

Examen d'algorithmique

Mercredi 4 novembre 2015 10h45-12h45 / Aucun document autorisé

Mode d'emploi : Le barème est donné à titre indicatif. **La qualité de la rédaction des algorithmes et des explications sera très fortement prise en compte pour la note.** On peut toujours supposer une question résolue et passer à la suite. Le Master Theorem est rappelé en annexe.

Exercice 1 : Dérouler un algorithme (3 points)

On considère l'algorithme ci-dessous :

```

Def P(tableau T, entiers: x , y) :
  Si (x >= y) Alors :
    Retourner 0
  Si (x == y-1) Alors :
    Retourner T[x]
  Sinon :
    m1 = (2*x+y)/3
    m2 = (x+2*y)/3
    Retourner P(T,x,m1)+P(T,m2,y)

```

1. Appliquer l'algorithme sur le tableau $[1, 5, 7, 8, 9, 19, 10, 8, 5]$ avec $x=0$ et $y=9$ (on suppose que le premier indice du tableau est 0). On précisera tous les appels récursifs effectués et leurs résultats.
2. En déduire l'équation vérifiée par sa complexité $T(n)$ où n est le nombre d'éléments dans la zone $x, \dots, y-1$.
3. Appliquer le Master Theorem et déduire la complexité de l'algorithme pour un tableau de taille n . $\Theta(n^2)$

Exercice 2 : Application du Master Theorem - 4 points

Donner les bornes asymptotiques obtenues grâce au Master Theorem pour les fonctions suivantes (en supposant qu'elles sont constantes pour $n \leq 2$).

1. $T(n) = 2 \cdot T(\frac{2n}{3}) + n$
2. $T(n) = 2 \cdot T(\frac{2n}{3}) + n^2$
3. $T(n) = 2 \cdot T(\frac{n}{2}) + n$
- ④ $T(n) = T(\frac{2n}{3}) + \log n$

$\log(n)$ 1

↗ 1^{er} cas
 — 2nd cas
 ↘ 3^{ème} cas

○ majeure
 ⊖ encadré
 ∩ mineure

Exercice 3 : MinMax - 5 points

Donner un algorithme "diviser-pour-régner" qui retourne une paire correspondant à l'indice du min et l'indice du max d'un tableau :

IndiceMinMax (tableau de valeurs T, entiers: bg , bd) : paire d'entiers
Donner sa complexité et justifier l'algorithme.

Exercice 4 : Sous-ensemble de somme exacte - 8 points

On cherche à résoudre le problème suivant : étant donné un ensemble de n valeurs entières $\mathcal{E} = \{a_1, a_2, \dots, a_n\}$ et une valeur cible V , on veut savoir si il est possible de trouver un sous-ensemble de $\mathcal{E}$ dont la somme des éléments vaut exactement V . Par exemple, pour $\mathcal{E} = \{2, 8, 3, 7\}$ et $V = 12$, c'est possible avec le sous-ensemble $\{2, 3, 7\}$, mais si on prend $V = 14$, ce n'est pas possible...

1. Proposer un algorithme de programmation dynamique pour résoudre ce problème. On cherchera à construire un tableau Booléen à deux dimensions $T[i, v]$ où $0 \leq i \leq n$ et $0 \leq v \leq V$ tel que $T[i, v]$ vaut VRAI si et seulement si il est possible de trouver un sous-ensemble de $\{a_1, \dots, a_i\}$ de somme v . Justifier l'algorithme et donner sa complexité.
2. Appliquer l'algorithme sur l'exemple $\mathcal{E} = \{2, 8, 3, 7\}$ et $V = 15$. On donnera le tableau complet $T[i, v]$.
3. Proposer un algorithme qui permet de reconstituer une solution à partir du tableau T .
4. Adapter l'algorithme de la première question au cas où on cherche un multi-ensemble (où un même élément a_i peut apparaître plusieurs fois). Par exemple, pour $\mathcal{E} = \{2, 8, 3, 7\}$ et $V = 14$, c'est possible en prenant $\{7, 7\}$.

Annexe**Master theorem :**

Soient $a \geq 1$, $b > 1$, $f(n)$ une fonction positive et
$$t(n) = \begin{cases} a \cdot t(\frac{n}{b}) + f(n) & \text{si } n > 1 \\ \Theta(1) & \text{si } n = 1 \end{cases};$$

- Si $f(n) = O(n^{\log_b a - \epsilon})$ avec $\epsilon > 0$, alors $t(n) = \Theta(n^{\log_b a})$
- Si $f(n) = \Theta(n^{\log_b a})$, alors $t(n) = \Theta(n^{\log_b a} \cdot \log n)$
- Si $f(n) = \Omega(n^{\log_b a + \epsilon})$ pour $\epsilon > 0$ et si $\exists c < 1$ tel que $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$ pour n assez grand, alors $t(n) = \Theta(f(n))$