TP programmation dynamique : le problème du sac à dos

I. Le problème du sac à dos

1) Le problème

On dispose de n objets de masses m_k et de valeurs v_k, $1 \leqslant k \leqslant n$. On souhaite en emporter une partie dans un sac à dos mais la masse totale qu'on peut porter doit être inférieure à une valeur m fixée.

Comment emporter la plus grande valeur possible tout en respectant cette contrainte ?

2) Formalisation du problème

Dans la suite nous allons numéroter les objets de 1 à n : chaque objet sera donc identifié précisément par son numéro. Soit $E = \{1, ..., n\}$ l'ensemble de ces objets. Nous supposons de plus, sans perte de généralité, que la masse et la valeur de chaque objet sont des entiers strictement positifs.

Pour tout entier i tel que $0 \leqslant i \leqslant n$, nous définissons les ensembles :

$$E_0 = \varnothing \ ; \ E_1 = \{1\} \ \ \ldots \ \ E_i = \{1, ..., i\} \quad \text{et} \quad E_n = \{1, ..., n\} = E$$

Concentrons-nous sur E_i avec i quelconque mais fixé. Pour tout partie $J \subset E_i$ nous posons :

$$m(J) = \begin{cases} 0 & \text{si} \quad J = \varnothing \\ \displaystyle\sum_{j \in J} m_j & \text{si} \quad J \neq \varnothing \end{cases} \quad \text{et} \quad v(J) = \begin{cases} 0 & \text{si} \quad J = \varnothing \\ \displaystyle\sum_{j \in J} v_j & \text{si} \quad J \neq \varnothing \end{cases}$$

Ainsi $m(J)$ et $v(J)$ sont respectivement la masse totale et la valeur totale des objets qui appartiennent à J.

Pour tout entier $m \geqslant 0$, nous posons :

$$\mathscr{F}_i(m) = \{\, J \subset E_i \mid m(J) \leqslant m \,\}$$

Exercice 1 : a) De quoi est constitué $\mathscr{F}_0(m)$? b) Pour i quelconque compris entre 0 et n, de quoi est constitué $\mathscr{F}_i(0)$? c) Montrer que pour tout $i \in \{0, ..., n\}$, $\varnothing \in \mathscr{F}_i(m)$.

Posons en outre :

$$V_i(m) = \{\, v(J) \mid J \in \mathscr{F}_i(m) \,\} \subset \mathbb{N}$$

Exercice 2 : pour tout i tel que $0 \leqslant i \leqslant n$ et tout $m \geqslant 0$: a) Montrer que $V_i(m)$ n'est jamais vide. b) Donner un majorant de $V_i(m)$.

En tant que partie non vide et majorée de $\mathbb{N}$, $V_i(m)$ admet un **plus grand élément** que nous notons $c(i, m)$.

Résoudre le problème du sac à donc consiste donc, la masse m étant fixée :

1. à trouver $c(n, m)$ (valeur optimale) ;
2. à trouver toutes les parties $J \subset E_n = E$ telles que $v(J) = c(n, m)$.

3) Algorithme force brute

L'algorithme force brute procède de la façon suivante :

1. Déterminer toutes les parties J de $E = \{1, ..., n\}$. Pour chacune d'entre elles calculer sa masse $m(J)$.
2. Sélectionner les parties qui vérifient $m(J) \leqslant m$ et pour chacune d'entre elles calculer $v(J)$.
3. Chercher le plus grand $v(J)$ et lister les parties J qui lui sont associées.

Exercice 3 : quel est le nombre de parties d'un ensemble E de cardinal n ? Exemple avec $n = 50$:

a) Est-il possible de stocker dans la mémoire vide (RAM) d'un ordinateur possédant 8 Go de RAM toutes les parties de E ?

b) Essayons d'utiliser un disque dur pour stocker les données au fur et à mesure qu'on les obtient. Sachant qu'un très bon disque dur actuel a un capacité de 8 To (tera-octets), cela sera-t-il suffisant ?

4) Programmation dynamique

On se donne une masse $m \geqslant 0$ et on va commencer par chercher une relation de récurrence sur les $c(i, m)$ définis au 2).

Proposition

1. $c(0, m) = 0$ et, pour tout i tel que $0 \leqslant i \leqslant n$, $c(i, 0) = 0$.
2. Supposons $i \geqslant 1$ et notons m_i la masse de l'objet numéro i et v_i sa valeur :
   a) Si $m_i > m$ alors $c(i, m) = c(i-1, m)$;
   b) Si $m_i \leqslant m$ alors :

$$c(i, m) = \max\left(c(i-1, m), c(i-1, m-m_i) + v_i \right)$$

Démonstration :

1. Montrons que $c(0, m) = 0$:

$$c(0, m) = \max\{ v(J) \mid J \in \mathscr{F}_0(m) \} = \max\{ v(J) \mid J = \varnothing \}$$

puisque $\mathscr{F}_0(m) = \{ \varnothing \}$. Comme $v(\varnothing) = 0$ on en déduit que :

$$c(0, m) = \max\{0\} = 0$$

On aurait pu s'en douter puisque $c(0, m)$ représente la plus grande valeur qu'on peut emporter dans le sac à dos avec zéro objet. Elle est donc égal à 0.

