TP 5 : Dictionnaires et programmation dynamique (partie 2)

1 Traversée d'un tableau de nombres

Soit T un tableau de nombres réels formé de n lignes et p colonnes.
Voici un exemple de tableau qui nous servira dans cet exercice.

$$Ttest = \begin{pmatrix} 1 & 5 & 2 & 5 & 7 & 9 \\ 7 & 3 & 4 & 1 & 2 & 4 \\ 1 & 0 & 4 & 7 & 2 & 1 \\ 2 & 6 & 2 & 1 & 0 & 5 \\ 0 & 1 & 3 & 7 & 9 & 3 \\ 5 & 0 & 8 & 1 & 2 & 5 \end{pmatrix}$$

On souhaite aller du coin (haut, gauche) numéroté $(0, 0)$ au coin (bas, droite) numéroté $(n - 1, p - 1)$ du tableau, en ne se déplaçant que vers la droite ou vers le bas (on parle de chemin admissible traversant), de manière à minimiser la somme des valeurs rencontrées.

1. Pour un tableau de taille 100×200, combien de chemins admissibles traversants existe-t-il ? Sachant que ce nombre est de l'ordre de 10^{80}, que penser d'une exploration exhaustive de tous ces chemins pour déterminer le coût minimal ?

2. a) Sur l'énoncé, surligner le chemin qui correspond par un algorithme glouton (on se déplace toujours vers la case de plus petite valeur). *(on trouve un coût de 30).*

   b) Écrire une fonction `glouton` d'argument un tableau `T` de flottants (liste de listes) et qui retourne la valeur du coût calculé par l'algorithme glouton. *Vérifier que* `glouton(Ttest)` *retourne* `30`.

   c) Surligner d'une autre couleur le chemin obtenu en partant d'en bas à droite et calculer son coût. *L'algorithme glouton ne donne pas le même résultat si l'on part d'une extrémité ou de l'autre et ne donne a priori pas la solution optimale.*

3. On note $c(i, j)$ le coût minimal d'un chemin allant de $(0, 0)$ à (i, j) (pour $0 \leqslant i \leqslant n - 1$ et $0 \leqslant j \leqslant p - 1$).

   a) Justifier $c(i, j) = T_{i,j} + \min\{c(i - 1, j), c(i, j - 1)\}$ en dehors des bords.

   b) Utiliser la mémoïsation pour créer une fonction `coutMinimal` d'arguments `(i,j)` qui renvoie la valeur de $c(i, j)$. *Vérifier que le coût minimal d'un chemin admissible traversant pour* `Ttest` *est* `28`.

4. Utiliser la fonction `coutMinimal` pour déterminer un chemin admissible traversant de coût minimal pour `Ttest`. *On pourra introduire un dictionnaire* `chemins` pour lequel `chemins[(i,j)]` est la liste des coordonnées des cases par lesquelles passer pour joindre (i, j) avec un coût minimal.

5. Déterminer la valeur de la réponse à la question 1. *Vous devez trouver* 92714636861952391188035307164468351845718305590716805098395399271009059717369200 0.

2 Plus longue sous-séquence commune

Une sous-séquence d'une chaîne de caractères est obtenue en retirant certains caractères. Par exemple, `'gothique'` est une sous-séquence de `'algorithmique'` tout comme l'est `'lghqe'`.
On considère deux chaînes de caractères `S` et `T`. Leurs longueurs sont notées respectivement `m` et `n`. On cherche à déterminer la plus longue sous-séquence commune à `S` et `T`.
Pour cela, pour $0 \leqslant$`i`$<$`m`-1 et $0 \leqslant$`j`$<$`n`-1, on note `PLSSC[i][j]` la longueur de la plus longue sous-séquence :

- commune à `S` et `T`,
- terminant avant le caractère de rang `i` (exclu) de `S`,
- terminant avant le caractère de rang `j` (exclu) de `T`.

1. On note `Q` la plus longue sous-séquence commune à `S` et `T` et on note k sa longueur. Pour visualiser, on a `Q`$=$`Q[0]`$\cdots$`Q[k-1]`. Justifier que :
   - si `S[m-1]`$=$`T[n-1]` alors `Q[k-1]`$=$ `S[m-1]`$=$`T[n-1]` et `Q[:k-2]` est la plus longue sous séquence commune à `S[:m-1]` et `T[:n-1]`,
   - si `S[m-1]`$\neq$`T[n-1]` et `Q[k-1]`$\neq$`S[m-1]` alors `Q` est la plus longue sous séquence commune à `S[:m-1]` et `T`,
   - si `S[m-1]`$\neq$`T[n-1]` et `Q[k-1]`$\neq$`T[n-1]` alors `Q` est la plus longue sous séquence commune à `S` et `T[:n-1]`.
2. En déduire une formule de récurrence qui permet d'exprimer `PLSSC[i][j]` en fonction de `PLSSC[i-1][j-1]`, `PLSSC[i][j-1]` et `PLSSC[i-1][j]`.
3. Écrire une suite d'instructions qui retourne la longueur d'une plus grande sous-séquence commune à deux chaînes de caractères `S` et `T` en utilisant la programmation dynamique.