Arbres Couvrants de Poids Minimum

Exercice 1 (Connexité et coûts minimaux)

Nous allons reprendre la problématique de notre ISP *MarWaNet* (souvenez vous du TD sur la connexité). Le comptable voulait avoir le nombre minimum de liaisons tout en gardant la connexité du réseau. Il veut maintenant minimiser le coût global du réseau. On dispose du coût (annuel) de chaque liaison.

Notre objectif est donc de minimiser les coûts tout en conservant la connexité du réseau. Le comptable fournit un tableau avec les différentes liaisons possibles avec le prix associé. On rappelle que les liaisons sont *bi-directionnelles*. La figure 1 représente le graphe correspondant.

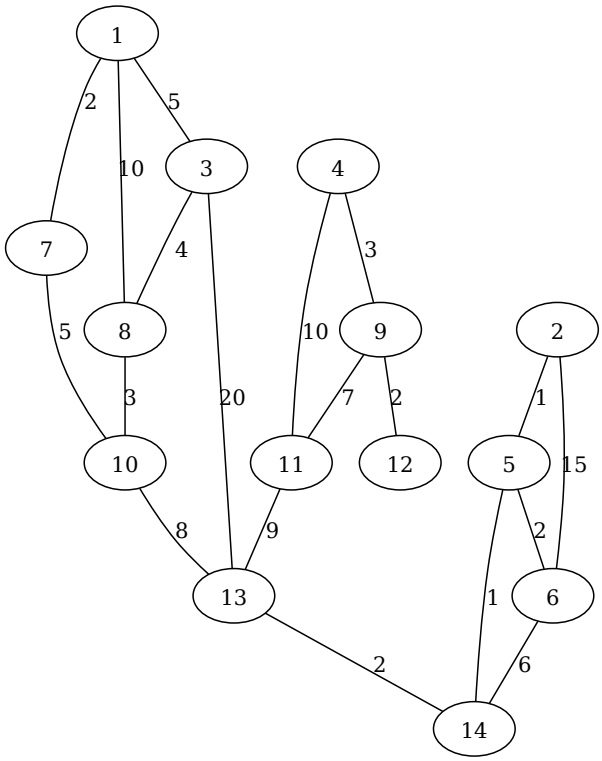

FIGURE 1 – Le graphe avec les coûts

1. (*Rappel*) Quel *type* de graphes permet de conserver la connexité tout en minimisant les liaisons ?

2. On intègre maintenant la notion de coût : Pour répondre au problème, quelles sont les contraintes qu'il faut satisfaire ?

3. Proposer une solution pour le graphe de la figure 1 (liste des arêtes à conserver). Dessiner le graphe résultat.

Nous allons maintenant essayer de construire un algorithme qui trouve une solution.

L'approche la plus classique est incrémentale [1] : on ajoute progressivement des arêtes à la solution (en partant d'un ensemble vide). À chaque étape de l'algorithme, on obtient un ensemble d'arêtes qui préservent en partie les propriétés finales attendues. En théorie, un algorithme incrémental peut *redémarrer* d'une solution partielle intermédiaire.

1. En fait, il existe des algorithmes basés sur la suppression d'arêtes comme : *reverse-delete algorithm*

Il y a deux stratégies pour résoudre ce problème selon la propriété que l'on préserve à chaque choix.

Exercice 2 (Algorithme de Prim)

1. Quelle propriété est maintenue à chaque itération de l'algorithme de Prim ?
2. On suppose déjà construit une partie de la solution (un ensemble d'arêtes dont les sommets forment un sous-graphe connexe), comment choisir une arête du graphe d'origine pour que le sous-graphe ainsi formé reste connexe et sans cycle ?
3. On veut maintenant garantir que lorsque l'algorithme termine, la solution construite soit minimale. Comment choisir le sommet de départ de la prochaine arête à ajouter ?
4. On déduire le principe d'un algorithme construisant la solution à notre problème.
5. Appliquer ce principe au graphe de la figure 1 en utilisant le sommet 13 comme point de départ.
6. L'algorithme que l'on vient de décrire ressemble à un algorithme de recherche de plus court chemin, lequel ? En déduire la structure de données *la plus pratique* pour choisir les sommets à traiter.
7. Écrire l'algorithme correspondant.

Exercice 3 (Algorithme de Kruskal)

Dans un premier temps, on considère que l'on travaille avec un ensemble d'arêtes déjà trié.

1. Quelle propriété est maintenue à chaque itération de l'algorithme de Kruskal ?
2. Soit A l'ensemble des arêtes du graphe et T une solution partielle minimale (un ensemble d'arêtes sans cycle dont le coût est minimal), montrer que si l'on ajoute à T la plus petite arête e de l'ensemble $A - T$ qui n'introduit pas de cycle dans T, alors l'ensemble $T \cup \{e\}$ reste minimal (il n'existe pas d'ensemble sans cycle avec autant d'arêtes de coût inférieur).
3. On considère maintenant le problème des cycles : comment déterminer que la nouvelle arête n'introduit pas de cycle ?
4. En déduire le principe de l'algorithme de Kruskal.
5. Appliquer ce principe au graphe de la figure 1.
6. 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.
7. Construire un algorithme qui à partir du graphe, construit un ARPM (représenté par une matrice).