Correction TD6

1. a) $d_{0,j} = j$ et $d_{i,0} = i$ il faut faire j (ou i) ajouts

b) si `ch1[i-1]=ch2[j-1]`, $d_{i,j} = d_{i-1,j-1}$: les modifications sont à faire sur `ch1[:i-1]` et `ch2[:j-1]` sans modifier le dernier caractère de `ch1[:i]` et `ch2[:j]`

c) Sinon $d_{i,j} = 1+\min\{d_{i-1,j}, d_{i,j-1}, d_{i-1,j-1}\}$: $1+d_{i-1,j}$ correspond à supprimer le dernier caractère de `ch1[:i]`, $1 + d_{i,j-1}$ à supprimer le dernier caractère de `ch2[:j]` (ou à l'ajouter à la fin de `ch1[:i]`) et $1 + d_{i-1,j-1}$ à remplacer le dernier caractère de `ch1[:i]` par le dernier de `ch2[:j]` (qui est `ch2[j-1]`).

d)
```
def tableauLEV(ch1,ch2) :
    n,m = len(ch1),len(ch2)
    T = [[j for j in range(m+1)] for i in range(n+1)]
    for i in range(1,n+1) :
        T[i][0] = i
        for j in range(1,m+1) :
            if ch1[i-1] == ch2[j-1] :
                T[i][j] = T[i-1][j-1]
            else :
                T[i][j] = 1+min([T[i-1][j],T[i][j-1],T[i-1][j-1]])
    return T
```

e) Il suffit de renvoyer le dernier élément de la dernière ligne du tableau précédent

```
def distLEV(ch1,ch2) :
    return tableauLEV(ch1,ch2)[-1][-1]
```

f) La complexité de `tableauLEV` est $O(n \times m)$, celle de `distLEV` aussi.

2. On utilise un dictionnaire pour la mémoïzation, dont les clefs sont les longueurs de sous-chaînes de `ch1` et `ch2` :

```
def distLEV_rec(ch1,ch2) :
    d = {}
    def dist(ch1,ch2) :
        i,j = len(ch1),len(ch2)
        if (i,j) not in d :
            if i*j == 0 :
                r = i+j
            elif ch1[-1] == ch2[-1] :
                r = dist(ch1[:-1],ch2[:-1])
            else :
                r1 = dist(ch1[:-1],ch2)
                r2 = dist(ch1,ch2[:-1])
                r3 = dist(ch1[:-1],ch2[:-1])
                r = 1+min(r1,r2,r3)
            d[(i,j)] = r
        return d[(i,j)]
    return dist(ch1,ch2)
```

3. a) On cherche le chemin permettant de remonter de la case en bas à droite à celle en haut à gauche :

- La distance à trouver est $d_{8,7} = 6$. Comme $d_{8,7} = 1 + d_{7,7}$, on peut commencer par supprimer le dernier caractère de `'distance'` qui devient `'distanc'` et donc la distance à `'édition'` vaut 5
- Comme $d_{7,7} = 1 + d_{6,7}$, on refait pareil : on cherche les 4 transformations pour passer de `'distan'` à `'édition'`
- $d_{6,7} = d_{5,6}$ car les deux dernières chaînes ont le même dernier caractère que l'on ne modifie donc pas et on cherche les 4 transformations pour passer de `'dista'` à `'éditio'`
- $d_{5,6} = 1 + d_{4,5}$ donc cette fois c'est une substitution (du `'a'` par le `'o'`, dernier caractère de `'éditio'`) et on continue avec les 3 transformations pour passer de `'dist'` à `'éditi'`
- $d_{4,5} = 1+d_{4,4}$ donc cette fois, on ajoute le dernier caractère de `'éditi'` et on cherche les 2 transformations pour passer de `'dist'` à `'édit'`
- $d_{4,4} = d_{3,3}$ donc on laisse le dernier caractère et on cherche les 2 transformations pour passer de `'dis'` à `'édi'`
- $d_{3,3} = d_{2,2}$ donc idem et on recommence avec `'di'` et `'éd'`
- $d_{2,2} = 1 + d_{1,2}$ donc on supprime le `'i'` et on recommence avec `'d'` et `'éd'`
- $d_{1,2} = d_{0,1}$, on laisse le `'d'` et il reste `''` et `'é'`
- on est sur le bord du haut ($i = 0$) il reste à ajouter le $j = 1$ caractères de `'é'` à `''`

En réalisant ces opérations à l'envers, on retrouve une chaîne de transformations optimale (elle n'est pas unique).

Il y a des cas ambigus : on a par exemple dans ce tableau $d_{3,3} = d_{2,2} = 1 + d_{2,3}$; l'égalité $d_{3,3} = d_{2,2}$ pourrait laisser penser qu'on peut ne pas toucher au caractère d'indice $2 = 3 - 1$ mais ce n'est pas le cas car ils ne sont pas égaux. Pour lever cette ambiguïté, soit il faut aussi vérifier l'égalité `ch1[i-1]==ch2[j-1]`, soit garder ce cas en dernier recours (donc dans le `else` final).

b) On suit la démarche précédente, en modifiant les caractères de `ch1` au fur et à mesure :

```
def transfLEV(ch1,ch2) :
    n,m = len(ch1),len(ch2)
    T = tableauLEV(ch1,ch2)
    L = [ch1]
    while n>0 and m>0 :
        if T[n][m] == 1+T[n-1][m] :      # on supprime le caractère
                                         # d'indice n-1 de ch1
            ch1 = ch1[:n-1]+ch1[n:]
            n = n-1
            L.append(ch1)
        elif T[n][m] == 1+T[n][m-1] :    # on ajoute le caractère
                                         # d'indice m-1 de ch2 à ch1
            ch1 = ch1[:n]+ch2[m-1]+ch1[n:]
            m = m-1
            L.append(ch1)
        elif T[n][m] == 1+T[n-1][m-1] : # on remplace le caractère
                                         # d'indice n-1 de ch1 par celui
                                         # d'indice m-1 de ch2
            ch1 = ch1[:n-1]+ch2[m-1]+ch1[n:]
            n,m = n-1,m-1
            L.append(ch1)
        else :                           # on a forcément ch1[n-1]==ch2[m-1]
            n,m = n-1,m-1
    if n == 0 :                          # ch1 est vide, on rajoute le
                                         # début de ch2
        while m>0 :
            ch1 = ch1[:n]+ch2[m-1]+ch1[n:]
            m = m-1
            L.append(ch1)
    else :                               # ch2 est vide donc on supprime
                                         # tout ce qui reste
        while n>0 :
            ch1 = ch1[:n-1]+ch1[n:]
            n = n-1
            L.append(ch1)
    return L
```