CS367 Homework 5

Lecture 1, Spring 2018

Due by 11:59 pm on Friday, April 20, 2018 (not accepted late)

In this page: Questions | Handing in Related pages: Assignments

Questions

What do I need to answer?

1. Shaker sort is a bi-directional bubble sort. The shaker sort algorithm is:

```
begin = 0, end = A.length-1

// At the beginning of every iteration of this loop, we know that the
// elements in A are in their final sorted positions from A[0] to A[begin-1]
// and from A[end+1] to the end of A.  That means that A[begin] to A[end] are
// still to be sorted.
do
    for i going from begin to end-1
        if A[i] and A[i+1] are out of order, swap them
    end--

    if no swaps occurred during the preceding for loop, the sort is done

    for i going from end to begin+1
        if A[i] and A[i-1]  are out of order, swap them
    begin++
 until no swaps have occurred or begin >= end
```

   Part A: What is the **best-case time complexity** of shaker sort? **Briefly explain** the reasoning behind your answer. The best-case behavior would be achieved if the values in the array were *(complete this part of the description)*.

   Part B: What is the **worst-case time complexity** of shaker sort? **Briefly explain** the reasoning behind your answer. The worst-case behavior would be achieved if the values in the array were *(complete this part of the description)*.

2. Sometimes you want to sort lots of data, but you only need the smallest (or largest) K items instead of all items. For example, maybe you want to give awards to the ten students with the best GPAs. If you have thousands of students, you could waste time sorting all of them just to find the top ten. Perhaps we can adapt an existing sorting algorithm so that it works faster if we just want to find and sort the smallest (or largest) K items.

   Suppose we can start with either insertion sort or selection sort.

   Part A: Which one is easier to modify so that it **efficiently** gives us only the smallest **K** items (in sorted order) instead of all items?

   Part B: Describe the changes needed to make this happen.

   Part C: Briefly explain why the other algorithm cannot be modified to do what we want efficiently.

3. Using median-of-three pivot selection, **show the successive changes** that are made as quicksort partitions the following list of integers. Note you need only show the changes that occur during the the first partitioning, stopping just before the recursive calls.

80	90	50	10	80	70	30	40	70	50	40	20	60

Handing in

Please include your name at the top your file.

Put your answers to the questions into one file named `Homework5` with the appropriate file extension, e.g., `Homework5.pdf` (see File Format for acceptable file formats).

Electronically submit your work to the Homework 5 tab on Canvas.