INSTITUT FRANCOPHONE INTERNATIONAL

PROBLÈME D'AFFECTATION DE TÂCHES A l'aide de L'ALGORITHME DE FORD-FULKERSON Dans un GRAPHE BIPARTI(Rercherhe operationel)

PROMOTION 23 RSC
ANNÉE ACADÉMIQUE 2019-2020

ENSEIGNANT :
PROF.HOANG-THACH NGUYEN

RÉDIGÉ PAR :
DIBWE FITA
VIGAN SILAS
VITOFODJI JEAN CLAUDE

Table des matières

Table des figures

1 INTRODUCTION

La recherche opérationnelle étant l'ensemble des méthodes et techniques rationnelles orientées vers la recherche du meilleur choix dans la façon d'opérer en vue d'aboutir au résultat visé ou au meilleur résultat possible. Elle fait partie des « aides à la décision» dans la mesure où elle propose des modèles conceptuels en vue d'analyser et de maitriser des situations complexes pour permettre aux décideurs de comprendre, d'évaluer les enjeux et d'arbitrer ou de faire les choix les plus efficaces.

2 OBJECTIF

L'objectif principal de notre travail est d'implémenter l'algorithme de Ford-Forlkerson utilisant un BFS avec le langage de programmation java permettant de résoudre le problème d'affectation de tâches au sein d'un groupe de personnes.

3 L'ALGORITHME DE FORD-FOLKERSON

3.0.1 Définition

La méthode Ford - Fulkerson ou algorithme Ford – Fulkerson (FFA) est un algorithme glouton qui calcule le flux maximal dans un réseau de flux. On l'appelle parfois une "méthode" au lieu d'un "algorithme" car l'approche permettant de trouver des chemins d'extension dans un graphe résiduel n'est pas complètement spécifiée ou spécifiée dans plusieurs implémentations avec des temps d'exécution différents.

3.0.2 Principe

- L'idée de l'algorithme : trouver un chemin augmentant et augmenter le flot sur ce chemin - Initialisation : f= 0. - Alternance de deux phases :

1. Phase de marquage (recherche d'un chemin augmentant)
2. Phase d'augmentation (augmenter le flot sur le chemin trouvé en phase démarquage) Ces deux phases sont répétées jusqu'au moment où il n'existe plus de chemin augmentant.

3.0.3 Complexité de l'algorithme de ford-fulkerson

En ajoutant le chemin d'augmentation de flux au flux déjà établi dans le graphique, le flux maximum sera atteint lorsqu'il ne restera plus aucun chemin d'augmentation de flux dans le graphique. Cependant, rien ne garantit que cette situation sera un jour atteinte. Le mieux que l'on puisse garantir, c'est que la réponse sera correcte si l'algorithme se termine. Dans le cas où l'algorithme fonctionne à l'infini, le flux pourrait même ne pas converger vers le flux maximal. En effet voici la complexité maximal de l'algorithme de ford-fulkerson la complexité maximale totale est O(F•(|N|+|N|+|A|)) =O(F•(|N|+|A|)).

