ISM MP, Informatique
Année 2024/2025

Corrigé du TP Informatique 11

Exercice 1

1. On saisit :

```
def lev1(X,Y):
    """Distance de Levenshtein entre X et Y
    récursivement"""
    n,m=len(X),len(Y)
    if min(n,m)==0:
        return max(n,m)
    elif X[0]==Y[0]:
        return lev_rec(X[1:],Y[1:])
    else:
        X_trunc=X[1:]
        Y_trunc=Y[1:]
        return 1+min(lev1(X_trunc,Y),lev1(X,Y_trunc),lev1(X_trunc,Y_trunc))
```

2. On saisit :

```
def lev2(X,Y):
    """Distance de Levenshtein entre X et Y
    récursivement avec mémoïsation"""
    def lev(X,Y):
        if (X,Y) not in memo:
            n,m=len(X),len(Y)
            if min(n,m)==0:
                res=max(n,m)
            elif X[0]==Y[0]:
                res=lev(X[1:],Y[1:])
            else:
                X_trunc=X[1:]
                Y_trunc=Y[1:]
                res=1+min(lev(X_trunc,Y),lev(X,Y_trunc),lev(X_trunc,Y_trunc))
            memo[(X,Y)]=res
        return memo[(X,Y)]
    memo={}
    return lev(X,Y)
```

3. On a $\forall i \in [\![0\,;n]\!] \quad \mathrm{d}(i,0)=i \quad$ et $\quad \forall j \in [\![0\,;m]\!] \quad \mathrm{d}(0,j)=j$

puisque si l'une des deux chaînes est vide, il faut insérer tous les caractères de l'autre pour réaliser la transformation attendue.

4. Une séquence optimale transformant X en Y transforme également X[:: -1] en Y[:: -1] et réciproquement. Ainsi, on a

$$\mathrm{lev}(\mathrm{X},\mathrm{Y}) = \mathrm{lev}(\mathrm{X}[::-1],\mathrm{Y}[::-1])$$

On en déduit

$$\mathrm{lev}(\mathrm{X},\mathrm{Y}) = 1+\min\left(\mathrm{lev}(\mathrm{X}[:-1],\mathrm{Y}),\mathrm{lev}(\mathrm{X},\mathrm{Y}[:-1]),\mathrm{lev}(\mathrm{X}[:-1],\mathrm{Y}[:-1]) - \delta_{\mathrm{X}[-1],\mathrm{Y}[-1]}\right)$$

et par conséquent

$$\forall (i,j) \in [\![1\,;n]\!]\times[\![1\,;m]\!] \quad \mathrm{d}(i,j) = 1+\min\left(\mathrm{d}(i-1,j),\mathrm{d}(i,j-1),\mathrm{d}(i-1,j-1) - \delta_{\mathrm{X}[i-1],\mathrm{Y}[j-1]}\right)$$

5. On saisit :

```
def lev3(X,Y):
    n,m=len(X),len(Y)
    tab=[[0]*(m+1) for i in range(n+1)]
    for i in range(n+1):
        tab[i][0]=i
    for j in range(m+1):
        tab[0][j]=j
    for i in range(1,n+1):
        for j in range(1,m+1):
            tab[i][j]=1+min(tab[i-1][j],tab[i][j-1],
                            tab[i-1][j-1]-int(X[i-1]==Y[j-1]))
    return tab[n][m]
```

Exercice 2

1. On saisit :

```
def lev_tab(X,Y):
    n,m=len(X),len(Y)
    tab=[[0]*(m+1) for i in range(n+1)]
    for i in range(n+1):
        tab[i][0]=i
    for j in range(m+1):
        tab[0][j]=j
    for i in range(1,n+1):
        for j in range(1,m+1):
            tab[i][j]=1+min(tab[i-1][j],tab[i][j-1],
                            tab[i-1][j-1]-int(X[i-1]==Y[j-1]))
    return tab
```

2. On saisit :

```
def lev_align(X,Y):
    """Détermination d'un alignement optimal des chaînes X et Y"""
    n,m=len(X),len(Y)
    tab=lev_tab(X,Y)
    i,j=n,m
    res="",""
    while i>0 or j>0:
        if tab[i-1][j-1]+int(X[i-1]!=Y[j-1])==tab[i][j]:
            i-=1
            j-=1
            res=X[i]+res[0],Y[j]+res[1]
        elif i>0 and tab[i-1][j]+1==tab[i][j]:
            i-=1
            res=X[i]+res[0],"-"+res[1]
        else:
            j-=1
            res="-"+res[0],Y[j]+res[1]
    return res
```

3. On saisit :

```
def lev_align_all(X,Y):
    """Détermination de tous les alignements optimaux des chaînes X et Y"""
    n,m=len(X),len(Y)
    tab=lev_tab(X,Y)
    file_coord=deque()
    file_align=deque()
    res=[]
    file_coord.append((n,m))
    file_align.append(("",""))
    while len(file_coord)>0:
        coord=file_coord.popleft()
        align=file_align.popleft()
        if coord==(0,0):
            res.append(align)
        else:
            i,j=coord
            if i>0 and tab[i-1][j]+1==tab[i][j]:
                file_coord.append((i-1,j))
                file_align.append((X[i-1]+align[0],"-"+align[1]))
            if j>0 and tab[i][j-1]+1==tab[i][j]:
                file_coord.append((i,j-1))
                file_align.append(("-"+align[0],Y[j-1]+align[1]))
            if i>0 and j>0:
                if (tab[i-1][j-1]+int(X[i-1]!=Y[j-1]))==tab[i][j]:
                    file_coord.append((i-1,j-1))
                    file_align.append((X[i-1]+align[0],Y[j-1]+align[1]))
    return res
```