TP

Sommes de sous-ensembles

1. Préliminaires

Dans la suite du sujet, nous aurons besoin de générer des suites d'entiers. Pour ce faire, on définit la suite $(v_n)_{n\in\mathbb{N}}$ de la façon suivante :

$$v_0 = 26 \qquad \text{et} \qquad \forall n \in \mathbb{N}^*, \quad v_n = 353 \times v_{n-1} \bmod 997$$

Question 1. Rédiger une fonction qui prend pour argument un entier $n \in \mathbb{N}$ et renvoie la liste $[v_0, v_1, \dots, v_{n-1}]$. Donner la valeur de :

a) v_5 b) v_{47} c) v_{86}

2. Stratégie optimale pour un jeu

Considérons un ensemble d'entiers positifs $v_0, v_1, \dots, v_{n-1}$ positionné dans cet ordre sur une ligne. On joue à un jeu contre un adversaire en jouant l'un après l'autre. À chaque tour, un joueur choisit soit le premier, soit le dernier nombre sur la ligne, l'enlève de façon permanente et reçoit ce nombre. L'objectif est de déterminer la somme maximale que l'on est certain de gagner lorsqu'on joue le premier.

Pour ce faire, on utilise la programmation dynamique : pour tout $0 \leqslant i \leqslant j \leqslant n-1$ on note $\mathrm{M}(i,j)$ la somme maximale que l'on est certain de gagner lorsqu'on joue le premier avec la liste $v_i, v_{i+1}, \dots, v_j$.

On note aussi $\mathrm{S}(i,j) = \sum_{k=i}^{j} v_k$.

Question 2. Rédiger une fonction `sommes(v)` qui prend pour argument la liste $v = [v_0, v_1, \dots, v_{n-1}]$ et qui renvoie le tableau des valeurs prises par $\mathrm{S}(i,j)$ pour $0 \leqslant i \leqslant j \leqslant n-1$. Ce tableau sera représenté par une liste de listes.
Exécuter la fonction `sommes` pour $n = 2\,000$ puis vérifier sa validité en donnant la valeur de $\mathrm{S}(i,j)$ pour :

a) $(i,j) = (5,30)$ b) $(i,j) = (100,199)$ c) $(i,j) = (1000,1999)$

Question 3. On suppose que c'est à notre tour de jouer à partir de la liste $v_i, \dots, v_j$.
Si on choisit de prendre la valeur v_i, exprimer en fonction de S et de M le gain maximal que l'on est en droit d'espérer lorsque le second joueur joue de manière optimale.
Répondre à la même question lorsqu'on choisit la valeur v_j, et en déduire l'expression de $\mathrm{M}(i,j)$ en fonction de $\mathrm{M}(i+1,j)$, $\mathrm{M}(i,j-1)$ et $\mathrm{S}(i,j)$.

Question 4. Rédiger alors une fonction `gain(v)` qui prend pour argument la liste $(v_0, \dots, v_{n-1})$ et qui renvoie la valeur maximale que l'on est certain de gagner.
Donner cette somme maximale pour :

a) $n = 100$ b) $n = 300$ c) $n = 500$

3. Sommes de sous-ensembles

Dans cette partie, indépendante de la précédente, nous nous intéressons au problème suivant : étant donnés une liste $v = [v_0, \dots, v_{n-1}]$ de n entiers naturels et un entier t, existe-t-il des éléments $v_{i_1}, \dots, v_{i_p}$ de v vérifiant $v_{i_1} + \cdots + v_{i_p} = t$, et si oui, combien y-a-t'il de solutions ?

On choisit de résoudre ce problème en calculant la plus grande somme d'éléments tirés de v qui soit *inférieure ou égale* à t, en appliquant l'algorithme suivant :

1. on pose $\mathrm{S}_0 = [0]$;
2. pour i variant de 1 à n :
   (a) on calcule la liste $\mathrm{T}_i = \left[s + v_{i-1} \;\middle|\; s \in \mathrm{S}_{i-1} \text{ et } s + v_{i-1} \leqslant t \right]$;
   (b) on pose $\mathrm{S}_i = \mathrm{S}_{i-1} + \mathrm{T}_i$ (le + désigne la concaténation)

Autrement dit, S_i est la liste des sommes que l'on peut obtenir en sommant des éléments de $v_0, \dots, v_{i-1}$ sans dépasser t.

Exemple. Lorsque $v = [2,3,7,4]$ et $t = 9$ on calcule successivement :

- $T_1 = [2]$ et $S_1 = [0,2]$;
- $T_2 = [3,5]$ et $S_2 = [0,2,3,5]$;
- $T_3 = [7,9]$ (les sommes $3+7$ et $5+7$ dépassent t) et $S_3 = [0,2,3,5,7,9]$;
- $T_4 = [4,6,7,9]$ (les sommes $7+4$ et $9+4$ dépassent t) et $S_4 = [0,2,3,5,7,9,4,6,7,9]$.

On constate que la somme $t = 9$ peut être obtenue de deux façons différentes ($9 = 2+7$ ou $9 = 2+3+4$).

Question 5. Donner le nombre de décompositions de t que l'on peut obtenir en sommant des éléments parmi $v_0, \ldots, v_{n-1}$ pour :

a) $(n,t) = (10, 4\,000)$ b) $(n,t) = (15, 4\,000)$ c) $(n,t) = (20, 5\,000)$

Question 6. Calculer la taille de la liste S_n pour :

a) $(n,t) = (10, 4\,000)$ b) $(n,t) = (15, 4\,000)$ c) $(n,t) = (20, 5\,000)$

Quelle est la complexité en temps et en mémoire de cet algorithme ?

Pour améliorer cette complexité, une solution consiste à représenter S_i non plus par une liste mais par un dictionnaire dont les clefs sont les sommes atteignables et la valeur leur multiplicité.

Question 7. Modifier en conséquence votre fonction, et donner le nombre de solutions pour :

a) $(n,t) = (40, 16\,000)$ b) $(n,t) = (45, 18\,000)$ c) $(n,t) = (50, 20\,000)$

Les réponses attendues

Question 1.

a) $v_5 =$ 636 *b)* $v_{47} =$ 407 *c)* $v_{86} =$ 667

Question 2.

a) $(i,j) = (5,30)$: 11 351 *b)* $(i,j) = (100,199)$: 48 688 *c)* $(i,j) = (1\,000, 1\,999)$: 498 790

Question 4.

a) $n = 100$: 23 968 *b)* $n = 300$: 78 581 *c)* $n = 500$: 123 889

Question 5.

a) $(n,t) = (10, 4\,000)$: 0 *b)* $(n,t) = (15, 4\,000)$: 9 *c)* $(n,t) = (20, 5\,000)$: 292

Question 6.

a) $(n,t) = (10, 4\,000)$: 1 020 *b)* $(n,t) = (15, 4\,000)$: 28 048 *c)* $(n,t) = (20, 5\,000)$: 584 459

Question 7.

a) $(n,t) = (40, 16\,000)$: 12 *b)* $(n,t) = (45, 18\,000)$: 6 294 *c)* $(n,t) = (50, 20\,000)$: 57 971