TP voyageur de commerce

Le problème du voyageur du commerce est un problème classique.

Il s'agit de trouver la plus petite distance effectuée en visitant des villes.

On s'interesse à une liste de 22 villes, en choisissant une ville de départ (qui sera la ville d'arrivée). La force brute permet de lister toutes les trajets possibles en partant d'une ville déterminée :

$21\times20\times19\times....\times4\times3\times2\times1 = 51\ 090\ 942\ 171\ 709\ 440\ 000$ soit environ 5×10^{19}

Il faudrait quelques dizaines de millions de disques durs actuels pour stocker ces trajets et il faudrait ensuite calculer les distances associées et récupérer la plus petite...

Techniquement, nous sortons des limites proposées par nos outils, tant en capacité qu'en coût de calcul.

Et il n'y a dans ce problème **que** 22 villes.

Donc nous allons utiliser une autre technique avec une solution la plus interessante à une étape donnée soit un algorithme glouton.

1 - Données du problème : *tpVoyageur.py*

Récupérer le fichier *tpVoyageur.py* et copier dans un dossier *tpVoyageurVotreNom*.

Q1 : quelle instruction dans la console peut-on utiliser pour **vérifier** que *tabDistance* et *tabVille* ont le même nombre d'éléments ?

tabDistance donne un tableau des distances routières entre les villes dont les noms sont dans la liste *tabVille*

Q2 : quelle est la distance proposée entre Auxerre et Bordeaux ?

2 - Distance entre deux villes

Q3 : complète la fonction **distanceAffiche(listA,listB,i,j)** pour qu'elle respecte les instructions de sa documentation

Q4 : même question avec **distanceEntreVille(listA,i,j)**

3 - L'algorithme glouton

On choisit de partir d'Annecy.

Le principe est simple : de manière naturelle, la ville à visiter sera celle la plus proche d'Annecy. C'est une solution **optimale** au sens où l'on ne peut pas faire mieux à ce stade.

Une fois cette ville déterminée, on cherche la ville la plus proche (en excluant Annecy dans notre recherche).

On continue jusqu'à ce que toutes les villes aient été visitées (et on revient à Annecy, histoire que notre voyageur retourne chez lui).

Q5 : applique cet algorithme `à la main` pour déterminer les deux villes visitées en partant d'Annecy.

4 - Programmation en partant d'Annecy

a - La fonction **indice_ville_min(i,listA,listR)**

Le but de cette fonction et de récupérer l'indice de la ville la plus proche de la ville d'indice i avec listA contenant le tableau des distances et listR contiendra la liste des indices des villes déjà visitées.

Q6 : Que vaut donc listR si on doit commencer par Annecy ?

Q7 : Retrouve les résultats de la **Q5** en indiquant les valeurs de listR et les indices obtenus à chaque étape.

b - La liste des villes

Maintenant que l'on a une fonction pour trouver la position de la ville la ville la plus proche à un stade donné, on va l'utiliser dans le pseudo-code suivant :

```
fonction trajet(listA): #listA est le tableau des distances
    indice=0 #on part d'Annecy
    listR=[0] #on initialise le trajet en indiquant que l'on est à Annecy
    pour i allant de 0 à ... : #je ne vais quand même pas tout vous donner
        ajoute indice_ville_min à listR
        indice devient alors indice_ville_min
    fin pour
    on ajoute 0 à listR #à la fin on rentre à Annecy
    retourner listR
```

Q8 : Ajoute cette fonction au fichier tpVoyageur.py et complète avec une documentation et en exemple la liste obtenue

c - La distance

Il s'agit maintenant de déterminer la longueur totale obtenue avec la liste obtenue à la question 8.

Q9 : Ajoute une fonction `distance_totale(listR)` qui permet de calculer la distance totale parcourue grâce au trajet déterminé précédemment. Donne le résultat obtenu.

5 - La meilleure ville de départ.

Quelle ville offre la meileure position de départ ?

Q10 : Modifie ton programme pour qu'il détermine les distances des parcours possibles et donne le nom de la ville la plus interessante et le circuit que doit alors faire le voyageur.