NSI *Projet Koh-Lanta*

par G. Coste

Contexte :

Dans l'émission Koh-Lanta, il y a évidemment l'épreuve mythique des poteaux mais il y a également une belle épreuve (d'immunité ou de confort) qui met en jeu *plusieurs* poteaux !

Chaque aventurier débute perché au sommet d'un poteau (nommé 'A'). Il dispose d'une poutre (de longueur `L`) qu'il peut déplacer et positionner en appui sur 2 poteaux (celui sur lequel il se trouve et celui sur lequel il souhaite aller) à condition qu'ils soient suffisamment proches (leur écartement doit être inférieur ou égal à L). En équilibre sur la poutre, l'aventurier doit passer sur l'autre poteau sans tomber (sinon, il recommence au *poteau initial* 'A').

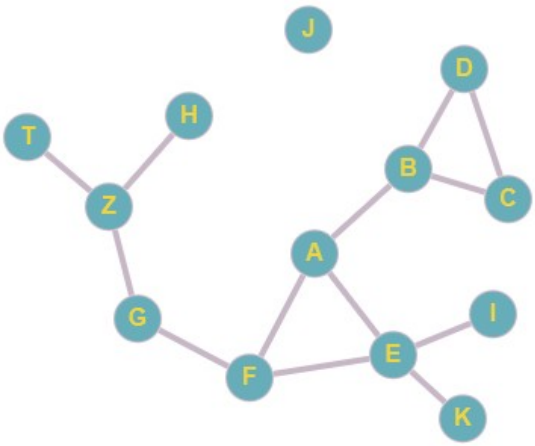
Le but de l'épreuve est d'atteindre en premier, en passant de poteau en poteau, le *poteau cible* (nommé 'Z'). Pour cela, l'aventurier a intérêt à minimiser le nombre de "traversées". En effet, la distance parcourue importe peu car c'est le fait de déplacer la poutre qui lui fait perdre du temps.

La zone de jeu est constituée d'un ensemble de `N` poteaux (tous ne servent pas forcément) dont on connait la position (x ; y).

Objectifs :

Le but de cette activité est de **cartographier** la zone de jeu, c'est-à-dire de tracer le *graphe* de tous les parcours possibles (tous ne mènent pas au poteau 'Z') et d'identifier le chemin (ou les chemins) le plus court (le moins de traversées possible) pour aller de 'A' vers 'Z'.

Voici un exemple de *graphe* :



Dans cet exemple, le chemin le plus court
de 'A' vers 'Z' est : 'A', 'F', 'G', 'Z'

Remarque : Le tracé du graphe ne présume en rien de la position *géographique* des poteaux. Il matérialise uniquement le fait que certains poteaux sont accessibles à partir d'autres poteaux en les reliant par des traits (non fléché car on peut effectuer la traversée dans les 2 sens).

Dans cet exemple, le poteau 'J' n'est accessible depuis aucun autre poteau.

Numérisation du jeu

Chaque poteau est codé par un dictionnaire (dict) ayant 3 clés :

- La clé "`nom`" dont la valeur est une chaine de caractères (`str`) : 'A' ou 'B' ou 'C'...
- La clé "`x`" dont la valeur, *abscisse* du poteau, est un nombre (`float`)
- La clé "`y`" dont la valeur, *ordonnée* du poteau, est un nombre (`float`)

On dispose d'une liste (`list`) dont chaque élément est un poteau (`dict`)
La place des poteaux dans cette liste ne présume en rien de la position géographique du poteau.
Cette liste est appelée `liste_des_poteaux`

La poutre a pour longueur `L` = 3 (unité de mesure arbitraire)

C'est à vous !

Utiliser les fichier ***koh_lanta.py*** pour effectuer les tâches 1, 2, 3 et 4.
La rédaction des *docstrings* et les *doctests* est attendue.

1- Quelle commande python "simple" permet de retourner dans la console (*shell*) le nombre de poteaux constituant la liste `liste_des_poteaux` ?

2- Ecrire la fonction **`distance(poteau1, poteau2)`** qui retourne la distance géométrique séparant le `poteau1` et le `poteau2` passé en paramètres
Rappel : la distance géométrique entre 2 points A et B est $AB=\sqrt{(x_A-x_B)^2+(y_A-y_B)^2}$

3- Ecrire la fonction **`est_accessible(poteau1, poteau2, d)`** qui retourne *`True`* si le `poteau1` et le `poteau2` (passés en paramètres) sont à une distance inférieure au nombre `d` (passé en paramètre)

```
>>> est_accessible(poteauA,poteauB,3)
True
>>> est_accessible(poteauA,poteauZ,3)
False
```

4- Ecrire la fonction **`liste_des_poteaux_voisins(poteau1, liste_complete, d)`** qui retourne la listes des noms des poteaux (parmi les poteaux de la `liste_complete` des poteaux passée en paramètre) accessibles depuis le `poteau1` (passé en paramètre) car ils sont situés à une distance du `poteau1` inférieure à `d` (passée en paramètre)

```
>>>liste_des_poteaux_voisins(poteauA,liste_des_poteaux,L)
['B','C','D','T']
>>>liste_des_poteaux_voisins(poteauJ,liste_des_poteaux,L)
[]
```

Remarque : Vous remarquerez que 'A' n'apparaît pas dans la liste des poteaux accessibles depuis le poteauA.

5- Utiliser la correction du projet « ours->cage » pour déterminer le *chemin optimal* (en termes de nombre de traversées) allant de A à Z.

6- Déterminer la *distance* entre A et Z.