DM - TNSI

Ex 1 (Récursivité vs Programmation dynamique)

On considère un tableau de nombres de n lignes et p colonnes.

Les lignes sont numérotées de 0 à n - 1 et les colonnes sont numérotées de 0 à p - 1.

La case en haut à gauche est repérée par (0, 0) et la case en bas à droite par (n - 1, p - 1).

On appelle chemin une succession de cases allant de la case (0, 0) à la case (n - 1, p - 1), en n'autorisant que des déplacements case par case : soit vers la droite, soit vers le bas. On appelle somme d'un chemin la somme des entiers situés sur ce chemin.

Par exemple, pour le tableau T suivant :

8	2	9	4
5	3	1	10
7	11	6	4

Un chemin est (0,0)-(0,1)-(1,1)-(1,2)-(2,2)-(2,3) et la somme de ce chemin vaut 24

L'objectif de cet exercice est de déterminer la somme maximale pour tous les chemins possibles allant de la case(0, 0) à la case (n - 1, p - 1).

1. En listant tous les chemins possibles allant de (0, 0) à (2, 3) du tableau T, déterminer un chemin qui permet d'obtenir la somme maximale et la valeur de cette somme
2. **(Spé maths uniquement)**

   Montrer qu'un chemin dans un tableau de n lignes et de p colonnes peut être codé par un mot de longueur n+p écrit avec les lettres D (pour aller à droite) et B (pour aller en bas).

   Par exemple le chemin donné plus haut est codé par DBDBD.

   En déduire que le nombre de chemins dans un tableau de n lignes et p colonnes vaut $\binom{n+p}{p} = \binom{n+p}{n}$
3. Par la suite on travaille dans le cas général d'un tableau de n lignes et p colonnes T tel que T[i][j] est un entier naturel pour i $\leqslant$ n-1 et j $\leqslant$ p-1

   On note S(n-1,p-1) la somme maximale pour tous les chemins possibles allant de la case (0, 0) à la case (n - 1, p - 1).

   Justifier en faisant une démonstration par l'absurde que :

   Tout sous-chemin allant de (0,0) à (i,j) avec i $\leqslant$ n-1 et j $\leqslant$ p-1, d"un chemin allant de (0,0) à (n-1,p-1) de somme maximale S(n-1,p-1), **est à son tour forcément optimal de somme S(i,j)**

   (Propriété de la sous-structure optimale)
4. En déduire la formule de récurrence suivante :

   S(0,i) = $\sum_{k=0}^{i}$ T[0][k]

   S(i,0) = $\sum_{k=0}^{i}$ T[k][0]

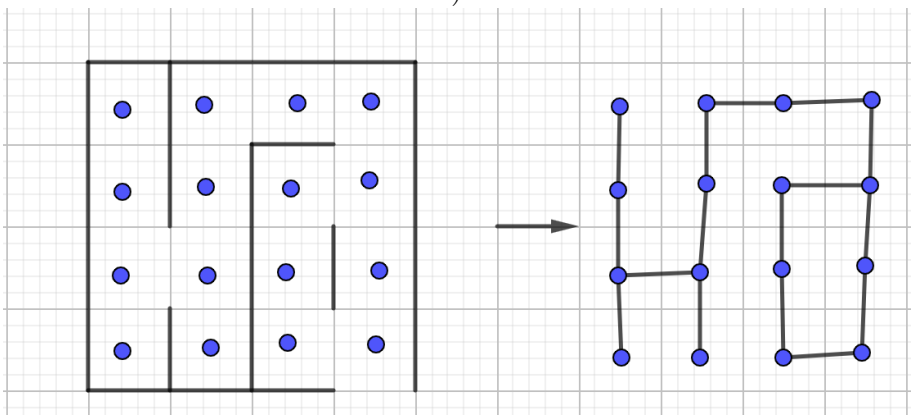
   S(i,j) = max(S(i-1,j) + T[i][j],S(i,j-1) +T[i][j]) = max(S(i-1,j) ,S(i,j-1)) +T[i][j]

5. Ecrire une fonction `somme_max(T,i,j)` récursive qui renvoie la somme maximale pour tous les chemins possibles allant de la case (0, 0) à la case (i, j).
6. Quel appel de fonction doit-on faire pour résoudre le problème initial ?
7. Quel problème pose cette fonction récursive ?
8. Pour remédier au problème précédent on vous demande d'écrire une fonction `somme_max_dyn(T,i,j)` en procédant cette fois ci de manière ascendante (programmation dynamique)
9. La fonction précédente renvoie la somme maximale **mais ne renvoie pas un chemin qui conduit à la somme maximale**.

   Modifier la fonction précédente en ajoutant une variable qui mémorise le chemin parcouru ascendant.

Ex 2 (Graphes et labyrinthes)

Un labyrinthe est représenté par un graphe non orienté de la manière suivante. (La sortie est le sommet en bas à droite.)



Le but est de sortir du labyrinthe en partant de n'importe quel sommet du labyrinthe.

1. Proposer une solution en faisant un parcours en profondeur en partant du point de départ (Ecrire une fonction Python en ce sens)
2. Proposer une solution en faisant un parcours en largeur en partant du point de départ .(Ecrire une fonction Python en ce sens)
3. Comparer les deux méthodes