I Problématique

Dans le cadre d'une politique ambitieuse concernant l'éducation, un élu souhaite construire de nouvelles écoles, modernes, dans la communauté d'agglomération dont il est responsable. Durant la campagne électorale, il a promis deux choses :

(Accessibilité) Chaque ville doit posséder son école, ou être directement reliée à une ville qui possède une école.

(Économie) Le coût du projet doit être le plus bas possible, ce qui signifie que le nombre d'écoles à construire doit être le plus petit possible.

Il nous demande de l'aider, en développant un logiciel qui permet :

1) de représenter les villes d'une communauté d'agglomération, et les routes qui les relient ;

2) de simuler la construction d'écoles dans les villes de la communauté ;

3) de s'assurer que la contrainte d'*accessibilité* est respectée ;

4) de calculer le coût d'une solution (le nombre d'écoles), et de le minimiser.

Nous réaliserons ces tâches de façon incrémentale, afin d'avoir à la fin du semestre un logiciel fonctionnel.

II Villes et Routes

Pour simplifier le problème, nous considérons que toutes les routes sont à double sens, et de même longueur. Une route est donc un point d'accès rapide entre une ville A et une ville B. Par conséquent, une communauté d'agglomération peut-être représentée par un graphe non-orienté, où les noeuds représentent les villes, et les arêtes représentent les routes. Un exemple est représenté en Figure 1.

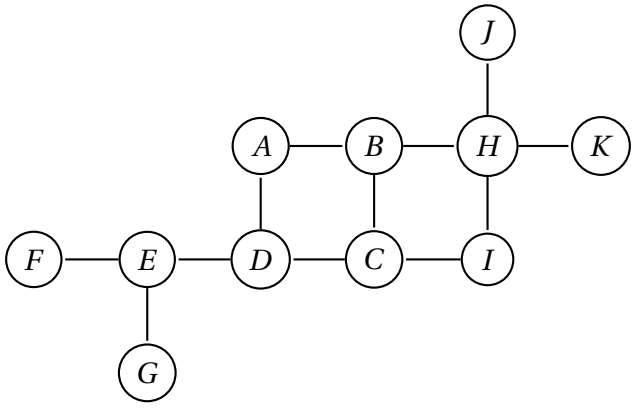


FIGURE 1 – Une communauté d'agglomération avec 11 villes.

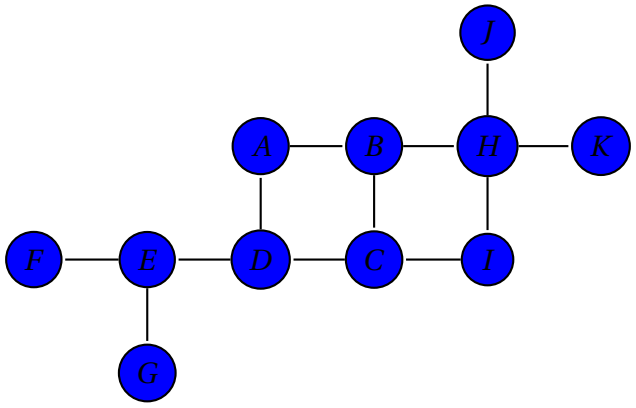


FIGURE 2 – Une communauté d'agglomération avec 11 villes, et 11 écoles.

Une solution naive pour satisfaire la contrainte d'*Accessibilité* consiste à placer une école dans chaque ville. On représente les villes possédant une école par des noeuds colorés en bleu, comme en Figure 2.
Bien entendu, cette solution ne satisfait absolument pas la contrainte d'*Économie*, puisqu'il serait possible de retirer certaines écoles. La Figure 3 présente une meilleure solution, où seules 4 écoles sont construites.

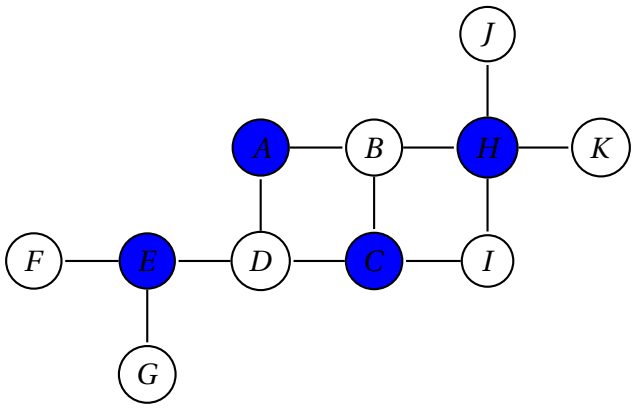


FIGURE 3 – Une communauté d'agglomération avec 11 villes, et 4 écoles.

