CS 61B | More Sorting
Fall 2020 | Discussion 13: November 16, 2020

1 Quicksort

(a) Sort the following unordered list using stable Quicksort. Assume that we always choose first element as the pivot and that we use the 3-way merge partitioning process described in lecture and lab. Show the steps taken at each partitioning step.

```
18, 7, 22, 34, 99, 18, 11, 4
```

(b) What is the best and worst case running time of Quicksort with Hoare Partitioning on N elements? Give an example of a list of 5 numbers that would result in best and worst case running time.

```
Best: ______________        Example list: ___ ___ ___ ___ ___

Worst: _____________        Example list: ___ ___ ___ ___ ___
```

(c) What are two techniques that can be used to reduce the probability of Quicksort taking the worst case running time?

2 Comparison Sorts Summary

(a) When choosing an appropriate algorithm, there are often several trade-offs that we need to consider. Complete the chart for the following sorting algorithms: give the expected time complexity in the worst case, in the best case, and whether or not each sort is stable.

	Time Complexity (Best)	Time Complexity (Worst)	Stability
Selection Sort			
Insertion Sort			
Heapsort			
Mergesort			
Quicksort (w/ Hoare Partitioning)			

(b) For each unstable sort, give an example of a list where the order of equivalent items is not preserved.

(c) In general, what are some other tradeoffs we might want to consider when designing or choosing a sorting algorithm?

(d) Notice that the worst-case runtime in the comparison sorts on an N element array listed above are lower bounded by $\Theta(N \log N)$. Can there be a sort that runs faster than $\Theta(N \log N)$ in the worst-case?

3 Radix Sorts

(a) Sort the following list using LSD Radix Sort with counting sort. Show the steps taken after each round of counting sort. The first row is the original list and the last two rounds are already filled for you.

	30395	30326	30392	30315
1				
2				
3				
4	30315	30326	30392	30395
5	30315	30326	30392	30395

(b) Sort the following list using MSD Radix Sort with counting sort. Show the steps taken after each round of counting sort. The first row is the original list and the first three rounds are already filled for you.

<table>
<tr><td></td><td>30395</td><td>30326</td><td>30392</td><td>30315</td></tr>
<tr><td>1</td><td>| $\underline{3}$0395</td><td>$\underline{3}$0326</td><td>$\underline{3}$0392</td><td>$\underline{3}$0315 |</td></tr>
<tr><td>2</td><td>| $\underline{30}$395</td><td>$\underline{30}$326</td><td>$\underline{30}$392</td><td>$\underline{30}$315 |</td></tr>
<tr><td>3</td><td>| $\underline{303}$95</td><td>$\underline{303}$26</td><td>$\underline{303}$92</td><td>$\underline{303}$15 |</td></tr>
<tr><td>4</td><td></td><td>|</td><td>|</td><td></td></tr>
<tr><td>5</td><td></td><td>|</td><td>|</td><td>|</td></tr>
</table>

(c) Give the best case runtime, worst case runtime, and whether or not the sort is stable for both LSD and MSD radix sort. Assume we have N elements, a radix R, and a maximum number of digits in an element W.

	Time Complexity (Best)	Time Complexity (Worst)	Stability
LSD Radix Sort			
MSD Radix Sort			

(d) Is radix sort always the best sort to use? Explain why or why not.

4 Bounding Practice *Extra*

Given an array of n elements, the heapification operation permutes the elements of the array into a heap. There are many solutions to the heapification problem. One approach is bottom-up heapification, which treats the existing array as a heap and rearranges all nodes from the bottom up to satisfy the heap invariant. Another is top-down heapification, which starts with an empty heap and inserts all elements into it.

(a) Why can we say that any solution for heapification requires $\Omega(n)$ time?

(b) Show that the worst-case runtime for top-down heapification is in $\Theta(n \log n)$. Why does this mean that the optimal solution for heapification takes $O(n \log n)$ time?

(c) In contrast, bottom-up heapification is an $O(n)$ algorithm. Is bottom-up heapification asymptotically-optimal?

(d) Show that the running time of bottom-up heapify is $\Theta(n)$.

Some useful facts:

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$$

Taking the derivative:

$$\frac{d}{dx}(\sum_{i=0}^{\infty} x^i) = \frac{1}{(1-x)^2}$$