UNIVERSITÉ CÔTE D'AZUR

MASTER INFORMATIQUE

Rapport de Projet

Minimum Cost Maximum Flow

Réalisé par
Dylann Batisse

Encadré par
Jean-Charles Régin

Master Informatique
à Université Côte d'Azur
Second semestre de l'année 2021-2022

Table des matières

1 Introduction

1.1 Problème de flot maximum

Le problème de flot maximum consiste à trouver, dans un réseau de flot, un flot réalisable depuis une source unique et vers un puits unique qui soit maximum.

1.2 Problème du flot de coût minimum

Le problème du flot de coût minimum est similaire au flot maximum sauf qu'il faut emprunter coût le plus minimal possible.

2 Choix de la structure de donnée

J'ai décidé de choisir une liste d'adjacence pour représenter mon graphe car je trouve que cela est beaucoup plus simple d'accéder à tout les nœuds. Par exemple si l'ont veut accéder à la capacité et au coût du nœud $0 \rightarrow 1$ nous devons accéder à capacities[i][j] et costs[i][j] où i = 0 (nœud 0) et j = 1 (nœud 1).
Par exemple :

— Graphe des capacités : $\begin{bmatrix} 0 & 40 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 8 & 5 & 0 & 0 \\ 0 & 0 & 0 & 20 & 4 & 10 & 12 \\ 0 & 0 & 0 & 0 & 15 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

— Graphe des coûts : $\begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 4 & 8 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 6 & 8 \\ 0 & 0 & 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

Alors capacities[0][1] renvoie 40 et costs[0][1] renvoie 2.

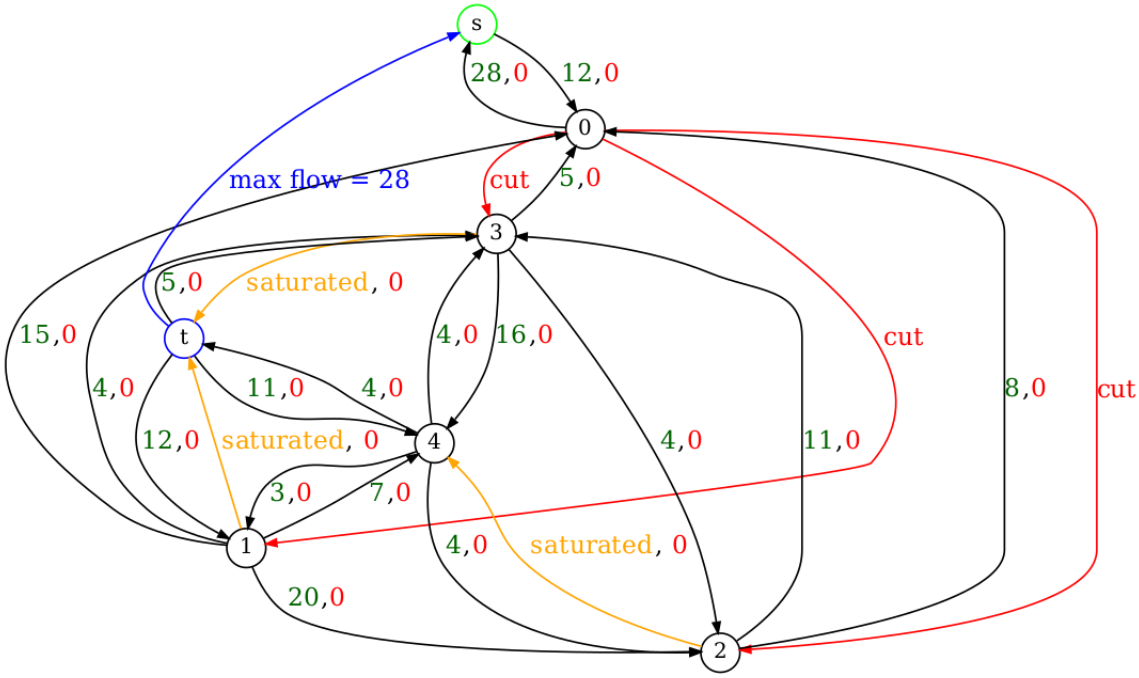
ATTENTION : Le graphe que j'utilise est réadapter pour que l'on puisse ajouter les nœuds s et t dans le graphe car je ne peux pas utiliser des lettres, la coupe minimum peut différer du graphe de base, le nœud s est donc 0 et le nœud 6 est t dans ce graphe.

