TD : Dictionnaires et programmation dynamique

I] Dictionnaire

Q1. Écrire une fonction `max_dico(D)` qui prend en paramètre un dictionnaire et qui renvoie la valeur maximale.
Indication : les valeurs du dictionnaire pouvant être aussi bien négative que positive et comme il n'y a pas de première valeur on pourra initialiser une variable `M` comme valant $-\infty$ avec : `M=-float('inf')`.
Écrire une fonction `clef_min_dico(D)` qui prend en paramètre un dictionnaire et qui renvoie la clef de valeur minimal.

Q2. On dispose de noms de villes avec les températures moyennes relevées à une date donnée. Les données sont enregistrées dans une liste de listes comme `[["Paris",14],["Lille",12],["Dunkerque",10],...]` On note n la longueur de la liste.

(a) Écrire une fonction `convert(L)` qui prend en argument une telle liste de liste et qui retourne un dictionnaire dont les clefs sont les noms des villes et les valeurs sont les températures.
Quels sont les avantages d'avoir ce type de représentation.

(b) Écrire une fonction `moyenne(d)` qui prend en argument un dictionnaire de température obtenue comme ci dessus et qui retourne la moyenne des températures.

Q3. Comptage des éléments d'une liste à l'aide d'un dictionnaire : Écrire une fonction `comptage` qui prend une liste comme argument et qui retourne un dictionnaire où chaque clé est un élément de la liste et où sa valeur est le nombre d'occurrence dans la liste.
Faire en sorte que la fonction marche aussi avec une chaine de caractères.

Q4. Au Scrabble, on associe à un mot un score égal à la somme des valeurs de chacune de ses lettres, dans sa version française on a des lettres qui comptent pour 1 (A,E,I,L,N,O,R,S,T et U), pour 2 (D, G et M), pour 3 (B, C et P), pour 4 (F, H et V), pour 8 (J et Q) et pour 10 (K, W, X, Y et Z).

(a) Définir un dictionnaire `D` dont les clés sont les lettres de l'alphabet et les valeurs sont les points qu'elles rapportent au Scrabble.
On utilisera des boucles pour définir le dictionnaire (ie on ne tapera pas `D={'A':1, ... }`).

(b) Écrire une fonction `score` qui prend en entrée une chaine de caractère et qui retourne son score associé (on utilisera le dictionnaire de la question (a)).

(c) Qui vaut le plus de points : "EXERCICE" ou "SYZYGIE" ?

Q5. Mettez le fichier `anagrammes.py` et `ods8.txt` (mots officiel du Scrabble dans sa version 8 de 2020, sans accent) dans le même répertoire.
Dans Pyzo, la première exécution du fichier anagrammes.py doit être fait avec Exécuter/Exécuter le script (run/run file as script) pour que le répertoire contenant le fichier py soit le répertoire courant.

(a) Écrire une fonction `est_anagramme()` qui détermine si deux mots sont des anagrammes l'un de l'autre.
Indication : On pourra créer, par exemple, une fonction `mot_to_dic` qui prend en argument un mot et qui retourne un dictionnaire dont les clés sont les lettres du mot et les valeurs sont le nombre d'apparition de la lettre dans le mot

(b) On va maintenant chercher le mot qui a le plus d'anagramme. Pour cela une méthode brutale n'est pas envisageable (la liste `liste_mots` contient 411 430 mots ...).
On définit la signature de la chaine de caractère `mot` comme étant le résultat de l'expression `"".join(sorted(mot))` (les lettres de la signature sont les mêmes, mais misent dans l'ordre alphabétique), ainsi deux mots sont anagramme l'un de l'autre si et seulement si ils ont la même signature.
Écrire une fonction `creer_dic` qui prend en argument une liste de mots et qui retourne deux dictionnaires, le premier où les clés sont les signatures des mots de la liste de mots et les valeurs sont les mots qui ont cette signature, et un second dictionnaire qui a les mêmes clés mais où les valeurs sont le nombre de mots ayant cette signature.
Quelle est le mot qui a le plus d'anagramme ?

II] Mémoïsation

Q6. Écrire une fonction qui calcul $\binom{n}{p}$ à l'aide de la formule de Pascal (et que $\binom{n}{p}$ vaut 1 si p vaut 0 ou n). On procèdera de manière récursif avec mémoïsation.

III] Méthode diviser pour régner

Q7. On considère un nuage de N points qu'on a stocké dans une liste (`L=[[x1,y1],[x2,y2], ...]`) et on suppose que les points sont triés avec l'ordre lexicographique (ie selon les abscisses puis les ordonnées) [1].

