

Institut Villebon - Georges Charpak T.P. 2

Savoir faire :

- Savoir effectuer un parcours de graphe.
- Savoir reconnaître une situation où l'algorithme de Dijkstra est utile.
- Savoir implémenter l'algorithme de Dijkstra en Python.

1 Condition de rendu du T.P.

Ce T.P. sera à rendre par mail à olivier.bouillot@villebon-charpak.fr, au plus tard le mardi 8 décembre à 23h59. Il sera à réaliser dans un premier temps seul, pendant la séance, puis à finir en binôme avec un membre de l'autre groupe.

Votre mail d'envoi aura pour sujet : [UE 5i4] RENDU DE TP 2 + **prenom_1** + **prenom_2**, où **prenom_1** et **prenom_2** seront bien entendu remplacé par les prénoms des membres du binômes. Il contiendra une et une seule pièce jointe qui sera une archive nommée **nom_1_nom_2_-_TP_2.zip**, où **nom_1** et **nom_2** correspondent aux noms de familles des membres du groupe, **nom_1** étant avant **nom_2** dans le dictionnaire.

Veillez bien à ne pas modifier les noms donnés dans l'énoncé, car une partie de votre T.P. sera corrigé automatiquement : si vous ne respectez pas cette consigne, vous aurez alors 0, car aucun des tests que votre code subira ne passera...

Enfin, n'oubliez pas de respecter des règles d'hygiènes correctes pour votre code : cela sera hautement pris en compte dans la notation. Voir annexes de l'énoncé du TP 1.

2 Amélioration de la classe Graph

On reprends la classe **Graph** écrite au cours du T.P. 1 pour y ajouter une implémentation de l'algorithme de Dijkstra. Pour cela, nous souhaitons pouvoir considérer des graphes non valués ainsi que des graphes valués aux arêtes.

► **Exercice 1.** Modifier le constructeur de la classe **Graph** (c'est-à-dire la méthode `__init__(self)` de sorte que :

- la signature du constructeur devienne `__init__(self, valued)`
- deux attributs soit ajouté à la classe :
 - un booléen **valued_graph** indiquant au programmeur si le graphe est valué aux arêtes ou non.
 - un dictionnaire **weight**, dont les clés seront des arêtes (donc des couples) et les valeurs associés seront les valeurs données aux arêtes.

Le constructeur doit uniquement permettre de créer le graphe vide et d'initialiser l'attribut `valued_graph` à la valeur passée en paramètre d'appel.

L'attribut `valued_graph` ne sera pas censé être modifié une fois initialisé.

► **Exercice 2.** Modifier la méthode `add_an_edge(self, from_node, to_node)` de la classe `Graph` permettant d'ajouter une arête au graphe courant entre les sommets `from_node` et `to_node` de sorte que :

- la signature de la méthode devienne `add_an_edge(self, from_node, to_node, weight = 0)` :
- l'argument `weight` soit toujours nul dans le cas où le graphe n'est pas valué aux arêtes : une exception de type `TypeError` sera levée dans le cas contraire.
- l'attribut `weight` de la classe soit mis à jour lorsque le graphe est valué aux arêtes.

► **Exercice 3.** Modifier la méthode `__str__(self)` permettant de représenter visuellement un graphe. Le comportement de celle-ci ne doit pas être affecté lorsque le graphe est non valué. Cependant, lorsque le graphe courant est valué, on souhaite que l'appel

```
print(guido_s_graph)
```

produise un affichage du type de celui ci-contre.

```
*****
* Display of the graph *
*****
Nodes:
----
A, E, C, D, F, B

Edges:
----
A ----> B : 5.0
A ----> C : 10.2
B ----> D : 2.1
B ----> C : 0.01
C ----> D : 3.0
D ----> C : 9.6
E ----> F : 2.4
F ----> C : 43.3
=====
```

► **Exercice 4.** Ecrire dans la classe `Graph` une méthode `Dijkstra(self, departure)` qui implémente l'algorithme de Dijkstra en partant du sommet `departure`. Si `departure` n'est pas un sommet du graphe courant, une exception de type `NameError` sera levée. Si le graphe courant n'est pas valué, une exception de type `TypeError` sera levée.

3 Application : un peu de théorie de l'évolution, à la Lewis Carroll

Un doublet, ou *échelle de mots* (Word Ladder Puzzle en anglais) est un jeu inventé par Lewis Carroll. La première mention de ce jeu apparaît dans son journal le 12 mars 1872. Le jeu est ensuite publié

pour la première fois le 29 mars 1879 dans le magazine britannique *Vanity Fair*. Le principe est de trouver une chaîne de mots reliant deux mots donnés, où à chaque étape les mots ne diffèrent que d'une et une seule lettre, sans changer la place des lettres. Par exemple, pour relier en anglais HEAD à TAIL, Lewis Carroll suggère la chaîne suivante :

HEAD
HEAL
TELL
TALL
TAIL

Dans l'article publié, il a été proposé aux lecteurs de l'époque un concours : marquaient le plus de points les joueurs ayant trouvé une chaîne entre deux mots donnés la plus courte possible.

Question 1. *Trouver la plus petite chaîne, en anglais, permettant de passer de FOUR à FIVE.*

Question 2. *Trouver la plus petite chaîne, en français, permettant de passer de SINGE à HOMME.*

Nous proposons dans cette partie d'écrire un programme python réalisant cette tâche pour nous. Pour cela, on travaillera dans un fichier `Word_Ladder_Puzzle.py`.

► **Exercice 5.** Ecrire une fonction `is_a_neighbour(word_1, word_2)` prenant en argument deux chaînes de caractères représentant chacune un mot, et renvoyant `True` si les deux mots diffèrent d'une et une seule lettre, et `False` sinon.

► **Exercice 6.** Ecrire une fonction `load_dictionary(file, length)` permettant de lire un dictionnaire (*i.e.* un fichier texte contenant un mot par ligne), et de retourner le graphe dont les sommets sont les mots de longueur `length` et dont les arêtes relient des mots pouvant être voisins dans une chaîne d'un doublet.

► **Exercice 7.** Ecrire une fonction `doublets(word_1, word_2)` prenant deux mots en arguments qui seront deux mots de la langue française, et qui renvoie la liste constituée de la plus petite chaîne permettant de passer du premier mot au second selon les règles de Lewis Carroll.

► **Exercice 8.** Ecrire un programme acceptant deux arguments sur la ligne de commande, qui seront deux mots de la langue française, et qui affiche en console tous les mots de la chaîne permettant de passer du premier mot au second selon les règles de Lewis Carroll

► **Exercice 9.** Ecrire une fonction `usage()` affichant un texte en console expliquant à l'utilisateur comment utiliser le script précédent.

► **Exercice 10. (Bonus !)** Modifier légèrement le programme pour que celui-ci accepte une option `-a` permettant d'accepter deux mots de la langue anglaise.

Question 3. 1. *Trouver la chaîne passant de SINGE à HOMME.*

2. *Trouver des chaînes sympathiques : un bonus sera accordé au(x) groupe(s) trouvant la plus longue chaîne !!!*