3 Flot maximum

Pour résoudre le problème du flot maximum nous allons utiliser l'algorithme de Ford-Fulkerson. Le principe est le suivant : Tant qu'il existe un chemin entre la source et le puits dans le graphe résiduel, il faut envoyer le minimum des capacités résiduelles sur ce chemin.
Dans l'algo, dans un premier temps l'utilisation de BFS est nécessaire pour savoir s'il existe un chemin de s à t et ainsi faire les opérations sur les arêtes pour les mettre à jour. Si un chemin existe alors nous cherchons le flot minimum de t à s dans le chemin que nous avons trouvé. Une fois trouvé, ce flot est ajouté au flot maximum.

Une deuxième passe de t à s est nécessaire pour mettre à jour les flots et cela va provoquer la saturation de certaines arêtes.

Dans le graphe généré par mon programme ci-dessous nous pouvons voir un exemple applicatif de l'algo qui tourne du nœud 0 au nœud 6. Par soucis de structure de donnés mon programme n'acceptant pas l'alphabet mais n'utilisant que les indices j'ai donc dû faire en sorte d'ajouter deux nœuds, 0 et 6 qui seront au final s et t.



On peut dire que le flot maximum est la somme du flot de la coupe minimum qui dans notre cas est $0 \to 1, 0 \to 3, 0 \to 2$ donc $5 + 15 + 8 = 28$, ce graphe à un flot maximum de 28.
La sortie du programme est la suivante :

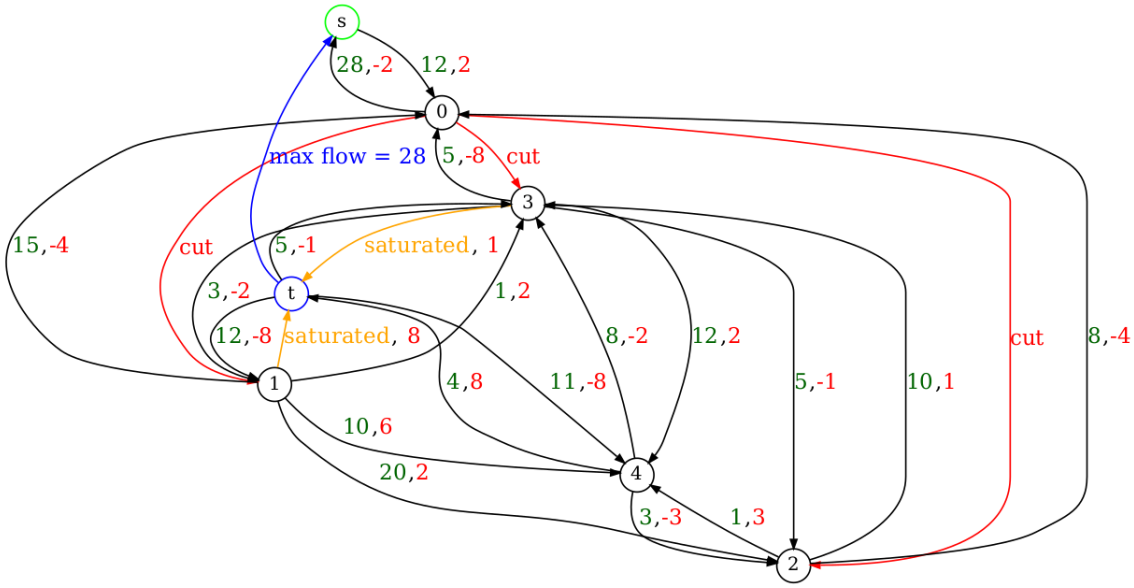
```
Max flow: 28
Min cost: 0
Attention le resultat de la min cut differe du graphe donne de base car il est
     readapter pour rajouter s et t
Min cut |(1 → 4)|(1 → 2)|(1 → 3)|
```

