T NSI	Structures de données	*Et si on construisait un réseau électrique*

I. A la recherche d'une solution optimale :

I.1. Le problème :

Le but du jeu est de construire un réseau électrique qui relie toutes les villes et dont le coût de construction soit le plus petit possible.
Un graphe G est utilisé pour représenter le réseau. Chaque ville est représentée par un sommet (vertex) et les arêtes (edge) sont les lignes électriques.
Chacune de ces lignes a un coût, ce coût est modélisé par la pondération w. (weigth)

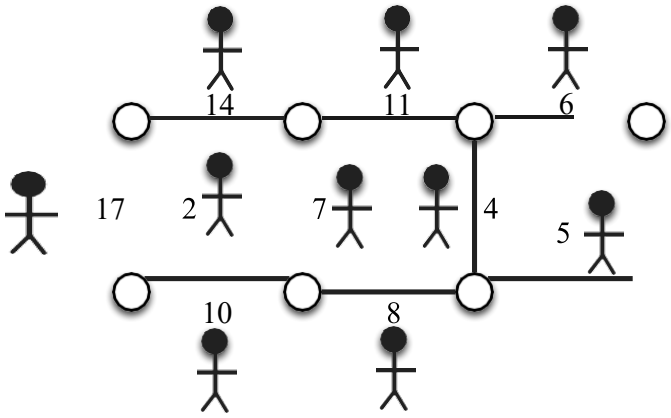
Pour câbler de façon optimale ce réseau, chaque sommet doit avoir accès au réseau et le coût du câblage doit être minimum.

Qu'est-ce qu'un arbre T (Tree) ?
Un arbre est un graphe qui ne contient **aucun cycle** : Entre deux sommets quelconques d'un arbre, il existe un unique chemin.

On appelle arbre couvrant de poids minimum (ACPM) de G tout arbre couvrant dont la somme des poids des arêtes le constituant est minimal.
La solution à notre problème est donc un Arbre Couvrant de Poids Minimum.

I.2. Le jeu :

Considérons le graphe pondéré G(V,E, w) suivant :



Nombre de joueurs. Il y a autant de joueurs que d'arêtes dans le graphe. Le jeu est collaboratif (il n'y a pas de gagnant ni de perdant). Dans notre exemple, il y a dix arêtes et donc dix joueurs.

Avant le jeu. Les joueurs se positionnent sur les arêtes du graphe (un joueur par arête).

Déroulement du jeu. Le jeu est divisé en étapes.
À chaque étape, une arête va être sélectionnée et possiblement ajoutée à la solution (si elle ne crée pas de cycle avec les arêtes de la solution).

Lors de la première étape, l'arête de plus petite valeur est sélectionnée et ajoutée à la solution.

Q1. Quel est l'ordre n de G (nombre total de sommets)
Q2. Après plusieurs essais proposer une méthode.
Q3. Combien d'étapes faut-il pour trouver une solution ?

II. Un exemple de solution : L'algorithme de KRUSKAL (1956)

I.3. Principe

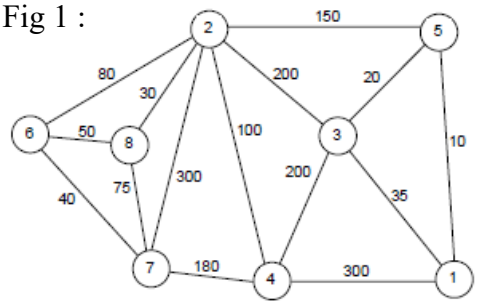
Nous allons partir d'un sous-graphe contenant initialement les n sommets du graphe mais ne possédant aucune arête et nous allons lui rajouter des arêtes une par une.

Tant qu'on n'a pas un arbre, on ajoute l'arête la plus légère si elle ne crée pas de cycles.
Une fois qu'une arête e a été traitée on ne reviendra pas dessus. De plus, à chaque étape l'on effectue un choix que l'on espère (et qui l'est d'ailleurs) optimal. Ces deux caractéristiques font que cet algorithme est un algorithme glouton.

Propriété :

Si n est l'ordre du graphe, cet algorithme procède en au moins n−1 itérations. Nous voulons en effet un arbre couvrant, i.e. un arbre possédant les n sommets du graphe d'origine donc cet arbre doit nécessairement comporter n−1 arêtes.

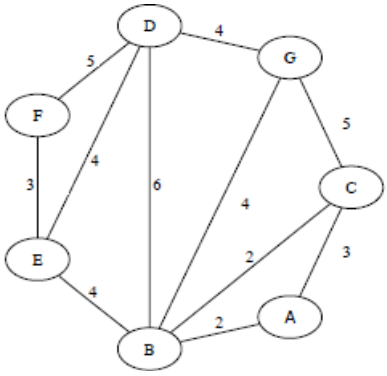
Fig 1 :



Q4. Appliquer ce principe au graphe de la figure 1.

Q5. Quel est le poids de l'arbre couvrant de poids minimal trouvé ?

Fig 2 :



Q6. Deviner un arbre couvrant minimum pour le graphe de la figure 2.

Q7. L'arbre couvrant de poids minimal est il forcément unique ?

Q8. Selon l'ordre dans lequel les arêtes de même poids sont examinées, l'algorithme de Kruskal retourne-t-il le même ACM ? donner un exemple

I.4. Pseudo-code :

Soit E l'ensemble des arêtes du graphe et F (forêt) une solution partielle minimale (un ensemble d'arêtes sans cycle dont le coût est minimal),

Q9. Compléter le pseudo-code suivant :

```
E ensemble des arêtes de G triées dans l'ordre croissant
F = {}; ensemble des arêtes de l'arbre couvrant
Pour chaque élément e de E et tant que |F|<|V|-1 faire
    …..
```

Q10. Il existe de nombreuses techniques pour trier un ensemble, mais dans notre cas nous ne disposons pas vraiment de l'ensemble. Quelle structure de données pourrait être utilisée pour construire cet ensemble et récupérer efficacement l'arête minimale à chaque étape.

Q11. Quel algorithme de tri pourrait on utiliser ?