TD7

Soient `ch1` et `ch2` deux chaînes de caractères. On cherche à transformer `ch1` en `ch2` en n'utilisant que trois transformations élémentaires :

- la substitution d'un caractère de `ch1` par un autre caractère de l'alphabet
- l'insertion d'un caractère de l'alphabet à n'importe quel endroit dans `ch1`
- la suppression de n'importe lequel des caractères de `ch1`

La distance de Levenshtein (ou distance d'édition) entre `ch1` et `ch2` est le nombre minimal de transformations nécessaires pour passer de `ch1` à `ch2`.
Par exemple, la distance de Levenshtein de `'distance'` à `'édition'` est 6 :

$$\texttt{'distance'} \overset{\text{supp}}{\to} \texttt{'distanc'} \overset{\text{supp}}{\to} \texttt{'distan'} \overset{\text{ins}}{\to} \texttt{'distaon'} \overset{\text{modif}}{\to} \texttt{'distion'} \overset{\text{supp}}{\to} \texttt{'dition'} \overset{\text{ins}}{\to} \texttt{'édition'}$$

1. Compléter le code suivant de façon à ce que la fonction `dist(ch1:str,ch2:str)->int` renvoie la distance de `ch1` à `ch2`

```
def dist(ch1,ch2) :
    if len(ch1)*len(ch2) == 0 :
        # return
    elif ch1[-1] == ch2[-1] :
        # return
    else :
        r1 = dist(ch1[:-1],ch2)
        r2 = dist(ch1,ch2[:-1])
        r3 = dist(ch1[:-1],ch2[:-1])
        # return
```

2. Pourquoi une telle fonction est-elle inefficace ? Pour illustrer cette inefficacité, vous pourrez examiner son effet sur les chaines `ch1 = 'psimontaigne'` et `ch2 = 'montaignepsi'`.

3. On note, pour $0 \leqslant i \leqslant n =$ `len(ch1)` et $0 \leqslant j \leqslant m =$ `len(ch2)`, $d_{i,j}$ la distance d'édition de `ch1[:i]` à `ch2[:j]`.

a) Que valent $d_{0,j}$ et $d_{i,0}$?

b) Si `ch1[i-1] == ch2[j-1]` que vaut $d_{i,j}$ en fonction de $d_{i-1,j-1}$?

c) Sinon, quelle est la valeur de $d_{i,j}$ en fonction de $d_{i-1,j}$, $d_{i,j-1}$ et $d_{i-1,j-1}$?

d) En déduire une fonction `tableauLEV(ch1:str,ch2:str)->list` qui prend en argument deux chaînes de caractères et qui renvoie le tableau `D` (liste de liste) tel que `D[i][j]` $= d_{i,j}$

e) Écrire une fonction `distLEV(ch1:str,ch2:str)->int` qui renvoie la distance d'édition de `ch1` à `ch2`.

f) Évaluer la complexité de cette fonction en fonction de n et m.

4. Écrire une fonction `transfoLev(ch1:str,ch2:str)->None` qui affiche les transformations à faire pour passer de `ch1` à `ch2`.

Par exemple `transfoLEV('distance','édition')` pourra afficher

```
distance
supp :  distanc
supp :  distan
ins :  distaon
modif :  distion
supp :  dition
ins :  édition
```

En plus des deux indices `i` et `j`, on pourra introduire une variable `reste` contenant une chaîne de caractères (initialement vide) telle que $d_{i,j}$ soit le nombre de transformations à faire pour passer de `ch1[:i]+reste` à `ch2[:j]+reste`.