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 compute the maximum element in an array of integers. You may assume the existence of a function \( \max(a, b) \) that returns the maximum of two integers \( a \) and \( b \).

Solution: 

\begin{algorithm}
\begin{algorithmic}[1]
\Function{Find-Array-Max}{} \end{algorithmic}
\begin{algorithmic}[1]
\Function{Find-Array-Max}{A, n} \end{algorithmic}
\begin{algorithmic}[1]
\Function{Find-Array-Max}{A[1]} \end{algorithmic}
\begin{algorithmic}[1]
\Function{Find-Array-Max}{\max(A[n], \text{Find-Array-Max}(A, n-1))} \end{algorithmic}
\end{algorithm}

\textbf{Algorithm 1.1:} Finding the maximum in an array of \( n \) elements

2. Argue that your algorithm is correct.

\textbf{Solution:} We first need to formulate the proposition for algorithm correctness. In this case, we let \( P(n) \) stand for the proposition that Algorithm (1.1) finds and returns the maximum integer in the locations \( A[1] \) through \( A[n] \). Accordingly, we have to show that \( \forall n, P(n) \) is true.

\textbf{Basis:} When there is only one element in the array, i.e., \( n = 1 \), then this element is clearly the maximum element and it is returned on Line 2. We thus see that \( P(1) \) is true.

\textbf{Inductive Step:} Assume that Algorithm (1.1) finds and returns the maximum element, when there are exactly \( k \) elements in \( A \).

Now consider the case in which there are \( k+1 \) elements in \( A \). Since \( (k+1) > 1 \), Line 4 will be executed. In this step, we first make a recursive call to \text{Find-Array-Max} with exactly \( k \) elements. From the inductive hypothesis, we know that the maximum elements in \( A[1] \) through \( A[k] \) is returned. Now the maximum element in \( A \) is either \( A[k+1] \) or the maximum element in \( A[1] \) through \( A[k] \) (say \( r \)). Thus, returning the maximum of \( A[k+1] \) and \( r \) clearly gives the maximum element in \( A \), thereby proving that \( P(k) \rightarrow P(k+1) \). By applying the first principle of mathematical induction, we can conclude that \( \forall n, P(n) \) is true, i.e., Algorithm (1.1) is correct. 

3. What is the exact comparison complexity of your algorithm? Derive a recurrence relation and solve it to justify your answer.

\textbf{Solution:} Observe that the function \( \max(a, b) \) uses exactly one comparison. Thus, the comparison complexity of Algorithm (1.1) can be described the recurrence relation:

\[
\begin{align*}
T(1) &= 0 \\
T(n) &= T(n-1) + 1, \quad n > 1
\end{align*}
\]
This recurrence can be expanded as \( T(n) = 1 + 1 + \ldots + 1 \) \((n-1)\) times to give \( T(n) = n - 1 \). □

4. Argue using induction that the exact solution to the recurrence relation:

\[
T(1) = 0 \\
T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n, \quad n \geq 2
\]

is \( T(n) = n \cdot \log n \).

**Solution:**

**Basis:** At \( n = 1 \), both the closed form and the recurrence relation agree \((0 = 0)\) and so the basis is true.

**Inductive Step:** Assume that \( T(r) = r \cdot \log r \) for all \( 1 \leq r \leq k \). Now consider \( T(k + 1) \). As per the recurrence relation, we have,

\[
T(k + 1) = 2 \cdot T\left(\frac{k + 1}{2}\right) + (k + 1), \quad \text{since} \ (k + 1) \geq 2
\]

\[
= 2 \cdot \left(\frac{k + 1}{2}\right) \cdot \log \left(\frac{k + 1}{2}\right) + (k + 1) \quad \text{as per the inductive hypothesis, since} \ \frac{k + 1}{2} < k
\]

\[
= (k + 1) \cdot (\log(k + 1) - \log 2) + (k + 1)
\]

\[
= (k + 1) \cdot \log(k + 1) - (k + 1) + (k + 1)
\]

\[
= (k + 1) \cdot \log(k + 1)
\]

We can therefore apply the second principle of mathematical induction to conclude that the exact solution to the given recurrence is \( n \cdot \log n \). □

5. Show that \( \log(n!) \in O(n \cdot \log n) \).

**Solution:** Observe that

\[
\log 1 \leq \log 2 \leq \log 3 \leq \ldots \leq \log n
\]

Adding up both the LHS and the RHS, we get

\[
\sum_{i=1}^{n} \log i \leq n \cdot \log n
\]

\[
\Rightarrow \log(1 \cdot 2 \cdot \ldots \cdot n) \leq n \cdot \log n
\]

\[
\Rightarrow \log(n!) \leq n \cdot \log n
\]

Yet another way of proving the above identity is observing that \( \log n \) is an increasing function of \( n \) and hence we can use the bound

\[
\log(n!) = \log 1 + \log 2 + \ldots + \log n
\]

\[
= \sum_{i=1}^{n} \log i
\]

\[
\leq \int_{1}^{n+1} \log x \, dx
\]

From elementary calculus, we know that \( \int \log x = x \log x - x \), from which it follows that \( \log(n!) \leq (n + 1) \cdot \log(n + 1) - n \) and hence \( \log(n!) \in O(n \cdot \log n) \). □