TP #8 — Graphes #2 : Recherche de plus-court-chemins

Exercice 1. *Saute-canton du misanthrope* (un exercice de J.-B. Bianquis)
Le but de cet exercice est de calculer plusieurs types de plus-court-chemins à travers les communes de France.

1. Téléchargez les fichiers communes.csv et adjacences.csv.
   - `communes.csv` contient, pour chaque commune, un identifiant entier unique, le code INSEE (identfiant alphanumérique unique), le nom, le département et la population ;
   - `adjacence.csv` contient la liste des communes immédiatement adjacentes (l'identifiant utilisé est l'identifiant entier unique du fichier `communes.csv`). Chaque paire de communes adjacentes n'est présente qu'une seule fois (s'il y a une ligne pour x, y, la ligne y, x n'est pas présente).
2. Écrivez une fonction `lire_communes()` qui construit et renvoie une `list` `com` telle que `com[i]` contient un tuple représentant les informations de la commune d'identifiant unique i. On fera bien attention au fait que l'information de population doit être stockée comme un entier. (Au besoin, allez revoir le sujet du TP6 pour une documentation succincte sur la lecture de fichiers textes en Python. Pour convertir une chaîne de caractères `s` « écrivant » un entier en l'entier correspondant, vous pouvez utiliser `int(s)`.)
   (Vous pouvez dans cette fonction « coder en dur » l'emplacement où vous avez sauvegardé le fichier *communes.csv*.)
3. Testez.
4. Écrivez une fonction `construire_graphe()` qui construit et renvoie le graphe non-orienté et non pondéré représentant les liens d'adjacence entre communes. Le graphe renvoyé devra être représenté par « liste de listes d'adjacences ».
   (Si vous trouvez cela plus pratique, vous pouvez ajouter des arguments à *construire_graphe*.)
5. Testez.

Le jeu Saute canton consiste à partir d'une commune aléatoire et à passer de commune adjacente en commune adjacente en essayant d'arriver le plus rapidement possible à une commune d'au moins 50 000 habitants.

6. Écrivez une fonction `saute_canton(com, adj, u)` qui renvoie un chemin de longueur minimale reliant la commune passée en argument à une commune (quelconque) d'au moins 50 000 habitants.

Vous êtes libre de choisir la représentation du chemin renvoyé (par exemple une liste du nom des communes).

Comment procéder. Un simple parcours en largeur convient ici, puisque l'on cherche à minimiser le nombre de sauts (et pas par exemple la distance à vol d'oiseau). Vous pouvez essayer d'implémenter ce parcours (ainsi que la reconstruction de chemin) par vous-même (en utilisant par exemple une `deque` de `collections` pour la file), ou bien adapter le parcours générique du cours.
Bien entendu, il ne sert à rien de continuer le parcours une fois qu'une solution a été trouvée. Si aucune solution n'est trouvée (ce qui est possible), vous pouvez vous contenter de renvoyer une liste vide.

7. Testez (par exemple en jouant à *saute canton*, mais soyez raisonnable !)
8. Déterminez la (ou l'une des) commune la plus « perdue » de France suivant le critère de saute canton (celle pour laquelle le chemin minimal vers une « grande » commune est le plus long possible).

Remarque. On trouve un chemin de longueur 32. Cependant, si vous êtes à jour des dernières (?) fusions de commune vous constaterez que le fichier utilisé dans ce TP n'est plus correct, de même que la solution trouvée :(

On s'intéresse désormais au cas d'un voyageur misanthrope : il souhaite voyager d'une commune A à une commune B (toutes deux fixées), mais tient absolument à rencontrer le moins de personnes possible en route. Autrement dit, il cherche un chemin minimisant la somme des populations des communes traversées.

9. Expliquez comment construire un graphe permettant de résoudre ce problème à l'aide de l'algorithme de Dijkstra.
10. Écrivez une fonction `construire_graphe2(com)` effectuant cette construction
11. Écrivez une fonction `saute_canton_misanthrope(com, adj, u, v)` permettant de résoudre ce problème.
    Si vous le souhaitez, vous pouvez utiliser l'implémentation de file de priorité `PriorityQueue` du module Python queue (notamment les fonctions `PriorityQueue`, `pq.empty`, `pq.put`, `pq.get`). N'hésitez pas à demander de l'aide si nécessaire.
12. Quel chemin conseillez-vous au misanthrope pour relier Villeurbanne à La Mulatière ? Montrouge à Aubervilliers ?

Remarque. Les chemins les plus courts sont respectivement Villeurbanne — Lyon — La Mulatière et Montrouge — Paris — Aubervilliers, mais notre ami misanthrope est prêt à de très longs détours (on trouve des chemins de longueur 22 et 67, respectivement).