CS2123 Data Structures Summer 2017

Assignment 5: Asymptotic Notation and Searching

Due 7/24/17 by 11:59pm

1. Growth analysis (8 points)

For each of the below code snippets, identify the bounding function (the Big-Oh) of the number of times the indicated line is run (justify your answer briefly):

```
(1) for(i = 1; i < n; i+=2) {
       printf("Insert critical work here.\n");  /* THIS LINE */
    }

(2) for(i = 1; i < n; i++) {
       for(j = 1000/i; j > 0; j--) {
             arr[j]++;                         /* THIS LINE */
       }
    }

(3) for(i = 0; i < n; i++) {
       for(j = i; j < n; j++) {                /* CAREFUL WITH THIS LINE! */
          while( j<n ){
             arr[i] += arr[j];                 /* THIS LINE */
             j++;
          }
       }
    }

(4) m = pow( 2, n );                     /* i.e., m = 2^n */
    while( m > 1 ) {
       printf("Important message!\n");  /* THIS LINE */
       m = m/2;
    }
```

2. Insertion sort (8 points)

Write a C program that implements a simple array-based insertion sort. This problem is like problem 3 below but much simpler (and is therefore a good starting point for solving problem 3). Your program must read in the integers from the accompanying data file and use insertion sort to store the sorted data into an array. If you must insert an element between two existing values, then you must also move (or shift) the elements all elements with an index >= the index where you wish to insert the new element. Note that you can find the insertion sort algorithm in the textbook and the slides.

ADDITIONAL REQUIREMENT: Your program must output the total # of assignments performed in your sorting function from insertion and array element shifting (don't worry about assignments like control variable updates in loops, etc). This # will be used in problem #4 below. Include this # in the written document portion of your assignment submission. It can be something simple like:

Problem 2: # of sorting assignments =<your answer here>

You should also print out the sorted array to confirm that the data was correctly sorted.

3. Two-way Insertion Sort (10 points)

The two way insertion sort is a modification of your simple insertion sort. Suppose you are sorting the array x. A separate output array of size $2n+1$ is set aside. Initially $x[0]$ is placed into the middle element of the array n.

Continue inserting elements until you need to insert between a pair of elements in the array. As before you need to make room for the new element by shifting elements. Unlike before, you can choose to shift all smaller elements one step to the left or all larger elements one step to the right since there is additional room on both sides of the array. The choice of which shift to perform depends on which would require shifting the smallest amount of elements.

ADDITIONAL REQUIREMENT: Your program must output the total # of assignments performed in your sorting function from insertion and array element shifting (don't worry about assignments like control variable updates in loops, etc). This # will be used in problem #4 below. Include this # in the written document portion of your assignment submission. It can be something simple like:

Problem 3: # of sorting assignments =<your answer here>

You should also print out the sorted array to confirm that the data was correctly sorted.

4. Evaluation of #2 and #3 (4 points)

Which implementation seems to be more efficient with respect to assignments from both insertion of the new element and shifting elements in the array? By how much? Note: you can calculate the % speedup as:

$$100 * (1 - (\text{smaller \# of assignments}/\text{larger \# of assignments}))$$

Do you think the Big-Oh runtimes are different for your two implementations? Why or why not? You don't have to write much, but please write in complete sentences.

Deliverables:

The answers to homework problems 1 and 4 should be typed and submitted in a single document. You must also include in your written document the # of assignments made in problems 2 and 3.

The source code for problems 2 and 3 should be submitted each as one or more C source code files.

Archive and submit the files in UTSA's Blackboard system (learn.utsa.edu) under Assignment 5.

Remember:

The program you submit should be the work of only you. Cheating will be reported to judicial affairs. Both the copier and copiee will be held responsible.