Analysis of Algorithms - Quiz I (Solutions)

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Problems

1. Show that $\log(n!) \in \Omega(n \cdot \log n)$.

   Solution: Without loss of generality, we assume that the logarithm base is 2.

   Observe that

$$\begin{aligned}
\log(n!) &= \log(1 \cdot 2 \cdot 3 \ldots n) \\
&= \log 1 + \log 2 + \log 3 + \ldots \log n \\
&= \sum_{i=1}^{n} \log i \\
&\geq \sum_{i=\frac{n}{2}+1}^{n} \log i \\
&\geq \sum_{i=\frac{n}{2}+1}^{n} \log \frac{n}{2} \\
&= \frac{n}{2} \log \frac{n}{2} \\
&= \frac{n}{2} \log n - \frac{n}{2} \\
&\geq \frac{1}{4} n \cdot \log n, \ \forall n \geq 4
\end{aligned}$$

   Thus $\log_2(n!) \in \Omega(n \cdot \log_2 n)$ with constant $c = \frac{1}{4}$ and $N_0 = 4$. □

2. Let T be a proper binary tree with height h having n nodes. Show that $\log_2(n+1) - 1 \leq h$, i.e., the height of any proper binary tree having n nodes is at least $\log_2(n+1) - 1$.

   Solution: In class, we argued that the maximum number of nodes at Level i of a proper binary tree is 2^i.

   It follows that the maximum number of nodes in a binary tree of height h is $\sum_{i=0}^{h} 2^i$, which is $2^{h+1} - 1$, as per the formula for the sum of a geometric progression. Thus, if T has n nodes, we have

$$\begin{aligned}
n &\leq 2^{h+1} - 1 \\
\Rightarrow 2^{h+1} &\geq n + 1 \\
\Rightarrow h + 1 &\geq \log_2(n+1) \\
\Rightarrow h &\geq \log_2(n+1) - 1
\end{aligned}$$

   □

3. Argue using mathematical induction that the solution to the recurrence

$$\begin{aligned} T(n) &= 1, \text{ if n} = 1 \\ &= T(n-1)+n, \text{ n} \geq 2 \end{aligned}$$

is $G(n) = \frac{n\cdot(n+1)}{2}$.

Solution: BASIS: At $n = 1$, $T(1) = 1$, by definition and $G(1) = \frac{1\cdot(2)}{2} = 1$. Thus $T(1) = G(1)$ and the basis is proven.

INDUCTIVE STEP: Assume that the conjecture is true for some $k \geq 1$, i.e., assume that $T(k) = G(k) = \frac{k\cdot(k+1)}{2}$. Now note that

$$\begin{aligned} T(k+1) &= T(k)+(k+1), \text{ by definition} \\ &= G(k)+(k+1), \text{ by the inductive hypothesis} \\ &= \frac{k\cdot(k+1)}{2}+(k+1) \\ &= (k+1)\cdot[\frac{k}{2}+1] \\ &= (k+1)\cdot[\frac{k+2}{2}] \\ &= \frac{(k+1)\cdot(k+2)}{2} \\ &= G(k+1) \end{aligned}$$

We have thus shown that $(T(k) = G(k)) \Rightarrow (T(k+1) = G(k+1))$ and thus by applying the first principle of mathematical induction, we can conclude that $T(n) = G(n), \forall n \geq 1$. □

4. Write a recursive algorithm for the *Post-order* Traversal of a binary tree. The input to your function is the root of the tree. Argue using induction that all the non-null nodes in the tree are visited by your algorithm.

Solution:

```
Function POST-ORDER(T)
1: if (T ≠ null) then
2:     POST-ORDER(T → lchild)
3:     POST-ORDER(T → rchild)
4:     print (T → key)
5: end if
```

Algorithm 1.1: Post-order traversal of a binary tree

Assume that **T** is the root of a null tree. In this case, Line (4) of Algorithm (1.1) is not reached, i.e., the node is not visited and thus the algorithm functions correctly.

Now assume that **T** has precisely one node. In this case, Line (2) and Line (3) are called on null trees and terminate without visiting their respective nodes, as per the discussion above. This is followed by Line (4) in which the root is visited.

Assume that Algorithm (1.1) works correctly when called on trees having height at most k.

Consider the case in which Algorithm (1.1) is called on a tree of height $(k+1)$. Note that Line (2) and Line (3) represent recursive calls on trees of height at most k. By the inductive hypothesis, all the nodes in the left subtree and the right subtree are visited. When Line (4) is executed, the root is also visited and thus all the nodes in the tree are visited.

From the second principle of mathematical induction, it follows that Algorithm (1.1) visits all the nodes of every tree. □

5. Assume that you are given a rudimentary programming language which contains only four operators, viz., $+$, $-$, abs and div. $+$ and $-$ have their usual meanings, while $div(a, b)$ returns the quotient of $\frac{a}{b}$ and $abs(a)$ returns the absolute value of a. Write a function $\max(a, b)$ that takes two integers a and b as input and returns the maximum of the two. Note that you can only use the operators provided; in particular, the constructs "**if**", "**while**", or "**for**" are not available.

   Solution: Let us study the function $f(a, b) = div(((a + b) + abs(b - a)), 2)$. We consider the following three cases:

   (i) $a > b$ - In this case $abs(b - a) = (a - b)$ and hence $f(a, b) = div(((a + b) + (a - b)), 2) = div(2a, 2) = a$.
   (ii) $b > a$ - In this case $abs(b - a) = (b - a)$ and hence $f(a, b) = div(((a + b) + (b - a)), 2) = div(2b, 2) = b$.
   (iii) $b = a$ - In this case $abs(b - a) = 0$ and hence $f(a, b) = div(((a + a)), 2) = div(2a, 2) = a$.

   We see that in all three cases $f(a, b) = \max(a, b)$, so we can indeed produce the maximum of two integers using only the operators provided! □