Le problème du Sac-à-Dos.

Le problème du Sac-à-Dos est un problème classique en informatique. Il modélise une situation analogue au remplissage d'un sac. Une personne veut remplir un sac à dos ne pouvant pas supporter plus d'un certain poids $C \in \mathbb{N}$, et elle dispose de n objets (On note l'ensemble des objectspar $\mathcal{O} = \{1, \ldots, n\}$). Chaque objet i a une valeur $v_i \in \mathbb{N}\backslash\{0\}$ et un poids $p_i \in \mathbb{N}\backslash\{0\}$. Le problème est de trouver un ensemble d'objets tel que

- tous les objets de cet ensemble puissent être mis dans le sac.
- la somme des valeurs de ces objets soit maximale.

Exercice 1 Programmation dynamique

Nous allons construire un tableau T dans lequel les lignes seront indexées par les objets et les colonnes par les valeurs. L'élément $T[i, j]$ représentera la valeur maximale pour un sac-à-dos de capacité j à l'aide des i premiers objets.

Question 1.1 Donner la formule de récurrence.

Correction: Soit OPT une solution optimale avec un sac à dos de capacité C et un ensemble $\mathcal{O}$ de n éléments. Notons $T[i, j]$ représentera la valeur maximale pour un sac-à-dos de capacité j à l'aide des i premiers objets.

1. Si le $n$ième élément appartient à la solution optimale, cela signifie, que la solution optimale privée du $n$ième élément est aussi une solution optimale avec un sac à dos de capacité $C - p_n$ et un ensemble $\mathcal{O}$ privé du $n$ième élément.

$$T[n, C] = v_n + T[n-1, C-p_n]$$

2. Si le $n$ième élément n'appartient pas à la solution optimale, cela signifie, que la solution optimale OPT est aussi une solution optimale avec un sac à dos de capacité C et un ensemble $\mathcal{O}$ privé du $n$ième élément.

$$T[n, C] = T[n-1, C]$$

Donc par récurence on peut en déduire pour tout $i \in \{1, \ldots, n\}$

$$\text{Si } j < v_i, \text{ alors } T[i, j] = 0 \text{ sinon } T[i, j] = max(T[i-1, j], T[i-1, j-p_i] + v_i)$$

On supposera que $\forall j \in \{1, \ldots, C\}$, on a $T[0, j] = 0$

C'est la fin de la correction.

Question 1.2 Donner l'algorithme utilisant la programmation dynamique.

Correction:

Entrée : un ensemble d'objects $\mathcal{O} = \{1, \ldots, n\}$. L'objet i a une valeur v_i et un poids p_i.

Sortie : un entier

1. Intialiser tous les éléments du tableau T à zéro.
2. Pour tout i allant de 1 à n
   (a) Pour tout j allant de 1 à C

   2 (a).1 Si $j < v_i$, alors $T[i, j] = 0$
   sinon $T[i, j] = max(T[i-1, j], T[i-1, j-p_i] + v_i)$
3. Retourner $T[n, C]$

C'est la fin de la correction.

Question 1.3 Donner la complexité de cet algorithme.

Correction: La complexité de cet algorithme est $\mathcal{O}(nC)$ car

1. l'initialisation du tableau se fait en $\mathcal{O}(nC)$ opérations ;
2. L'instruction 2 (a).1 coûte $\mathcal{O}(1)$ opérations. Et elle est exécutée nC fois.

C'est la fin de la correction.