 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 donnée 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.

Nous présentons ci-dessous les résultats de notre programme après avoir implémenté les trois différentes classes qui nous a été demandé, Il s'agit de : Couplage.java, Graphe.java et Nœud.java ;

B. Résultats

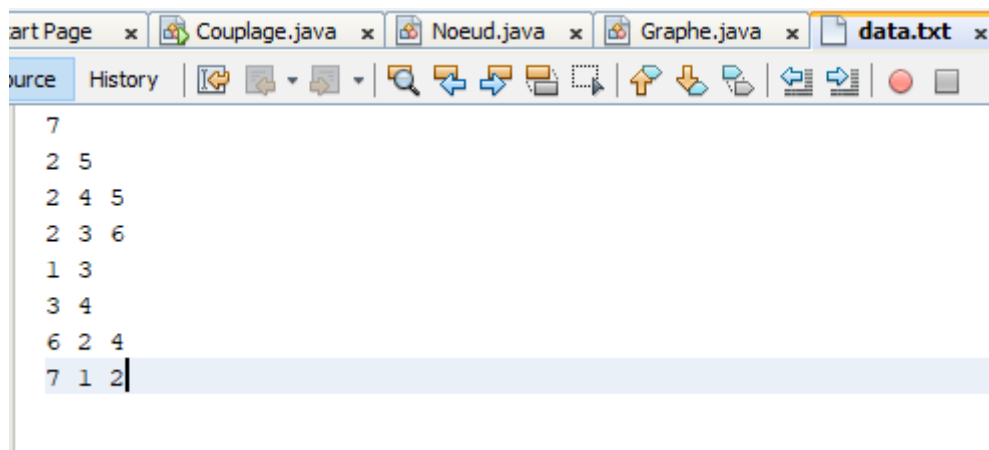


Figure 2 : Fichier d'entrée de tâches souhaitées par personne.

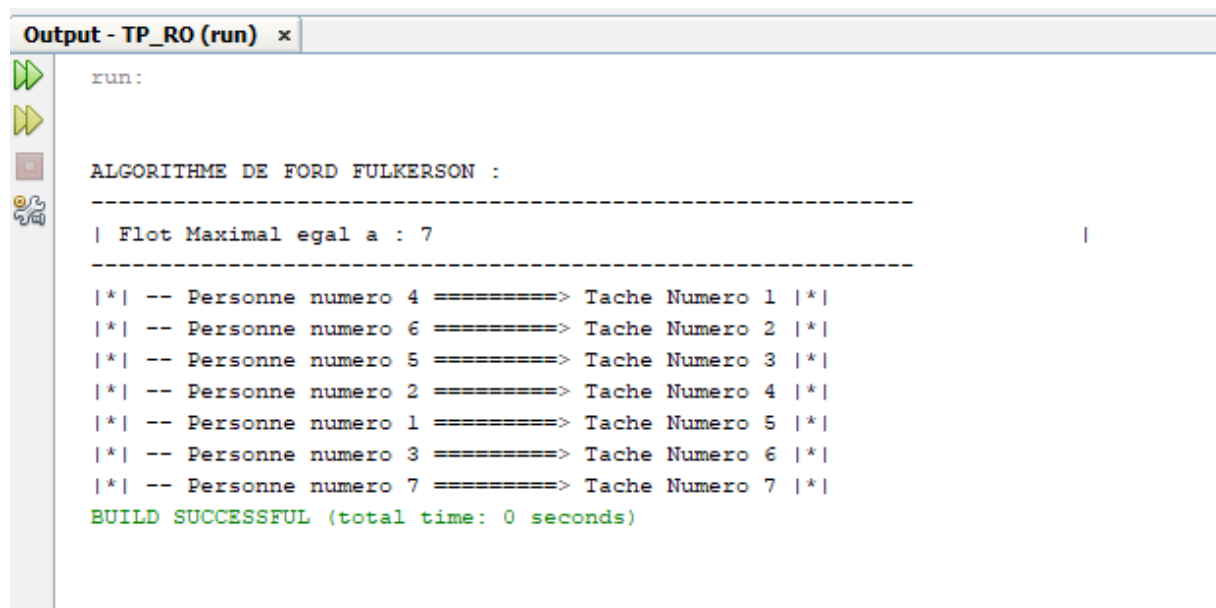
La figure 2 ci-dessus montre les contenus du fichier data.txt qui liste les tâches souhaitées par personne.

