Analysis of Algorithms - Homework I (Solutions)

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

1 Problems

1. Write a recursive algorithm to check whether an integer x, exists in an array $\mathbf{A}$ of n integers.

   Solution: Algorithm 1.1 searches for an integer x in array $\mathbf{A}[n]$. It returns $\mathtt{true}$, if $x \in \mathbf{A}$ and **false** otherwise.

```
Function ARRAY-SEARCH(A, x, n)
 1: if (n = 1) then
 2:    if (A[1] = x) then
 3:       return(true)
 4:    else
 5:       return(false)
 6:    end if
 7: else
 8:    if (A[n] = x) then
 9:       return(true)
10:    else
11:       ARRAY-SEARCH(A, x, n − 1)
12:    end if
13: end if
```

Algorithm 1.1: Array Search

   □

2. Argue the correctness of your algorithm using induction.

   Solution: Let $P(n)$ denote the proposition that Algorithm 1.1 correctly searches for x in an array of size n.

   Base case ($n = 1$): Observe that when $n = 1$, only lines 1 through 5 of Algorithm 1.1 are executed. If $x \in \mathbf{A}$, then the conditional in the **if** statement of line 2 is satisfied and hence **true** is returned. Likewise, if $x \notin \mathbf{A}$, the conditional is falsified and hence line 5 of Algorithm 1.1 is executed, i.e., **false** is returned by Algorithm 1.1. We have thus established that when there is only one element in array $\mathbf{A}$, Algorithm 1.1 functions correctly.

   Inductive Hypothesis: Assume that $P(k)$ is **true**, i.e., assume that when Algorithm 1.1 is presented with an integer x and an array $\mathbf{A}$ of exactly k elements, then it returns **true** when $x \in \mathbf{A}$ and **false** otherwise.

   Now consider the case in which Algorithm 1.1 is presented with an array of size $k + 1$ and asked to search for the presence of an integer x. We consider the following two cases:

   (i) $x = A[k + 1]$ - In this case, line 9 of Algorithm 1.1 is executed. Since $x \in \mathbf{A}$, the algorithm functions correctly.

(ii) $x \neq A[k+1]$ - In this case, Algorithm 1.1 recurses over the first k elements of $\mathbf{A}$. From the inductive hypothesis, we know that Algorithm 1.1 functions correctly, when $\mathbf{A}$ has exactly k elements, i.e., if $x \in \mathbf{A}$, then **true** is returned and if $x \notin \mathbf{A}$, then **false** is returned.

We thus see that if Algorithm 1.1 functions correctly on arrays of size k, then it also functions correctly on arrays of size $k+1$; using the first principle of mathematical induction, we conclude that Algorithm 1.1 functions correctly for all $n \geq 1$.

□

3. Provide upper and lower bounds on $S = \sum_{i=1}^{n} i \cdot \log i$.

Solution: We use the integration bounds discussed in class. Observe that $i \cdot \log i$ is an increasing function of i. Hence, we must have,

$$S \leq \int_{i=1}^{n+1} x \log x \, dx$$

Observe that

$$\begin{aligned} \int x \log x \, dx &= \log x \int x - \int [\frac{d}{dx}(\log x) \cdot \int x \, dx] \, dx \\ &= \frac{x^2 \log x}{2} - \int \frac{x}{2} \, dx \\ &= \frac{x^2 \log x}{2} - \frac{x^2}{4} \end{aligned}$$

Thus, $S \leq [\frac{(n+1)^2 \log(n+1)}{2} - \frac{(n+1)^2}{4} - \frac{1}{4}$.

For the lower bound, observe that $S = 1 \log 1 + \sum_{i=2}^{n} i \cdot \log i$. Applying the lower bound from calculus, we conclude that

$$\begin{aligned} S &\geq \int_{1}^{n} x \cdot \log x \, dx \\ &= \frac{n^2 \log n}{2} - \frac{n^2}{4} - \frac{1}{4}. \end{aligned}$$

□

4. Let T denote a proper binary tree with n nodes having height h. Formally establish that $h \leq \frac{n-1}{2}$.

We use induction on the number of nodes n in the tree T.

Base case $n = 1$**:** In this case, the only node in T and therefore the height h of T is zero. Since $h \leq \frac{1-1}{2}$, the base case is proven.

Inductive hypothesis: Assume that whenever the number of nodes in a proper binary tree is at most k, the height of the tree is at most $\frac{k-1}{2}$. Now consider a proper binary tree having $(k+1)$ nodes; since T is proper, we know that:

(i) There are at least two leaves at level h.

(ii) We can group the leaves at level h into pairs, such that both leaves of a pair are children of the same node at level $h-1$.

Remove one such leaf pair, say (l_1, l_2), which are children of node l at level $(h-1)$. The resultant binary tree T' is still proper and has $k-1$ nodes; let h' denote the height of T'. As per the inductive hypothesis, we know that $h' \leq \frac{k-2}{2}$. There are precisely two possibilities to consider:

(i) $h' = h$ - In this case, $h = h' \leq \frac{k-2}{2} \leq \frac{k}{2}$.

(ii) $h' < h$ - In this case l_1 and l_2 were the only nodes at level h and hence $h' = h - 1$. As per the inductive hypothesis, $h' \leq \frac{k-2}{2}$ and hence, $h = h' + 1 \leq \frac{k-2}{2} + 1 = \frac{k}{2}$.

In either case, we have established that $h \leq \frac{k}{2}$ and using mathematical induction, we can conclude that the height h of a proper binary tree T having n nodes is at most $\frac{n-1}{2}$.

5. Consider the following recursive definition of $T(n)$.

$$\begin{aligned} T(1) &= 1 \\ T(n) &= n \cdot T(n-1),\ n \geq 2. \end{aligned}$$

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

Solution: Observe that $T(n)$ is in fact $n!$ and hence you are asked to show that $\log n! \in \Omega(n \cdot \log n)$.

Note that

$$\begin{aligned} \log n! &= \sum_{i=1}^{n} \log i \\ &= \sum_{i=2}^{n} \log i \\ &\geq \int_{1}^{n} \log x\, dx \\ &= [x \cdot \log x]_1^n - [x]_1^n \\ &= (n \cdot \log n - (n-1)] \\ &\geq n \cdot \log n - n \\ &\geq n \cdot \log n - \frac{n}{2} \log n \\ &= \frac{1}{2} n \cdot \log n \end{aligned}$$

We can then conclude that $\log T(n) \in \Omega(n \cdot \log n)$. $\square$