TP Informatique 10

Problème

On complètera directement le fichier `TP10.py` en ligne sur le site de la classe.

Dans ce problème, on cherche à remplir un sac-à-dos d'une capacité maximale notée C (poids maximal supporté par le sac-à-dos) avec des objets caractérisés par leur utilité et leur poids en maximisant le remplissage, c'est-à-dire en maximisant la somme des utilités des objets placés dans le sac. Tous les poids considérés sont des entiers.

On note $\mathrm{X}_n = [(v_1, p_1), \ldots, (v_n, p_n)]$ la liste des objets considérés avec n entier non nul. Pour $k \in [\![1 \,;\, n]\!]$, la valeur v_k désigne l'utilité du k-ième objet et p_k désigne son poids. On suppose C entier et

$$\forall k \in [\![1 \,;\, n]\!] \qquad v_k \geqslant 0 \quad \text{et} \quad p_k \in \mathbb{N}^*$$

Le *problème du sac-à-dos* (*KP*, *Knapsack Problem*) consiste à réaliser

$$\mathrm{KP}(\mathrm{X}_n, \mathrm{C}) = \ \mathrm{Max}\ \left\{ \sum_{k=1}^{n} x_k v_k,\ (x_1, \ldots, x_n) \in \{0,1\}^n \ :\ \sum_{k=1}^{n} x_k p_k \leqslant \mathrm{C} \right\}$$

On note $\mathrm{X}_i = [(v_1, p_1), \ldots, (v_i, p_i)]$ pour $i \in [\![1 \,;\, n]\!]$ et $\mathrm{X}_0 = \varnothing$. On a le résultat suivant :

Proposition 1. *On a*

$$\mathrm{KP}(\mathrm{X}_n, \mathrm{C}) = \begin{cases} 0 & \text{si } n = 0 \\ \mathrm{KP}(\mathrm{X}_{n-1}, \mathrm{C}) & \text{si } p_n > \mathrm{C} \\ \max(\mathrm{KP}(\mathrm{X}_{n-1}, \mathrm{C}), v_n + \mathrm{KP}(\mathrm{X}_{n-1}, \mathrm{C} - p_n)) & \text{sinon} \end{cases}$$

On définit la matrice $\mathrm{T} = \left(t_{i,j}\right)_{(i,j) \in [\![0 \,;\, n]\!] \times [\![0 \,;\, \mathrm{C}]\!]}$ par

$$\forall (i,j) \in [\![0 \,;\, n]\!] \times [\![0 \,;\, \mathrm{C}]\!] \qquad t_{i,j} = \mathrm{KP}(\mathrm{X}_i, j)$$

Toutes les fonctions pourront être testées pour une capacité maximale $\mathrm{C} = 15$ et la liste d'objets

```
X=((1,2),(2,5),(3,7),(7,12),(10,9),(11,15),(1,1),(2,1))
```

On saisit `X` en tant que `tuple` qui est un type hachable mais on pourra le confondre avec une liste de couples.

1. Écrire une fonction `KP_rec(X,C)` d'arguments `X` une liste de couples (utilité,poids) et `C` une capacité maximale entière qui renvoie la valeur $\mathrm{KP}(\mathrm{X}, \mathrm{C})$ en réalisant le calcul de manière descendante sans mémoïsation.
2. Écrire une fonction `KP_memo(X,C)` d'arguments `X` une liste de couples (utilité,poids) et `C` une capacité maximale entière qui renvoie la valeur $\mathrm{KP}(\mathrm{X}, \mathrm{C})$ en réalisant le calcul de manière descendante avec mémoïsation.
3. Préciser $t_{0,j}$ pour $j \in [\![0 \,;\, \mathrm{C}]\!]$ puis déterminer une relation entre $t_{i,j}$ et des valeurs de la matrice T d'indices de lignes et/ou colonnes inférieurs.

4. Écrire une fonction `KP_tab(X,C)` d'arguments `X` une liste de couples (utilité,poids) et `C` une capacité maximale entière qui renvoie la matrice T en la construisant de manière ascendante.
5. Écrire une fonction `KP_comp(X,C)` d'arguments `X` une liste de couples (utilité,poids) et `C` une capacité maximale entière qui renvoie une composition optimale du sac en utilisant le résultat de `KP_tab(X,C)`.
6. Compléter la fonction `KP_all(X,C)` d'arguments `X` une liste de couples (utilité,poids) et `C` une capacité maximale entière pour qu'elle renvoie toutes les compositions optimales du sac.