1. Si `L` n'est pas triée il suffit de faire `L.sort()` pour la trier avec cet ordre

On recherche deux points qui réalisent la plus petite distance, pour cela on va utiliser l'algorithme récursif suivant :
— si $N \leq 3$ on utilise la méthode brute et on retourne le résultat.
— sinon on considère la sous-liste `Lg` des $N//2$ points situés les plus à gauche et la sous-liste `Ld` complémentaire.
  — On détermine les solutions du même problème sur `Lg` et `Ld` et on note δ_0 la plus petite distance obtenue.
  — La solution au problème est soit δ_0 soit la distance entre un point de `Lg` et un point de `Ld`, pour cela on considère la list `Lm` des points dont l'abscisse est compris entre $x_m - \delta_0$ et $x_m + \delta_0$, où $x_m = \frac{x_g+x_d}{2}$ et où x_g est la plus grande abscisse de `Lg` et x_d la plus petite abscisse de `Ld`, on note δ_1 la plus petite distance entre deux points de `Lm` (on utilise la méthode brute), la solution au problème est donc le minimum entre δ_0 et δ_1.

Pour faire des tests on générera une liste de points aléatoires :

```
def nuage_alea(N,xmax,ymax):
    L=[]
    for k in range(N):
        x=np.random.uniform(0,xmax)
        y=np.random.uniform(0,ymax)
        L.append([x,y])
    L.sort()
    return L
L=nuage_alea(1024,100,100)
```

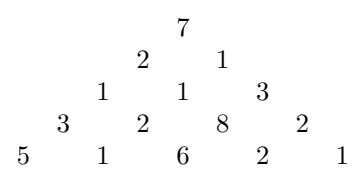
(a) Écrire une fonction `distance` qui prend en argument deux points et qui retourne la distance de ces deux points.
(b) Écrire une fonction `ppdBrute` qui prend en argument une liste de points et qui retourne le couple `R0,d0` où `d0` est la plus petite distance entre deux points de la liste et `R0` une liste de deux points qui réalise cette plus petite distance.
(c) Écrire une fonction récursive `ppd` qui met en œuvre l'algorithme présenté.

IV] Programmation dynamique

Q8. On considère une scierie dont le prix de vente p_i d'une planche de longueur $i \in [\![0, 10]\!]$ est donnée par `prix[i]` où `prix=[0,1,5,8,9,10,17,17,20,24,30]`, Lorsqu'elle reçoit une planche de longueur $n \in [\![1, 10]\!]$ elle souhaite la découper en k planches de longueur $i_1, \ldots, i_k$ avec $i_1 + \ldots + i_k = n$ et donc peut la vendre au prix $p_{i_1} + \ldots + p_{i_k}$. Bien entendue elle désire en tirer le meilleur prix.

(a) Déterminer, à la main (sans ordinateur), les différentes possibilités pour une planche de longueur 4 et en déduire la découpe la plus rentable.
(b) On note r_n le profit maximal que l'on peut avoir pour une planche de longueur n (et on pose $r_0 = 0$), donner une relation de récurrence reliant r_n et des r_i pour $i < n$ (et éventuellement p_i).
(c) Écrire une fonction `coupe(prix,n)` qui retournera le meilleur prix qu'on peut obtenir avec une planche de longueur n.
On procèdera de haut en bas (version mémoïzée) et de bas en haut.
(d) Modifier l'une des fonctions précédentes pour pouvoir construire une solution

Q9. On considère une pyramide de nombres. En partant de celui du haut, le but est de trouver le chemin vers le bas qui maximise la somme de tous les nombres parcourus. Chaque entier peut être suivi d'une des deux cases inférieures.
Par exemple si on considère l'exemple :



On peut espérer obtenir : $25 = 7 + 1 + 3 + 8 + 6$.
Pour représenter la pyramide précédente, on utilise une liste de listes : `P=[[7],[2, 1],[1, 1, 3],[3, 2, 8, 2],[5, 1, 6, 2, 1]]`.
On cherche à écrire une fonction prenant comme argument la liste `P`, une ligne `i` et une colonne `j`, et renvoyant la somme maximale vers le bas de la pyramide à partir du nombre en d'indices i et j. On notera $P_{i,j}$ cette somme maximale.

(a) Dans l'exemple, quel est la valeur de `P[3][1]` ? Quels sont les deux nombres en dessous de `P[i][j]`
(b) Que vaut $P_{i,j}$ si i est l'indice de la dernière ligne ?
(c) À partir de la case (i, j) quelles sont les deux possibilités à envisager ? En déduire une formule de récurrence donnant $P_{i,j}$ en fonction de `P[i,j]`, $P_{i+1,j}$ et $P_{i+1,j+1}$

(d) Écrire la fonction `somme_max(P,i,j)` qui renvoie la la somme maximale vers le bas de la pyramide à partir du nombre d'indices i et j.

(e) Comment obtenir la somme maximale de la pyramide ?

(f) Quel est la complexité de votre fonction, comment l'améliorer ? Le faire.

Q10. Soit X une liste d'entiers à n éléments et soit t un entier. Existe-t-il une sous liste de X telle que la somme vaut t ?