3.0.4 Exemple

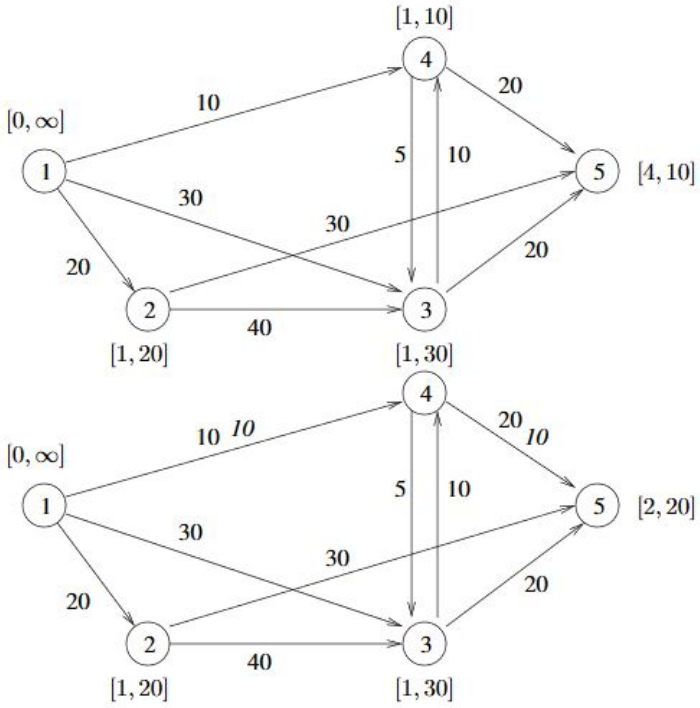

FIGURE 1 – Graphe orienté

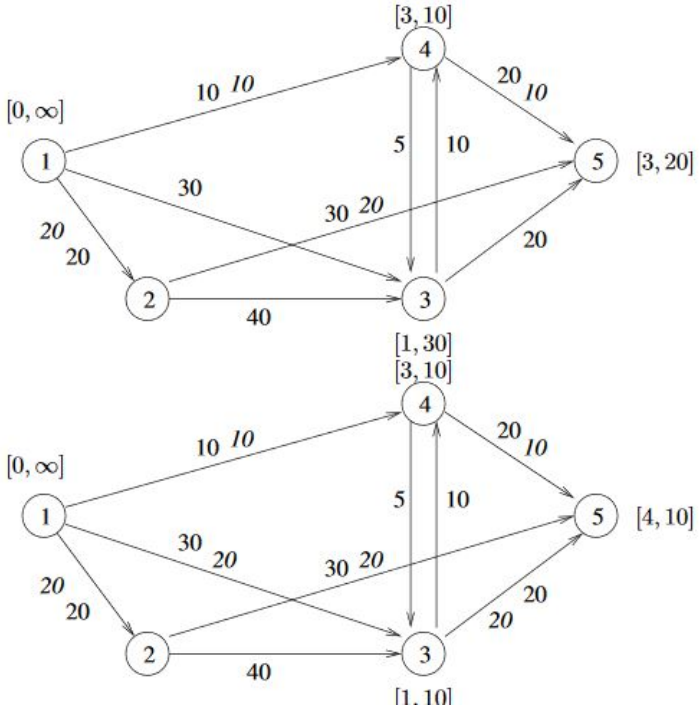

FIGURE 2 – Graphe orienté

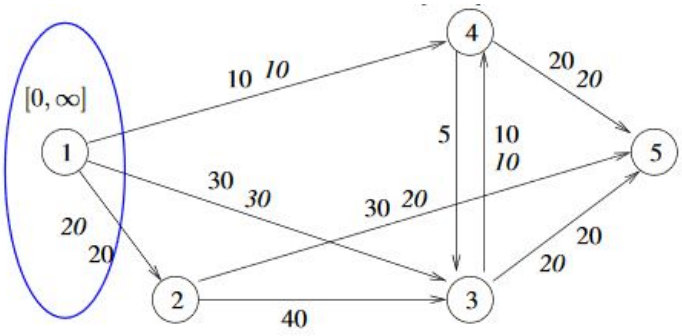

FIGURE 3 – Graphe orienté

Les nœuds marqués à la dernière itération définissent la coupe minimale

4 Algorithme de BFS

4.0.1 Définition

L'algorithme de parcours en largeur (ou BFS, pour Breadth First Search en anglais) 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.

