CMPS 201: Summer 2003
Homework Assignment 3

1. (20 Points) Prove the correctness of Quicksort(A, p, r) by induction on the length \( m = r - p + 1 \) of the subarray \( A[p \ldots r] \).

2. (20 Points) Problem 7-3 on p.161. (In the first edition this is problem 8-3 on p.169.)
Professors Howard, Fine, and Howard have proposed the following “elegant” sorting algorithm:

\[
\text{Stooge-Sort}(A, i, j) \\
2. \quad A[i] \leftrightarrow A[j] \\
3. \text{if } i + 1 \geq j \\
4. \quad \text{return} \\
5. \quad k \leftarrow \lfloor (j - i + 1)/3 \rfloor \\
6. \quad \text{Stooge-Sort}(A, i, j - k) \\
7. \quad \text{Stooge-Sort}(A, i + k, j) \\
8. \quad \text{Stooge-Sort}(A, i, j - k)
\]

a. (10 Points) Argue that, if \( n = \text{length}[A] \), then Stooge-Sort(A, 1, length[A]) correctly sorts the input array \( A[1 \ldots n] \)

b. (5 Points) Give a recurrence for the worst-case running time of Stooge-Sort and a tight asymptotic (Θ-notation) bound on the worst-case running time.

c. (5 Points) Compare the worst-case running time of Stooge-Sort to that of insertion sort, merge sort, heapsort, and quicksort. Do the professors deserve tenure?

3. (30 Points) Design a variation of MergeSort which, instead of recurring until the subarray has length 0 or 1, recurs at most a constant \( k \) times, then calls InsertionSort (see section 2.1) on the resulting \( 2^k \) subarrays of length \( n/2^k \). Call this algorithm Depth-\( k \)-MergeSort.

For Example let \( q = \left\lfloor \frac{1+n}{2} \right\rfloor \), \( u = \left\lfloor \frac{1+q}{2} \right\rfloor \), and \( v = \left\lfloor \frac{(q+1)+n}{2} \right\rfloor \). We can picture the operation of Depth-2-MergeSort by the following recursion tree.

```
A[1...n]
  /    \
A[1...q]  A[(q+1)...n]
 /     /     \
A[1...u] A[(u+1)...q] A[(q+1)...v] A[(v+1)...n]
```

Depth
0

1

2
The nodes at depth $i$ represent subarrays of length $n/2^i$. Depth-2-MergeSort calls InsertionSort on the 4 subarrays at Depth 2. Depth-1-MergeSort recurs down to depth 1 and calls InsertionSort on the 2 subarrays at Depth 1. Depth-0-MergeSort simply calls InsertionSort on the full array.

a. (10 Points) Write pseudo-code for this algorithm. (Hint: Notice that each call to Depth-$k$-MergeSort must know its own level in the recursion tree in order to know whether it should recur again, or call InsertionSort.)

b. (10 Points) Determine the asymptotic run time of Depth-$k$-MergeSort as a function of both $k$ and $n$. In order to simplify the analysis, you may assume that $n$ is always an exact power of 2. (Hint: First review the discussion of InsertionSort in the text, and note that InsertionSort runs in time $\Theta(n^2)$.)

c. (10 Points) Determine $k$ as a function of $n$ such that nothing is gained in terms of asymptotic run time by recurring more than $k$ times. (Hint: You will see from your answer to part (b) that if $k$ is a fixed constant, then Depth-$k$-MergeSort runs in time $\Theta(n^2)$). Observe that $k = \lg(n)$ gives ordinary MergeSort, which recurs down the level at which all arrays are of length 1, and runs in time $\Theta(n\lg(n))$. You are to determine a function $k = g(n)$ (which is less than $\lg(n)$ in absolute terms) such that Depth-$k$-MergeSort also runs in time $\Theta(n\lg(n))$.

4. (20 Points) Design an algorithm called Extrema($A, p, r$) on the divide and conquer paradigm which finds and returns the maximum and minimum values in $A[p \ldots r]$. Your algorithm should perform exactly $\lceil 3n/2 \rceil - 2$ comparisons, on an input array of length $n$.

a. (10 Points) Prove the correctness of your algorithm.

b. (10 Points) Write a recurrence for the number of comparisons performed on arrays of length $n$, and solve it exactly.