CSE 373 Section Handout #9: Comparison Sorting

1. Use merge sort to sort the following input: [7, 5, 2, 4, 1, 6, 3, 0]. Show each step.

2. Perform radix sort on the following array of integers: [36, 9, 0, 25, 1, 49, 64, 16, 81, 4]. Show your work.

3. The following arrays are partially sorted, the result of a malicious TA interrupting the sorting algorithm being performed on each array. Use your knowledge of comparison based sorting to determine which algorithm was being used on each array.

Choose from the following types of sorts (each appears exactly once):
Heap Sort, Insertion Sort, Merge Sort, Selection Sort

Array: Sort Used:

-5	2	19	53	44	91	87	35

14	42	17	72	12	10	5	1

29	35	44	114	37	30	28	46

6	10	3	50	15	60	1	34

4. Answer the following:
    a. We are expecting the majority of the data that we are sorting to be “almost” in order. What would be a good sorting algorithm to use?

    b. Our mobile application needs to sort an array of comparable elements. Being a mobile application, we would like to use as little extraneous memory as possible. Which sorting algorithm should we use?

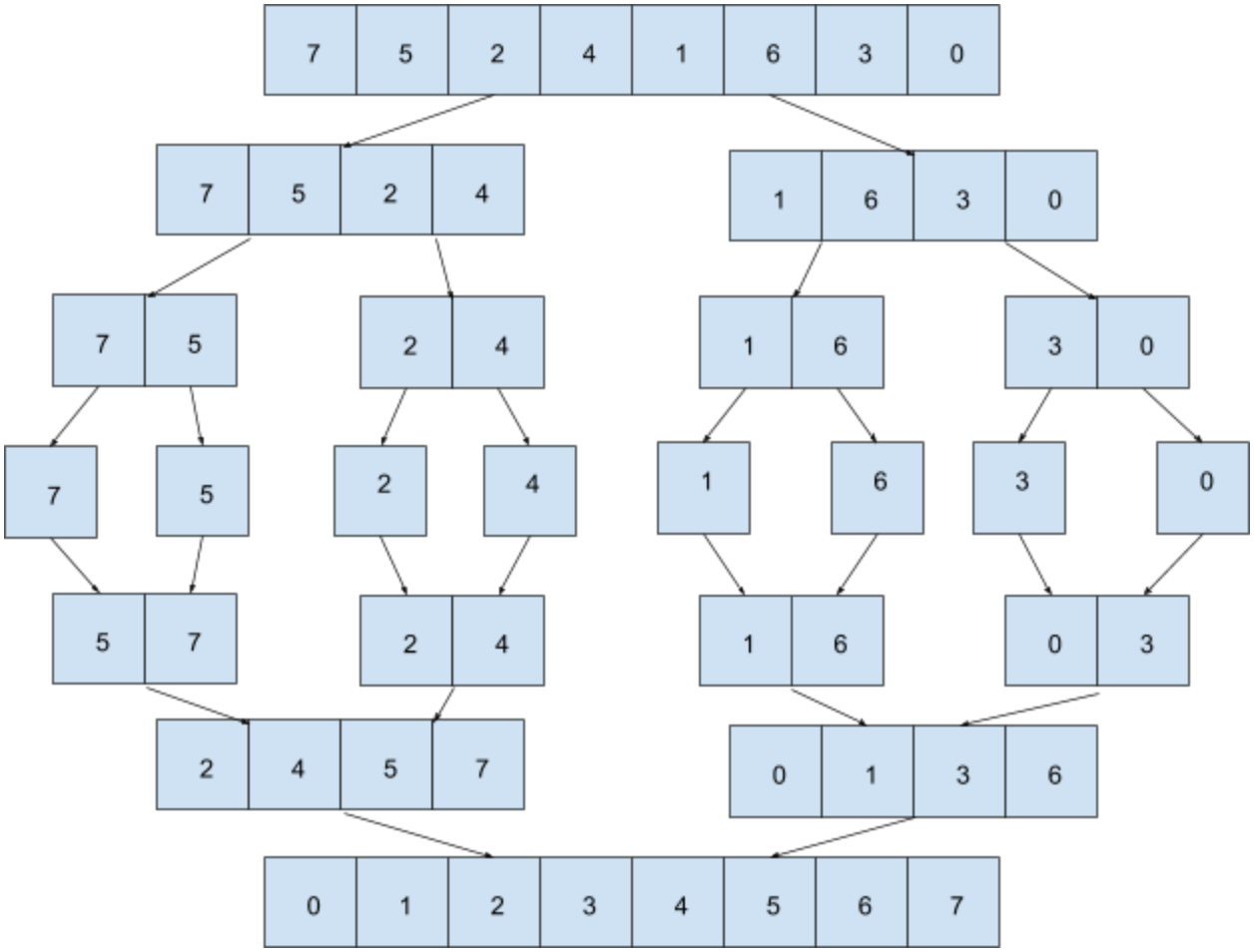
    c. If our data was guaranteed to always be in reverse order, what’s the **worst** sorting algorithm we could possibly use (assuming we NEED it in order)?

