CSC-321 Design and Analysis of Algorithms
Section 401
Fall 2019-20

Instructor: Iyad Kanj
Office: CDM 832
Phone: (312) 362-5558
Email: ikanj@cs.depaul.edu
Office Hours: Monday & Wednesday 3:00 - 4:30
Course Website: https://d2l.depaul.edu/

Assignment #2

(Due October 7)

1. (10 points) Textbook, pages 71-72, exercise 2.5 parts (a), (c), (d), and (g). You do not need to give a $\Theta()$ bound for (a), (c), (d); it suffices to give the $O()$ bound that results from applying the Master theorem.

2. (10 points) Illustrate the execution of Merge Sort on the array $A = \langle 20, 17, 35, 30, 15, 14, 36, 12 \rangle$. Regarding the level of illustration, follow the level of illustration done in class but complete the illustration until the end (array is sorted).

3. (10 points) Illustrate the execution of Quick Sort on the array $A = \langle 20, 17, 35, 30, 15, 14, 36, 12 \rangle$. Use the version of Quick Sort given in the lecture notes. Regarding the level of illustration, follow the level of illustration done in class but complete the illustration until the end (array is sorted).

4. (10 points) Give a recursive version of the algorithm **Insertion-Sort** based on the following paradigm: to sort $A[1..n]$, we first sort $A[1..n-1]$ recursively and then insert $A[n]$ in its appropriate position. Write a pseudocode for the recursive version of **Insertion-Sort** and analyze its running time by giving a recurrence relation for the running time and then solving it.

5. (10 points) Let A be an array of n numbers. Suppose that you have an algorithm **Algo**() that finds the median of any array of numbers[1] in time linear in the number of elements of the array (that is, in $O(\#\text{elements in the array})$). Show how, using **Algo**(), **Quick Sort** can be modified so that it runs in $O(n \lg n)$ time in the worst case. Write a pseudocode for the modified **Quick Sort**, and analyze its running time by describing its recurrence relation and solving it. You can use the subroutine **Algo**(), which takes as input an array and returns the position of the median element in the array, as a black box in the modified **Quick Sort**.

6. (10 points) Suppose that we are given an array $A[1..n]$ of integers (note that integers can be negative) such that $A[1] < A[2] < \ldots < A[n]$. Give an $O(\lg n)$ time algorithm to decide if there exists an index $1 \leq i \leq n$ such that $A[i] = i$.

7. (40 points) Program the following sorting algorithms: **InsertionSort**, **MergeSort**, and **QuickSort**. There are 9 test files uploaded for you on D2L (under the same folder as the assignment), each containing a list of numbers. There are 3 files for each of the input sizes 10, 100, and 1000, and for each input size, there are 3 files that contain the same numbers, but arranged in different orders: sorted, sorted in reverse order, and random. Your program should read the input from the test files, sort the numbers in the file using each of the above sorting algorithms, and output the sorted numbers to the screen. You can use any standard programming language such as **C**, **C++**, **Visual C++**, **C#**, **Java**, **Python**.

   Upload a single zipped file on D2L containing (1) the source code of your program, and (2) a PDF file of your answers to problems 1-6 in this assignment.

[1] A median of an array of n numbers is the element that appears in position $n/2$ in the sorted array.