TD2 - Programmation dynamique

Optimisation et Recherche Operationnelle

M1 Info - semestre d'automne 2021-2022

Université Claude Bernard Lyon 1

Christophe Crespelle
christophe.crespelle@univ-lyon1.fr

Florian Ingels
florian.ingels@ens-lyon.fr

Exercice 1. *Sur l'autoroute du profit*

Vous implentez une chaine de restaurants sur les aires de repos d'une section d'autoroute. Les aires sont numerotees de 1 a n dans l'ordre dans lequel on les rencontre sur l'autoroute. Pour chaque aire i, on connait :

- sa position x_i, exprimee en km depuis le debut de la section autoroutiere $0 < x_1 < x_2 < \cdots < x_n$, et
- le revenu espere $r_i > 0$ en placant un de vos restaurants dans cette aire, qui depend de la frequentation de l'aire.

Les x_i et les r_i sont donnes dans deux tableaux separes indexes par i. La societe qui gere l'autoroute, l'AFN (Autoroutes de France et de Navarre), impose une contrainte sur l'implentation des restaurants : deux restaurants de la meme chaine (y compris la votre) doivent etre espaces d'au moins 50km. On veut faire un algorithme qui retourne un placement de vos restaurants qui ait un revenu escompte maximum, note OPT, compte tenu de la contrainte imposee par l'AFN.

a. On note $p(i)$ le numero de l'aire la plus proche de i qui est situee avant i ($x_{p(i)} < x_i$) et a au moins 50 km de l'aire i. Donnez un algorithme de complexite $O(n)$ qui calcule $p(i)$ pour tout $i \in [\![1, n]\!]$. On prendra pour convention $p(i) = 0$ lorsqu'il n'existe pas d'aire a au moins 50km de i avant i.

b. Si on decide de placer un restaurant sur l'aire i, sur quelles aires j avant i, c.a.d. $j < i$, peut-on eventuellement placer un autre restaurant ?

On s'interesse maintenant au sous-probleme $Restau(j)$ dans lequel on ne place des restaurants que sur les aires $i \in [\![1, j]\!]$, pour un $j \in [\![1, n]\!]$ fixe, et on note $OPT(j)$ le revenu maximum qu'on peut atteindre dans ce sous probleme (on etend cette notation en posant par convention $OPT(0) = 0$).

c. Soit S_j^* une solution de revenu maximum au sous-probleme $Restau(j)$ telle que $j \notin S_j^*$. Exprimez le revenu $r(S_j^*)$ de cette solution en fonction des $OPT(j')$ pour $j' < j$.

d. Meme question lorsque $j \in S_j^*$.

e. Donnez une formule de recurrence qui exprime $OPT(j)$ en fonction des $OPT(j'), j' < j$.

f. Donnez un algorithme de complexite $O(n)$ pour calculer OPT, le revenu escompte maximum, et un placement de vos restaurants correspondant.

Exercice 2. *Un sac de valeur*

Dans le probleme du sac a dos, on donne une collection d'objets numerotes de 1 a n et chaque objet a une valeur $v_i \in \mathbb{R}^+$ et un poids $w_i \in \mathbb{R}^+$, pour $i \in [\![1, n]\!]$. Le probleme est a valeurs entieres si de plus les valeurs v_i sont des entiers, c.a.d. $\forall i \in [\![1, n]\!], v_i \in \mathbb{N}$. Pour une collection d'objets $S \subseteq [\![1, n]\!]$, on note $v(S) = \sum_{i \in S} v_i$ la valeur de la collection S et $w(S) = \sum_{i \in S} w_i$ son poids (avec par convention $v(\varnothing) = 0$ et $w(\varnothing) = 0$). On donne aussi un poids limite $W \geq 0$ pour le sac a dos et on demande la valeur maximum OPT d'une collection d'objets S telle que $w(S) \leq W$. En clair, on veut maximiser la valeur de ce que l'on prend en ayant une limite ferme sur le poids total. Ce probleme est un grand classique de l'optimisation combinatoire, utilise pour modeliser de nombreux problemes pratiques. Il est NP-difficile et on se propose de faire un algorithme pseudopolynomial pour le resoudre de maniere exacte, en utilisant l'approche de la programmation dynamique. Cet algorithme a une complexite theorique exponentielle mais est tres efficace en pratique lorsque les valeurs entieres restent relativement petites (c'est a dire du meme ordre de grandeur que le nombre d'objets).
On considere le sous-probleme $PoidsSac(i, V)$ suivant : quel est le poids limite minimum d'un sac qui peut recevoir une collection d'objets de valeur au moins V, avec $V \leq \sum_{j=1}^{i} v_j$, qui sont choisis uniquement parmi les objets d'indice $j \leq i$? Ce poids minimum est note $\overline{OPT}(i, V)$. Soit O une solution qui atteint le poids minimum $\overline{OPT}(i, V)$.

a. Que vaut $\overline{OPT}(i, V)$ si $i \in O$ et i est l'unique objet de O ?

b. Que vaut $\overline{OPT}(i, V)$ si $i \in O$ et i n'est pas l'unique objet de O ?

c. Montrer que dans le cas ou $i \in O$, on a toujours $\overline{OPT}(i, V) = w_i + \overline{OPT}(i - 1, \max\{0, V - v_i\})$.

d. Que vaut $\overline{OPT}(i, V)$ si $i \notin O$?

e. Donnez une formule de recurrence pour $\overline{OPT}(i, V)$ dans le cas ou $V > \sum_{j=1}^{i-1} v_j$?

f. Donnez une formule de recurrence pour $\overline{OPT}(i, V)$ dans le cas ou $V \leq \sum_{j=1}^{i-1} v_j$?

g. En utilisant les formules des deux questions precedentes, ecrivez un algorithme qui calcule $\overline{OPT}(i, V)$ pour toutes les valeurs possibles et pertinentes de i et V et qui retourne OPT, la valeur de la solution optimale au probleme du sac a dos a valeurs entieres.

On note $v^* = \max_{1 \leq i \leq n} \{v_i\}$.

h. Donnez la complexite de votre algorithme en fonction de n et v^*.

On note aussi $w^* = \max_{1 \leq i \leq n} \{w_i\}$.

i. Exprimez la taille t, en nombre de bits, de l'entree de l'algorithme en fonction de n, v^* et w^*.

j. La valeur de v^* est-elle polynomiale en fonction de t ?