Synthèse du cours 5 : programmation dynamique (2)

12 octobre 2021

François Laroussinie

NB : Ces synthèses ont pour but de compléter les notes prises en cours. Elles ne les remplacent pas ! En particulier, la plupart des preuves n'y figurent pas. Rappel : il faut programmer les algorithmes vus en cours.

1 Le problème du sac à dos

Input : Un poids maximal $W \in \mathbb{N}$, un ensemble de n objets ayant chacun un poids w_i et une valeur v_i pour $i = 1, \ldots, n$. On suppose $w_i > 0$ et $v_i > 0$.
Output : La **valeur** maximale pouvant être stockée dans le sac sans dépasser sa capacité W.

On peut distinguer deux variantes selon que l'on autorise à mettre plusieurs fois un même élément dans le sac (on a dans ce cas plutôt une liste de types d'objets) ou pas. La variante la plus classique est celle sans répétition.

Exemple avec $W = 10$ et

w	5	5	6
v	20	2	3

La valeur maximale pour un sac de capacité 10 est 22 pour la version sans répétition (avec les objets 1 et 2) et 40 si on autorise des répétitions (avec deux objets 1).

1.1 Sans répétition

Chacun des n objets peut être mis au plus une fois dans le sac.

On va calculer le tableau $K[i, w]$, avec $0 \leq i \leq n$ et $0 \leq w \leq W$, défini comme étant la **valeur** maximale pouvant être stockée dans un sac de **capacité** w avec les objets $1, 2, 3, \ldots, i$. Le résultat du problème est alors la valeur $K[n, W]$: la valeur maximale pour un sac de capacité W en prenant des objets parmi tous les objets possibles.

On a clairement $K[0, w] = 0$ car si aucun objet n'est disponible, le sac sera vide, donc de valeur 0. De même on a $K[j, 0] = 0$ car tout objet à un poids strictement positif et aucun de peut être mis dans un sac de capacité nulle. Il reste à établir le cas général :

$$K[j, w] \stackrel{\text{def}}{=} \max\Big(K[j-1, w], v_j + K[j-1, w-w_j]\Big)$$

D'où l'algo :

Procédure `SaDssRepet` (O, W)
begin
- `//`O `correspond à la liste des` n `objets de poids` w_i `et de valeur` v_i
- `K` : tableau de booléens de taille $(n+1) \times (W+1)$
- **pour** $w = 1 \dots W$ **faire** $\mathtt{K}[0, w] := 0$
- **pour** $i = 0 \dots n$ **faire** $\mathtt{K}[i, 0] := 0$
- **pour** $i = 1 \dots n$ **faire**
  - **pour** $w = 1 \dots W$ **faire**
    - **si** $w_i \leq w$ **alors** $\mathtt{K}[i, w] := \max(\mathtt{K}[i-1, w], v_i + \mathtt{K}[i-1, w-w_i])$
    - **else** $\mathtt{K}[i, w] := \mathtt{K}[i-1, w]$
- **return** $\mathtt{K}[n, W]$

Que l'on peut améliorer en utilisant une seule ligne du tableau (en veillant à énumérer correctement les indices !) :

Procédure `SaDssRepet2` (O, W)
begin
- `//`O `correspond à la liste des` n `objets de poids` w_i `et de valeur` v_i
- `K` : tableau de booléens de taille $W+1$
- **pour** $w = 0 \dots W$ **faire** $\mathtt{K}[w] := 0$
- **pour** $i = 1 \dots n$ **faire**
  - **pour** $w = W \dots w_i$ **faire**
    - $\mathtt{K}[w] := \max(\mathtt{K}[w], v_i + \mathtt{K}[w-w_i])$
- **return** $\mathtt{K}[n, W]$

1.2 Avec répétition

Là encore, on peut calculer des coefficients $[i, w]$. On a toujours $K[0, w] = 0$ et $K[j, 0] = 0$ et le cas général s'énonce ainsi :

$$K[j, w] \stackrel{\text{def}}{=} \max\Big(K[j-1, w], v_j + K[j, w-w_j]\Big)$$

NB : c'est de prendre $K[j, w-w_j]$ (et non $K[j-1, w-w_j]$) qui permet de mettre plusieurs occurrences du même objet j !

D'où l'algo :

Procédure `SaDavecRepet` (O, W)
begin

```
//O correspond à la liste des n objets de poids w_i et de valeur v_i
K : tableau de booléens de taille (n+1) × (W+1)
pour w = 1...W faire K[0,w] := 0
pour i = 0...n faire K[i,0] := 0
pour i = 1...n faire
    pour w = 1...W faire
        si w_i ≤ w alors K[i,w] := max(K[i-1,w], v_i + K[i,w-w_i])
        else K[i,w] := K[i-1,w]
return K[n,W]
```

Et son amélioration (en veillant à énumérer correctement les indices !) :

Procédure `SaDavecRepet2` (O, W)
begin

```
//O correspond à la liste des n objets de poids w_i et de valeur v_i
K : tableau de booléens de taille W+1
pour w = 0...W faire K[w] := 0
pour i = 1...n faire
    pour w = w_i...W faire
        K[w] := max(K[w], v_i + K[w-w_i])
return K[n,W]
```