Graphe résiduel :

$$\begin{bmatrix} 0 & 12 & 0 & 0 & 0 & 0 & 0 \\ 28 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 0 & 20 & 4 & 7 & 0 \\ 0 & 8 & 0 & 0 & 11 & 0 & 0 \\ 0 & 5 & 0 & 4 & 0 & 16 & 0 \\ 0 & 0 & 3 & 4 & 4 & 0 & 4 \\ 0 & 0 & 12 & 0 & 5 & 11 & 0 \end{bmatrix}$$

4 Flot maximum de coût minimum

Pour le problème du flot de coût minimum le principe est presque le même, à la place d'utiliser BFS nous utilisons un algorithme de plus court chemin nommé Bellman-Ford qui permet de détecter les cycles négatifs et gérer les coûts négatifs. Le principe est le suivant : tant qu'il existe un plus court chemin de s à t, on cherche le flot minimum du chemin et nous l'ajoutons au flot maximum. Ensuite, nous mettons à jour les capacités de flot et nous rajoutons le coût de l'arête fois le flot minimal du plus court chemin actuel.



Pour ce graphe nous avons un flot maximum de 28 et un coût minimum de 413 avec pour coupe minimum $0 \to 1, 0 \to 3, 0 \to 2$.

Sortie du programme :

```
Max flow: 28
Min cost: 413
Attention le resultat de la min cut differe du graphe donn  de base car il
    est readapter pour rajouter s et t
Min cut |(1 → 3)|(1 → 2)|(1 → 4)|
```

Graphe résiduel :
$$\begin{bmatrix} 0 & 12 & 0 & 0 & 0 & 0 & 0 \\ 28 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 0 & 20 & 1 & 10 & 0 \\ 0 & 8 & 0 & 0 & 10 & 1 & 0 \\ 0 & 5 & 3 & 5 & 0 & 12 & 0 \\ 0 & 0 & 0 & 3 & 8 & 0 & 4 \\ 0 & 0 & 12 & 0 & 5 & 11 & 0 \end{bmatrix}$$

Graphe des coûts :
$$\begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 4 & 4 & 8 & 0 & 0 \\ 0 & -4 & 0 & 2 & 2 & 6 & 8 \\ 0 & -4 & 0 & 0 & 1 & 3 & 0 \\ 0 & -8 & -2 & -1 & 0 & 2 & 1 \\ 0 & 0 & 0 & -3 & -2 & 0 & 8 \\ 0 & 0 & -8 & 0 & -1 & -8 & 0 \end{bmatrix}$$

5 Coupe minimum

Une coupe minimum d'un graphe est un ensemble de sommets contenant un nombre minimal d'arêtes. Lors du déroulement des algos de flot maximum nous allons ajouter dans une liste toutes les arêtes qui sont saturés pour ensuite pouvoir déterminer la coupe minimum.

Pour trouver la coupe minimum d'un graphe il va falloir chercher parmi nos arcs saturés s'il existe un chemin de s à u et s'il n'existe pas de chemin de s à v, si cette condition est validé alors cet arc fait partie de la coupe minimum. L'algo procède comme suit : on vérfie chaque arête (u, v) qui ont été saturés et si BFS(s, u) trouve un chemin de s à u et qu'il n'existe pas de chemin BFS(s, v) de s à v alors cet arc fait partie de la coupe minmum.

6 Conclusion

Les problèmes rencontrés dans ce projet se prononcent surtout sur le choix de la structure de donnée et la détection de la coupe minimum pour savoir quels arêtes il fallait garder.