Fall_2017_INFO6205_Quiz_8 45 minutes

Question - 1

What does MergeSort have in common with QuickSort

SCORE: **5 points**

Using your knowledge of MergeSort and QuickSort, choose which of the following statements is true.

● They each tend to be O(n log n) in the average case; they are both recursive, operating on two or more partitions.

○ They are both "stable"sorts

○ They both use a lot of extra memory

○ Unlike Quicksort, MergeSort can benefit from having a recursion cutoff which uses InsertionSort.

Question - 2

Quick sort randomization step

SCORE: **4 points**

Why does quick sort shuffle the input before starting work? Isn't that counter-productive?

Question - 3

Minimum number of compares

SCORE: **5 points**

What is the theoretical minimum number of compares that a sort algorithm must perform when data is randomly shuffled prior to the sort?

○ n

○ n^2

● n lg n

○ n!

Question - 4

Quick Sort Three Way Implementation

SCORE: **30 points**

Implement the quick sort using 3-way partitioning implementation, also you are required to do the insertion sort cutoff using the provided insertion sort function.

```
Cutoff to insertion sort: As with most recursive
algorithms, an easy way to improve the
performance of quicksort is based on the
```

```
following two observations:
        ■ Quicksort is slower than insertion sort
for tiny subarrays.
        ■ Being recursive, quicksort’s sort() is
certain to call itself for tiny subarrays.
Accordingly, it pays to switch to insertion sort
for tiny subarrays.
```

```
Sample Input
5
5 4 3 1 2

The first line is size of the array.
The second line is the value of the intergers in
the array, ie. The elements in the array
```

```
Sample output:
1 2 3 4 5
The output is the sorted array
```

Please note: you are not allowed to use the Arrays.sort() method.