1 Problème de CVRP

1.1 Rappel du problème

On s'intéresse au problème de tournées de véhicules avec contraintes de capacités.

Données :
— un ensemble de clients à visiter. Chaque client i a une demande d_i
— une matrice de distances entre les clients
— un ensemble infini de véhicules identiques de capacité C
— un unique dépôt 0

Objectif : trouver des tournées pour les véhicules qui minimisent la distance totale parcourue tout en visitant tous les sommets avec une demande non nulle. On notera que l'on n'est pas obligé d'utiliser tous les véhicules.

1.2 Heuristique

On se propose d'utiliser l'heuristique de Clarke et Wright.

Algorithme 1 : Algorithme de Clarke et Wright

Créer une liste vide L de tournées ;
pour *chaque client* i **faire**
| créer une nouvelle tournée contenant uniquement i et l'ajouter à L
fin
while *il reste deux tournées concaténables* **do**
| fusionner les deux tournées de L qui maximise le gain de la concaténation (saving)
end

Note : pour une implémentation efficace :
— on peut calculer a priori tous les gains possibles en fusionnant une tournée finissant par i et une tournée commençant par j puisqu'ils ne changent pas au cours de l'algorithme
— on peut obtenir le meilleur gain à chaque fois en utilisant la structure de tas (file de priorité)
— il faut créer toutes les structures de données nécessaires pour ne pas générer de la complexité non nécessaire

1.3 Code fourni

On vous fournit le parser (Java ou C++) qui permet de constuire une instance de CVRP à partir d'un fichier de données. La documentation est donnée sur moodle.

En Java, on vous fournit en plus une implémentation simplifiée de liste chaînée qui permet de concaténer des listes en temps constant. La classe `list` de C++ le permet.

1.4 Travail à réaliser

Question 1. Implémentez l'heuristique de Clarke et Wright pour le CVRP.
Question 2. Testez votre heuristique sur les instances fournies.

1.5 Giant tour

On se propose d'implémenter l'heuristique du "giant tour". Pour cela, on commence par résoudre avec une méthode heuristique ou exacte le problème de voyageur de commerce sur l'instance de CVRP. On découpe ensuite la tournée en plusieurs tours à l'aide de la programmation dynamique.

Algorithme 2 : Algorithme du tour géant

Résoudre le problème de TSP correspondant à la matrice de distances du VRP ;
La solution est une permutation $v_1 = 0, v_2, \dots v_n$;
Construire un sommet par client ;
Ajouter un arc entre un sommet v_i et v_j $(j > i)$ s'il existe une tournée réalisable passant par les sommet $v_{i+1}, \dots, v_j$. Le coût de l'arc est le coût de la tournée;
Résoudre le problème du chemin de plus petite valeur entre v_1 et v_n ;

Question 3. Implémentez l'heuristique du giant tour pour le CVRP.
Question 4. Testez votre programme sur les instances de CVRP.

2 Bonus : problème de CARP

2.1 Rappel du problème

Données :
— un ensemble d'arcs avec une demande
— un dépôt
— une capacité unique pour les véhicules
Objectif : trouver des tournées pour les véhicules qui minimise la distance totale parcourue tout en visitant tous les arcs avec une demande.

2.2 Transformation d'un problème de CARP en problème de CVRP

Pour transformer un problème CARP en problème CVRP, on procède de la manière suivante.
On crée $3m + 1$ sommets :
— le sommet 0 (dépôt)
— trois sommets par arc (i, j) à visiter
  — s_{ij}
  — m_{ij}
  — s_{ji}
On explique maintenant comment on construit la matrice de distances $W = (w_{ij})$.
La demande de l'arc (i, j) est répartie sur les trois sommets qui le représentent de manière arbitraire.
Les distances entre les sommets sont calculées de la manière suivante. On note $\text{dist}(i, j)$ la valeur d'un chemin de plus petite valeur entre le sommet i et le sommet j dans le graphe initial, et $c(i, j)$ la distance correspondant à l'arc (i, j).

$$w(s_{ij}, s_{kl}) = \begin{cases} \frac{1}{4}\big(c(i,j) + c(k,l)\big) + \text{dist}(i,k) & \text{si } (i,j) \neq (k,l) \\ 0 & \text{sinon} \end{cases} \tag{1}$$

$$w(0, s_{ij}) = \frac{1}{4}c(i,j) + \text{dist}(i,0) \tag{2}$$

$$w(m_{ij}, v) = \begin{cases} \frac{1}{4}c(i,j) & \text{si } v = s_{ij} \text{ ou } v = s_{ji} \\ +\infty & \text{sinon} \end{cases} \tag{3}$$

2.3 Code fourni

On vous fournit un parser pour lire les instances de CARP.
Pour calculer les plus courts chemins, vous pourrez utiliser la bibliothèque de graphes de votre choix.

2.4 Travail à réaliser

Question 5. Implémentez la méthode permettant de transformer une instance de CARP en instance de CVRP.
Question 6. Testez votre méthode sur les instances fournies.