    d. Which sorting algorithm would be best to sort integers in the domain of [-50, 50]?

5. During a job interview, you are asked to implement a functioning, comparison-based sorting algorithm. The runtime of your implementation doesn’t matter - All that is important is that you provide a working sorting algorithm. Choose one of the sorting algorithms we talked about in class and implement it here.

CSE 373 Section Handout #9: Solutions

1. Use merge sort to sort the following input: [7, 5, 2, 4, 1, 6, 3, 0]. Show each step.



2. Perform radix sort on the following array of integers: [36, 9, 0, 25, 1, 49, 64, 16, 81, 4]. Show your work.

First pass:

0	1	2	3	4	5	6	7	8	9
0	1 81			64 4	25	36 16			9 49

Second pass:

0	1	2	3	4	5	6	7	8	9
0 1 4 9	16	25	36	49		64		81	

3. The following arrays are partially sorted, the result of a malicious TA interrupting the sorting algorithm being performed on each array. Use your knowledge of comparison based sorting to determine which algorithm was being used on each array.

For some arrays, multiple sorting algorithms could have been used - only a few are listed.

Array:								Sort Used:
-5	2	19	53	44	91	87	35	**Selection sort, as the first few of values are sorted "globally"**
14	42	17	72	12	10	5	1	**Heap sort. The first 4 values are a heap, and the last 4 are in reverse sorted order.**
29	35	44	114	37	30	28	46	**This is insertion sort. The first 5 values are in locally sorted order.**
6	10	3	50	15	60	1	34	**Merge sort, as we have multiple sorted sub-portions**

4. Answer the following:
    a. We are expecting the majority of the data that we are sorting to be “almost” in order. What would be a good sorting algorithm to use?

    Insertion sort is great for when data is almost sorted.

    b. Our mobile application needs to sort an array of comparable elements. Being a mobile application, we would like to use as little extraneous memory as possible. Which sorting algorithm should we use?

    Here we want an in-place sorting algorithm. Heap sort or quick sort come to mind as efficient in-place sorting algorithms.

    c. If our data was guaranteed to always be in reverse order, what’s the **worst** sorting algorithm we could possibly use (assuming we NEED it in order)?

    Reverse sorted order is the worst case for insertion sort!

    d. Which sorting algorithm would be best to sort integers in the domain of [-50, 50]?

    We know that we are sorting integers, and our domain of values, so we can use bucket sort! (store counts in an array, where val goes to arr[val+50])

5. During a job interview, you are asked to implement a functioning, comparison-based sorting algorithm. The runtime of your implementation doesn’t matter - All that is important is that you provide a working sorting algorithm. Choose one of the sorting algorithms we talked about in class and implement it here.

Various answers