Université d'Ottawa
École de science informatique et de génie électrique
CSI 3505, Automne 2021 Prof: Nejib Zaguia

Devoir #1 Solution

Solution Question 1.

Utilisez les notations adéquates parmi O et Θ afin de comparer les comportements asymptotiques des paires des fonctions suivantes **(Prouver chacune des relations)**

- 2^{n+1} et 2^n **[$2^{n+1} = 2 * 2^n$ et donc $2^{n+1} = \Theta(2^n)$]**
- 4^n et 2^n **[$\lim_{n\to\infty} \frac{4^n}{2^n} = 2^n = \infty$ et donc $2^n \in O(4^n)$]**
- 2^{n+1} et n^6 **[en appliquant la règle de l'hôpital, plusieurs fois consécutives on obtient : $n^6 = O(2^{n+1})$]**

- n log n et $\log^3 n$

  $\lim_{n\to\infty} \frac{\log^3 n}{n \log n} = \lim_{n\to\infty} \frac{\log^2 n}{n} = \lim_{n\to\infty} \frac{\log^2 e * \ln^2 n}{n}$

  en appliquant la règle de l'hôpital, on obtient:

  $\lim_{n\to\infty} \frac{\log^2 e * \ln^2 n}{n} = \lim_{n\to\infty} \frac{\log^2 e * 2 * 1/n * \ln n}{n} = 2 * \log^2 e * \lim_{n\to\infty} \frac{\ln n}{n2} = 0$ **(en appliquant la règle de l'hôpital une deuxième fois) donc $\log^3 n \in O(n \log n)$**

- $3n^2$ et $15n(n+1)$ $\lim_{n\to\infty} \frac{3n^2}{15n(n+1)} = 3/15$ **et donc $3n^2 = \Theta(15n(n+1))$]**

Solution Question 2

$$a) \sum_{i=2}^{n-2} \sum_{j=i+1}^{n-1} 2 = 2 \sum_{i=2}^{n-2} [(n-1) - (i+1) + 1] = 2 \sum_{i=2}^{n-2} (n-i-1) = 2[\sum_{i=2}^{n-2} n - \sum_{i=2}^{n-2} i - \sum_{i=2}^{n-2} 1]$$

$$= 2 * [(n-2-2+1)n - \left(\frac{(n-2).(n-1)}{2} - 1\right) - (n-2-2+1)]$$

$$= 2(n^2 - 3n) - (n^2 - 3n) - 2(n-3) = (n^2 - 5n + 6)$$

$$\leq c.n^2 \quad \in O(n^2)$$

$$b) \quad \sum_{i=1}^{n} (\log n + 1) = \sum_{i=1}^{n} (\log n) + n = \log(n!) + n$$

$$\in \Theta(n \log n)$$

Note: En supposant que n est une puissance de 2 (i.e. $n = 2^k$), la boucle « while » se fait exactement (k+1) fois.

$$c) \quad \sum_{i=1}^{n-1} \sum_{k=1}^{i} \sum_{j=1}^{2^i} 1 = \sum_{i=1}^{n-1} \sum_{k=1}^{i} (2^i - 1 + 1) = \sum_{i=1}^{n-1} \sum_{k=1}^{i} 2^i$$

$$= \sum_{i=1}^{n-1} (i - 1 + 1)(2^i) = \sum_{i=1}^{n-1} i.2^i$$

Dapres la formule $\sum_{i=1}^{n} i.2^i = (n-1)2^{n+1} + 2$

Donc le nombre total de multiplications: $(n-2)2^n + 2 = n2^n - 2^{n+1} + 2$

Solution Question 3

a) Entrée : S tableau trié S[1...n] contient n nombres et un nombre x.

Sortie : la position de x dans A (-1 si x n'existe pas dans A.)

```
DEBUT
i = 4
Tant que i < n
  Si S[i] > x
    Pour j = 0 jusqu'a 3
      Si S[i − j] = x
        Retourne i-j
  i = i + 4
Retourne -1
FIN
```

b) Analyse du pire cas

Taille de l'entrée : n, le nombre des entiers dans S, et n=4k (multiple de 4)

Opérations élémentaires à compter : Comparaisons de x avec les éléments de S.

