Analysis of Algorithms - Quiz I (Solutions)

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

1 Problems

1. **Recurrences:**

   (a) Solve the following recurrence. You may assume any convenient form for n.

   $$\begin{aligned} T(1) &= 0 \\ T(n) &= T(\frac{n}{2}) + 1,\ n > 1 \end{aligned}$$

   (b) Consider the following recurrence relation:

   $$\begin{aligned} T(1) &= 4 \\ T(n) &= T(n-1) + 4 \end{aligned}$$

   Argue using mathematical induction that $T(n) = 4 \cdot n$. Note that you *must* use induction to establish the solution.

   Solution:

   (a) Assume that $n = 2^k$, so that $k = \log n$. Accordingly, the original recurrence can be rewritten as:

   $$\begin{aligned} T(2^0) &= 0 \\ T(2^k) &= T(2^{k-1}) + 1,\ k > 1 \end{aligned}$$

   Now observe that,

   $$\begin{aligned} T(2^k) &= T(2^{k-1}) + 1 \\ &= T(2^{k-2}) + 1 + 1 \\ &= T(2^{k-3}) + 1 + 1 + 1 \\ \vdots &= \vdots \\ &= T(2^{k-(k-1)}) + (k-1) \\ &= T(2^{k-k}) + k \end{aligned}$$

   Finally, we note that $T(2^{k-k}) = T(2^0) = 0$ and hence, $T(2^k) = k$, from which it follows that $T(n) = \log n$.

   (b) **Base case:** When $n = 1$, as per the definition, $T(n) = 4$, while as per the conjecture $T(n) = 4 \cdot 1 = 4$. Since definition and conjecture coincide, the base case is proven.
   Assume that $T(k) = 4 \cdot k$, for some $k > 1$.

Let us focus on $T(k+1)$; as per the definition,

$$\begin{aligned} T(k+1) &= T(k) + 4 \\ &= 4 \cdot k + 4 \\ &= 4 \cdot (k+1) \\ &= \text{conjecture} \end{aligned}$$

Since assuming that conjecture and definition coincide at $n = k$ implies that conjecture and definition coincide at $n = k+1$, we apply the first principle of mathematical induction to conclude that $T(n) = 4 \cdot n, \forall n \geq 1$.

□

2. **Asymptotics:**

   (a) Show that if $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$.

   (b) Show that for any function $f(n)$, the set $\omega(f(n)) \cap o(f(n))$ is empty.

   Solution:

   (a) Since, $f(n) \in O(g(n))$, it follows that there exist constants c_1 and N_1, such that for all $n \geq N_1$, $f(n) \leq c_1 \cdot g(n)$. Likewise, there exist constants c_2 and N_2, such that for all $n \geq N_2$, $g(n) \leq c_2 \cdot h(n)$.
   Set $N_3 = \max(N_1, N_2)$ and $c_3 = c_1 \cdot c_2$. It is clearly the case, that for all $n \geq N_3$,
   $f(n) \leq c_1 \cdot g(n)$ and $g(n) \leq c_2 \cdot h(n)$. It thus follows that for all $n \geq N_3$, $f(n) \leq c_1 \cdot c_2 \cdot h(n) = c_3 \cdot h(n)$, i.e., $f(n) \in O(h(n))$.

   (b) Let $h(n)$ denote a function that belongs to the set $\omega(f(n)) \cap o(f(n))$. Since $h(n) \in \omega(f(n))$, we know that $\lim_{n\to\infty} \frac{f(n)}{h(n)} = 0$. Likewise, since $h(n) \in o(f(n))$, we have $\lim_{n\to\infty} \frac{h(n)}{f(n)} = 0$. Thus, $h(n)$ simultaneously grows faster and slower than $f(n)$, which is a contradiction, proving that $h(n)$ cannot exist.

□

3. **Greedy:** In class, we established the correctness of Kruskal's algorithm by assuming that all edges had distinct weights. Can you modify the proof to prove that if the edge weights are distinct, the Minimum Spanning Tree of the input graph is *unique*?

   Solution: Let T denote the Minimum Spanning Tree (MST) produced by Kruskal's algorithm and let T' denote an MST, which is distinct from T, such that $w(T) = w(T')$, where $w(T)$ and $w(T')$ denote the respective weights of T and T' respectively.

   Since the two trees are distinct and both contain exactly $(n-1)$ edges, where n denotes the number of vertices in the input graph, it must be the case that there exists at least one edge in T, which is not in T'; pick the lightest such edge, say e.

   Insert e into T'; as per the discussion in class, we know that a cycle C is created in T'. Further all the edges in the cycle C cannot belong to T, since in this case, e cannot be part of T either. Pick any edge $e' \in C$, such that $e' \notin T$.

   Let S denote the set of edges that are lighter than e and are part of T. As per the above discussion, the edges in S are also part of T' (since e is the lightest edge in T that is not part of T'). Since $S \cup e' \subseteq T'$, $S \cup e'$ does not contain a cycle.

   We now argue that e' is heavier than e. First note that e' and e cannot have the same weight, since all edge weights are distinct. Then note that if e' is indeed lighter than e, it would have been considered before e, by Kruskal's algorithm and made part of T, since it does not create a cycle with S! Inasmuch as e' does not create a cycle with S and e' is not part of T, it follows that e' is heavier than e. Thus, removing e' from T' maintains the spanning structure of T',while lowering its weight. The fact that we could lower the weight of T' immediately implies that T' is not an MST, i.e., the MST of the input graph must be unique.

   □

4. **Binary Trees:** Let T denote a proper binary tree of height h having n nodes. Formally establish that the number of internal nodes in T is at least h and at most $2^h - 1$.

   Solution:

   Observe that there is at least one internal node at each of the levels 0 through $(h-1)$; it follows that the tree has at least h internal nodes. (Curiously enough, a similar argument does not work as far as a lower bound on the number of external nodes is concerned. In this case, mathematical induction is necessary. As an exercise, I strongly urge you to prove the lower bound on the number of internal nodes using induction.)

   The internal nodes of T are maximized, when all the nodes in the levels 0 through $(h-1)$ are internal. Since the maximum number of nodes at level i is 2^i, the maximum number of internal nodes in T cannot exceed $\sum_{i=0}^{h-1} 2^i = 2^h - 1$.

   □

5. **Sorting and Selection:** Given and unordered array $\mathbf{A}$ of n elements, describe how you would find both the maximum and minimum elements of $\mathbf{A}$ using at most $\frac{3}{2}n - 2$ element to element comparisons. You may assume that n is an even number.

   Solution: Let $n = 2 \cdot m$; we divide the elements in $\mathbf{A}$ into m pairs (a_i, b_i), $i = 1, 2, \ldots, m$.

   A single comparison determines the maximum and minimum element of each pair. Construct the set U using the maximum elements of each pair and the set L using the minimum element of each pair; thus, U and L are built using a total of m comparisons.

   The critical observation is that the maximum element of U is the maximum element of $\mathbf{A}$ and the minimum element of L is also the minimum element of $\mathbf{A}$.

   The maximum element of U can be determined using $(m-1)$ comparisons and likewise the minimum element of L can be determined using $(m-1)$ comparisons.

   It follows that the maximum and minimum elements of $\mathbf{A}$ can be determined in $m + (m-1) + (m-1) = \frac{3}{2}n - 2$ element to element comparisons.

   □