

Examen d'algorithmique

Mercredi 13 janvier 2016 12h–15h / Aucun document autorisé

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

Exercice 1 : Dérouler et analyser (2,5 points)

On considère l'algorithme ci-dessous :

```
Def P(T,bg,bd) :  
  Si bg>bd Alors Retourner 0  
  Si bg==bd Alors Retourner T[bg]  
  Sinon  
    m = (bg+bd)/2  
    x1 = P(T,bg,m)  
    x2 = P(T,m+1,bd)  
  Retourner x1+x2+T[m]
```

1. Décrire ce que fait l'algorithme P appelé sur le tableau $[4, 1, 8, 7, 5, 3, 2, 9, 14, 17, 6]$ avec $bg = 0$ et $bd = 10$. On détaillera tous les appels récursifs effectués. Quelle est sa complexité pour un tableau de taille n ?
2. Et que devient la complexité si on remplace les lignes 6 et 7 par :

```
x1 = P(T,bg,(bg+m)/2)  
x2 = P(T,(m+1+bd)/2 +1,bd)
```

Exercice 2 : MinMax (2,5 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 3 : Plus longue sous-séquence stable (5 points)

Étant donné un tableau d'entiers T d'indices $0, \dots, |T| - 1$, on cherche à calculer la taille du sous-tableau stable ie plus long de T. Un sous-tableau stable est une suite continue d'indices du tableau contenant la même valeur. Par exemple, avec le tableau $T = [1, 6, 8, 8, 8, 4, 2, 1, 1, 8, 4, 4]$, la valeur recherchée est 3 et elle correspond au sous-tableau d'indices (2, 3, 4) de taille 3.

1. Proposer un algorithme *Diviser-pour-régner* pour résoudre ce problème. L'algorithme sera de la forme $\text{algo1}(T, l, u)$ et renverra la valeur recherchée (la taille du sous-tableau stable le plus long) pour la partie de T entre les indices l et u . Appliquer votre algorithme sur $T = [1, 6, 8, 8, 8, 4, 2, 1, 1, 8, 4, 4]$. Donner sa complexité.
2. Amélioration : proposer un autre algorithme *Diviser-pour-régner* $\text{algo2}(T, l, u)$ qui renvoie comme résultat un triplet d'entiers (p, t, s) où t est la taille du sous-tableau stable le plus long pour la partie de T entre les indices l et u (à vous de voir ce que peuvent être p et s !). Quelle est sa complexité? L'appliquer sur l'exemple.
3. Proposer un algorithme de programmation dynamique pour résoudre ce problème. On pourra construire des tableaux $\text{Nb}[-]$ et $\text{Nbf}[-]$ tels que :
 - $\text{Nb}[i]$ est la taille d'un sous-tableau stable le plus long dans la partie de T restreinte aux indices $0 \leq j \leq i$ et

2. Après plusieurs consultations chez le kiné, nos déménageurs renoncent aux économies de cartons et changent de stratégie : désormais, ils cherchent à répartir *équitablement* les livres dans les k cartons. On supposera qu'on peut toujours les ranger dans les k cartons et on ne s'intéressera donc plus à la capacité maximale M de chaque carton. Il s'agit donc de répartir les n livres en k paquets "équitables".

Pour partager équitablement les livres (toujours en gardant l'ordre alphabétique !), on va procéder comme suit : on va chercher à répartir tous les livres dans les k cartons en minimisant le poids du plus carton le plus lourd. Le problème se définit donc comme suit :

Données : une séquence de n livres avec des poids (entiers positifs) $\{p_1, \dots, p_n\}$ et un nombre k de cartons.

Résultat : une répartition en k paquets respectant l'ordre initial des livres et pour laquelle le poids du carton le plus lourd est minimal.

