Devoir surveillé d'informatique

Remarques et consignes importantes

- On pourra toujours librement utiliser une fonction demandée à une question précédente même si cette question n'a pas été traitée.
- Veillez à présenter vos idées et vos réponses partielles même si vous ne trouvez pas la solution complète à une question.
- La clarté et la lisibilité de la rédaction et des programmes sont des éléments de notation.

❑ Exercice 1 : *Problème du sac à dos*

On dispose d'un sac à dos pouvant contenir un poids maximal noté P et de n objets ayant chacun un poids $(p_i)_{0\leqslant i\leqslant n-1}$ et une valeur $(v_i)_{0\leqslant i\leqslant n-1}$. On cherche à remplir le sac à dos de manière à maximiser la valeur totale des objets contenus dans le sac sans dépasser le poids maximal P. Par exemple, si on dispose des objets suivants :

— un objet de poids $p_0 = 4$ et de valeur $v_0 = 20$,
— un objet de poids $p_1 = 5$ et de valeur $v_1 = 28$,
— un objet de poids $p_2 = 6$ et de valeur $v_2 = 36$,
— un objet de poids $p_3 = 7$ et de valeur $v_3 = 50$,

et qu'on suppose que le poids maximal du sac est 10 alors un choix possible serait de prendre l'objet 3, aucun autre objet ne rentre alors dans le sac et la valeur du sac est de 50 avec un poids de 7. Une autre possibilité plus intéressante serait de choisir les objets 0 et 2, la valeur totale serait alors de 56 et le poids du sac de 10.

Dans toute la suite de l'exercice on supposera que les poids et les valeurs des objets sont fournis sous la forme d'une liste de Python contenant les tuples `(poids, valeur)` représentant les objets. Par exemple, les objets précédents seraient représentés par la liste suivante :

```python
objets = [(4, 20), (5, 28), (6, 36), (7, 50)]
```

On propose de représenter un choix d'objets par une liste contenant des 0 et des 1. Si le i-ème élément de la liste vaut 1 alors l'objet i est choisi, s'il vaut 0 alors l'objet i n'est pas choisi. Par exemple, pour les objets précédents, le choix de prendre uniquement l'objet 3 serait représenté par la liste `[0, 0, 0, 1]` et le choix de prendre les objets 0 et 2 serait représenté par la liste `[1, 0, 1, 0]`.

■ Partie I : Approche par recherche exhaustive

La recherche exhaustive consiste à énumérer tous les choix possibles d'objet et à calculer la valeur ainsi que le poids pour chaque choix, on retient alors le choix qui maximise la valeur du sac sans dépasser le poids maximal.

Q1– Ecrire une fonction `maximum` qui prend en argument une liste non vide d'entiers et renvoie le maximum des valeurs de cette liste.

```python
def maximum(liste):
    m = liste[0]
    for x in liste:
        if x>m:
            m = x
    return m
```

Q2– Justifier rapidement que le nombre possible de choix d'objet est 2^n.

Q3– Ecrire une fonction `valeur_poids` qui prend en arguments, une liste d'objets ainsi qu'un choix d'objet (sous la forme indiquée ci-dessus) et qui renvoie le poids et la valeur du sac correspondant à ce choix. Par exemple avec la liste `objets` donnée en exemple plus haut, `valeur_poids(objets, [1, 0, 1, 0])` doit renvoyer $(56, 10)$.

Q4– Donner la complexité de la fonction `valeur_poids` en fonction de n.

Q5– En déduire la complexité d'une méthode qui pour chaque choix possible d'objet calculerait la valeur du sac ainsi que son poids et renverrait le choix optimal.

■ **Partie II** : Stratégie gloutonne

On considère la stratégie gloutonne suivante : on trie les objets par ordre décroissant de leur rapport valeur/poids et on les prend dans cet ordre jusqu'à ce que le poids maximal soit atteint.

Q6– Vérifier qu'en appliquant cette stratégie à la liste d'objets :

```
[(4, 30), (5, 34), (6, 36), (7, 49), (10, 74)]
```

on n'obtient pas la meilleure solution.

■ **Partie III** : Approche par programmation dynamique

On propose de résoudre le problème du sac à dos par programmation dynamique. Pour tout $i \in [\![0;n]\!]$, on note $V(i,p)$ la valeur maximale que l'on peut obtenir avec les objets i à $n-1$ et un poids maximal p.

Q7– Donner $V(i,0)$ pour $i \in [\![0;n-1]\!]$ et $V(n,p)$ pour $p \in [\![0;P]\!]$.

Q8– Donner une relation de récurrence liant $V(i,p)$ à $V(i+1,p)$ et $V(i+1,p-p_i)$.
Indication : on pourra considérer deux cas, celui où l'objet i est n'est pas pris et celui où il l'est (dans ce cas on a nécessairement $p \geqslant p_i$).

Q9– Écrire une fonction récursive `sac_dynamique` qui prend en arguments, une liste d'objets, un indice i et un poids maximal p et qui renvoie la valeur maximale que l'on peut obtenir avec les objets i à $n-1$ et un poids maximal p.

Q10– Proposer une version de la fonction précédente utilisant la mémoïsation afin de ne pas recalculer les instances du problème déjà résolues.