Data Structures and Algorithms (EE 220): Homework 2 Solutions

Contact TA for any Queries about the Solutions

Posted 02/14/2003

Problem 1: (5 pts) We show that

1. $\log(\log n) = O(\log^2 n)$
2. $\log^2 n = O(e^{\log n})$
3. $e^{\log n} = O(\log(n!))$
4. $\log(n!) = O((\log n)^{\log n}$

Part 1:

$$\begin{aligned}\lim_{n\to\infty} \frac{\log(\log n)}{\log^2 n} &= \lim_{n\to\infty} \frac{\frac{1}{\log n}\frac{1}{n}}{2\log n \frac{1}{n}} \quad \text{By L'Hospital} \\ &= \lim_{n\to\infty} \frac{1}{2\log^2 n} = 0.\end{aligned}$$

This shows that $\log(\log n) = O(\log^2 n)$.

Part 2:

$$\begin{aligned}\lim_{n\to\infty} \frac{\log^2 n}{e^{\log n}} &= \lim_{n\to\infty} \frac{\log^2 n}{n} \\ &= \lim_{n\to\infty} 2\log n \frac{1}{n} \quad \text{By L'Hospital} \\ &= \lim_{n\to\infty} \frac{2}{n} \quad \text{By L'Hospital} \\ &= 0.\end{aligned}$$

This shows that $\log^2 n = O(n)$.

Part 3: To prove this part we use the bound $n! \geq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \geq \left(\frac{n}{e}\right)^n$. Since log is a monotone function, this bound implies that $\log(n!) \geq n\log\left(\frac{n}{e}\right)$. Now,

$$\lim_{n\to\infty} \frac{n}{\log(n!)} \leq \lim_{n\to\infty} \frac{n}{n\log\left(\frac{n}{e}\right)} = \lim_{n\to\infty} \frac{1}{\log\left(\frac{n}{e}\right)} = 0.$$

This shows that $n = O(\log(n!))$.

Part 4: In this part we use yet another important inequality $n! \leq n^n$. This inequality implies that $\log(n!) \leq n \log n$.

Hence

$$\begin{aligned}
\lim_{n\to\infty} \frac{\log(n!)}{(\log n)^{\log n}} &\leq \lim_{n\to\infty} \frac{n \log n}{(\log n)^{\log n}} \\
&= \lim_{n\to\infty} \frac{n}{(\log n)^{\log n - 1}} \\
&\leq \lim_{n\to\infty} \frac{n}{e^{\log n - 1}} \qquad \forall n \geq e^3 \\
&= \lim_{n\to\infty} \frac{en}{n} = e < \infty.
\end{aligned}$$

This shows that $\log(n!) = O((\log n)^{\log n})$.

Problem 2: (5 pts)

Part 1: Proof by Counter Example.

Let $f(n) = n$ and $g(n) = n^2$. Hence $\min\{f(n), g(n)\} = f(n) = n$. Now observe that $f(n) + g(n) = \Theta(n^2) \neq \Theta(n)$. This shows that the claim is false in general.

Part 2(a): It is given that $f(n) = O(g(n))$. Hence from the definition of "O" notation it implies that $\exists\, c > 0$ and an index n_0 such that $\forall n \geq n_0$

$$0 \leq f(n) \leq cg(n).$$

Since log is a monotone function and $f(n) \geq 1$, the above equation implies that

$$0 \leq \log(f(n)) \leq \log c + \log(g(n)). \tag{1}$$

Note that $\log c + \log(g(n)) = O(\log(g(n)))$. This implies that $\exists c' > 0$ and n'_o such that $\forall n \geq n'_0$

$$0 \leq \log(g(n)) + \log c \leq c' \log(g(n)). \tag{2}$$

Now choose $\hat{n}_0 = \max\{n_0, n'_0\}$ and observe from (1) and (2) that

$$0 \leq \log(f(n)) \leq c' \log(g(n)) \quad \forall n \geq \hat{n}_0.$$

This shows that by definition $\log(f(n)) = O(\log(g(n))$.

Part 2(b): Proof by Counter Example.

Let $f(n) = 2^n$. Then, $f(\frac{n}{2}) = 2^{\frac{n}{2}}$. Now, observe that

$$\lim_{n\to\infty} \frac{f(n)}{f(\frac{n}{2})} = \lim_{n\to\infty} \frac{2^n}{2^{\frac{n}{2}}} = \lim_{n\to\infty} 2^{\frac{n}{2}} = \infty.$$

This shows that $2^n \neq \Theta(2^{\frac{n}{2}})$. Hence the claim is not true in general.

Problem 3: (5 pts) Consider the given recursion

$$\begin{aligned} T(n) &= T(n-a)+T(a)+n \quad \text{iteration } i=1 \\ &= T(n-2a)+2T(a)+2n-a \quad \text{iteration } i=2 \\ &= T(n-3a)+3T(a)+3n-3a \quad \text{iteration } i=3 \\ &= T(n-ia)+iT(a)+in-\sum_{j=1}^{i-1} ja \quad \text{for general iteration } i \end{aligned}$$

Verify the expression for general i using Induction. Observe that we need to iterate till $\frac{n}{i} > a$, i.e for $i < \frac{n}{a}$. Hence,

$$\begin{aligned} T(n) &\leq T(a)+\frac{n}{a}T(a)+\frac{n^2}{a}-a\sum_{j=1}^{\frac{n}{a}-1} j \\ &= \left(\frac{n}{a}+1\right)\Theta(1)+\frac{n}{2}+\frac{n^2}{2a} \\ &= O(n^2). \end{aligned}$$

Observe that since a is a fixed constant $T(a) = \Theta(1)$.

Problem 3: (10 pts) Since the arrays were sorted our task of obtaining maximum subsequence sum is much simplified. We only need to obtain $\max\{\sigma_1, \sigma_2, \sigma_3\}$, where σ_1 is the sum of all positive numbers of $LIST_1$, σ_2 is the sum of all the positive elements of $LIST_2$ and σ_3 is the sum of the positive elements of $LIST_1$ and all the elements of $LIST_2$. We can simply obtain these sums using following algorithm. Let $A[]$ be the combined array.

Initialize: $\sigma_1 = \sigma_2 = \sigma_3 = 0$
For $(i = 0$ to $m)$
    If $(A[m+n-i] > 0)$
        $\sigma_2 = \sigma_2 + A[m+n-i]$
    else
        $\sigma_3 = \sigma_3 + A[m+n-i]$
End For Loop
For $(i = 0$ to $n)$
    If $(A[n-i] > 0)$
        $\sigma_1 = \sigma_1 + A[n-i]$
    else
        EXIT For Loop
End For Loop
$\sigma_3 = \sigma_3 + \sigma_1 + \sigma_2$
Max_Subseq_Sum $= \max\{\sigma_1, \sigma_2, \sigma_3\}$.

Note that the complexity of algorithm is $O(m+\sqrt{n})$. In the case when m is much smaller than n, it becomes $O(\sqrt{n})$.

Caution: It is given that the number of positive elements in the Lists are $O(\sqrt{n})$ and $O(sqrtm)$, respectively. This does not mean that the lists have exactly $\sqrt{n}$ or

exactly $\sqrt{m}$ elements. The number of elements can be any function $f(n)$ such than $f(n) = O(\sqrt{n})$.