DM #6 — Sac à dos binaire

On rappelle le problème du *sac à dos binaire* (*subset sum*) : soit $\mathcal{S}$ un ensemble de n (supposé pair) entiers naturels et $t \in \mathbb{N}$, on cherche à décider s'il existe un sous-ensemble de $\mathcal{S}$ dont les éléments somment à t. Il serait par ailleurs facile d'adapter l'algorithme auquel on aboutira au problème de recherche sous-jacent, c'est à dire *trouver* un tel sous-ensemble s'il en existe un.
On propose d'étudier en détail la résolution de ce problème par une approche *rencontre au milieu*. Soient $\mathcal{S}_1$, $\mathcal{S}_2$ deux sous-ensembles de taille $n/2$ formant une partition de $\mathcal{S}$, on définit $\mathbb{T}_1 = \left\{\sum_{x\in\sigma} x, \sigma \in 2^{\mathcal{S}_1}\right\}$, $\mathbb{T}_2 = \left\{t - \sum_{x\in\sigma} x, \sigma \in 2^{\mathcal{S}_2}\right\}$.

1. Montrez qu'il existe $\sigma \in 2^{\mathcal{S}}$ tel que $\sum_{x\in\sigma} x = t$ ss.'il existe $\tau_1 \in \mathbb{T}_1$, $\tau_2 \in \mathbb{T}_2$ avec $\tau_1 = \tau_2$.
2. Donnez les tailles de $\mathbb{T}_1$ et $\mathbb{T}_2$.

On suppose que la taille de l'écriture en base 2 des éléments de $\mathbb{T}_1$ et $\mathbb{T}_2$ est du même ordre de grandeur que n (par exemple si $n = 256$, alors les éléments de $\mathbb{T}_1$ & $\mathbb{T}_2$ sont des entiers d'environ 256 bits). On suppose également que $\mathbb{T}_1$ et $\mathbb{T}_2$ sont représentés par des tableaux `t1` et `t2`.

3. Quel algorithme de tri choisiriez vous pour trier `t1` & `t2` ? Quel serait son coût ?

On suppose `t1` et `t2` triés par ordre de croissant.

4. Esquissez un algorithme pour décider s'il existe une collision entre `t1` et `t2` (c'est à dire une paire d'indices `i` et `j` tels que `t1[i]` est égal `t2[j]`). Quel est son coût pire cas en fonction de n ?
5. Montrez que l'on peut résoudre le problème du sac à dos binaire en temps et espace $\widetilde{O}(2^{n/2})$ (où la notation $\widetilde{O}$ indique que l'on « ignore » les facteurs (au plus) polynomiaux (ici, en n)).

Un algorithme dont le coût en espace est égal au coût en temps est en général peu attractif : il est par exemple bien plus accessible de réaliser 2^{40} opérations élémentaires (votre téléphone peut probablement le faire en temps raisonnable) que de stocker 2^{40} octets.

6. Esquissez une adaptation de l'algorithme de résolution de la question précédente qui permet de résoudre le problème du sac à dos binaire en temps T et espace $M \leq T$, sous la contrainte $T \times M = \widetilde{O}(2^n)$. (Par exemple, on doit pouvoir prendre $M \approx 2^{n/3}$ et $T \approx 2^{2n/3}$.)

L'algorithme « de Schroeppel-Shamir » offre une solution plus attractive que celle de la question précédente pour diminuer le coût en espace de l'approche par rencontre au milieu. Il consiste à utiliser des représentations compressées de $\mathbb{T}_1$ et

$\mathbb{T}_2$ de tailles seulement $O(2^{n/4})$, qui permettent néanmoins de toujours parcourir leurs éléments par ordre croissant pour un coût total $\widetilde{O}(2^{n/2})$. Ces représentations peuvent alors être substituées à `t1` et `t2` dans l'étape de recherche de collision ci-dessus, ce qui donne un algorithme en temps $\widetilde{O}(2^{n/2})$ et espace seulement $O(2^{n/4})$!

Dans ces représentations compressées, les éléments sont stockés dans des files de priorité (min) de taille au plus $2^{n/4}$.

7. Donnez une expression (fonction de n) du coût pire-cas des opérations d'insertion et d'extraction dans ces files de priorité, lorsqu'elles sont implémentées par des tas binaires.