L'entrée qui cause le pire cas : Lorsque $s[n-3] \leq x < s[n-2]$.

Pour le pire cas l'algorithme va comparer x avec S[4j] pour j=1, 2, 3, ..., k,

Découvre que x < A[4k=n] et puis va comparer x avec S[i] pour i = n-1, n-2 et n-3.

Au total l'algorithme va dans le pire cas faire k + 3 comparaisons. Donc

$$W(n) = n/4 + 3 = O(n).$$

c) Analyse du moyen cas:

Les exemplaires de ce problème sont divisé sur 2n+1 classes I_j pour j= 1, …, 2n+1.

Pour tout j = 1, 2, …, 2n+1 la probabilité $p(I_j) = 1/(2n+1)$ pour j = 1, 2, …, 2n+1. Il faut trouver $t(I_j)$ pour chaque valeur de j, ensuite faire la somme en utilisant la formule adéquate.

Pour faire la somme un peu plus facilement, considérant les éléments de A comme des regroupements de 4 éléments :

A[4j-3] A[4j-2] A[4j - 1] A[4j]

On a ces regroupements de 4 nombres pour j =1, j=2, ..., j=k, où n=4k. Alors on a k groupes. Ci-dessous sont les valeurs possibles de t(I) pour les positions différentes de x dans cet interval (ceci inclus le cas I_{2n})

	A[4j-4]	A[4j-3]	A[4j-2]	A[4j-1]	A[4j]			
	↑	↑	↑	↑	↑	↑	↑	↑
	x	x	x	x	x	x	x	x
t(I)	j+3	j+3	j+3	j+2	j+2	j+1	j+1	j

Il est aussi clair que $t(I_{2n}) = 4$ et $t(I_{2n+1}) = k$.

En utilisant notre formule pour le moyen cas, et la sommation des 8 cas pour toutes les valeurs de j plus le dernier cas x > A[n] (le cas I_{2n} est inclus dans le calcul suivant), on a:

$$A(n) = \sum_{r=1}^{2n+1} p(I_r)t(I_r)$$

$$= \frac{1}{2n+1} \sum_{j=1}^{k} [3(j+3) + 2(j+2) + 2(j+1) + j] + \frac{k}{2n+1}$$

$$= \frac{1}{2n+1} \sum_{j=1}^{k} (8j + 15) + \frac{k}{2n+1} \quad \text{(Simplification)}$$

$$= \frac{8}{2n+1} \sum_{j=1}^{k} j + \frac{15k}{2n+1} + \frac{k}{2n+1}. \quad \text{(Simplification)}$$

$$= \frac{8}{2n+1} \frac{k(k+1)}{2} + \frac{16k}{2n+1} \quad \text{(formule de sommation des k premiers nombres)}$$

$$= \frac{4k(k+1) + 16k}{2n+1} \quad \text{(Simplification)}$$

$$= \frac{4k^2 + 20k}{2n + 1} \quad \text{(Simplification)}$$

$$= \frac{4(n/4)^2 + 20(n/4)}{2n + 1} \quad \text{(k=n/4 et Simplification)}$$

$$= \frac{n^2 + 20n}{4(2n+1)}$$

Ceci est asymptotiquement équivalent à n/8 et donc d'ordre n.

Solution Question 4.

a) L'algorithme n'est pas correcte. Soit p1=5, p2=10, s1=9 et s2=14. L'algorithme va assigner le ski s2 à p1 et le ski s1 à p2, avec un coût de 10. Par contre si on donne s1 a p1 et s2 a p2 on a un coût de 8.

b) L'algorithme est correcte. On fera une preuve par contradiction. Supposons que les skieurs et les skis sont tries par ordre croissant $P_1, \ldots, P_n$ et $S_1, \ldots S_n$ et donc l'algorithme dans b) nous donne la distribution G = $(P_1, S_1), \ldots, (P_n, S_n)$. Supposons que cet algorithme n'est pas optimal pour cet ensemble de skis et skieurs. Donc, il existe une autre distribution T des skis qui est plus efficace :

T = (P_1, $S_{\alpha(1)}$), …, (P_n, $S_{\alpha(n)}$)
Les deux algorithmes différent a partir d'un certain index i. Dans G le skieur i a le ski Si, et dans T il a le ski Sj. Si on échange les skis pour les skieurs i et j, on a une nouvelle distribution T'.