4.0.2 Fonctionalité de l'algorithme de BFS

- BFS est un algorithme de parcours de graphe par recherche en largeur d'abord (Breadth First Search).
- Fonctionne avec les graphes orientés et non-orientés.
- Permet la détection de cycle si on considère le graphe non-orienté.
- Ne permet pas la détection de circuit (la version "stricte" n'utilise pas de tableau des précédents).
- Permet la détection de Composantes Simplement Connexes si il reste des sommets non traités après un premier passage (plus rapide en pratique que l'algorithme de Warshall si nous avons une seule cfc)

4.0.3 Caractéristiques de l'algorithme de 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.

4.0.4 Exemple du parcours BFS

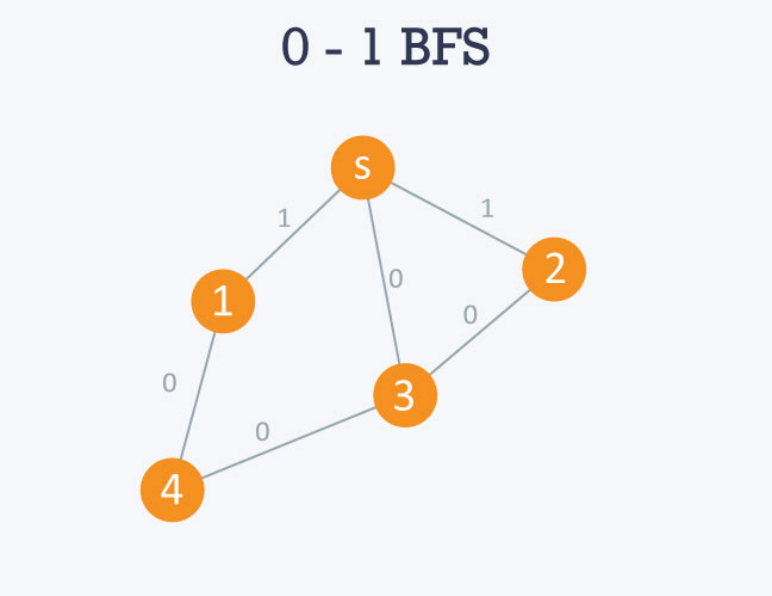

FIGURE 4 – BFS

5 IMPLEMENTATION

5.0.1 Graphe biparti

Lorsqu'on lance l'application, il prend en paramètre un fichier représentant les liaisons entre les éléments de deux parties d'un graphe biparti. Cependant un graphe est dit biparti si son ensemble de sommets peut être divisé en deux sous-ensembles disjoints et tels que chaque arête ait une extrémité dans et l'autre dans . Un graphe biparti permet notamment de représenter une relation binaire

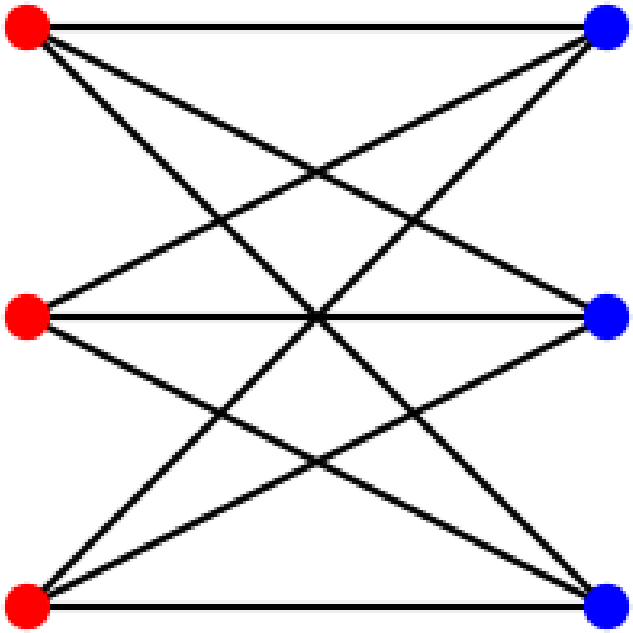

FIGURE 5 – Graphe biparti

En effet nous avons utilisé les classe ci-après :

