Exam Statistics

130 students took the exam; there were 3 no-shows recorded as 0. The range of scores was 0–90, with a mean of 44.76 (this includes the no-shows), a median of 48, and a standard deviation of 18.05. Very roughly speaking, if I had to assign final grades on the basis of this exam only, above 60 would be an A (25), 46–59 a B (46), 30–45 a C (34), 20–29 a D (16), below 20 an E (12). Every student should have been able to get full credit, or nearly so, on the first, fourth and last problems, plus a few points on the other two problems; that is, no score should have been much below 60.

Problem Solutions

1. (a) $T(n)$ grows linearly because the annihilator is $(E - 1)^2$ so $T(2n) = 2T(n)$; hence the running time doubles.
   (b) $T(n)$ grows cubically because the annihilator is $(E - 1)^4$ so $T(2n) = 8T(n)$; hence the running time is multiplied by 8.
   (c) $T(n)$ grows like the Fibonacci numbers; $F_n = \Theta(\phi^n)$ where $\phi = (1 + \sqrt{5})/2 \approx 1.61801$ so $T(2n) = \Theta(\phi^{2n}) = \Theta(T(n)^2)$. Hence the running time is roughly squared.
   (d) The recurrence tells us that $T(2n) = T(n) + 1/(2n)$ so the running time is only negligibly affected.

2. Let the elements of list $S$ (ordered by starting value) be $[a_i, b_i]$ and let the elements of list $E$ (ordered by ending value) be $[\hat{a}_i, \hat{b}_i]$. We merge $S$ and $E$ looking at the starting values $a_i$ of $S$ and the ending values $\hat{b}_i$ of $E$ into a list of $2n$ values—think of the $a_i$ as opening parentheses and the $\hat{b}_i$ as closing parentheses. We start a counter at 0 and, as the merge is being done, we increment the counter when an $a_i$ is put into the merged output and decrease the counter when a $\hat{b}_i$ is put into the merged output. Of course the counter is 0 at the end (why?). The maximum value attained by the counter over the course of the merge is the maximum nesting depth of the intervals.

3. (a) If $n$ is even, then because the elements are in decreasing order the element 1 is in an even position; the restricted swaps mean that it can only be moved to other even positions, never to the first position as needed to sort the elements.
(b) Suppose \( n = 2k + 1 \) so that \( k = (n - 1)/2 \). Use insertion sort separately on the \( k + 1 \) odd positions 1, 3, \ldots, 2k + 1 and then on the \( k \) even positions 2, 4, \ldots, 2k; the resulting array is now fully sorted.

(c) In the worst case every comparison results in a swap. As we saw in the notes for Lecture 3 (January 18), the worst case for insertion sort to sort the \( k + 1 \) odd-indexed values takes \( (k + 1)(k)/2 \) comparisons; the worst case to sort the \( k \) even-indexed values takes \( (k)(k - 1)/2 \) comparisons. In total then \( k^2 = [(n - 1)/2]^2 = (n - 1)^2/4 \) comparisons (and hence swaps) are used in the worst case.

(d) Following the hint, we count the number of inversions in the given input. The \( k + 1 \) odd-indexed elements are in reverse order, and hence have \( 0 + 1 + 2 + \cdots + k = (k)(k+1)/2 = k(k+1)/2 \) inversions. Similarly, the \( k \) even-indexed elements have \( 0 + 1 + 2 + \cdots + (k-1) = k(k-1)/2 \) inversions. There are thus \( k(k+1)/2 + k(k-1)/2 = k^2 \) inversions, each of which must be undone by a swap; hence \( k^2 = (n - 1)^2/4 \) swaps are needed in the worst case.

4. For \( n = 1 \) the tree has a single leaf at depth 0, so \( \sum_{i=1}^{n} 2^{-l_i} = 2^0 = 1 \). Now suppose the inequality is true for all tree with fewer than \( n \) leaves; that means that in a tree of \( n \) leaves both the left and right subtrees satisfy the inequality so

\[
\sum_{\text{leaves } l \text{ in left subtree}} 2^{-(\text{depth}(l)-1)} \leq 1
\]

and

\[
\sum_{\text{leaves } l \text{ in right subtree}} 2^{-(\text{depth}(l)-1)} \leq 1
\]

because the leaves are one level shallower with respect to the roots of the subtrees. Adding these two inequalities and dividing by 2 give the desired result,

\[
\sum_{\text{leaves } l \text{ in tree}} 2^{-\text{depth}(l)} \leq 1
\]

5. We know from Lemma 1 of Lecture 6 (January 30) that the binary tree \( T \) with the least external path length has all of its leaves on levels \( l \) and \( l + 1 \) for some value of \( l \). So coloring all internal nodes at level \( l \) red and all other internal nodes black gives every leaf a black depth of \( l \). There are clearly no two parent/child red nodes and the root is black. Hence this is a proper red-black coloring of the tree.