Dans tout ce qui suit, on suppose sans perte de généralité que n est divisible par 4. La représentation compressée de $\mathbb{T}_1$ fonctionne de la façon suivante :

— on partitionne $\mathcal{S}_1$ en deux sous-ensemble $\mathcal{S}_{11}$ et $\mathcal{S}_{12}$ de tailles $n/4$, et l'on représente les $2^{n/4}$ éléments de $\mathbb{T}_{11} := \left\{\sum_{x\in\sigma} x, \sigma \in 2^{\mathcal{S}_{11}}\right\}$ & $\mathbb{T}_{12} := \left\{\sum_{x\in\sigma} x, \sigma \in 2^{\mathcal{S}_{12}}\right\}$ sous forme de tableaux triés par ordre croissant `t11` et `t12` en réalité, `t12` n'a pas besoin d'être trié ;
— on initialise une file de priorité min `pq1` avec les $2^{n/4}$ couples *(priorité, donnée)* : `(t11.(0) + t12.(j), (0, j))`, pour $j \in [\![2^{n/4}]\!]$;
— pour parcourir les éléments de $\mathbb{T}_1$ par ordre croissant, on itère la séquence suivante (donnée avec une syntaxe OCaml) tant que possible :

```
let x, (i, j) = pop pq1 in
let ip1 = i + 1 in
next := x ;
if ip1 < tpn4 then
    push pq1 (t11.(ip1) + t12.(j), (ip1, j))
```

où `tpn4` contient la taille des tableaux `t11` & `t12` et la référence `next` est modifiée avec la valeur de $\mathbb{T}_1$ à considérer.

Les questions suivantes visent à montrer que la suite des valeurs prises par `!next` énumère bien les éléments de $\mathbb{T}_1$ par ordre croissant.

On note $\mathbb{V}_1$ l'ensemble des couples (i, j) qui ont été extraits de `pq1` au terme d'un certain nombre (éventuellement nul) d'itérations de la séquence ci-dessus.

8. Montrez que la boucle itérant la séquence ci-dessus satisfait l'invariant qu'il existe une suite finie d'entiers (ℓ_k) telle que pour tout $j \in [\![2^{n/4}]\!]$:

— $\mathbb{V}_1$ contient l'ensemble des couples $(i < \ell_j, j)$;
— et `pq1` contient au plus une donnée (i, j), qui si elle existe satisfait $i = \ell_j$.

Par construction, les éléments de $\mathbb{T}_1$ sont donnés par les $2^{n/2}$ sommes `t11.(i) + t12.(j)` pour $(i, j) \in [\![2^{n/4}]\!] \times [\![2^{n/4}]\!]$. Pour tout $\mathbb{X} \subseteq [\![2^{n/4}]\!] \times [\![2^{n/4}]\!]$, on note $\mathbb{T}_1^{\mathbb{X}}$ l'ensemble $\{$`t11.(i) + t12.(j)`$, (i, j) \in \mathbb{X}\}$.

9. Montrez que pour la même suite (ℓ_k) qu'à la question précédente, l'on a l'invariant de boucle que pour tout $j \in [\![2^{n/4}]\!]$, ou bien $\mathbb{V}_1 \supseteq [\![2^{n/4}]\!] \times \{j\}$, ou bien :

$$\min \mathbb{T}_1^{[\![2^{n/4}]\!]\times\{j\}} \setminus \mathbb{T}_1^{\mathbb{V}_1} = \mathbb{T}_1^{\{\ell_j, j\}}$$

(C'est à dire que l'on a ou bien extrait tous les éléments de la forme $(\_, j)$ de `pq1`, ou bien l'élément de $\mathbb{T}_1$ minimum (pour j fixé) qui peut s'écrire comme une somme `t11.(i) + t12.(j)` avec $(i, j) \notin \mathbb{V}_1$ est obtenu pour $i = \ell_j$.)

10. Conclure que la suite des valeurs prises par `!next` en itérant la séquence ci-dessus est bien celle des éléments de $\mathbb{T}_1$ par ordre croissant.

11. Discutez brièvement du coût de mise en place et d'opération de cette représentation compressée de $\mathbb{T}_1$.

12. Expliquez comment construire et opérer de façon similaire une file de priorité permettant d'énumérer (avec le même coût en temps et espace) les éléments de $\mathbb{T}_2$ par ordre croissant.