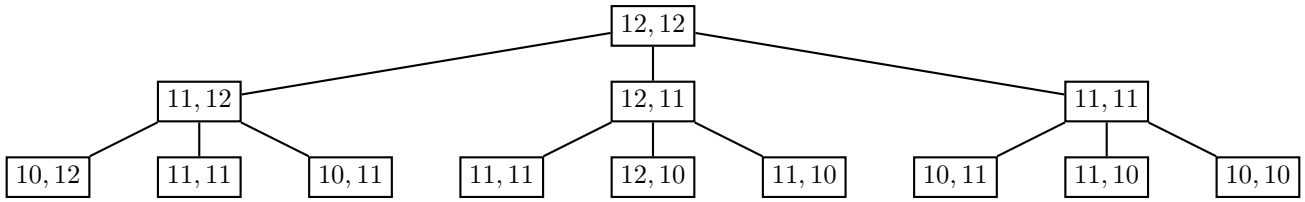
Correction TD7

1.
```
def dist(ch1,ch2) :
    if len(ch1)*len(ch2) == 0 :
        # on renvoie la longueur non nulle : nombre d'ajouts à faire
        return len(ch1)+len(ch2)
    elif ch1[-1] == ch2[-1] :
        # il reste à transformer les chaînes sans leur dernier caractère
        return dist(ch1[:-1],ch2[:-1])
    else : # une modification doit être faite sur les derniers caractères
        # soit on supprime le dernier caractère de ch1
        r1 = dist(ch1[:-1],ch2)
        # soit celui de ch2
        r2 = dist(ch1,ch2[:-1])
        # soit on modifie le dernier caractère de ch1 en le remplaçant par le
            dernier de ch2
        r3 = dist(ch1[:-1],ch2[:-1])
        # et on garde le meilleur des trois cas
        return 1+min(r1,r2,r3)
```

2. Ce programme récursif présente de nombreux problèmes de recouvrement : si on représente les chaînes auxquelles la fonction `dist` est appliquée, on a le schéma suivant (on représente les couples `len(ch1),len(ch2)`)



On voit qu'il faut déjà calculer trois fois `dist('psimontaign','montaigneps')` (cases 11, 11)

3. 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.

4. Il faut remonter depuis la case en bas à droite du tableau vers un bord en retrouvant le chemin qui a amené à la valeur de $d_{i,j}$ et reconnaître la transformation correspondante (en faisant évoluer la variable `reste` en conséquence) :

```
def transfoLEV(ch1,ch2) :
    n,m = len(ch1),len(ch2)
    T = tableauLEV(ch1,ch2)
    reste = ''
    print(ch1)
    while n>0 and m>0 :
        if T[n][m] == 1+T[n-1][m] :
            n -= 1
            print('supp : ',ch1[:n]+reste)
        elif T[n][m] == 1+T[n][m-1] :
            reste = ch2[m-1]+reste
            print('ins : ',ch1[:n]+reste)
            m -= 1
        elif T[n][m] == 1+T[n-1][m-1] :
            reste = ch2[m-1]+reste
            print('modif : ',ch1[:n-1]+reste)
            n -= 1
            m -= 1
        else :
            n -= 1
            m -= 1
            reste = ch1[n] + reste
    while n > 0 :
        n -= 1
        print('supp : ',ch1[:n]+reste)
    while m > 0 :
        reste = ch2[m-1]+reste
        print('ins : ',ch1[:n]+reste)
        m -= 1
```