NTU CSIE 2016 Fall Algortihm 1st Miterm Solutions

Chih Yao Chang

2016/11/01

1 Problem 3

We showed an $O(n)$-time algorithm for finding the k-th largest number in an array of n distinct numbers via an initial division of the input into groups of five numbers. What would the time complexity of the algorithm be if the initial group size is (1) three, (2) seven, and (3)$\lceil log_2 n \rceil$? Justify your answers.

1. group size = 3
   (a) $T(n) = T(\frac{1}{3}n) + max(|X_>|, |X_<|) + O(n) = T(\frac{1}{3}n) + T(\frac{2}{3}n) + O(n)$ (1 points)
   (b) $T(n) = T(\frac{1}{3}n) + T(\frac{2}{3}n) + O(n) \neq O(n)$ (4 points)
2. group size = 7
   (a) $T(n) = T(\frac{1}{7}n) + max(|X_>|, |X_<|) + O(n) = T(\frac{1}{7}n) + T(\frac{5}{7}n) + O(n)$ (1 points)
   (b) $T(n) = T(\frac{1}{7}n) + T(\frac{5}{7}n) + O(n) = O(n)$ (4 points)
3. group size = $\lceil log_2 n \rceil$
   (a) $T(n) = T(\frac{n}{\lceil log_2 n \rceil}) + max(|X_>|, |X_<|) + O(n) = T(\frac{n}{\lceil log_2 n \rceil}) + T((1 - \frac{(\lfloor \lceil log_2 n \rceil + 1)/2 \rfloor}{2 \times \lceil log_2 n \rceil})n) + O(n) \leq T(\frac{n}{\lceil log_2 n \rceil}) + T((1 - \frac{\lceil log_2 n \rceil}{4 \times \lceil log_2 n \rceil})n) + O(n) = T(\frac{n}{\lceil log_2 n \rceil}) + T(\frac{3}{4}n) + O(n)$ (5 points)
   (b) $T(\frac{n}{\lceil log_2 n \rceil}) + T(\frac{3}{4}n) + O(n) = O(n)$ if $\lceil log_2 n \rceil > 4$ (5 points)

Please refer slides *algo2016fall05* p.31∼34 for the proof of part(a) and p.23∼30 for the proof of part(b).

2 Problem 4

Prove of disprove the recurrence relation

$$T(n) = \begin{cases} 1, & \text{if } n \leq 2 \\ \sqrt{n} \cdot T(\sqrt{n}) + n, & \text{if } n \text{ otherwise} \end{cases}$$

implies $T(n) = O(n\text{loglog}n)$.

By definition, we have

$$\begin{cases} T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n \\ T(\sqrt{n}) = \sqrt[4]{n} \cdot T(\sqrt[4]{n}) + \sqrt{n} \\ \cdots \\ T(\sqrt[2^k]{n}) = 1, \text{ where } k = \lceil \text{loglog} n \rceil \end{cases} \quad \text{(10 points)}$$

$$\Rightarrow \begin{cases} T(\sqrt[2^{k-1}]{n}) = 2 + \sqrt[2^{k-1}]{n} \leq 2 \times \sqrt[2^{(k-1)}]{n} \\ T(\sqrt[2^{k-2}]{n}) = \sqrt[2^{k-1}]{n} \cdot T(\sqrt[2^{k-1}]{n}) + \sqrt[2^{k-2}]{n} \leq 3 \times \sqrt[2^{k-2}]{n} \\ \cdots \\ T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n \leq (k+1) \times n = O(n\text{loglog}n) \end{cases} \quad \text{(10 points)}$$

3 Problem 5

The purpose of this question is to understand the process of potential method. You get 10 points if we can see a complete framework of potential method from your answer, full credits if your proof is correct with potential method.

The complete potential method process includes:

- a potential function $\Phi(i)$
- the relation between actual cost and amortized cost $\hat{t}_i = t_i + \Phi_i - \Phi_{i-1}$
- use the summation of amortized cost as the upper bound of the actual total cost $\sum_{i=1}^{n} \hat{t}_i = \sum_{i=1}^{n} t_i + \Phi_0 - \Phi_n$, and $\Phi_0 - \Phi_n$ should satisfy some condition to bound the actual cost reasonably (typically, $\Phi_0 - \Phi_n \leq 0$)
- prove the upper bound $\sum_{i=1}^{n} \hat{t}_i$ to be lower enough