La première ligne contient un chiffre représentant le nombre de tâches qui doit être égal au Nombre de personne. Dans le cas présent 7 ; soit 7 personnes et 7 tâches.

Les lignes suivantes contiennent les tâches de préférence de chaque personne. Le tableau ci-dessous liste les préférences de chaque personne :

	Tâche 1	Tâche 2	Tâche 3	Tâche 4	Tâche 5	Tâche 6	Tâche 7
Personne 1		*			*		
Personne 2		*		*	*		
Personne 3		*	*			*	
Personne 4	*		*				
Personne 5			*	*			
Personne 6		*		*		*	
Personne 7	*	*					*

Tableau 1 : Préférences de tâches de chaque personne.



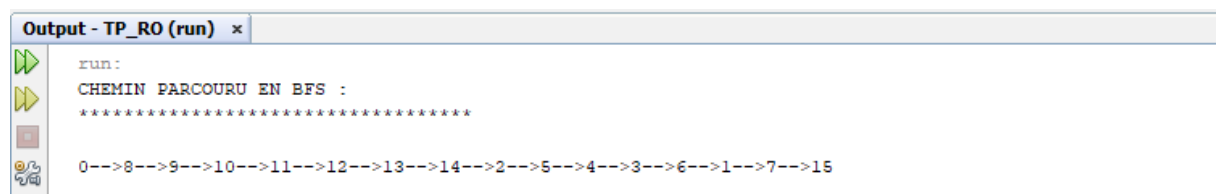
```
Output - TP_RO (run) x
run:
ALGORITHME DE FORD FULKERSON :
-----
| Flot Maximal egal a : 7 |
-----
|*| -- Personne numero 4 =====> Tache Numero 1 |*|
|*| -- Personne numero 6 =====> Tache Numero 2 |*|
|*| -- Personne numero 5 =====> Tache Numero 3 |*|
|*| -- Personne numero 2 =====> Tache Numero 4 |*|
|*| -- Personne numero 3 =====> Tache Numero 5 |*|
|*| -- Personne numero 1 =====> Tache Numero 6 |*|
|*| -- Personne numero 7 =====> Tache Numero 7 |*|
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 3 : Résultat après implémentation.

La figure 3, illustre le résultat obtenu avec après l'entrée du fichier data.txt illustrer par la figure 2.

Nous pouvons constater que chaque personne a obtenu une tâche parmi ses tâches souhaitées. Ainsi nous avons satisfait à l'objectif de l'algorithme de Ford-Fulkerson étant d'obtenir le flot maximum ; nous pouvons dire que le problème d'affectation de taches est résolu avec le flot max égal à 7.

✓ Chemin BFS



```
Output - TP_RO (run) x
run:
CHEMIN PARCOURU EN BFS :
*****
0-->8-->9-->10-->11-->12-->13-->14-->2-->5-->4-->3-->6-->1-->7-->15
```

Figure 4 : présente le chemin parcouru par BFS.

✓ Chemin inverse BFS

```
'  
CHEMIN PARCOURU EN BFS INVERSE:
```

```
*****
```

```
15 -->1 -->2 -->3 -->4 -->5 -->6 -->7 -->11 -->14 -->8 -->9 -->10 -->13 -->12 -->0
```

Figure 5 : présente le chemin inverse parcouru par BFS.

6. Conclusion

Nous estimons avoir atteint l'objectif de notre travail en implémentant l'algorithme de Ford-Fulkerson utilisant le BFS avec le langage de programmation java afin de résoudre le problème d'affectation de tâches au sein d'un groupe de personnes.

7. Références et Liens

Notes de cours (Recherche Opérationnelle).

<https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>(20/04/2020).

<https://tutorialedge.net/artificial-intelligence/breadth-first-search-java>(16/04/2020).