Spring 2013 **CMSC 351: Homework 6** Clyde Kruskal

Due at the start of class Thursday, March 28, 2013.

Problem 1. Consider an array of size eight with the numbers $70, 30, 20, 80, 50, 10, 60, 40$. Assume you execute quicksort using the version of partition from CLRS (and class). Note that in this algorithm an element might exchange with itself (which counts as one exchange).

(a) Show the array after the first partition. How many comparisons and exchanges are used?

(b) Show the left side after the next partition. How many comparisons are used? How many exchanges?

(c) Show the right side after the next partition on that side. How many comparisons are used? How many exchanges?

Problem 2. Assume you execute quicksort. Assume it turns out that at the odd levels of recursive calls you get medium lucky and partition on the $n/4$th smallest element. At the even levels you get very lucky and partition on the median element.

For example, let's say $n = 4$. The first pivot will be at index 2, leaving arrays of sizes 1 and 2. It will take three comparisons. The size 1 array will take no more comparisons. In the size 2 array you will pivot on either element, which is as close to the median as we can get, giving one more comparison. Overall, this will take $3 + 1 = 4$ comparisons.

Analyze how many comparisons this lucky version of quicksort does. Just get the exact value for the high order term. You do not have to worry about floors and ceilings and you can make reasonable simplifying assumptions.

Problem 3. Assume you execute quicksort. Assume it turns out that at every other level of recursive calls you get lucky and partition on the median element. At the other levels you partition on a random element. So odd levels of recursion look random and even levels are lucky. Analyze how many comparisons this lucky version of quicksort does on average.

For example, let's say $n = 4$. The first pivot will be random: Indices 1 and 4 are symmetric and 2 and 3 are symmetric. In either case it will take three comparisons. If it is 1 or 4 you will end up with arrays of sizes 0 and 3. In the array of size 3 you will always pivot on the median element (because it is an even level of recursion), for two more comparisons. If it is 2 or 3, you will end up with arrays of sizes 1 and 2. In the size 2 array you will pivot on either element, which is as close to the median as we can get, giving one more comparison. This will all average out to $3 + (1/2)(2 + 1) = 9/2$ comparisons.

Just get the exact value for the high order term. You do not have to worry about floors and ceilings and can make reasonable simplifying assumptions. (It turns out to not be much harder to keep track of the floors and ceilings, and to take into account that the median is slightly different depending on whether the list size is even or odd.)

You must use the method of writing a recurrence as we did in class. Show your work.

HINT: When you write your recurrence, combine the work of pairs of even/odd levels of the recursive calls.

Problem 4.

(a) Give pseudo code for finding the minimum and maximum element in a list of size n, where n is even. Minimize the number of comparisons.

(b) Draw the decision tree for this method applied to the four elements A, B, C, and D.

Problem 5. (Challenge Problem) Solve Problem 3 using the method of analysis from the book.