CS 61B Exam Prep 11: Sorting Spring 2020

1 Identifying Sorts

Below you will find intermediate steps in performing various sorting algorithms on the same input list. The steps do not necessarily represent consecutive steps in the algorithm (that is, many steps are missing), but they are in the correct sequence. For each of them, select the algorithm it illustrates from among the following choices: insertion sort, selection sort, mergesort, quicksort (first element of sequence as pivot), and heapsort.

Input list:1430, 3292, 7684, 1338, 193, 595, 4243, 9002, 4393, 130, 1001

(a)

1430, 3292, 7684, 193, 1338, 595, 4243, 9002, 4393, 130, 1001

1430, 3292, 193, 1338, 7684, 595, 4243, 9002, 130, 1001, 4393

193, 1338, 1430, 3292, 7684, 130, 595, 1001, 4243, 4393, 9002

(b)

1338, 193, 595, 130, 1001, 1430, 3292, 7684, 4243, 9002, 4393

193, 595, 130, 1001, 1338, 1430, 3292, 7684, 4243, 9002, 4393

130, 193, 595, 1001, 1338, 1430, 3292, 4243, 9002, 4393, 7684

(c)

1338, 1430, 3292, 7684, 193, 595, 4243, 9002, 4393, 130, 1001

193, 1338, 1430, 3292, 7684, 595, 4243, 9002, 4393, 130, 1001

193, 595, 1338, 1430, 3292, 7684, 4243, 9002, 4393, 130, 1001

(d)

1430, 3292, 7684, 9002, 1001, 595, 4243, 1338, 4393, 130, 193

7684, 4393, 4243, 3292, 1001, 595, 193, 1338, 1430, 130, 9002

130, 4393, 4243, 3292, 1001, 595, 193, 1338, 1430, 7684, 9002

2 Conceptual Sorts

Answer the following questions regarding various sorting algorithms that we've discussed in class. If the question is T/F and the statement is true, provide an explanation. If the statement is false, provide a counterexample.

(a) [True/False] Quicksort has a worst case runtime of $\Theta(N\log N)$, whereN is the number of elements in the list that we're sorting.

(b) We have a system running insertion sort and we find that it's completing faster than expected. What could we conclude about the input to the sorting algorithm?

(c) Give a 5 integer array such that it elicits the worst case running time for insertion sort.

(d) [True/False] Heapsort is stable.

(e) Give some reasons as to why someone would use mergesort over quicksort

(f) You will be given an answer bank, each item of which may be used multiple times. You may not need to use every answer, and each statement may have more than one answer.

A. QuickSort (nonrandom, inplace using Hoare partitioning, and choose the leftmost item as the pivot)

B. MergeSort

C. Selection Sort

D. Insertion Sort

E. HeapSort

N. (None of the above)

List all letters that apply. List them in alphabetical order, or if the answer is none of them, use N indicating none of the above. All answers refer to the entire sorting process, not a single step of the sorting process. For each of the problems below, assume that N indicates the number of elements being sorted.

_____ Bounded by $\Omega(N\log N)$lower bound.

_____ Has a worst case runtime that is asymptotically better than Quicksort's worstcase runtime.

_____ In the worst case, performs $\Theta(N)$ pairwise swaps of elements.

_____ Never compares the same two elements twice.

_____ Runs in best case $\Theta(\log N)$time for certain inputs

3 Counting Inversions

Given an array of size N, find the number of inversions in O(NlogN) time. Hint: Use merge sort.

```
public static int countInversions (int[] arr, int left, int right) {
    int count = 0;
    int middle = ____________________;

    if (left<right) {
        count+=countInversions(____________________);
        count+=countInversions(____________________);
        count+=mergeAndCount(____________________);
    }
    return count;
}

public static int mergeAndCount(int[] arr, int left, int middle, int
   right) {
    int count = 0;
    int[] leftArr = new int[____________________];
    int[] rightArr = new int[____________________];
    System.arraycopy(arr,________,leftArr,________,________);
    System.arraycopy(arr,________,rightArr,________,________);

    int leftPointer = 0;
    int rightPointer = 0;
    int mergePointer = left;
    while (leftPointer < leftArr.length && rightPointer <
       rightArr.length) {
        if (____________________) {
            ________________________________________;
            ________________________________________;
            ________________________________________;
            ________________________________________;
        } else {
            ________________________________________;
            ________________________________________;
        }
    }

    //if uneven split
    while (____________________) {
        ____________________;
        ____________________;
        ____________________;
    }
    while (____________________) {
        ____________________;
        ____________________;
        ____________________;
    }
    return __________;
}
```

4 Sorted Runtimes

We want to sort an array of N distinct numbers in ascending order. Determine the best case and worst case runtimes of the following sorts -

(a) Once the runs in merge sort are of $size <= N/100$, we perform bubble sort on them.

(b) We can only swap adjacent elements in selection sort.

(c) We use a linear time median finding algorithm to select the pivot in quicksort.

(d) We implement heapsort with a min-heap instead of a max-heap. You may modify heapsort but must have constant space complexity.

(e) We run an optimal sorting algorithm of our choosing knowing:

- There are at most N inversions
- There is exactly 1 inversion
- There are exactly $(N^2 - N)/2$ inversions