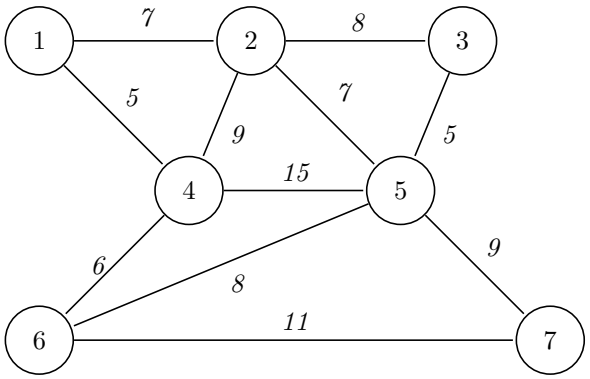
Algorithme d'optimisation

Graphes

1 Echauffement

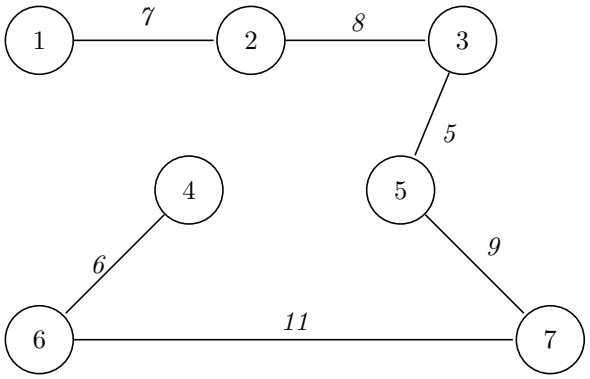
Exercice 1

1. *Une société souhaite relier ses 7 sites avec la fibre pour un coût minimal. Le coût de chaque ligne (en milliers d'euros) est donné ci-dessous :*



Ci-dessous, une proposition de travaux permet de relier les sites pour un coût de :

$$7+8+5+9+11+6=46$$



La proposition ci-dessus peut être optimisée. Trouver la solution optimale (c'est à dire celle qui minimise le coût des travaux et qui permet aux 7 sites d'être connectés)

2. *Ci-dessous, la matrice d'adjacence d'un graphe pondéré qui modélise le raccordement des 8 sites d'une autre société.*
Trouver la solution optimale pour cette autre société.

$$\begin{matrix}[[0,3,5,0,0,0,0,0],\\ [3,0,0,4,0,0,0,0],\\ [5,0,0,12,0,0,0,0],\\ [0,4,12,0,9,0,0,8],\\ [0,0,0,9,0,4,5,1],\\ [0,0,0,0,4,0,6,0],\\ [0,0,0,0,5,0,0,20],\\ [0,0,0,8,1,0,20,0]]\end{matrix}$$

2 Implémentation d'un algorithme

Exercice 2 *Voici un algorithme qui permet de résoudre les deux problèmes ci-dessus :*

> ***G*** *: un graphe pondéré*
> ***NewG*** *: une copie de* ***G*** *mais avec des poids tous nuls*
> ***Sol*** *: une liste d'arêtes (initialement vide)*
> ***L*** *: Liste des arêtes de* ***G*** *triées dans l'ordre croissant*
>
> *Tant que* ***L*** *n'est pas vide :*
> ***a*** *le premier éléments de* ***L*** *(celui avec le poids le plus petit)*
> *Supprimer* ***a*** *de* ***L***
> *Ajouter* ***a*** *au graphe* ***NewG***
> *Si* ***NewG*** *présente un cycle :*
> *Supprimer l'arête* ***a*** *de* ***NewG***
> *else :*
> *Ajouter l'arête* ***a*** *à* ***Sol***
> *Afficher* ***Sol***
> *Afficher la somme des pondérations des arêtes de* ***Sol***

1. *Appliquer cet algorithme avec l'exemple de l'exercice 1.*

2. *Implémenter cet algorithme en python.*

Aide et précisions : Chaque points ci-dessous est détaillé dans le fichier : lien

1. ***G*** *sera une matrice d'adjacence*
2. ***NewG*** *est un tableau de même taille que* ***G*** *qui ne contient que des 0. On pourra compléter la fonction* ***InitMatrice***
3. *L'arête* ***a*** *sera un tuple de 3 valeurs* $(sommet1, sommet2, ponderation)$.
4. *Compléter la fonction* ***ListeAretes*** *qui permet de générer la liste des arêtes (sous la forme d'un tuple) à partir de la matrice d'adjacence.*
5. *Pour trier une liste de tuples : lien. On pourra compléter la fonction* ***TrierAretes***
6. *Compléter la fonction* ***AddEdge*** *qui ajoute une arête dans un graphe donnés.*
7. *Compléter la fonction* ***SuppEdge*** *qui supprime une arête dans un graphe donnés.*
8. *Nous avons déjà a disposition la détection de cycle mais cette fonction prend en paramètre un dictionnaire des successeurs et non une matrice d'adjacence. On évitera de modifier cette fonction mais on écrira une fonction* ***convert_adj_dico*** *qui a partir d'une matrice d'adjacence renvoie un dictionnaire des successeurs (On perdra l'information des poids mais celle-ci ne nous est pas utile pour détecter nu cycle)*
9. *Vérifier la bonne implémentation avec les graphes de l'exercice 1.*

3 Application

Exercice 3 *Vous êtes chargé par le gouvernement kazakh d'équiper le pays en accès internet à haut débit. Pour cela, vous devez relier les 16 plus grandes villes du Kazakhstan avec des câbles de fibres optiques. Après une étude préliminaire, vous estimez les coûts suivants (en millions de KZT) de connexion entre les villes.*

Quelles sont les liaisons à réaliser pour connecter toutes les villes au moindre coût ?