

Corrigé de l'Epreuve de Rattrapage : Complexité et algorithmique avancée

Exercice 1

On considère deux ensembles d'entiers $A = \{a_1, \dots, a_n\}$ et $B = \{b_1, \dots, b_m\}$ tels que $a_1 < \dots < a_n$ et $b_1 < \dots < b_m$. On rappelle que la différence symétrique de A et B, notée $A \Delta B$, est le sous ensemble des éléments de A ou de B qui ne sont pas communs à A et B. On a donc $A \Delta B = (A \cup B) \setminus (A \cap B)$. L'objet de l'exercice est de proposer un algorithme pour le calcul de $A \Delta B$.

1. Quelle est la valeur de $A \Delta B$ si $A = \emptyset$? Même question si $B = \emptyset$. (1 pts)

On suppose ici que $A \neq \emptyset \neq B$ et l'on note $A' = \{a_2, \dots, a_n\}$ et $B' = \{b_2, \dots, b_m\}$.

2. Démontrer que : (1 pts)

$A \Delta B =$

$$\{A' \Delta B'\} \quad \text{si } a_1 = b_1 \quad \{a_1\} \cup (A' \Delta B) \quad \text{si } a_1 < b_1 \quad \{b_1\} \cup (A \Delta B') \quad \text{si } a_1 > b_1$$

3. Proposer un algorithme itératif pour le calcul de $A \Delta B$. Calculer sa complexité (3 pts).
4. En vous basant sur la propriété démontrée à la question 2, proposer un algorithme récursif pour le calcul de $A \Delta B$. Calculer sa complexité (3 pts)

Solution:

1. Si $A = \emptyset$ alors $A \Delta B = B$ et si $B = \emptyset$ alors $A \Delta B = A$;

2. Démontrons que : $A \Delta B =$
 $\{A' \Delta B'\} \quad \text{si } a_1 = b_1 \quad \{a_1\} \cup (A' \Delta B) \quad \text{si } a_1 < b_1 \quad \{b_1\} \cup (A \Delta B') \quad \text{si } a_1 > b_1$

Si $a_1 = b_1$ alors $a_1 \in A \cap B \Rightarrow A \Delta B = A' \Delta B'$

Si $a_1 < b_1$ et puisque les deux ensemble A et B sont triés dans l'ordre croissant, alors $a_1 < b_j \forall j \in$

$[1 \dots m] \Rightarrow a_1 \notin B \Rightarrow a_1 \in A \Delta B$ et il reste à déterminer les autres éléments de $A \Delta B$ à partir de A' et B

Si $a_1 > b_1 \Rightarrow b_1 < a_1$ et on retombe sur le cas qu'on vient tout juste de démontrer

$\Rightarrow b_1 \in A \Delta B$, et de même que pour le point précédent, les autres éléments de $A \Delta B$ seront calculé à partir de A et B' .

(c.q.f.d)

3. Algorithme itératif pour le calcul de $A \Delta B$.

Début

/* A et B sont deux tableaux d'entiers en entrée et C contient la différence symétrique de A et B */

i := 1 ; j := 1 ; k := 1 ;

Tant que (i < n ou j < m) faire si A[i] < B[j] alors C[k] := A[i] ;

k := k+1 ; i := i+1 ;

Sinon si A[i] > B[j] alors C[k] := B[j] ;

k := k+1 ; j := j+1 ;

fsi ;

Sinon i := i+1 ; j := j+1 ;

fsi ;

Fait ;

Fin.

Corrigé de l'Epreuve de Rattrapage : Complexité et algorithmique avancée

Pire cas : les éléments de l'ensemble A sont tous inférieurs (respectivement supérieurs) aux éléments de l'ensemble B.

Nombre d'itération dans ce cas = $n + m < 2 * \max(n, m) \Rightarrow$ la complexité = $O(\max(n, m))$.

4. Algorithme récursif

Diff_symetrique(i,j,k : entier) ;

Début

Si (i <= n ou j <= m) alors Si A[i] < B[j] alors C[k] := A[i] ;

Diff_symetrique(i+1, j, k+1);

Sinon Si A[i] > B[j] alors C[k] := B[j] ;

Diff_symetrique(i, j+1, k+1) ;

Sinon Diff_symetrique(i+1, j+1, k) ;

Fsi ;

Fsi ;

Fsi ;

Fin ;

$$\begin{cases} (A \Delta B) & \text{si } a_1 = b_1 \\ \{a_1\} \cup (A' \Delta B) & \text{si } a_1 < b_1 \\ \{b_1\} \cup (A \Delta B') & \text{si } a_1 > b_1 \end{cases}$$

La complexité est la même que pour le cas itératif $O(\max(n, m))$.

Exercice 2 <12 points>

Considérons le problème de partition d'un ensemble d'entier suivant :

Instance : Un ensemble de n entier $S = \{s_1, s_2, \dots, s_n\}$ et un entier positif k.

Question : Peut-on partitionner l'ensemble d'entiers S en k sous-ensembles distincts de même somme ?

On s'intéresse à la variante du problème ou $k = 2$. La question à laquelle doit répondre le problème devient :

Question : Peut-on partitionner l'ensemble d'entiers S en (2) sous-ensembles distincts S_1 et S_2 tel que :

$$\sum_{i=1}^m S_{i1} = \sum_{j=1}^p S_{j2}$$

Sachant que : $m + p = n$, et S_{i1} : représente les éléments de l'ensemble S_1 et S_{j2} représente les éléments de l'ensemble S_2 .