3.1 Sample Solution

Define $h_i = log_2 n - log_2 t_i$ *s.t.* the time of i^{th} operation is $h_i O(1)$.
Define potential function

$$\Phi_i = \begin{cases} 0 & \text{if } i = 0 \\ \sum_{x=0}^{i} [\log_2 t_x - \log_2 i] & \text{if } 1 \leq i \leq n \end{cases}$$

In other words, $\Phi_i - \Phi_{i-1} = \log_2 t_i - \log_2 i \quad \text{for } i > 0$.
Let $\hat{h_i} = h_i + \Phi_i - \Phi_{i-1}$ be the amortized cost of i^{th} operations. The total cost:

$$\sum_{i=1}^{n} h_i = \sum_{i=1}^{n} \hat{h_i} + \Phi_0 - \Phi_n$$

By definition,

$$\begin{aligned}
&\Phi_0 - \Phi_n \\
&= -\sum_{i=0}^{n}(\log_2 t_i - \log_2 i) \\
&= -\left(\sum_{i=0}^{n}\log_2 t_i - \sum_{i=0}^{n}\log_2 i\right) \\
&= 0
\end{aligned}$$

And

$$\begin{aligned}
&\sum_{i=1}^{n}\hat{h_i} \\
&= \sum_{i=1}^{n}[h_i + \Phi_i - \Phi_{i-1}] \\
&= \sum_{i=1}^{n}\log_2 n - \log_2 i \\
&= \sum_{i=1}^{n}\log_2 \frac{n}{i} \\
&= O(n) \quad \text{Please refer slides } \textit{algo2016fall03} \text{ p.53}\sim\text{55 for elaboration of the last equation}
\end{aligned}$$

By using the potential method, the amount of the time of the n operations is

$$\left(\sum_{i=1}^{n}h_i\right)O(1) = O(n)$$

3.2 Another Sample Solution from Student's Answer

The total time of n operations:

$$\sum_{i=1}^{n}O(\log_2 n - \log_2 t_i) = \sum_{i=1}^{n}O(\log_2 n - \log_2 i)$$

Define Φ_i as the potential function:

$$i(\log_2 i - \log_2 n)$$

Define h_i as the actual cost, $\hat{h}_i$ as the amortized cost *s.t.* each operation cost $t_iO(1)$ time, and

$$\hat{h}_i = h_i + \Phi_i - \Phi_{i-1}$$

We have

$$\sum_{i=1}^{n}h_i = \sum_{i=1}^{n}\hat{h_i} + \Phi_0 - \Phi_n$$

By definition,

$$\begin{aligned}&\Phi_0 - \Phi_n\\&= 0 - 0\\&= 0\end{aligned}$$

And

$$\begin{aligned}&\sum_{i=1}^{n} \hat{h_i}\\&= \sum_{i=1}^{n} [h_i + \Phi_i - \Phi_{i-1}]\\&= \sum_{i=1}^{n} \log_2 n - \log_2 i + i(\log_2 i - \log_2 n) - (i-1)[\log_2(i-1) - \log_2 n]\\&= \sum_{i=1}^{n} \log_2 n - \log_2 i - \log_2 n + i\log_2 i - (i-1)\log_2(i-1)\\&= \sum_{i=1}^{n} \log_2 n - \log_2 i - \log_2 n + \log_2 i + (i-1)\log_2 i - (i-1)\log_2(i-1)\\&= \sum_{i=1}^{n} (i-1)\log_2 \frac{i}{i-1}\\&= \sum_{i=1}^{n} O(1)\end{aligned}$$

By using the potential method, the amount of the time of the n operations is

$$\left(\sum_{i=1}^{n} h_i\right) O(1) = O(n)$$

4 Problem 6

Let $h(x) = \max\{f(x), g(x)\}$
we choose $c_1 = 1$ and $c_2 = 1$ to satisfy the inequality:

$$c_1 h(x) \leq h(x) \leq c_2 h(x)$$

And note that $f(x)$ and $g(x)$ should be non-negative for x large enough.So,

$$c_1 h(x) \leq h(x) \leq f(x) + g(x) \leq 2h(x) \leq 2c_2 h(x) \quad \text{for x large enough}$$

For the inequality above, $f(x) + g(x) = \Theta(\max\{f(x), g(x)\})$ holds.

You get half credits if you ignore the assumption of $f(x)$ and $g(x)$ being non-negative for x large enough, and disprove the statement logically.