Informatique PT. 2024 - 2025

Mathématiques & Informatique - F. Gaunard
http://frederic.gaunard.com
Lycée Voltaire, Paris 11e.

6 Travaux Pratiques : Programmation dynamique

Contexte. C'est l'été, vous avez prévu de vous rendre dans un grand festival de musique où jouent tou.te.s les artistes du moment que vous rêvez de voir.
Vous voulez absolument passer un moment inoubliable en ne passant à côté de rien.

Deux problématiques apparaissent, que l'on va résoudre dans deux parties indépendantes, à l'aide d'une approche de programmation dynamique :

- ✗ Comment *optimiser* le planning des concerts que vous allez voir pour repartir complètement satisfait.e de cette parenthèse musicale ?
- ✗ Comment optimiser le rangement de votre sac à dos, de sorte qu'il contienne un maximum d'objets dont vous aurez besoin tout en contrôlant son poids ?

On téléchargera le fichier **TP6.py** qu'on complètera tout au long de la séance.

1 Optimisation du planning

Le festival comporte plusieurs scènes, si bien que des artistes peuvent jouer en même temps ou des concerts se chevaucher.
Dans toute la suite, n est un entier supérieur ou égal à 1 et représente le nombre de concerts du festival.
Le planning du festival est implémenté par une liste C dont les éléments $C_1, C_2, ..., C_n$ sont des *tuple* à quatre éléments : le nom du concert, un horaire de début de concert d_i, un horaire de fin f_i et un indice de valeur v_i que vous avez attribué à chaque artiste.

On pourrait utiliser l'algorithme introduit ans le cours pour obtenir la plus longue chaîne sous-séquence de dominos. Cet algorithme est optimal si... les valeurs des concerts sont toutes égales à 1.
On adopte alors une autre approche ici.

On décide donc de numéroter les concerts par date de fin croissante

$$f_1 \leq f_2 \leq ... \leq f_n.$$

Deux concerts C_i et C_j (avec $i < j$) sont alors dits **compatibles** si $f_i \leq f_j$ (on estime que le temps de se rendre d'une scène à l'autre est négligeable).
On note, lorsque cela a du sens,

$$m_i = \max\left(\{j \in [\![1, i-1]\!] : f_j < d_i\}\right).$$

On appelle **programme** toute (sous-)séquence $(C_{i_1}, C_{i_2}, ...C_{i_k})$ de concerts successifs compatibles..
La **valeur** d'un programme est alors définie comme la somme des valeurs des concerts qui composent ce programme.
On note ℓ_i la valeur maximale des programmes construits à partir des concerts $C_1, ..., C_i$.

C'est à dire *la valeur d'une solution optimale* restreinte aux i premiers concerts.

1. **a**. Que vaut ℓ_1 ?
b. Justifier la relation de récurrence, pour $i \geq 2$,

$$\ell_i = \max\left(\{v_i + \ell_{m_i}, \ell_{i-1}\}\right).$$

2. Écrire, en `Python`, une fonction d'en-tête `dernier_concert_compatible(C, i)` qui prend en argument un planning de concerts triés `C` et un indice $i \in [\![2, n]\!]$ et qui renvoie m_i.
La fonction renverra `None` si aucun concert n'est compatible.

3. Écrire une fonction d'en-tête `valeur_max_programme(C)` qui prend en argument un planning de concerts C **déjà triés** et qui renvoie la liste $L = [\ell_1, \ell_2, .., \ell_n]$.

4. Écrire une fonction **récursive** `programme_optimal(C)` qui prend en argument un planning de concerts C et renvoie le programme optimal, c'est à dire la liste des concerts $C_{i_1}, C_{i_2}, ..., C_{i_k}$ correspondant au programme ayant une valeur maximale.
Quel est alors le programme pour le planning `PT_en_fete` ?

2 Remplir son sac à dos

Vous disposez d'un sac à dos dont la charge maximale est P, et d'un ensemble d'objets $O = \{0, 1, \ldots, n-1\}$, tel que l'objet d'indice i objet ait pour poids p_i et pour valeur v_i. On cherche à remplir le sac à dos avec un ensemble d'objets de valeur maximale dont le poids ne dépasse pas P.
En d'autres termes on cherche un ensemble $I \subset [\![0, n-1]\!]$ qui maximise $\sum_{i \in I} v_i$ tout en respectant la contrainte $\sum_{i \in I} p_i \leqslant P$.

Soient $i, p \in \mathbb{N}$. Notons $W(i, p)$ la valeur maximale que l'on peut obtenir en prenant un sous-ensemble d'objets parmi $[\![0, i-1]\!]$ dont le poids total est inférieur à p.

5. Que vaut $W(i, p)$ lorsque $i = 0$ ou lorsque $p = 0$?

Observons alors la chose suivante ; si on souhaite trouver la solution optimale pour $i+1$ objets et un poids maximal P :

- ✗ soit on ne prend pas l'objet d'indice $i-1$, et on se ramène à l'étude du problème avec $i-1$ objets et le même poids,
- ✗ soit on prend l'objet d'indice $i-1$, et on se ramène à l'étude du problème avec $i-1$ objets et un poids maximal pour le sac valant $P - p_{i-1}$.

6. Écrire une relation de récurrence entre $W(i+1, p)$, $W(i, p)$, v_i et $W(i, P - p_i)$.

7. Écrire une fonction d'en-tête `tableau_dynamique(O, P)` qui prend en argument un dictionnaire d'objets O et un poids maximal P et renvoie le tableau des valeurs $W(i, p)$, $(i, p) \in [\![0, n]\!] \times [\![0, P]\!]$.

8. On veut alors utiliser ce tableau pour reconstruire la liste des objets qu'on met dans le sac à dos et répondre au problème initial.

a. Soit $(i, p) \in [\![0, n]\!] \times [\![0, P]\!]$. Interpréter :

- ✗ $W(i+1, p) = W(i, p)$
- ✗ $W(i+1, p) = v_i + W(i, p - p_i)$

b. En déduire une fonction d'en-tête `sac_a_dos(O, P)` qui prend en argument un dictionnaire d'objets O et un poids maximal P et renvoie la liste optimale des objets à emporter.