TD n° 4 : Programmation dynamique

EXERCICE 1 *Nombre de sous-problèmes communs*

Choisir les sous-problèmes adaptés pour résoudre un problème à l'aide de la programmation dynamique peut se révéler difficile et demande de la pratique et de l'intuition. Il y a cependant un certain nombre de cas classiques que l'on rencontre souvent. Donner le nombre de sous-problème pour les trois cas suivants :

1. Le problème est de la forme $x_1, x_2, \dots, x_n$ et les sous-problèmes sont de la forme $x_1, x_2, \dots, x_i$.
2. Le problème est de la forme $(x_1, x_2, \dots, x_n), (y_1, y_2, \dots, y_m)$ et les sous-problèmes sont de la forme $(x_1, x_2, \dots, x_i), (y_1, y_2, \dots, y_j)$.
3. Le problème est de la forme $x_1, x_2, \dots, x_n$ et les sous-problèmes sont de la forme $x_i, x_{i+1}, \dots, x_j$.

EXERCICE 2 *Nombre de chemins dans une grille*

On considère un quadrillage $n \times p$. On part du coin inférieur gauche (point A de coordonnées $(0, 0)$) et on cherche à rejoindre le coin supérieur droit (point B de coordonnées (n, p)). Les déplacements autorisées sont : horizontal vers la droite ; vertical vers le haut ; diagonal dans la direction supérieure droite.

Écrire une fonction `nb_chemins : int -> int -> int` telle que `nb_chemins n p` calcule le nombre de chemins différents allant de A à B.

EXERCICE 3 *Project Euler n° 67*

En partant du sommet du triangle ci-dessous et en se déplaçant vers un nombre adjacent de la ligne inférieure, la somme totale que l'on peut obtenir en reliant le sommet à la base est égal à 23.

$$\begin{array}{ccccccc} & & & 3 & & & \\ & & 7 & & 4 & & \\ & 2 & & 4 & & 6 & \\ 8 & & 5 & & 9 & & 3 \end{array}$$

Écrire une fonction CAML permettant de calculer la somme maximale que l'on peut obtenir par un chemin reliant le sommet à la base d'un tel triangle de hauteur n. Les valeurs du triangle seront représentées par un tableau `triangle` bi-dimensionnel $n \times n$. Par exemple la case `triangle.(i).(j)` contient la $(j+1)^e$ valeur de la $(i+1)^e$ ligne. On ignore simplement les cases inutiles.

EXERCICE 4 *Carrés de* 1 *dans une matrice*

On considère une matrice $M = (m_{i,j}) \in \mathcal{M}_{n,p}(\mathbb{R})$ à coefficients dans $\{0, 1\}$ et on cherche le plus grand « carré » composé uniquement de 1 dans cette matrice.

Par exemple le plus grand carré de 1 de la matrice suivante est de côté 3 (lignes 2, 3 et 4 et colonnes 3, 4 et 5) :

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Décrire un algorithme, linéaire en la taille de la matrice, résolvant ce problème.

EXERCICE 5 *Partie de somme maximale*

On considère un ensemble $E = \{n_1, n_2, \dots, n_k\}$ d'entiers strictement positifs et un entier $M \in \mathbb{N}^*$. Décrire un algorithme qui calcule la partie de E dont la somme est maximale parmi les parties de E dont la somme est inférieure ou égale à M. On convient que la somme d'une partie vide est nulle. Quelle est sa complexité temporelle et spatiale ?