Feuille d'exercices 6
Programmation dynamique

Exercice 1 Chemin de poids maximal

Dans l'exemple du cours du chemin de poids maximal reliant le bord haut gauche au bord bas droit, compléter le code du cours afin que la fonction renvoie le poids maximal ainsi que la liste des couples d'indices des cases parcourues :

```
def matricePoids(A):
    n, m = len(A), len(A[0])
    M = [ [0] * m for k in range(n) ]
    M[0][0] = A[0][0]
    for i in range(1,n):
        M[i][0] = M[i-1][0] + A[i][0]
    for j in range(1,m):
        M[0][j] = M[0][j-1] + A[0][j]
    for i in range(1,n):
        for j in range(1,m):
            M[i][j] = A[i][j] + max(M[i-1][j],M[i][j-1])
    Chemin = [(n-1,m-1)]
    # ...
    # à completer
    #...
    return ..., Chemin
```

Exercice 2 Rendu de monnaie

Le problème du rendu de monnaie consiste à déterminer le nombre minimal de pièces/billets à utiliser dans un système monétaire donné pour produire une somme s donnée.
Les paramètres en sont la somme s à produire, et la liste L des valeurs des pièces et billets dans le système monétaire considéré. Par exemple en euros la liste L est :

$$L = [0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500]$$

La fonction f renvoyant le nombre minimal de pièces/billets à utiliser pour atteindre une somme s, avec la liste L des valeurs du système monétaire, vérifie la relation de récurrence :

$$f(0, L) = 0 \quad \text{et} \quad f(s, L) = 1 + \min\{f(s - v, L) \mid s - v \geqslant 0,\ v \in L\}$$

1. Utiliser cette relation de récurrence pour écrire le code par programmation dynamique qui renvoie le résultat de $f(s, L)$.
2. Écrire une version de la fonction utilisant un algorithme glouton du plus grand d'abord. Vérifier sur des exemples que le résultat retourné n'est pas optimal.

Exercice 3 Le problème du sac à dos

Le problème consiste, étant donnés n objets de valeurs $v_1, \ldots, v_n$ et de poids respectifs $w_1, \ldots, w_n$ de remplir un sac à dos avec une partie I de ces objets en maximisant la valeur transportée sous la

contrainte que la somme des poids soit inférieure à la capacité $W_{max} \in \mathbb{N}$ du sac à dos. I s'agit donc de déterminer une partie $I \subset [\![1, n]\!]$ réalisant le maximum :

$$\max_I \sum_{i \in I} v_i \quad \text{tel que} \quad \sum_{i \in I} w_i \leqslant W_{max}$$

1. Résoudre le problème à l'aide d'un algorithme glouton ; le critère de priorité pour les objets à prendre (ou heuristique) consiste à choisir d'abord les objets avec $\frac{v_i}{w_i}$ maximal.
2. Résoudre le problème par programmation dynamique. Si $f(k, W)$ désigne la valeur maximale que l'on peut atteindre avec les k premiers objets et une capacité de W, alors si l'objet k figure dans la liste d'objets optimale alors $w_k \leqslant W$ et $f(k, W) = v_k + f(k-1, W - w_k)$ et s'il n'y est pas alors $f(k, W) = f(k-1, W)$. On en déduit :

$$f(k, W) = \begin{cases} \max(v_k + f(k-1, W - w_k), f(k-1, W)) & \text{si } w_k \leqslant W \\ f(k-1, W) & \text{sinon} \end{cases}$$

   (a) Écrire le code de la fonction, procédant de bas en haut par itération en complétant une matrice de taille $(n+1) \times (W_{max}+1)$ et renvoyant la valeur maximale $f(n, W_{max})$ pouvant être transportée.

   (b) Écrire le code de la fonction procédant de haut en bas par récursivité et avec mémoisation à l'aide d'un dictionnaire et renvoyant la valeur maximale $f(n, W_{max})$ pouvant être transportée.

   (c) A l'aide du dictionnaire complété par la fonction précédente, écrire une fonction qui renvoie la liste des objets à transporter pour une solution optimale.

3. Confronter les deux approches pour vérifier que l'approche gloutonne proposée est non optimale mais très performante.