Parcours des graphes

Objectif : L'objectif de ce tp est de rendre l'élève capable :

- charger un graphe avec une structure donnée (liste de listes, dictionnaire de listes, dictionnaire de dictionnaires,...)
- parcourir le graphe en largeur ou en profondeur
- déterminer les distances d'un sommet à tous les autres sommets
- déterminer le plus court chemin entre deux sommets

1 Réseau autoroutier français

Le Tp GRAPHES-Tp-3 a permis de stoker dans une variable `graf`, le réseau autoroutier français.

Q - 1 : *Charger le fichier joint* `autoroutes.py` *et affecter* `graf` *à la variable* `reseau`.

Q - 2 : *Afficher dans une figure* ***Python*** *l'image* `autoroutes.jpg`.

Q - 3 : *Implémenter un algorithme de Dijkstra.*

Q - 4 : *Déterminer les distances des nœuds du reseau par rapport à Dijon.*

Algorithm 1 Algorithme de Dijkstra

```
entrée: graphe, sommet
résultat: d_fin, pere
dijkstra(graphe, sommet)
d_ini = {s : float('inf') for s in graphe}
d_ini[sommet] = 0
d_fin, pere = {}, {}
tant que taille(d_ini) >0 faire
    s_k = mini_dic(d_ini)
    pour s_v dans graphe[s_k] faire
        w = graphe[s_k][s_v]
        si s_v n'est pas dans d_fin et d_ini[s_v] > d_ini[s_k] + w alors
            d_ini[s_v] = d_ini[s_k] + w
            pere[s_v] = s_k
        fin si
    fin pour
    d_fin[s_k] = d_ini[s_k]
    supprime(d[s_k])
fin tant que
renvoi: d_fin, pere
```

2 Recherche du chemin le plus court

Algorithm 2 Plus court chemin entre deux sommets

```
entrée: graphe, depart et arrivee
résultat: chemin
dijkstra_pcc(graphe, depart, arrivee)
d_ini = {s : float('inf') for s in graphe}
d_ini[depart] = 0
d_fin, pere = {}, {}
tant que taille(d_ini) >0 et arrivee n'est pas dans d_fin faire
    s_k = mini_dic(d_ini)
    pour s_v dans graphe[s_k] faire
        w = graphe[s_k][s_v]
        si s_v n'est pas dans d_fin et d_ini[s_v] > d_ini[s_k] + w alors
            d_ini[s_v] = d_ini[s_k] + w
            pere[s_v] = s_k
        fin si
    fin pour
    d_fin[s_k] = d_ini[s_k]
    supprime(d[s_k])
fin tant que
renvoi: trajet(pere, depart, arrivee)
```

Algorithm 3 Obtention du chemin

entrée: `pere`, `depart` et `arrivee`
résultat: `chemin`
trajet(`pere`, `depart`, `arrivee`)

```
chemin = [arrivee]
tant que arrivee différent de depart faire
    arrivee = pere[chemin[-1]]
    ajouter arrivee à chemin
fin tant que
inverser chemin
renvoi: chemin
```

Q - 5 : *Déterminer le plus court chemin pour aller de Dijon à Nevers.*