Montrons maintenant que pour tout i tel que $0 \leqslant i \leqslant n$, $c(i, 0) = 0$:

On a :

$$c(i, 0) = \max\{ v(J) \mid J \in \mathscr{F}_i(0) \} = \max\{ v(J) \mid J = \varnothing \}$$

puisque $\mathscr{F}_i(0) = \{ \varnothing \}$. Comme $v(\varnothing) = 0$ on en déduit que :

$$c(0, m) = \max\{0\} = 0$$

Ici aussi, une formulation simple du problème en français permettait de voir la solution : $c(i, 0)$ représente la plus grande valeur qu'on peut emporter dans le sac à dos avec des objets à choisir parmi $1, ..., i$ et dont la masse totale est nulle. La seule possibilité est de ne pas prendre d'objet du tout : la valeur maximale est donc nulle.

2. Supposons $i \geqslant 1$ et notons m_i la masse de l'objet numéro i et v_i sa valeur :
   a) Dans le cas où $m_i > m$:

$$c(i,m) = \max\{\, v(J) \mid J \in \mathscr{F}_i(m) \,\}$$

Or, $\forall\, J \in \mathscr{F}_i(m)$, $i \notin J$ car $m_i > m$. Il s'ensuit que $J \subset E_{i-1}$ avec $m(J) \leqslant m$ et donc $J \in \mathscr{F}_{i-1}(m)$. On a donc :

$$\boxed{c(i,m) = \max\{\, v(J) \mid J \in \mathscr{F}_{i-1}(m) \,\} = c(i-1,m)}$$

Une traduction en français de cette propriété est que si $m_i > m$, alors pour remplir le sac avec des objets appartenant à $\{1,...,i\}$ on ne peut choisir que des objets appartenant à $\{1,...,i-1\}$.

   b) Supposons enfin que $m_i \leqslant m$:

$$c(i,m) = \max\{\, v(J) \mid J \in \mathscr{F}_i(m) \,\}$$

Lorsque J parcourt $\mathscr{F}_i(m)$ on peut rencontrer deux situations :

- Soit $i \in J$ et on pose alors $J = J' \cup \{i\}$ avec $J' \subset E_{i-1}$. On a alors :

$$m(J') = m(J) - m_i \leqslant m - m_i$$

  ce qui montre que $J' \in \mathscr{F}_{i-1}(m - m_i)$.
- soit $i \notin J$ et dans ce cas $J \subset E_{i-1}$ avec $m(J) \leqslant m$, ce qui implique que $J \in \mathscr{F}_{i-1}(m)$.

On a donc :

$$\begin{aligned}&\{\, v(J) \mid J \in \mathscr{F}_i(m) \,\}\\ &= \underbrace{\{\, v(J) = v(J') + v_i \mid J' \in \mathscr{F}_{i-1}(m - m_i) \,\}}_{\text{ensemble noté } E_1} \cup \underbrace{\{\, v(J) \mid J \in \mathscr{F}_{i-1}(m) \,\}}_{\text{ensemble noté } E_2}\end{aligned}$$

Le plus grand élément de E_1 est $c(i-1, m-m_i) + v_i$ et le plus grand élément de E_2 est $c(i-1,m)$. Il s'ensuit que $c(i,m)$ est aussi le plus grand élément de la réunion de E_1 et E_2, ce qui s'écrit :

$$\boxed{c(i,m) = \max(\, c(i-1, m-m_i) + v_i, c(i-1,m) \,)}$$

CQFD.

On voit donc à nouveau apparaître la sous-structure optimale. On décrit maintenant des algorithmes permettant de calculer $c(n,m)$.

a) Versions récursives de l'algorithme

On suppose que les masses des objets et leurs valeurs sont stockées dans deux listes `masses` et `valeurs` de tailles $n+1$, définies comme variables globales, mais nous n'allons utiliser que les alvéoles d'indices 1 à n qui contiennent les masses et les valeurs des différents objets.

Pour tester vos fonctions vous pourrez utiliser par exemple :

`masses = [0, 1, 2, 5, 6, 7]` et `valeurs = [0, 1, 6, 18, 22, 28]` (le premier élément de chacune de ces deux listes n'a pas d'importance : on l'a donc mis arbitrairement à 0).

Exercice 4 : écrire une version récursive naïve `c1(n,m)` qui prend en paramètres le nombre n d'objets ainsi que la masse maximale m et qui renvoie la valeur maximale $c(n, m)$ correspondante.

Solution sur le script Python joint dans le dossier.

On ne dessinera ici pas un arbre des appels récursif mais une fois de plus, le chevauchement des sous-problèmes est important. On en vient donc à :

Exercice 5 : écrire une version `c2(n,m)` de cette fonction avec mémoïsation. On utilisera un dictionnaire `dico` formé des couples `((n,m) : c(n,m))` (les clés sont donc les tuples `(n,m)`).

