
Le problème de la somme d'un ensemble

Johanne Cohen

`Johanne.Cohen@loria.fr`

Laboratoire LORIA

Énoncé du problème

Instance:

$S = \{a_1, \dots, a_n\}$ un ensemble d'entiers positif

t un entier positif

Objectif:

Trouver un sous ensemble de S tel que sa somme est exactement t .

Théorème 1: Le problème de la somme d'un ensemble est NP-complet.

Notation

- $L + a = \{\ell + a : \ell \in L\}$
- $\text{union}(L, L') = \{\ell : \ell \in L \vee L'\}$
- $\text{union} - \text{trie}(L, L') = \{\ell_i : (\ell \in L \vee L') \wedge (\ell_i \leq \ell_{i+1})\}$ avec L' et L deux listes triées.
 - Complexité = $O(|L'| + |L|)$
 - Si $L' = \{2, 15\}$ et $L = \{1, 16, 20\}$, alors $\text{union} - \text{trie}(L, L') = \{1, 2, 15, 16, 20\}$

Algorithme pseudo-polynomial ($\mathcal{B}$)

Description de l'algorithme Entrée: $S = \{a_1, \dots, a_n\}, t$

Sortie: un entier

Algorithme:

1. $L_0 := \langle 0 \rangle$
2. Pour i allant de 1 à $|S|$ faire
 - (a) $L_i := \text{union} - \text{trie}(L_{i-1}, L_{i-1} + a_i)$
 - (b) supprimer tout les éléments a de L_i tel que $a > t$
3. retourner le plus grand élément de $L_{|S|}$

Complexité: $=O(t \times |S|)$

Illustration de l'algorithme ($\mathcal{B}$)

Entrée: $S = \{2, 3, 6\}$, $t = 10$

1. $L_0 := \langle 0 \rangle$

2. $L_1 := \langle 0, 2 \rangle$

3. $L_2 := \langle 0, 2, 3, 5 \rangle$

4. $L_3 := \langle 0, 2, 3, 5, 6, 8, 9, 11 \rangle$

5. Suppression d'éléments a tel que $a > 10$:

$$L_3 := \langle 0, 2, 3, 5, 6, 8, 9 \rangle$$

6. retournez 9

Comment réduire la complexité.

Objectif:

- Réduire le nombre d'éléments dans la liste $L_{|S|}$
- Construire une liste J où chaque solution (élément de $L_{|S|}$) de l'instance a un représentant de J .

Liste épurée de L par rapport à δ : Une liste épurée L' de L par rapport à δ est une liste telle que

$$\forall y \in L, \exists z \in L', (1 - \delta)y \leq z \leq y$$

Exemple:

- $L = \langle 0, 2, 3, 5, 6, 8, 9 \rangle$ avec $\delta = 0.2$
- Liste épurée de $L = \langle 0, 3, 6, 9 \rangle$

Algorithme d'épuration

Entrée: $L = \{\ell_1, \dots, \ell_n\}$ triée, un réel δ

Sortie: une liste $L' = \{z_1, \dots, z_m\}$

Algorithme:

1. $L' := \langle \ell_1 \rangle$
2. $dernier := \ell_1$
3. Pour i allant de 2 à $|L|$ faire
 - (a) Si ($dernier < (1 - \delta)\ell_i$) alors
 - i. $L' := L' \cup \{\ell_i\}$
 - ii. $dernier := \ell_i$
4. retourner L'

Complexité: $=O(|L|)$

Illustration d'algorithme d'épuration

Entrée: $L = \langle 0, 2, 3, 5, 6, 8, 9 \rangle$ avec $\delta = 0.2$

$$L'_1 := \langle 0 \rangle \text{ dernier}_1 := 0$$

$$1. (0 < (1 - 0.2) \times 2), L'_2 := \langle 0, 2 \rangle, \text{dernier}_2 := 2$$

$$2. (2 < (1 - 0.2) \times 3), L'_3 := \langle 0, 2, 3 \rangle, \text{dernier}_3 := 3$$

$$3. (3 < (1 - 0.2) \times 5), L'_4 := \langle 0, 2, 3, 5 \rangle, \text{dernier}_4 := 5$$

$$4. (5 > (1 - 0.2) \times 6), L'_5 := \langle 0, 2, 3, 5 \rangle, \text{dernier}_5 := 5$$

$$5. (5 < (1 - 0.2) \times 8), L'_6 := \langle 0, 2, 3, 5, 8 \rangle, \text{dernier}_6 := 8$$

$$6. (8 > (1 - 0.2) \times 9), L'_7 := \langle 0, 2, 3, 5, 8 \rangle, \text{dernier}_7 := 8$$

Algorithme FPTAS nommé $\mathcal{A}$

Entrée: $S = \{a_1, \dots, a_n\}$, t , un réel ϵ

Sortie: un entier ℓ

Algorithme:

1. $S_0 := \langle 0 \rangle$
2. Pour i allant de 1 à n faire
 - (a) $S_i := \text{union} - \text{trie}(S_{i-1}, S_{i-1} + a_i)$
 - (b) $S_i := \text{epuration}(S_i, \frac{\epsilon}{n})$
 - (c) supprimer tous les éléments a de L_i tel que $a > t$
3. retourner le plus grand élément de S_n

