TP n° 4 d'Informatique
Prog. dynamique : Distance de Levenshtein
Éléments de correction

```python
def levenshtein(s:str, t:str) -> int:
    """
    Calcul de la distance d'édition entre deux mots. Retourne le nombre
    d'opérations élémentaires nécessaires pour passer de s à t.
    """
    n = len(s)
    p = len(t)
    # Création d'un tableau contenant les distances, de taille (n+1) * (p+1)
    # avec d[i][0] = i pour tout i et d[0][j] = j pour tout j
    d = []
    for i in range(n + 1):
        ligne = [i]
        for j in range(p):
            ligne.append(0)
        d.append(ligne)
    for j in range(p + 1):
        d[0][j] = j
    # ou :
    # d = [ [ j for j in range(p+1) ] ]
    # for i in range(n):
    #     L = [ i+1 ] + [0] * p
    #     d.append(L)
    # Remplissage dynamique d'après l'équation de Bellman
    for i in range(1, n+1):
        for j in range(1, p+1):
            if s[i-1] != t[j-1]:
                delta = 1 # Substitution à réaliser entre les deux lettres
            else:
                delta = 0
            d[i][j] = min( d[i-1][j] + 1, d[i][j-1] + 1, d[i-1][j-1] + delta )
    return d[n][p]

# Essai
print('Distance entre les mots "rouge" et "rose" :',levenshtein('rouge','rose'))

def mindico(dico:dict):
    """
    Recherche de la clé de la valeur minimale à l'intérieur d'un dictionnaire
    """
    minimum = float('inf')
    cle_min = []
    for cle in dico:
        if dico[cle] < minimum:
            minimum = dico[cle]
            cle_min = [cle]
        elif dico[cle] == minimum:
            cle_min.append(cle)
    return cle_min
```

```python
# Essai
exemple = {'a':2,'b':4,'c':1,'d':2,'e':1}
print("Clé du minimum du dictionnaire",exemple,":",mindico(exemple))

def autocorrection_simple(mot:str, liste:list) -> str:
    """
    Recherche et retourne l'élément de "liste" le plus proche de "mot"
    """
    # Génération du dictionnaire contenant les distances avec mot
    dico = {}
    for m in liste:
        dico[m] = levenshtein(mot,m)
    # Récupération du mot le plus proche
    return mindico(dico)[0]

# Essai
exemple = ['essai','test','encore','maison']
print("Autocorrection de \"esai\"
   parmi",exemple,":",autocorrection_simple('esai',exemple))

def autocorrection(recherche:str) -> list:
    """
    Recherche et retourne la liste des mots du dictionnaire les plus proches
    de la chaîne "recherche"
    """
    fichier = open("TP4-liste.txt")
    distances = {}
    for mot in fichier:
        mot = mot[0:-1] # Suppression du caractère "retour à la ligne" à la fin
        distances[mot] = levenshtein(recherche,mot)
    fichier.close()
    return mindico(distances)

# Essai
print("Propositions d'autocorrection de 'aison' dans le dictionnaire
   :",autocorrection('aison'))
```

Résultat

```
Distance entre les mots "rouge" et "rose" : 2
Clé du minimum du dictionnaire {'a': 2, 'b': 4, 'c': 1, 'd': 2, 'e': 1} : ['c',
   'e']
Autocorrection de "esai" parmi ['essai', 'test', 'encore', 'maison'] : essai
Propositions d'autocorrection de 'aison' dans le dictionnaire : ['bison',
   'maison', 'raison', 'saison']
```