Algorithms
Computer Science 140 & Mathematics 168
Spring 2009
Homework 2a
Due Thursday, January 29

Please use LaTeX to typeset your solutions to this assignment.

1. **[20 Points] Curly, Mo, and Larry's Totally Excellent (?) Sorting Algorithm!** Professors Curly, Mo, and Larry of the Pasadena Institute of Technology have proposed the following sorting algorithm: First sort the first two-thirds of the elements in the array. Next sort the last two thirds of the array. Finally, sort the first two thirds again. (Notice that this algorithm is similar to Mergesort except that it uses three recursive calls rather than two and there is no merging step! As a consequence, this algorithm is very easy to implement!) The code is given below. Notice that the floor function, $\lfloor x \rfloor$, simply rounds down to the nearest integer. This is just used to compute the appropriate two-thirds and round to an integer so that we don't use non-integer indices into our array!

```
Stooge-sort (A,i,j)
begin
    if A[i] > A[j] then
        swap A[i] and A[j]
    if i + 1 ≥ j then
        return
    k = ⌊(j − i + 1)/3⌋.
    Stooge-sort(A, i, j − k)      Comment: Sort first two-thirds.
    Stooge-sort(A, i + k, j)      Comment: Sort last two-thirds.
    Stooge-sort(A, i, j − k)      Comment: Sort first two-thirds again!
end
```

   (a) Give an informal but convincing explanation (not a rigorous proof by induction) of why the approach of sorting the first two-thirds of the array, then sorting the last two-thirds of the array, and then sorting again the first two-thirds of the array yields a sorted array. A few well-chosen sentences should suffice here.

   (b) Find a recurrence relation for the worst-case running time of Stooge-sort. To simplify your recurrence relation, you may assume each

of the recursive calls is on a portion of the array that is *exactly* two-thirds the length of the original array.

(c) Next, solve the recurrence relation using the work tree method. **Show all of your work.** Do not use the "Master Theorem" in the book. In your analysis, it will be convenient to choose n to be c^k for some fixed constant c. (For example, we used $c = 2$ when analyzing Mergesort. Here you will want to use a different value of c. The value of c that you choose might not even be an integer! This may seem a bit strange, but it allows us to significantly simplify the analysis!)

(d) We made several simplifying assumptions: The array could be divided perfectly into thirds and that the length of the array was a power of c which wasn't even an integer. Explain as clearly and convincingly as you can why the asymptotic running time of the algorithm remains the same even when these simplifying assumptions do not hold.

(e) How does the worst-case running time of Stooge-Sort compare with the worst-case running times of the other sorting algorithms that we've seen so far?