Ex: $S = \langle 204, 101, 5, 200 \rangle$, $t = 320$, $\epsilon = 0.2$

1. $S_0 := \langle 0 \rangle$ et $\frac{\epsilon}{n} = 0.05$
2. $i = 1$; (2.a) $S_1 := \langle 0, 204 \rangle$; (2.b), (2.c) $S_1 := \langle 0, 204 \rangle$
3. $i = 2$; (2.a), (2.b), (2.c) $S_2 := \langle 0, 101, 204, 305 \rangle$;
4. $i = 3$;
 (2.a) $S_3 := \langle 0, 5, 101, 106, 204, 209, 305, 310 \rangle$
 (2.b,c) $S_3 := \langle 0, 5, 101, 204, 305 \rangle$
 (2.a) $\langle 0, 5, 101, 200, 204, 205, 304, 305, 505 \rangle$
5. $i = 4$; (2.b,c) $S_4 = \langle 0, 5, 101, 200, 204, 304, 505 \rangle$
 (2.b,c) $S_4 = \langle 0, 5, 101, 200, 204, 304 \rangle$

Conclusion: rés. = 304, rés. opt. = 310, $\rightarrow 310(1 - 0.2) \leq 304$

Qualité de la solution

Théorème 1: Soit C la solution de l'algo: $C^* \times (1 - \epsilon) \leq C$

Preuve:

Soit S_i la i ème liste épuré par l'algorithme

Soit L_i la i ème liste de toutes les solutions possibles pour les i èmes premiers éléments de S .

• on a $\forall y \in L_i, \exists z \in S_i$ tel que $(1 - \frac{\epsilon}{n})^i y \leq z \leq y$.

□

Qualité de la solution

Théorème 1: Soit C la solution de l'algo: $C^* \times (1 - \epsilon) \leq C$

Preuve:

Soit S_i la i ème liste épuré par l'algorithme

Soit L_i la i ème liste de toutes les solutions possibles pour les i èmes premiers éléments de S .

- on a $\forall y \in L_i, \exists z \in S_i$ tel que $(1 - \frac{\epsilon}{n})^i y \leq z \leq y$.
- Comme $C^* \in L_n$, il existe C tel que

$$(1 - \frac{\epsilon}{n})^n C^* \leq C \leq C^*$$

Comme la fonction $(1 - \frac{\epsilon}{n})^n$ est $\nearrow$ en fonction de n ,

$$C^* \times (1 - \epsilon) \leq (1 - \frac{\epsilon}{n})^n C^* \leq C$$

Plus en détail,

Proposition 2: Soit S_i la i ème liste épurée par l'algorithme $\mathcal{A}$. Soit L_i la liste de toutes les solutions possibles pour les i èmes premiers éléments de S . On a

$$\forall y \in L_i, \exists z \in S_i \text{ tel que } (1 - \frac{\epsilon}{n})^i y \leq z \leq y.$$

Preuve: Par induction

- pour $i = 1$, $S_1 = \langle 0, a_1 \rangle$, $L_1 = \langle 0, a_1 \rangle$
- Hl: pour $\forall k < i$, $\forall y \in L_k$, $\exists z \in S_k / (1 - \frac{\epsilon}{n})^k y \leq z \leq y$.
- soit $y' \in L_i$, il existe $y \in L_{i-1}$ tq $y' = y$ ou $y' = y + a_i$
 1. Soit $y \in L_{i-1}$, par Hl, $\exists z \in S_{i-1}$ tq $(1 - \frac{\epsilon}{n})^{i-1} y \leq z \leq y$.
 2. par $\mathcal{A}$, avant l'épuration on a $z \in S_i$ et $z + a_i \in S_i$
 3. Faire une étude de cas en fct du résultat de l'épuration.

Complexité de l'algorithme $\mathcal{A}$.

- Complexité = $O(n \times \text{complexité la boucle for})$.
- complexité la boucle for) dépendant de la cardinalité de S_i .
 1. si $z \in S_i$, alors $z \leq t$ (implique $|S_i| \leq t$).
 2. Après chaque épuration, quelle est la distance entre deux éléments consécutifs z' et z de S_i :
$$z'/z > 1/(1 - \frac{\epsilon}{n})$$
 3. majoration du nombre d'éléments de S_i

$$\log_{1/(1-\frac{\epsilon}{n})} t = \frac{\log(t)}{-\log(1 - \frac{\epsilon}{n})}$$

- Complexité total = $O(n \times \log(t))$ (comme l'entier t est codé en binaire).

En conclusion

Théorème 1: L'algorithme $\mathcal{A}$ est un algorithme entièrement polynomial d'approximation pour le problème de la somme d'un ensemble.

Car:

- Soit C la solution de l'algorithme $\mathcal{A}$: $C^* \times (1 - \epsilon) \leq C$ avec C^* la solution optimale.
- Complexité total = $O(n \times \log(t))$