? À rendre mercredi 26 novembre 2025
Devoir Maison n°4 : Programmation dynamique

- Le DM devra être rendu en **manuscrit**. Je vous encourage à y réfléchir dessus avec un crayon et un papier, sans ordinateur.
- Les algorithmes devront être commentés.
- Pour toute question, n'hésitez pas à me contacter par mail : nvalade.pcsi@gmail.com

À l'approche des fêtes de fin d'année, vous allez devoir préparer votre sac/valise pour y aller. Vous n'avez qu'un petit sac, et vous voulez emmener de nombreux objets plus ou moins utiles et plus ou moins encombrants : cours, livres, cadeaux, nourritures, ... Se pose donc la question : quels objets prendre dans votre sac, et quels objets abandonner dans votre chambre universitaire / d'internat,...

Dans la suite, on considère n objets, de valeurs v_0,..., v_{n-1} et de masses m_0, ..., m_{n-1}, ainsi qu'un sac à dos de capacité W. On considérera que la capacité, les masses et les valeurs sont des entiers strictement positifs.

On cherche alors à déterminer une partie I de $[\![0, n-1]\!]$, telle que $\sum_{i\in I} m_i \leq W$, et pour laquelle $\sum_{i\in I} v_i$ est maximale.

Les parties II et III sont indépendantes l'une de l'autre, mais dépendent toutes les deux de la partie I.

Partie I Algorithme glouton

C'était l'objet de l'exercice n°3 du DM n°3.

Partie II Programmation dynamique : mise en équation

Pour tout $w \in \mathbb{N}$, et tout $i \in [\![0, n]\!]$, notons $V(i, w)$ la valeur maximale des objets qu'il est possible de mettre dans un sac à dos de capacité w en ne prenant que des objets parmi les i premiers (c'est-à-dire dont les indices sont dans $[\![0, i-1]\!]$).

Q1. Donner en justifiant la valeur de $V(i, 0)$ et de $V(0, w)$ pour tout $w \in [\![0, W]\!]$, et tout $i \in [\![0, n]\!]$.

Q2. Justifier que pour tout $w \in \mathbb{N}$, et tout $i \in [\![0, n-1]\!]$:

— si $m_i > w$, alors $V(i+1, w) = V(i, w)$;

— sinon $V(i+1, w) = \max\left\{V(i, w), v_i + V(i, w - m_i)\right\}$

Partie III Programmation de bas en haut

Dans cette partie, on on décrira les matrices à coefficients entiers comme des listes de lignes, chaque ligne étant une liste d'entiers. Par exemple, la matrice $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ sera codée en `python` ainsi `M = [[1,2,3],[4,5,6]]`.

Q3. Dans cette question seulement, on considère que $W = 6$, $m = [3, 3, 2]$ et $v = [4, 2, 3]$.

Recopier et compléter le tableau suivant, donnant la valeurs des $V(i, w)$, pour $0 \leq i \leq 3$ et $0 \leq w \leq 6$.

	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$
$i=0$							
$m_0=3$ $v_0=4$ $i=1$							
$m_1=3$ $v_1=2$ $i=2$							
$m_2=2$ $v_2=3$ $i=3$							

Q4. Écrire une fonction `matrice_nulle(n:int,p:int)->list[list]` prenant en argument deux entiers strictement positifs `n` et `p` et renvoyant la matrice de `n` lignes et `p` colonnes dont tous les coefficients sont nuls.

Q5. Écrire une fonction `matrice_sac_a_dos(v:list, m:list, W:int)->list[list]` prenant en argument la liste `v` des valeurs, la liste `m` des masses et la capacité `W` totale du sac à dos, et renvoyant une matrice `V` de taille $(n+1, W+1)$ telle que pour tout couple $(i, w) \in [\![0, n]\!] \times [\![0, W]\!]$, `V[i][w]` contienne la valeur de $V(i, w)$.

