CS&E A311, Spring 2020
Assignment 2
58 points
Due 7 Feb 2020 at 11:59pm?, 1 document submitted in person, 1 file to *transformer*

Since most of this assignment lends itself to paper and pen, please submit hardcopies of this assignment to me. If you are not able to attend class, you may submit to Blackboard prior to 10am on the due date.

1. (5 pts) Illustrate (similar to what we did in class) the operation of merge sort on the array A = [3, 41, 52, 26, 38, 57, 9, 49].

2. (5 pts) Consider simple linear search, where the input is a sequence of n numbers in an array A and a value v. The output is the index of the array where v matches A[index], and NULL if v does not appear in A, where the index is found by looping through the elements of A. How many elements of the input sequence need to be checked on the average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst case? What are the average-case and worst-case running times of linear search in Theta notation? Justify your answers.

3a. (5 pts) Referring back to the linear searching problem, observe that if the sequence A is sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. Binary search is an algorithm that repeats this procedure, halving the size of the remaining portion of the sequence each time. Write pseudocode for a recursive binary search. Write the recurrence relation for this algorithm. (Hint: see **RecursiveFunctionsReview.pdf** attached to the Blackboard assignment for a refresher on recursive functions.)

3b. (5 pts) Argue that the worst-case running time of binary search is $\Theta(\lg n)$ using the recurrence relation above.

4. (5 pts) Write a program in C++ (filename **binarySearch.cpp**) that implements your binary search algorithm with an array of sample numbers. Don't just copy this from online, use the pseudo-code your wrote above to implement it (your variable names and structure of your function should match the pseudo code you came up with). Include several print statements that demonstrate the algorithm works when the number sought is present and absent. Be sure your file compiles on the *transformer* server, and place it in a subdirectory of your home directory named exactly: **csce311/a2/**.

5. (6 pts) For each of the following, be sure to show your answer is correct using the notation provided by the in-class definitions.

a) What is big-O for $f(n) = 3n^2 + 5n - 3$?
b) What is Θ for $f(n) = \log(n/5)$?
c) What is Ω for $f(n) = 2^{\lg(n)}$?

6. (5 pts) You work for a factory making bricks. A coworker accidentally contaminated one of the bricks with a lightweight material and it must be removed. Given a pile of 50 bricks and a balance scale, how can you find the one that weighs less? The scale can hold any number of bricks on each side of the scale at one time, and it will tell you if the two sides weigh the same, or which side is lighter if they do not weigh the same. Describe the algorithm to find the contaminated, lightweight brick. How many weighings will you do? Find an algorithm that uses the least number of weighings possible. State and justify your answer as a function of *n*, the total number of bricks**.**

7. (8 pts) Although merge sort runs in O(*n*lg*n*) time, and insertion sort runs in O(n^2) time, the constant factors in insertion sort make it faster for small n. Thus, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined. (Hint: it is helpful to diagram this by choosing reasonable values for n and k.) In this problem, we will analyze the runtime of this hybrid algorithm to determine an appropriate size for k.

a) Show that the n/k sublists, each of length k, can be sorted by insertion sort in O(nk) worst-case time.
b) Show that the sublists can be merged in O(n lg(n/k)) worst-case time.
c) From a) and b), we can see that the modified algorithm runs in O(nk + nlg(n/k)) worst case time. What is the largest asymptotic (O notation) value of k as a function of n for which the modified algorithm has the same asymptotic running time as standard merge sort?
d) How should k be chosen in practice?

8. (5 pts) Determine a tight upper bound for T(n) = 3T(n/2)+n using the iterative substitution method. Assume T(1) = Θ(1).

9. (9 pts) Use the master method to give tight asymptotic bounds for the following recurrences.

a) $T(n) = 4T(\frac{n}{2}) + n$

b) $T(n) = 4T(\frac{n}{2}) + n^2$

c) $T(n) = 4T(\frac{n}{2}) + n^3$