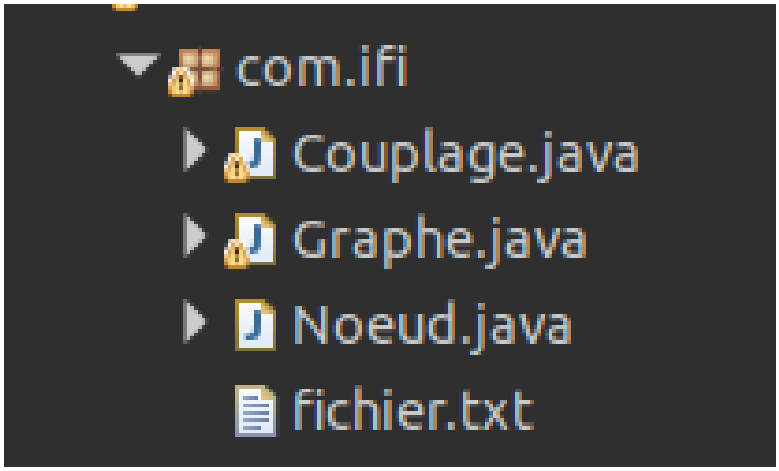

FIGURE 6 – classes java

5.0.2 Résultat

voici le couplage maximum de personnes à leurs tâches dans la capture ci-dessous

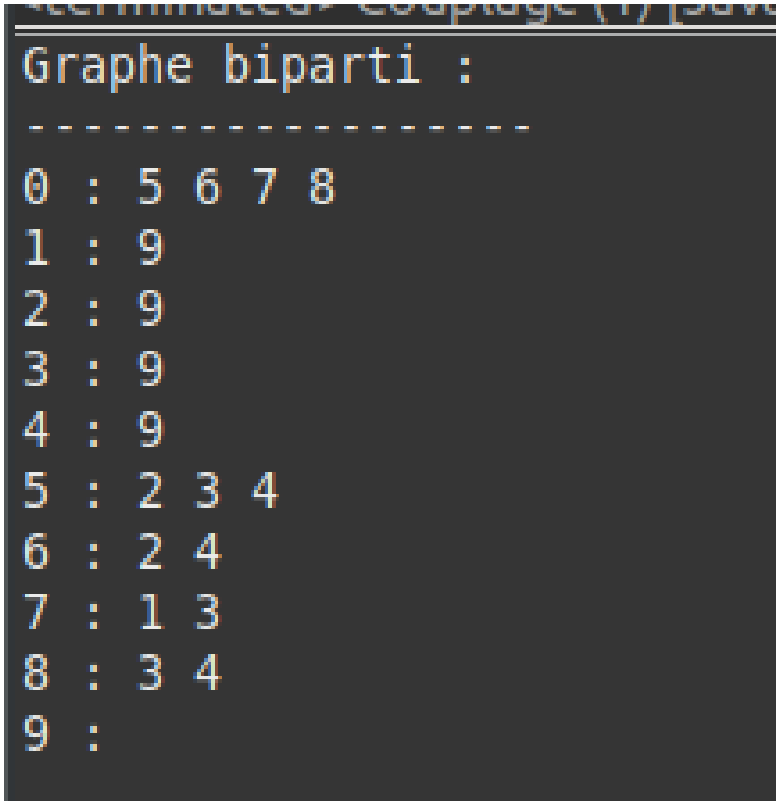

FIGURE 7 – couplage maximun

voici le résultat obtenu avec les donnée du couplage dans la capture ci-dessous

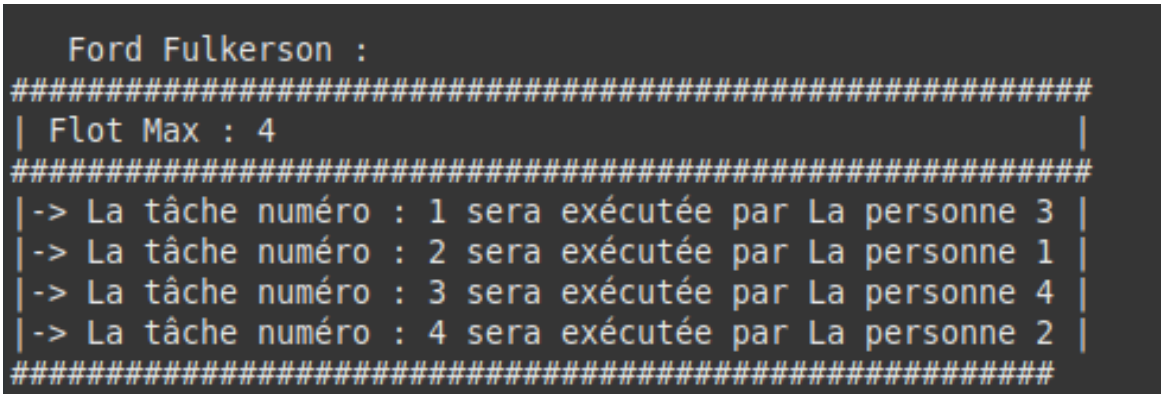

FIGURE 8 – résultat de l'algo de ford-folkerson

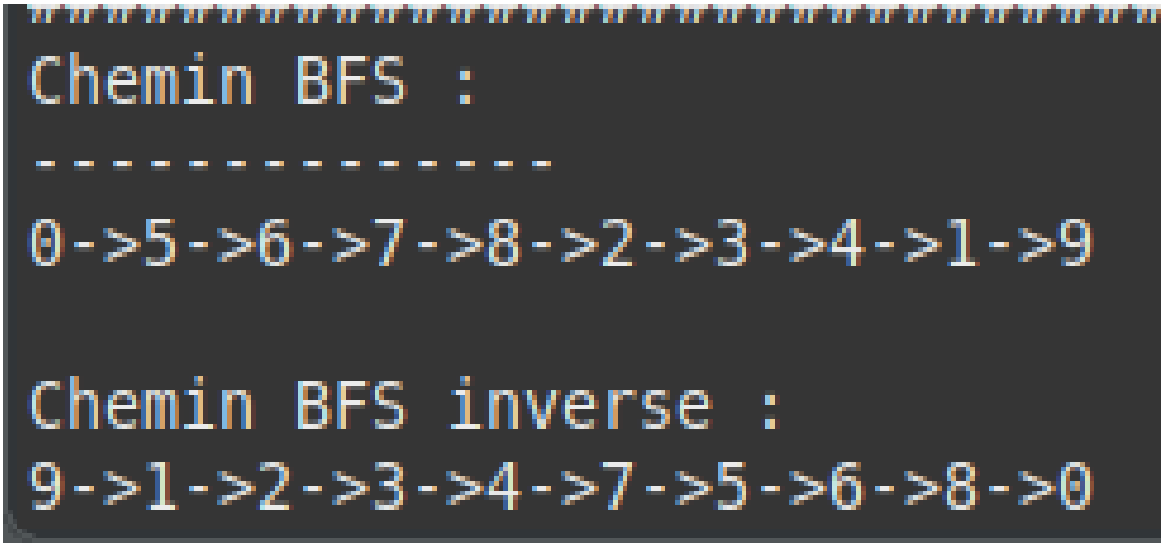

FIGURE 9 – BFsetInverse

Conclusion

Nous avons implémenté l'algorithme de ford-Fulkerson utilisant le BFS afin de résoudre le problème d'affectation de tâche au sein d'un groupe de personnes. En effet nous avons débuté par une introduction de notre travail. Dans la deuxième et troisième partie nous avons parlé de l'algorithme de Ford-Fulkeson et l'algorithme de Breadth First Search (BFS) et nous avons fini par une démonstration des résultats et l'architecture de notre travail

BIBLIOGRAPHIE

— Notes de cours de Recherche Opérationnelle
— https ://fr.wikipedia.org
— www.dunod.com