Synthèse du cours 4 : programmation dynamique (1)

4 octobre 2022

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.

0.1 Subset sum problem

Le problème est le suivant :

Input : Un ensemble E de n entiers $x_1, \ldots, x_n$ et un entier V.
Output : existe-t-il un sous-ensemble $E' \subseteq E$ de poids exactement V, *ie* tel que $\sum_{x \in E'} x = V$.

Pour résoudre ce problème, on peut utiliser un backtracking en énumérant tous les sous-ensembles de E et en s'arrêtant dès qu'un de ces sous-ensembles est de poids V. A faire...

Une autre approche consiste à utiliser une table de booléens K de taille $n+1 \times V+1$ et de la construire de manière à vérifier : $\mathsf{K}[i,s] = \top$ ssi il existe un sous-ensemble de $\{x_1, x_2, \ldots, x_i\}$ de poids s. On a d'abord : $\mathsf{K}[0,s] = \bot$ pour $s > 0$, et $\mathsf{K}[i,0] = \top$ pour tout $0 \leq i \leq n$. Ensuite, pour le cas général on a :

$$\mathsf{K}[i,s] \quad = \quad \begin{cases} \mathsf{K}[i-1,s] & \text{si } x_i > s \\ \mathsf{K}[i-1,s] \vee \mathsf{K}[i-1,s-x_i] & \text{sinon} \end{cases}$$

Cela donne un algorithme en $O(n \cdot V)$:

Procédure `Subset1` (V,E)
begin
- `K` : tableau de booléens de taille $(n+1) \times (V+1)$
- **pour** $s = 1 \ldots V$ **faire** $\mathtt{K}[0,s] := \bot$
- **pour** $i = 0 \ldots n$ **faire** $\mathtt{K}[i,0] := \top$
- **pour** $i = 1 \ldots n$ **faire**
  - **pour** $s = 1 \ldots V$ **faire**
    - **si** $x_i \leq s$ **alors** $\mathtt{K}[i,s] := \mathtt{K}[i-1,s] \vee \mathtt{K}[i-1,s-x_i]$
    - **else** $\mathtt{K}[i,s] := \mathtt{K}[i-1,s]$
- **return** $\mathtt{K}[n,V]$

On peut améliorer la complexité en espace en remarquant que seule la ligne $i-1$ est utilisée pour calculer la ligne i. En faisant attention à parcourir les valeurs dans le bon sens (décroissant) pour ne pas utiliser plusieurs fois un même élément x_i, on obtient l'algorithme Subset2.

Procédure `Subset2` (V,E)
begin
  `K` : tableau de booléens de taille $(V+1)$
  pour $s = 1 \ldots V$ **faire** $\mathtt{K}[s] := \bot$
  $\mathtt{K}[0] := \top$ **pour** $i = 1 \ldots n$ **faire**
    pour $s = V \ldots x_i$ **faire**
      $\mathtt{K}[s] := \mathtt{K}[s] \vee \mathtt{K}[s - x_i]$
  return $\mathtt{K}[V]$

Construction d'une solution. A partir du tableau à deux dimensions, il est très facile de retrouver un ensemble solution (lorsqu'il en existe) :

Procédure `Solution` ($V, E,$ `K`)
begin
  si $\mathtt{K}[n, V]$ **alors**
    $s := V$
    `res` $:= \emptyset$
    pour $i = n \ldots 1$ **faire**
      si $(s \geq x_i \wedge K[i-1, s - x_i])$ **alors**
        `res` $:=$ `res` $\cup \{x_i\}$
        $s := s - x_i$
        si $(s == 0)$ **alors return** `res`
  return $\bot$

Mais on peut aussi coupler la construction de solutions avec la construction de la table K par l'algorithme « économe » en mémoire :

Procédure `Subset3` (V, E)
begin
  `K` : tableau de booléens de taille $(V+1)$
  `sol` : tableau d'ensembles de taille $(V+1)$
  pour $s = 1 \ldots V$ **faire**
    $\mathtt{K}[s] := \bot$, $\mathtt{sol}[s] :=$ `undef`
  $\mathtt{K}[0] := \top$, $\mathtt{sol}[0] := \emptyset$
  pour $i = 1 \ldots n$ **faire**
    pour $s = V \ldots x_i$ **faire**
      si $\mathtt{K}[s - x_i]$ **alors**
        $\mathtt{sol}[s] := \mathtt{sol}[s - x_i] \cup \{x_i\}$
        $\mathtt{K}[s] := \top$
  return $(\mathtt{K}[V], \mathtt{sol}[V])$