Solution sur le script Python joint dans le dossier.

b) Construction d'une solution

Nous cherchons maintenant à fournir une liste des objets qui maximiseront la valeur une fois dans le sac à dos tout en respectant la contrainte de masse.

Pour cela il faut modifier la fonction `c2(n,m)` précédente pour qu'elle renvoie en plus de $c(n, m)$ laquelle des deux valeurs $c(n-1, m)$ et $c(n-1, m-m_n)+v_n$ était la plus grande (le cas échéant) lors du calcul de $c(n, m)$.

Une façon de procéder consiste à faire renvoyer à `c(n,m)` un tuple formé de la valeur de $c(n, m)$ et d'un entier k qui vaut :

- 1 si $c(n-1, m) \geqslant c(n-1, m-m_n)+v_n$: dans ce cas, l'objet numéro n ne fait pas partie de la solution retenue ;
- 2 si $c(n-1, m) < c(n-1, m-m_n)+v_n$: dans ce cas l'objet numéro n doit être inclu dans la liste formant la solution ;
- 0 dans tous les autres cas de renvoi de la fonction.

On doit aussi veiller à modifier le dictionnaire `dico` qui est maintenant constitué des couples $((n, m) : (c(n, m), k))$.

Exercice 6 : écrire une version modifiée `c2_modif(n,m)` de la version récursive avec mémoïsation, permettant de réaliser cela.

Solution : sur le script Python joint dans le dossier.

Il est maintenant possible de construire une solution optimale grâce à la fonction `construire`que l'on écrit de façon récursive. La fonction renvoie la liste des objets (c'est à dire de leurs numéros) qui ont servi à obtenir $c(n, m)$.

Exercice 7 : compléter la fonction ci-dessous :

```
def construire(n,m) :
   masses = [0,1,2,5,6,7]
   if n == 0 :
      return []
   elif m == 0 :
      return []
   else :
      if masses[n] > m :
         return construire(n-1,m) :
      else :
         val, k = c2_modif(n,m)
         if k == 1 :
            return construire(n-1,m)
         else :
            return [n] + construire(n-1,m-masses[n])
```

Exemple :

Si on prend $m = 11$, que renvoient les fonctions `c(n,m)` et `construire` avec les listes données au **4)**a) ? Est-ce cohérent ?

Solution :

`c1(n,m)` renvoie 40 et `contruire(5,11)` renvoie `[4,3]` ce qui est cohérent puisque la plus grande valeur permise est alors $18 + 22 = 40$.

II. Ouverture : partition équilibrée d'un ensemble d'entiers strictement positifs

1) Le problème

On considère n personnes qui souhaitent monter à bord d'une grande barque un peu instable. Pour simplifier, on suppose que les personnes vont s'asseoir soit à bâbord, soit à tribord. Pour équilibrer le bateau, on voudrait que la somme des masses des personnes assises à bâbord soit la plus proche possible de la somme des masses des personnes assises à tribord.

En supposant que les masses des personnes sont des entiers strictement positifs comment placer les personnes sur le bateau de façon optimale ?

2) Formalisation du problème

Numérotons les personnes de 1 à n : chaque personne est donc spécifiée par son numéro et nous appelons $E = \{1, ..., n\}$ l'ensemble de toutes ces personnes. La personne numéro k a une masse m_k.

Notons B l'ensemble des personnes assises à bâbord et T l'ensemble des personnes assises à tribord. Si on appelle $m(B)$ la masse totale des personnes dans B et $m(T)$ celle des personnes dans T, on doit résoudre le problème suivant : trouver B et T de sorte que :

$$B \cup T = E \ ; \ B \cap T = \varnothing \quad \text{et} \quad |m(B) - m(T)| \text{ minimale} \qquad (1)$$

En y réfléchissant bien on peut adopter le point de vue suivant : soit M la masse totale des personnes dans E et $A \subset E$ une partie non vide de E telle que $m(A) \leqslant M/2$ ($m(A)$ étant bien sûr la masse totale des personnes dans A).

Exercice :

On dit A minimise sa distance à $M/2$ si et seulement si la valeur $M/2 - m(A)$ est minimale parmi tous les autres sous-ensembles possibles vérifiant $m(A) \leqslant M/2$.

a) Montrer que si on dispose d'une partie A qui minimise sa distance à $M/2$, alors la paire A et $A' = E \backslash A$ est une solution à notre problème de partitionnement équilibré.

b) Inversement, montrer que si B et T sont deux parties de E qui vérifient (1) alors l'un des deux (B ou T) a une masse inférieure à $M/2$ et minimise sa distance à $M/2$.

Le problème consiste donc à chercher toutes les parties non vides $A \subset E$ telles que $m(A) \leqslant M/2$ et $m(A)$ la plus grande possible.

c) Montrer qu'il s'agit d'un cas particulier du problème du sac à dos. Quelles sont les valeurs associées aux personnes ?