Nous considérons par ailleurs le concept de solution au sens large pour le problème. Une solution est dite positive (valide) si elle répond par l'affirmation à la question du problème. Elle est dite négative dans le cas contraire.

Soit l'ensemble d'entier suivant : $S = \{1, 2, 5, 10, 9, 15, 21, 19, -4, 7, 12, 6, 3\}$

- 1) A quoi correspond une solution potentielle au problème du 2-Partition ? Donner un exemple de solution positive et un autre de solution négative. (2 pts)
- 2) Proposer une structure de données pour représenter une solution. (2 pts)
- 3) On s'intéresse à construire des solutions potentielles. Ecrire un algorithme permettant d'engendrer une solution quelconque au problème. Calculer la complexité de l'algorithme proposé. (3 pts)
- 4) Ecrire un algorithme permettant de vérifier que la solution engendrée est positive ou négative. Calculer sa complexité. (3 pts)
- 5) Que peut-on conclure sur le problème de partition ? (2 pts).

Solution :

Corrigé de l'Epreuve de Rattrapage : Complexité et algorithmique avancée

- 1) Une solution potentielle au problème du 2-Partition est un partitionnement ou division de l'ensemble d'entier de départ noté S en deux sous ensemble distinct S1 et S2 tel que la somme des éléments de S1 et égale à la somme des éléments de S2.

Solution positive : $S1 = (21, -4, 5, 7, 9, 15) = S2 = (19, 1, 2, 6, 3, 10, 12) = 53$

Solution négative : $S1 = (21, 5, 7, 9, 15) = 57 \neq S2 = (19, 1, 2, 6, 3, 10, 12, -4) = 49$

- 2) Une structure de données pour représenter une solution au problème, Deux possibilités :

- Deux vecteurs d'entiers S1 et S2 de dimension respectivement m et p
- Un vecteur binaire noté Sol de dimension N

$$\text{Sol}[i] = \{1 \text{ si } S[i] \in S1 \quad 0 \text{ si } S[i] \in S2$$

- 3) Algorithme de résolution

Principe de l'algorithme :

Tout d'abord, construire deux sous ensemble distinct à partir d'un ensemble d'entiers sous la contrainte que la somme des éléments de S1 est égale à la somme des éléments de S2, signifie que :

$$\sum S = \sum S1 + \sum S2 \Rightarrow \sum S1 = \sum S2 = \frac{1}{2} * \sum S$$

L'algorithme commence par calculer la somme des éléments de l'ensemble S pour en déduire la suite la somme des éléments de chaque partition.

Ensuite, diviser l'ensemble S en deux sous ensemble de même taille. Avec ce premier partitionnement on va avoir un des deux sous-ensembles avec une somme $< S/2$ et le second est $> S/2$.

A partir de ce point, l'algorithme devra essayer toute les combinaisons possible entre S1 et S2 afin d'équilibrer la somme des deux sous ensemble.

Complexité de l'algorithme : nombre de solution possible = nombre de combinaison possible.

Construire un ensemble de P élément à partir de N est une combinaison de P parmi n noté : $C_n^P = \frac{N!}{P!(n-p)!}$

Mais dans notre cas la taille des ensembles S1 et S2 n'est pas fixé au préalable et vari de 1 à n-1

Donc le nombre de combinaison et de solutions possible est égale à :

$$\sum_{p=1}^{n-1} C_n^P = \sum_{p=1}^{n-1} \frac{N!}{P!(n-p)!} = 2^n$$

Ce qui implique que la complexité de l'algorithme de résolution est de l'ordre de $O(2^n)$.

- 4) Algorithme de validation

Principe de l'algorithme : l'algorithme de validation vérifie si une solution X donnée en entrer (en fonction des structures de données utilisées) répond au critère du problème de partition à savoir :

- Les éléments des deux sous-ensembles S1 et S2 appartiennent à l'ensemble de départ S.
- Chaque élément de S est soit dans l'ensemble S1 ou dans l'ensemble S2 et ne peut être dans les deux à la fois.
- Si on considère que m et p représente la taille respective des sous ensemble S1 et S2 alors la taille de l'ensemble S qui est noté n doit être égale à m+p.
- La somme des éléments de S1 et égale à la somme des éléments de S2.
- Un critère supplémentaire peut être aussi de vérifier si la somme globale des éléments de S peut être divisée en deux (paire).

Corrigé de l'Epreuve de Rattrapage : Complexité et algorithmique avancée

Si toutes ces conditions sont vérifiées alors la solution est dite valide sinon elle est dite non valide.

La complexité de cet algorithme est :

Pour la vérification de l'appartenance des éléments à S et que S1 et S2 sont distinct : n^2

Pour le calcul de la somme de S1 et S2 c'est $m+p = n$

Donc cet algorithme est de complexité $O(n^2)$.

- 5) L'algorithme de résolution est de complexité exponentielle $O(2^n)$, cet algorithme ne peut être mis en place de fait qu'il n'est pas pratique et produit un temps d'exécution irréalisable pour des instances de N très grand, i.e, pour des ensembles d'entiers très grand.

L'algorithme de validation quant à lui est de complexité $O(n^2)$, quadratique ou polynomiale. Cet ordre de complexité est réalisable.

L'ordre de complexité de l'algorithme de validation est polynomial, ce qui implique que le problème du 2-Partition est NP

Si nous voulons généraliser pour le problème de partition en K sous ensemble, nous pouvons déduire qu'il est de classe NP lui aussi.