	P_1,	P_2,	…	P_{i-1}	P_i	…	P_j	…	P_n
G :	S_1,	S_2,	…	S_{i-1}	S_i	…	S_j	…	S_n
T :	S_1,	S_2,	…	S_{i-1}	S_j	…	S_i	**…**	$S_{\alpha(n)}$
T' :	S_1,	S_2,	…	S_{i-1}	S_i	…	S_j	**…**	$S_{\alpha(n)}$

Coût (T') = Coût (T) – (|t(Pi)- t(Sj)| + |t(Pj)- t(Si)| - |t(Pi)-t(Si)| - |t(Pj)-t(Sj)|)
Il suffit de prouver que : (|t(Pi)- t(Sj)| + |t(Pj)- t(Si)| - |t(Pi)-t(Si)| - |t(Pj)-t(Sj)|) ≥ 0, ce qui indiquerai que T ne peut être optimale, puisque T' aurai un coût plus petit.

Il y a plusieurs cas simples a considérer. [On sait que t(Si) ≤ t(Sj) et t(Pi) ≤ t(Pj)]

1) t(Pi) ≤ t(Pj) ≤ t(Si) ≤ t(Sj)
2) t(Pi) ≤ t(Si) ≤ t(Pj) ≤ t(Sj)
3) t(Pi) ≤ t(Si) ≤ t(Sj) ≤ t(Pj)
4) t(Si) ≤ t(Sj) ≤ t(Pi) ≤ t(Pj)
5) t(Si) ≤ t(Pj) ≤ t(Sj) ≤ t(Sj)
6) t(Si) ≤ t(Pi) ≤ t(Pj) ≤ t(Sj)

Dans tous ces cas, (|t(Pi)- t(Sj)| + |t(Pj)- t(Si)| - |t(Pi)-t(Si)| - |t(Pj)-t(Sj)|) ≥ 0

Solution Question 5.

a) Le nombre minimum de segments devrait être le rapport donné par le total de la taille des fichiers par rapport à la taille des segments.

ie $\lceil \sum f_i / K \rceil$ pour i=1,2,…,n

b) K=2 et on a 8 fichiers
Apres avoir trié en ordre décroissant, on a {0.8, 0.8 ,0.8, 0.8, 0.6, 0.6, 0.6, 0.6 }
L'heuristique donnerait 4 segments
Segment 1 : | 0.8 | 0.8 |
Segment 2 : | 0.8 | 0.8 |
Segment 3 : | 0.6 | 0.6 | 0.6|
Segment 4 : | 0.6 |

Une meilleure solution pourrait utiliser 3 segments
Segment 1 : | 0.8 | 0.6 | 0.6|
Segment 2 : | 0.8 | 0.6 | 0.6|
Segment 3 : | 0.8 | 0.8 |

Solution Question 6.

Clairement on peut ignorer tous les A[i] et B[i] dès que i>k.
Comparons A[k/2] et B[k/2]
Si A[k/2] = B[k/2] alors on a trouvé le k nième plus petit éléments de l'union des deux listes A et B.

Si A[k/2] < B[k/2] alors pour tout i > k/2, B[i] est plus grand que plus ce que k valeurs et ne peut être la solution. Par conséquent on peut éliminer la moitié de la liste B « B[i] ou i>k/2 ».

Si A[k/2] > B[k/2] même argument permet d'éliminer la moitié de la liste A « A[i] ou i>k/2 ».

Donc avec une seule comparaison on passe d'un problème de taille k a un problème de taille 3k/4

Cas de base:

- Si la longueur d'une des deux listes est 0 alors la solution est le k nième terme de la deuxième liste.

Diviser:

- If mid index of *a* + mid index of *b* is less than *k*:
    - If mid element of *a* is greater than mid element of *b*, we can ignore the first half of *b*, adjust *k*.
    - Otherwise, ignore the first half of *a*, adjust *k*.
- If *k* is less than sum of mid indices of *a* and *b*:
    - If mid element of *a* is greater than mid element of *b*, we can safely ignore second half of *a*.
    - Otherwise, we can ignore second half of *b*.