More Sorting

Discussion 13

Announcements

Week 13

- ❏ Week 13 Survey, due Monday 04/19
- ❏ Pre-Final Form, due Monday 04/19
- ❏ Project 3 Phase 2 due Tuesday, 04/27 (no slip days)
- ❏ Homework 3 due Monday, 05/03 (no slip days)

Content Review

Some radix vocabulary

A **radix** can be thought of as the alphabet or set of digits to choose from in some system. Properly, it is defined as the base of a numbering system. The **radix size** of the English alphabet is 26, and the radix size of Arabic numerals is 10 (0 through 9).

Radix sorts work by using **counting sorts** to sort the list, one digit at a time.

LSD Radix Sort

LSD sorts numbers by sorting them by digit from lowest digit to largest digit. We'll see an example of this on the worksheet.

Runtime: $\Theta(W(N + R))$

MSD Radix Sort

MSD sorts numbers by sorting them by digit from largest digit to smallest digit. We'll see an example of this on the worksheet.

Runtime: $O(W(N + R))$

Quicksort - More review

3 Way Partitioning or 3 scan partitioning is a simple way of partitioning an array around a pivot. You do three scans of the list, first putting in all elements less than the pivot, then putting in elements equal to the pivot, and finally elements that are greater. This technique is NOT in place.

Hoare Partitioning is a very fast in-place technique for partitioning.

We use a pair of pointers that start at the left and right edges of the array and move towards each other. The left pointer likes items < the pivot, and the right likes items > the pivot.

The pointers walk until they see something they don't like, and once both have stopped, they swap items. After swapping, they continue moving towards each other, and the process completes once they have crossed.

Finally, we swap the pivot into the appropriate location, and the partitioning is completed

Worksheet

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot: 18

```
[ _   _   _   _   _   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot: 18

```
[ 7   11   4   _   _   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot: 18

```
[ 7   11   4   18   18   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot: 18

```
[ 7  11  4  18  18  22  34  99 ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot:

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot: 7

```
[ _   _   _   _   _   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot: 7

```
[ 4   _   _   _   _   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot: 7

```
[ 4   7   _   _   _   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot: 7

```
[ 4   7   11   _   _   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot:

```
[ 4   7   11   _   _   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot: 18

```
[ 7  11  4  18  18  22  34  99 ]
```

Pivot: N/A

```
[ 4  7  11  18  18  _  _  _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot:

```
[ 4   7   11   18   18   _   _   _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot: 22

```
[ 4   7   11   18   18    _    _    _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot: 22

```
[ 4   7   11   18   18   22    _    _ ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot: 22

```
[ 4   7   11   18   18   22   34   99 ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot:

```
[ 4   7   11   18   18   22   34   99 ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot:

```
[ 4   7   11   18   18   22   34   99 ]
```

Pivot: 34

```
[ 4   7   11   18   18   22   34   99 ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7  11  4  18  18  22  34  99 ]
```

Pivot:

```
[ 4  7  11  18  18  22  34  99 ]
```

Pivot: 34

```
[ 4  7  11  18  18  22  34  99 ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot: 18

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot:

```
[ 4   7   11   18   18   22   34   99 ]
```

Pivot: 34

```
[ 4   7   11   18   18   22   34   99 ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7   11   4   18   18   22   34   99 ]
```

Pivot:

```
[ 4   7   11   18   18   22   34   99 ]
```

Pivot: 34

```
[ 4   7   11   18   18   22   34   99 ]
```

1A Quicksort

Show the steps of running quicksort on the list below, selecting the first item as the pivot and using 3 way partitioning.

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

Pivot:

```
[ 7  11  4  18  18  22  34  99 ]
```

Pivot:

```
[ 4  7  11  18  18  22  34  99 ]
```

Pivot:

[4 7 11 18 18 22 34 99]

1BC Quicksort

What is the best and worst case running time of Quicksort with Hoare Partitioning on N elements? Given the two lists [4, 4, 4, 4, 4] and [1, 2, 3, 4, 5], assuming we pick the first element as the pivot every time, which list would happen to result in better runtime?

Best case:

Worst case:

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

2A Comparison Sorts Summary

	Best Time Complexity	Worst Time Complexity	Stability	In place
Selection Sort				
Insertion Sort				
Heapsort				
Mergesort				
Quicksort				

2BC Comparison Sorts Summary

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

c) Notice that the worst-case runtime in the comparison sorts on an N element array listed above are lower bounded by Θ(NLogN).. Can there be a sort that runs faster than Θ(Nlog N) in the worst-case?

3A Radix Sorts Sort [30395, 30326, 430392, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1				
2				
3				
4				
5				

3A Radix Sorts Sort [30395, 30326, 430392, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	43092	30395	30315	30326
2				
3				
4				
5				

3A Radix Sorts Sort [30395, 30326, 430392, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	43092	30395	30315	30326
2	30315	30326	43092	30395
3				
4				
5				

3A Radix Sorts Sort [30395, 30326, 430392, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	43092	30395	30315	30326
2	30315	30326	43092	30395
3	43092	30315	30326	30395
4				
5				

3A Radix Sorts Sort [30395, 30326, 430392, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	43092	30395	30315	30326
2	30315	30326	43092	30395
3	43092	30315	30326	30395
4	30315	30326	30395	43092
5				

3A Radix Sorts Sort [30395, 30326, 430392, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	43092	30395	30315	30326
2	30315	30326	43092	30395
3	43092	30315	30326	30395
4	30315	30326	30395	43092
5	30315	30326	30395	43092

3B Radix Sorts Sort [30395, 30326, 430392, 30315] using MSD Radix sort.

	30395	30326	43092	30315
1				
2				
3				
4				
5				

3B Radix Sorts Sort [30395, 30326, 430392, 30315] using MSD Radix sort.

	30395	30326	43092	30315
1	30395	30326	30315	43092
2				43092
3				43092
4				43092
5				43092

3B Radix Sorts Sort [30395, 30326, 430392, 30315] using MSD Radix sort.

	30395	30326	43092	30315
1	30395	30326	30315	43092
2	30395	30326	30315	43092
3				43092
4				43092
5				43092

3B Radix Sorts Sort [30395, 30326, 430392, 30315] using MSD Radix sort.

	30395	30326	43092	30315
1	30395	30326	30315	43092
2	30395	30326	30315	43092
3	30395	30326	30315	43092
4				43092
5				43092

3B Radix Sorts Sort [30395, 30326, 430392, 30315] using MSD Radix sort.

	30395	30326	43092	30315
1	30395	30326	30315	43092
2	30395	30326	30315	43092
3	30395	30326	30315	43092
4	30315	30326	30395	43092
5	30315	30326	30395	43092

3C Radix Sorts

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

3D Radix Sorts

We just saw above that radix sort has great runtime with respect to the number of elements in the list. Given this fact, should we say that radix sort is the best sort to use?

4A 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.

Why can we say that *any* solution for heapification requires Ω(n) time?

4B Bounding Practice *Extra*

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?

4C Bounding Practice *Extra*

Show that the worst-case runtime for top-down heapification is in $\Theta(n \log(n))$.

4D Bounding Practice *Extra*

Show that the worst-case runtime for bottom-up heapification is in $\Theta(n)$.