CprE 310 / Spring 2019

Recitation Solutions 12

- Here is a set of additional problems. They range from being very easy to very tough. The best way to learn the material in 310 is to solve problems on your own.
- Feel free to ask (and answer) questions about this problem set on Piazza.
- This is an **optional** problem set; do not turn this in for grading.
- While you don't have to turn this in, be warned that this material **can** appear in a quiz or exam.

---

1. Suppose that $F_0 = 0, F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$. Prove using induction that for every $n \geq 1$,

$$F_1^2 + F_2^2 + \ldots F_n^2 = F_n \cdot F_{n+1}.$$

Solution

Base case: For $n = 1$, $F_1^2 = 1$ and $F_1^2 \cdot F_2^2 = 1$.

Induction Hypothesis: For some k, $\sum_{i=1}^{k} F_I^2 = F_k F_{k+1}$.

Induction Step: Suppose the induction hypothesis is true. Then,

$$\begin{aligned}
\sum_{i=1}^{k+1} F_i^2 &= \sum_{i=1}^{k} F_i^2 + F_{k+1}^2 \\
&= F_k F_{k+1} + F_{k+1}^2 && (\because Induction\ Hypothesis) \\
&= F_{k+1}(F_k + F_{k+1}) && (\because Definition) \\
&= F_{k+1} F_{k+2}
\end{aligned}$$

2. Consider the following algorithm in pseudocode form:

function SQ(n):

$S \leftarrow 0, i \leftarrow 0$

while $i < n$

$S \leftarrow S + n$

$i \leftarrow i + 1$

return S

Two questions:

(i) Figure out what the algorithm is supposed to be doing, and using induction, prove that the algorithm is correct.

(ii) Analyze the algorithm's efficiency (in terms of running time), assuming that each add operation takes a unit amount.

Solution

The algorithm is calculating the square of the n.

Proof the correctness with Induction.

Base case $n = 0$. $0^2 = 0$.

Induction Hypothesis: Assume $SQ(k) = k^2$ for some k.

Induction Step: When $k \to k + 1$,

We adds extra 1 for each loop, and there are one more loop adding $k + 1$.

Therefore, the output of the algorithm is equal to $k^2 + 1 \times k + k + 1$ where k^2 comes from the induction hypothesis.

Since $k^2 + k + k + 1 = k^2 + 2k + 1 = (k + 1)^2$, the induction shows that the algorithm is correct.

Only considering the add operation, the total operation is equal to $2n$ from adding S and i.

3. Consider the following "algorithm" that prints out a bunch of things (it is unimportant what it prints out):

   for $(i = n; i \geq 1; i = i/2)$:

     for j in $[1, i]$:

       print xyz

   Assume that each print command takes 1 unit of time. Using big-Oh notation, estimate the running time of the algorithm as a function of n.

Solution

In the first for loop, there are $\lfloor \log_2 n \rfloor + 1$ cycles. And for the i^{th} cycle, the algorithm prints $\lfloor \frac{n}{2^{i-1}} \rfloor$ times.

The total counts of printing is equivalent to

$$\begin{aligned}
\sum_{i=0}^{\lfloor log_2 n \rfloor} n(\frac{1}{2})^i &= n \cdot \frac{1 - (1/2)^{\lfloor \log_2 n \rfloor + 1}}{1 - 1/2} \\
&\leq n \cdot \frac{1 - (1/2)^{(\log_2 n) + 1}}{1 - 1/2} \\
&= n \cdot 2 - 2(1/2)^{(\log_2 n) + 1} \\
&= n \cdot (2 - n^{-1}) \\
&= 2n - 1
\end{aligned}$$

Therefore, $\mathcal{O}(n)$.

The easier way to think about is that $\sum_{i=0}^{\infty}(1/2)^i = 2$ which is greater than any finite sum $\sum_{i=0}^{k}(1/2)^i$ for any k. Then we can put the inequality such that $\sum_{i=0}^{\lfloor log_2 n \rfloor} n(1/2)^i \leq \sum_{i=0}^{\infty} n(1/2)^i$.

4. Two algorithms (call them A and B) have different time complexities. The first algorithm exhibits time complexity $T_A(n) = 5n \log_{10} n$ microseconds, and the second exhibits time complexity $T_B(n) = 25n$ microseconds, for a problem of input size n. Which algorithm is better in the Big-Oh sense? For which problem sizes does it outperform the other?

   Solution

   In Big-Oh notation, $T_A(n)$ and $T_B(n)$ correspond to $\mathcal{O}(n \log_{10} n)$ and $\mathcal{O}(n)$ respectively. Therefore, in Big-Oh sense, $T_B(n)$ is better algorithm.

   But if we directly want to compare both algorithms with actual input size n, we can substract $T_B(n)$ with $T_A(n)$ and observe n's condition to make the substraction positive.

$$\begin{aligned} T_B(n) - T_A(n) &= 25n - 5n \log_{10} n \\ &= 5n(5 - \log_{10} n) \geq 0 \end{aligned}$$

   Therefore,

$$\begin{aligned} 5 - \log_{10} n &\geq 0 \\ 5 &\geq \log_{10} n \\ 10^5 &\geq n \end{aligned}$$

   So we can conclude that $T_A(n)$ is better than $T_B(n)$ if the input size is smaller than 10^5.