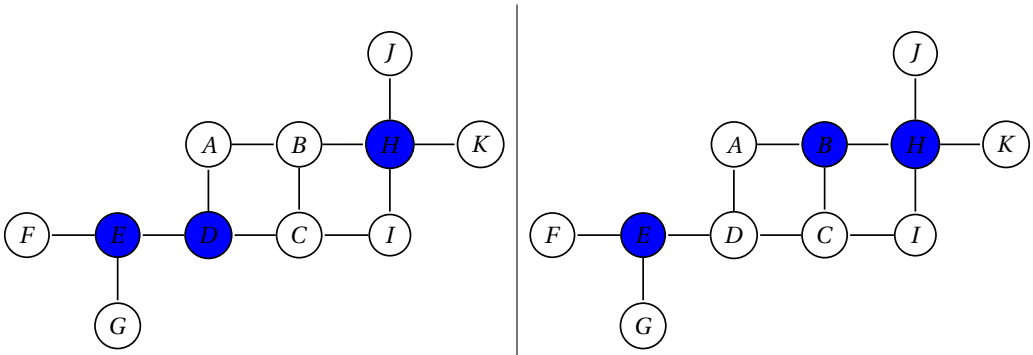
Cependant, cette solution n'est pas encore optimale. Il est possible de satisfaire la contrainte d'*Accessibilité* avec seulement 3 écoles, comme on le voit en Figure 4a. Pour toute communauté d'agglomération, il existe *au moins une* solution optimale. Il peut en exister plusieurs (c'est-à-dire plusieurs solutions qui ont le même nombre d'écoles n, tel qu'aucune solution avec $n-1$ écoles ne satisfait l'*Accessibilité*). C'est par exemple le cas, ici, avec la Figure 4b.

III Instructions

1. Tâches à réaliser

Pour la première étape du projet, vous devez développer un programme qui permet à un utilisateur de configurer les écoles d'une communauté d'agglomération. Au démarrage, le programme doit demander à l'utilisateur le nombre de villes. Pour l'instant, nous supposerons que le nombre de villes est inférieur à 26, par conséquent elles peuvent être nommées en fonction des lettres de l'alphabet. Retenez cependant que dans la suite du projet, il pourra y avoir un nombre quelconque de villes, avec des noms quelconques.
Une fois que le nombre de villes n est fixé, un menu s'affiche avec deux options :

1) ajouter une route ;



(a) Une solution optimale (b) Une deuxième solution optimale

FIGURE 4 – Deux solutions optimales pour la communauté d'agglomération

2) fin.

Dans le cas où l'option 1 est retenue, on demande à l'utilisateur les villes entre lesquelles il faut ajouter une route, puis on revient au menu précédent. Dans le cas où l'option 2 est retenue, l'utilisateur a terminé de représenter la communauté d'agglomération. Il peut maintenant essayer de trouver une solution au problème. La communauté d'agglomération est générée avec la solution naive qui consiste à placer une école dans chaque ville. Ensuite, l'utilisateur fait face à un menu qui propose trois options :

1) ajouter une école ;

2) retirer une école ;

3) fin.

Quand l'option 1 est sélectionnée, on demande à l'utilisateur dans quelle ville il souhaite ajouter une école, puis le programme ajoute l'école si c'est possible (c'est-à-dire s'il n'y a pas déjà une école dans cette ville).
Quand l'option 2 est sélectionnée, on demande à l'utilisateur dans quelle ville il souhaite retirer une école, puis le programme retire l'école si c'est possible (c'est-à-dire si ce retrait ne viole pas la contrainte d'*Accessibilité*). Dans le cas où le retrait est impossible, un message d'erreur doit indiquer à l'utilisateur pourquoi ce n'est pas possible (c'est-à-dire quelles villes se retrouveraient sans école dans leur voisinage).
Après chaque action de l'utilisateur, un message indique où se trouvent les écoles, par exemple la solution représentée en Figure 4b est représentée par :

```
B H E
```

qui indique que les villes B, H et E sont celles où se trouvent des écoles (l'ordre n'a pas d'importance).
Enfin, si l'option 3 est sélectionnée, le programme s'arrête.

2. Remise du projet
Ce projet est à réaliser par groupes de **deux ou trois étudiants**, issus du **même groupe de TD**. Votre code source, correctement documenté, sera à remettre sur Moodle au plus tard le 6 Novembre 2020, sous forme d'une archive jar (un seul dépôt par binôme/trinôme).
Des conseils sur l'implémentation seront fournis prochainement sur Moodle.