# TP4 - LE VOYAGEUR DE COMMERCE (SUITE ET FIN)

Vous rendrez ce TP sous forme d'une archive contenant fichier .c, .cpp ou .py et un compte-rendu en pdf, par mail le dimanche 28 avril au plus tard. La clarté et les commentaires du code sont évalués. Vous devez implémenter vos tris vous-même. Tout au long de ce TP, $n$ désigne le nombre de sommets de notre graphe, c'est-à-dire, le nombre de villes.

## ALGORITHME GLOUTON

Pour réaliser un algorithme glouton, on représente notre ensemble de villes sous forme d'un graphe, et l'on pondère les arêtes par les distances entre ces villes. On propose l'algorithme suivant :

1. Trier les arêtes du graphe en fonction de leur poids (la distance entre les villes).
2. Tant que l'on n'a pas sélectionné $n$ arêtes, sélectionner l'arête de poids minimal qui ne crée pas de sommet de degré 3, et qui ne ferme pas de cycle (sauf si c'est la dernière arête à ajouter).
3. Renvoyer la somme des poids des arêtes du cycle ainsi créé.

1. Implémenter l'algorithme glouton ci-dessus.
2. Quelle est la complexité de votre algorithme ?

## 2-APPROXIMATION DANS LE CAS EUCLIDIEN

Le but de cette partie est d'implémenter un algorithme efficace qui donne une solution approchée au problème du voyageur de commerce pour les graphes Euclidiens, c'est-à-dire, pour chaque triplet de villes $u, v, w$, on a $dist(u, w) \leq dist(u, v) + dist(v, w)$.
3. Expliquer pourquoi cette hypothèse est raisonnable dans le cas du voyageur de commerce.

Soit $G$ un graphe. Un arbre couvrant de $G$ est un ensemble d'arêtes de $G$ qui induit un arbre (si $G$ a $n$ sommets, c'est un ensemble de $n-1$ arêtes de $G$ qui n'induit pas de cycle).

Pour construire un arbre couvrant de $G$, on considère l'algorithme suivant :

1. On initialise notre arbre $T$ avec un sommet $r_0$ et aucune arête (le choix de $r_0$ n'importe pas).
2. Tant que $T$ n'est pas un arbre couvrant, ajouter à $T$ un sommet qui est à distance minimale de $T$ et l'arête correspondante à cette distance.
3. Renvoyer $T$.

4. Implémenter l'algorithme ci-dessus. On stockera $T$ sous forme d'un tableau $T$ tel que $T[i]$ est un tableau dynamique contenant la liste des enfants de $i$, 0 étant la racine.
5. Montrer que le poids total d'un arbre couvrant de poids minimal est plus petit que la solution optimale au problème du voyageur de commerce.
6. Implémenter un algorithme qui réalise un parcours en profondeur d'abord de l'arbre. Ce parcours est défini récursivement comme suit :

- $parcours(T) = [x]$ si $T = \{x\}$.
- $parcours(T) = [x] + \sum_{1 \leq i \leq k} parcours(f_i)$ si $T$ est un nœud $x$ ayant pour descendant $f_1, \ldots, f_k$.

7. Écrire une fonction *approx* qui prend en entrée un graphe $G$, en extrait un arbre couvrant, calcule un parcours de cet arbre partant de la racine, puis ajoute à ce parcours l'arête allant du dernier sommet du parcours à la racine. On renverra la liste des indices des sommets dans l'ordre d'exploration par le parcours. Cette liste doit donc commencer et se terminer par 0.
8. Montrer que la distance totale de la solution renvoyée par cet algorithme est plus petite que le double de la solution optimale au problème du voyageur de commerce. Discuter de la complexité de votre algorithme.