CS 477/677 Analysis of Algorithms

Homework 3

Due February 18, 2020

For the programming problems below, include in your hardcopy submission a printout of your algorithm and a screenshot of the output. Please follow attached submission instructions.

1. (U & G-required) [40 points] Consider the following algorithm.

> **ALGORITHM** *Enigma*(A[0..n − 1])
> //Input: An array A[0..n − 1] of integer numbers
> **for** $i \leftarrow 0$ **to** $n - 2$ **do**
>   **for** $j \leftarrow i + 1$ **to** $n - 1$ **do**
>   **if** $A[i] = A[j]$
>   **return false**
> **return true**

a) [15 points] What does this algorithm do?

b) [25 points] Compute the running time of this algorithm.

2. (U & G-required) [40 points]

(a) [20 points] Implement in C/C++ a divide and conquer algorithm for finding **the position of the largest element** in an array of *n* numbers. Show how your algorithm runs on the input A = [1 4 9 3 4 9 5 6 9 3 7].

(b) [10 points] What will be your algorithm's output for arrays with several elements of the largest value? Indicate the answer on the input given above.

(c) [10 points] Set up and solve a recurrence relation for the number of key comparisons made by your algorithm.

Note: Name your source file problem2.c or problem2.cpp.

3. (U & G-required) [20 points]

Implement in C/C++ an algorithm to rearrange elements of a given array of n real numbers so that all its negative elements precede all its positive elements. Your algorithm should be both time- and space-efficient. Show the output of your algorithm on the input array A = [4 -3 9 8 7 -4 -2 -1 0 6 -5]. **Note:** Name your source file problem3.c or problem3.cpp.

4. (G-required) [20 points]

Estimate how many times faster an average successful search will be in a sorted array of 100,000 elements if it is done by binary search versus sequential search.

Extra credit

5. [20 points] How can one use binary search for range searching, i.e., for finding all the elements in a sorted array whose values fall between two given values L and U (inclusively), $L \leq U$? What is the worst-case efficiency of this algorithm?