TSP, approximation et programmation linéaire

Le **problème du voyageur de commerce** (Traveling Salesman Problem, TSP) consiste à trouver le plus court circuit visitant un ensemble de villes exactement une fois et revenant au point de départ. Un tel circuit est appelé *hamiltonien*.

Formellement : étant donné un graphe complet à n sommets et une fonction de distance $d : V \times V \to \mathbb{R}^+$, trouver une permutation σ des sommets qui minimise $\sum_{i=0}^{n-1} d(\sigma(i), \sigma(i+1))$ (avec $\sigma(n) = \sigma(0)$).

On dit que le TSP est **métrique** si les distances vérifient l'inégalité triangulaire : pour tous sommets i, j, k, on a $d(i,k) \leq d(i,j) + d(j,k)$. Cela correspond au cas où les distances sont des vraies distances géométriques.

1. Application : on considère 4 villes A, B, C, D avec les distances suivantes (qui vérifient l'inégalité triangulaire) :

	A	B	C	D
A	0	10	15	20
B	10	0	35	25
C	15	35	0	30
D	20	25	30	0

   Combien y a-t-il de circuits hamiltoniens distincts ? Quel est le coût du circuit optimal ?

2. Proposer un algorithme glouton pour le TSP métrique basé sur la construction d'un arbre couvrant minimal (ACM). Appliquer cet algorithme sur l'exemple précédent.

3. Montrer que cet algorithme fournit une 2-approximation du TSP métrique, c'est-à-dire que le coût de la solution obtenue est au plus deux fois le coût optimal.

4. Montrer que le TSP peut se réécrire comme le problème de minimisation suivant :

$$\begin{array}{ll} \text{minimiser} & \sum_{i<j} d_{ij} x_{ij} \\ \text{sous} & \sum_{j \neq i} x_{ij} = 2 \quad \forall i \in V \\ & \sum_{i,j \in S, i<j} x_{ij} \leq |S| - 1 \quad \forall S \subset V, 2 \leq |S| \leq n-2 \\ & x_{ij} \in \{0,1\} \end{array}$$

5. Expliquez comment relaxer la condition $x_{ij} = \{0,1\}$ en $x_{ij} \in [0,1]$ permet de résoudre géométriquement le problème.

Matroïdes et optimalité gloutonne

Un matroïde est un couple $(E, \mathcal{I})$ où E est un ensemble fini et $\mathcal{I} \subseteq \mathcal{P}(E)$ est une famille de parties de E (appelées indépendantes) vérifiant :

(M1) $\varnothing \in \mathcal{I}$

(M2) Si $I \in \mathcal{I}$ et $J \subseteq I$, alors $J \in \mathcal{I}$ (hérédité)

(M3) Si $I, J \in \mathcal{I}$ avec $|I| < |J|$, alors $\exists e \in J \setminus I$ tel que $I \cup \{e\} \in \mathcal{I}$ (propriété d'échange)

1. Soit $G = (V, E)$ un graphe non orienté. On définit $\mathcal{I} = \{F \subseteq E : F \text{ est acyclique}\}$. Montrer que $(E, \mathcal{I})$ est un matroïde (matroïde graphique).
2. Montrer que pour $U = \sqcup U_i$ et $\mathcal{I}$ l'ensemble des parties contenant au plus un élément dans chacun des U_i on définit un matroïde, dit matroïde de partition.
3. Montrer qu'on peut remplacer la condition (M2) par l'une des deux conditions suivantes sans changer la définition :
   (a) Pour $\tilde{U} \subset U$, tous les ensembles indépendants maximaux dans U ont même cardinal.
   (b) Pour I, J indépendants tels que $|J \setminus I| = 2$ et $|I \setminus J| = 1$ il y a $x \in J \setminus I$ tel que $I + x$ est indépendant.
4. Soit une fonction de poids $w : E \to \mathbb{R}^+$. On considère l'algorithme glouton suivant pour maximiser le poids d'un ensemble indépendant. Montrer que cet algorithme retourne une solution optimale si et seulement si $(E, \mathcal{I})$ est un matroïde.
5. Application : retrouver l'algorithme de Kruskal comme cas particulier.
6. L'intersection de matroïdes est-elle un matroïde ? Donner une application de l'intersection des matroïdes définis ci-dessus pour pour la couverture d'un graphe biparti.

Détection de cycles dans les graphes

You are given an integer n. There is a complete binary tree with $2^n - 1$ nodes. The root of that tree is the node with the value 1, and every node with a value val in the range $[1, 2^n - 1 - 1]$ has two children where :

The left node has the value $2 * val$, and The right node has the value $2 * val + 1$.

You are also given a 2D integer array queries of length m, where $queries[i] = [a_i, b_i]$. For each query, solve the following problem :

— Add an edge between the nodes with values a_i and b_i.

— Find the length of the cycle in the graph.

— Remove the added edge between nodes with values a_i and b_i.