Par exemple, dans l'exemple ci-dessus des 12 livres ci-dessus avec 3 cartons, on obtiendrait la répartition suivante (et le poids du carton le plus lourd est 900)

1	2	3	4	5	6	7	8	9	10	11	12
200	150	400	100	500	300	100	100	150	75	50	100
carton 1 (850)				carton 2 (900)			carton 3 (875)				

L'objectif est ici d'écrire un algorithme de programmation dynamique pour résoudre ce problème. Pour cela, nous allons construire un tableau $M[i, c]$ pour $1 \leq i \leq n$ et $1 \leq c \leq k$ tel que $M[i, c]$ correspondra au poids **minimal** du carton le plus lourd pour toutes les répartitions des livres s_1, \dots, s_i dans c paquets.

- (a) Écrire un algorithme pour calculer le tableau $M[-, -]$ pour n livres et $k = 2$. Appliquer l'algorithme sur l'exemple ci-dessus.

On s'intéresse désormais au cas général avec $k \geq 1$:

- (b) Que vaut $M[1, c]$ pour $1 \leq c \leq k$?
- (c) Que vaut $M[i, 1]$ pour $1 \leq i \leq n$?
- (d) Exprimer $M[i, c]$ en fonction des éléments $M[j, c-1]$ avec $1 \leq j \leq i \leq n$ et $1 \leq c \leq k$ et des poids p_1, \dots, p_i . On regardera toutes les façons de remplir le dernier carton (numéro c) en réutilisant les calculs précédents pour les autres cartons.
- (e) Donner un algorithme pour calculer $M[i, c]$. Quelle est sa complexité ?
- (f) Illustrer ce calcul en donnant les valeurs de $M[-, -]$ pour l'exemple précédent des 12 livres mais avec $k = 4$.
- (g) Comment retrouver une répartition optimale à partir du tableau $M[-, -]$, des poids p_1, \dots, p_n et de k ?

Exercice 5 : Les vaches de Narayana (4 points)

On s'intéresse au problème posé par Narayana (un mathématicien indien) au 14^e siècle : "Chaque année, une vache met au monde un veau. À partir de la quatrième année,

chaque veau donne à son tour et au début de chaque année, naissance à un veau. Quel est le nombre de vaches et de veaux après une durée de 17 ans ?”

Au départ, on a une unique vache. Après l'année 1, on a donc une vache et un veau. Après l'année 2, on a une vache (toujours la même !), et deux veaux (un nouveau, et l'autre qui est dans sa seconde année). Après l'année 3, on a une vache et 3 veaux. Après l'année 4, on a deux vaches et 4 veaux (le premier veau est devenu vache et a donné naissance à un veau...). Etc.

On peut donc remplir un tableau comme ci-dessous :

année	vaches	veau de 1 an	veau de 2 ans	veau de 3 ans
	v_i	x_i^1	x_i^2	x_i^3
1	1	1	0	0
2	1	1	1	0
3	1	1	1	1
4	2	2	1	1
...
17	v_{17}	x_{17}^1	x_{17}^2	x_{17}^3

L'objectif est donc de calculer la somme $T_{17} = v_{17} + x_{17}^1 + x_{17}^2 + x_{17}^3$.

1. Compléter le tableau ci-dessus jusqu'à l'année 8.

2. Écrire les relations entre v_{n+1} , x_{n+1}^1 , x_{n+1}^2 , x_{n+1}^3 et les termes précédents des suites.

En déduire une définition simple de la suite v_n du nombre de vaches. Donner un algorithme pour calculer v_n et en déduire un calcul du nombre total d'animaux T_n après l'année n . Donner sa complexité.

Donner T_0 le nombre d'animaux après l'année 1?

3. Soit $Nb_{n,k}$ le nombre d'animaux de la k -ème génération après l'année n . Après l'année 1, il y a un animal de la première génération (la première vache) et un de la seconde génération (le veau). Après l'année 4, il y a un animal de la première génération, 4 de la seconde génération, et un de la troisième génération (le veau du premier veau devenu vache), etc.

Écrire la relation entre $Nb_{n,k}$ et les termes précédents $Nb_{n',k'}$ avec $k' \leq k$ et $n' \leq n$.

PS : On trouvera une jolie solution musicale à ce problème dans la pièce de Tom Johnson (voir sur youtube).

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 $t(n) = \Theta(f(n))$