Solutions for the First Midterm Exam Makeup Assignment

Problem 1. (8 pts. FindMax Recurrence) The algorithm makes to recursive calls, each on a list exactly half of the original list. Finally, the algorithm performs a single comparison between the winners of the two recursive calls. Assuming that \( n = 2^k \)

\[
T(2^k) = 2T(2^{k-1}) + 1 \\
= 2(2T(2^{k-2}) + 1) + 1 \\
= 4T(2^{k-2}) + 3 \\
= 8T(2^{k-3}) + 7 \\
= 2^iT(2^{k-i}) + 2^k - 1 \\
= 2^kT(1) + 2^k - 1 \\
= 2n - 1 \\
= O(n)
\]

Problem 2. (6 pts. Sequence) Admission of instructor error: the 8 in the sample list was supposed to be a 4. This yields a sequence defined by

\[
\sqrt{2^{2^{i-1}}}
\]

In this case the next number in the sequence would be \( 2^{16} \)

Problem 3. (6 pts. Big Oh definitions) Hopefully, you recognized that \( O() \) designates an upper bound; the algorithm will never take more time than \( f(n) \) times some constant. \( \Omega \) represents a lower bound; the algorithm will always use more time than \( f(n) \) times some constant. Both of these bounds allow for equality; they are not strict inequalities. That leaves \( \Theta \), which represents the intersection of the above two bounds; the algorithm is both \( O(f(n)) \) AND \( \Omega(f(n)) \).

Problem 4. (8 pts. Running times)

1. With the faster computer one can now solve a problem instance of size \( n = 2^{14} = 16k \).

2. With the faster computer one can now solve a problem instance that is essentially the same size. Or more precisely \( \sqrt{2^{16} + 12} = 256.023 \)