Indication : on remarquera que, après avoir initialisé `V` comme étant la matrice nulle, au vu de la question Q4, il suffit de parcourir cette matrice ligne par ligne (de haut en bas), puis de parcourir chaque ligne de gauche à droite

Q6. En déduire une fonction `valeur_maximale(v:list, m:list, W:int)->int` prenant en argument la liste `v` des valeurs, la liste `m` des masses et la capacité `W` totale du sac à dos et renvoyant la valeur maximale des objets qu'on peut mettre dans le sac à dos.

Q7. On souhaite connaître un ensemble d'objets permettant d'atteindre cette valeur maximale.

Pour cela, on se base sur la matrice `V` construite par la fonction `matrice_sac_a_dos`.

En effet, pour tout $w \in \mathbb{N}$ et tout $i \in [\![0, n-1]\!]$, si $V(i+1, w) = V(i, w)$, cela signifie qu'on ne prend pas l'objet i pour atteindre la valeur maximale pour la capacité w.

Compléter sur le document réponse la fonction `contenu_sac_a_dos(v:list,m:list,W:int)->list` prenant en argument la liste `v` des valeurs, la liste `m` des masses et la capacité `W` totale du sac à dos, et renvoyant la liste des indices correspondant aux objets à prendre pour atteindre la valeur maximale en utilisant la fonction `matrice_sac_a_dos`.

Partie IV Programmation de haut en bas avec mémoïsation

Dans cette partie, nous allons coder les matrices par des dictionnaires. Pour une matrice M, les clefs du dictionnaire seront les couples (i, j) d'indices, dont la valeur associée sera $M_{i,j}$. Les cases non présentes dans le dictionnaire seront considérés comme étant vides.

Par exemple, la matrice $\begin{pmatrix} 1 & . & 3 \\ 4 & . & . \end{pmatrix}$ où . désigne une case vide, sera codée en `python` de la manière suivante :

`M = { (0,0):1 , (0,2):3 , (1,0):4}` .

Q8. Écrire une fonction `initialisation(v:list, m:list, W:int)->dict` renvoyant la matrice des valeurs des $V(i, w)$ trouvées à la question Q1, pour $w = 0$ et $0 \leq i \leq n$ ainsi que pour $i = 0$ et $0 \leq w \leq W$.

Q9. **Compléter sur le document réponse** la fonction `valeur_maximale_memo(v:list, m:list, W:int)->int` prenant en argument `v` des valeurs, la liste `m` des masses et la capacité `W` totale du sac à dos et renvoyant la valeur maximale des objets que l'on peut mettre dans le sac à dos en utilisant une fonction récursive optimisée par mémoïsation.

DOCUMENT RÉPONSE À RENDRE

NOM : _____ Prénom : _____

Q7

```
def contenu_sac_a_dos(v:list,m:list,W:int)->list :
    objets_a_prendre = [] # liste des numéros des objets à prendre
    V = ..... # la matrice complétée
    w = ..... # initialisation de capacité du sac restante
    i = ..... # on part du dernier objet
    while ..... and ..... : # il reste des objets à étudier et de la place
        if ..... != ..... : # l'objet i est pris si ...
            ..... # on prend l'objet i
            w = ..... # la capacité restante diminue
        i = ..... # on remonte dans le tableau
    return objets_a_prendre
```

Q9

```
def valeur_maximale_memo(v:list, m:list, W:int)->int:
    """ fonction qui renvoie la valeur maximale des objets pouvant être emportés compte tenu de la capacité W du sac, de la liste m des masses et v des valeurs de ces objets """
    V=.....  # initialisation du dictionnaire qui sert à la mémoïsation
    def mem(i,w):
        """fonction récursive qui renvoie la valeur de V(i,w)"""
        if ..... :  # si V(i,w) a déjà été calculé
            return  .....
        else:
            if ..... : # si i ne peut pas être emporté
                .....
            else: # si i peut être emporté
                .....
                .....
                .....
            V[(i,w)]=..... # mémoïsation
            return
    return ..... # quel élément du dictionnaire V doit être renvoyé ?
```