CS 466/666 Spring 2014
Assignment 1
Due Noon, Tuesday May 20, 2014

Acknowledge your sources; do not copy

1. [6 marks] Extending the Master Theorem: Suppose we have a simple recursive algorithm that to solve a problem of size n we recursively solve two problems of size n/2 and then do a sort (in n lg n steps). A problem of size 1 takes 0 steps. Give a recurrence to describe the runtime and show this method takes $O(n \lg^2 n)$ time.

2. [10 marks] Refining the median algorithm of Blum et al: In class we discussed the simplest version of the Blum et al linear time algorithm to find the k^{th} largest of n elements in $\Theta(n)$ time in the worst case. For simplicity we assume all elements are distinct.

   Select(k,n)

   If n is "small"; sort and take the k^{th}

   Else {

   Sort elements in n/c "columns" of length c (c is an odd constant, it doesn't matter if one column has fewer elements))

   Recursively find median of column medians

   Determine where median of median fits in each column (so we have rank of median. Those falling on "wrong" side of median cannot be k^{th} so discard them)

   Do appropriate recursive selection on the others

   }

   Our analysis took the smallest value of c that works, namely 5. So let T(n) be the number of comparisons needed, for the worst value of k:

   Sort elements: n/5 * 7 (as 7 comparisons are enough to sort 5 elements)

   Recursively …: T(n/5)

   Determine …: 2n/5 (2 elements in each column are above/below the column median)

   Do appropriate …: T(n(c+(c-1)/2)/2c) = T(7n/10)

   So $T(n) \leq 9n/5 + T(n/5) + T(7n/10)$; by induction we want to prove $T(n) = \alpha n$ for some constant α.

   Then $T(n) \leq 9n/5 + 9\alpha n/10$; so setting $\alpha = 18$ and checking base cases, it all works (Clearly there was an arithmetic error in doing this in class).

   The goal of this assignment is to reduce the constant (18 above) by making some simple improvements to the algorithm. Clearly a different choice of c is one option; there are others.

Give suitable modifications/specifications to the algorithm above and show your version uses fewer comparisons in the worst case. Full marks will be given for a method using fewer than 8n comparisons and its proof.

Hint: For simplicity, you can assume that there are no repeated values in the input, and it also may be handy to add some dummy $\pm\infty$ values to simplify your algorithm and its analysis.

The best sorting algorithms for c elements take the following number of comparisons. (You should be able to determine which values of c are appropriate for reasons other than the values in the table … the table is this big so I don't give the "good choices" away".

c	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
sort	7	10	13	16	19	22	26	30	34	38	42	46	50	54	58	62	66	71	76