TP Mias 2 2000-2001, compléments.

Ecrire et tester un logiciel d'alignement de deux chaînes. Prévoir une procédure affichant la suite d'opérations correspondant au coût minimal, si l'utilisateur le demande, ainsi qu'une procédure traduisant une suite d'opérations en un grahe d'alignement, comme par exemple:

```
a)
null(t), null(o), substitution(t->u),null(t), substitution(o->i)
```

s'écrit:

```
totto
|||||
touti
```

```
b)
```

```
null(t), null(o), insertion(u) ,null(t), deletion(t),subst(o->i)
```

s'écrit:

```
to_tto
|||||
tout_i
```

Ci-dessous un rappel du cours:

```
----------------------------
```

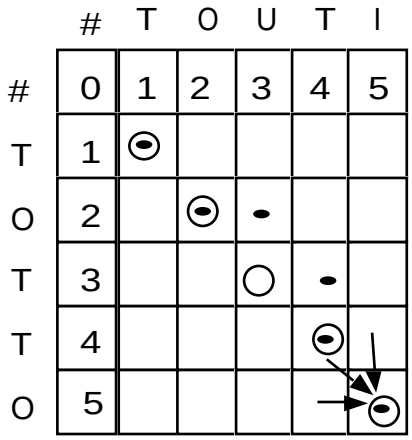
Alignement de deux chaînes par programmation dynamique

L'alignement de deux chaînes finies est la suite d'opérations permettant le passage d'une châine s1 à une chaîne s2. Au début les positions courantes sur les deux chaînes sont les positions 0 (avant le 1er élément).Les opérations sont:
- la substitution d'un caractère de s1 (de coût sub)
- l'insertion d'un caractère dans s1 (de coût ins)
- la délétion d'un caractère dans s1 (de coût del)

Par convention chaque opération correspond à un déplacement dans les deux chaînes.La substitution d'un caractère par lui même est de coût nul et sera notée "Null", les autres substitutions ont un coût sub, les insertions ont un coût ins, les délétions ont un coût del.On supposera par la suite que les substitutions, insertions et délétions ont un coût de 1.

Par exemple on passe de "totto" à "touti" par la suite d'actions
null(t), null(o), substitution(t->u),null(t), substitution(o->i).
qui est de coût = 2, mais aussi par la suite:
null(t), null(o), insertion(u) ,null(t), deletion(t),substitution(o->i)
qui est de coût = 3.

La distance d'édition distEd(s1,s2) est le coût minimum nécessaire pour passer de s1 à s2. Dans l'exemple il est égal à 2, et correspond à la première suite d'actions. Dans la représentation ci-dessous il est représenté par le chemin de ronds blancs (la diagonale). La deuxième suite d'actions correspond à un chemin de coût 3 représenté par les ronds noirs. Tout chemin commence à la case (0,0) correspondant au caractère vide #, et se termine à la case (longueur(s1),longueur(s2))..A la case (i,j) on a le coût du sous-chemin allant de (0,0) à (i,j).



Pour calculer le chemin de coût minimum nous faisons les remarques suivantes:
1) Sur la première ligne le coût en colonne j correspond à j insertions
2) Sur la première colonne le coût en ligne i correspond à i délétions
3) Le coût du meilleur sous-chemin en (i,j) dépend uniquement du sous-chemin correspondant à la case précédente (3 possibilités) et du coût du passage de celle-ci à la case (i,j).

Ce qui donne le principe récursif suivant:

```
coutMin(i,j)=  -  0 si i=j=0
                  i si j=0
                  j si i=0
                  Min (
                       coutMin(i-1,j-1)+ Sub { 0 s1(i)=s2(j) et
psub sinon}
                       coutMin(i,j-1)+ pins {insertion dans s1
à la position i}
                       coutMin(i-1,j)+ pdel     {délétion dans
s1 à la position j}
```

```
                              )
```

```
On a alors distEd(s1,s2) = coutMin(longueur(s1), longueur(s2))
```

On définira distEd telle que distEd(s1,s2) renvoie la distance d'édition entre les chaînes s1 et s2. On définira distEd comme une fonction dans laquelle les variables pins, pdel et psub sont locales.

Pour réellement béneficier de l'intérêt de la programmation dynamique il faut éviter de recalculer coutMin(i,j) plusieurs fois pour la même case (i,j): il faut donc ranger les valeurs calculées, ce qui donne une version réellement efficace. C'est à dire une version où le nombre d'appels à coutMin est taille(s1)*taille(s2) (c'est à dire le nombre de cases du tableau). La version sans mémorisation est exponentielle : chaque appel lance Trois appels qui eux memes lancent trois appels chacun, etc .....
Essayer distEd sur "totto"->"touti" et quelques autres exemples.

```
fonction sub(c1:caractere ; c2:caractere; Sub) : entier
     debut
     si c1=c2 alors
          retourner(0)
     finsi
     retourner (pSub)
     fin
fonction coutMin(var s1:chaîne ; i : entier; var s2: chaîne;
     j:entier; psub : réel; pins: réel; pdel: entier; var M:
     Mémoire):entier
debut
si j =0 alors
     retourner(i*pins)
finsi
si i =0 alors
     retourner(j*pdel)
finsi
Si M[i,j] = -1 alors
M[i,j] = TroisMin   (coutMin(s1,i-1,s2, j-1,psub, pins, pdel, M),
                         +sub(s1[i],s2[j], psub),
                    coutMin(s1,i,s2,j-1,psub, pins,pdel, M)+pins,
                    coutMin(s1,i-1,s2,j,psub, pins,pdel, M)+pdel
                    )
fin


fonction distEd(s1: chaîne; s2: chaîne): entier
```

```
var  M: Memoire
     (* Type Memoire = tableau[0..Nmax, 0..Nmax) de entier*)
Pour i<-0 à Taille(s1) faire
     Pour j = 0 à taille(s2) faire
          M[i,j]<- -1
     finPour
Finpour
retourner(coutMin(s1,taille(s1),s2, taille(s2), 1,1, 1, M),

fin


distEd["t","bt"]
1



1
```