More Sorting

Discussion 12

Announcements

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	4/15 Project 3A due					
	4/22 Project 3B/C due					

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. This contrasts with what we've learned with **comparison sorts**, which compares elements in the list directly.

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.

```
120
923
112
342
199
```

General Runtime: $\Theta(W(N + R))$, where:

- W = width of longest key in list
- N = # elements being sorted
- R = radix size

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.

```
120
923
112
342
199
```


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.

```
120        120
923        112
112   →    342
342        923
199        199
```

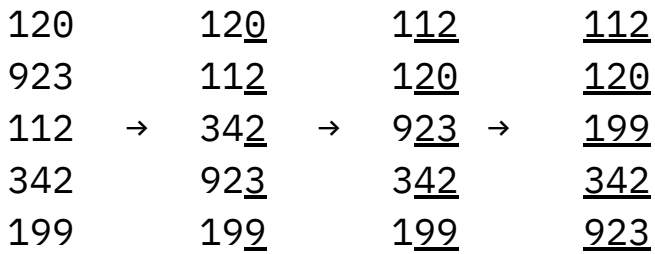

LSD Radix Sort

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

120		12**0**		1**12**
923		11**2**		1**20**
112	→	34**2**	→	9**23**
342		92**3**		3**42**
199		19**9**		1**99**

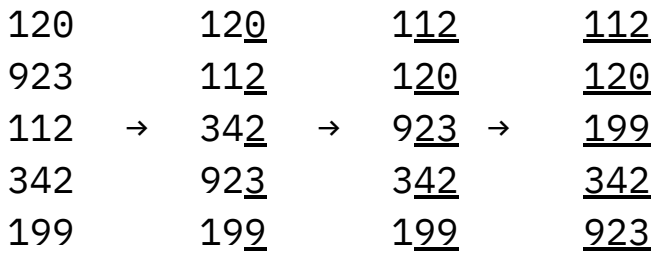
LSD Radix Sort

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



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.



General Runtime: Θ(W(N + R)), where:

- W = width of longest key in list
- N = # elements being sorted
- R = radix size

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.

```
120
923
112
342
199
```

General Runtime: O(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.

```
120
923
112
342
199
```


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.

120		120
923		112
112	→	199
342		342
199		923

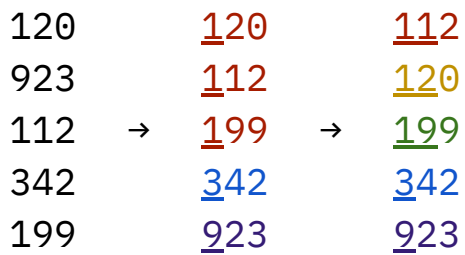
MSD Radix Sort

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

120		120		112
923		112		120
112	→	199	→	199
342		342		342
199		923		923

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.



General 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, but it is stable.

3 1 2 5 4

Quicksort - More review

Hoare Partitioning is an unstable, in place algorithm for partitioning. We use a pair of pointers that start at the left and right edges of the array, skipping over the pivot.

The left pointer likes items < the pivot, and the right likes items > the pivot. The pointers walk toward each other until they see something they don't like, and once both have stopped, they swap items.

Then they continue moving towards each other, and the process completes once they have crossed. Finally, we swap the pivot with the pointer that originated on the right, and the partitioning is completed.

3 1 2 5 4

Link to Hoare partitioning demo used in lecture

Comparison Sorts Summary

	Best case	Worst case	Stable?	In Place?
Selection Sort	$\Theta(N^2)$	$\Theta(N^2)$	no	yes
Insertion Sort	$\Theta(N)$	$\Theta(N^2)$	yes	yes
Heapsort	$\Theta(N)$	$\Theta(N\log N)$	no	yes
Mergesort	$\Theta(N\log N)$	$\Theta(N\log N)$	yes	no (usually)
Quicksort (w/ Hoare Partitioning)	$\Theta(N\log N)$	$\Theta(N^2)$	no (usually)	yes (logN space)

Comparison sorts cannot run faster than $\Theta(N\log N)$! What about counting sorts?

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]

1B 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:

1B 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: $\Theta(N \log N)$ on [4, 4, 4, 4, 4]

Worst case: $\Theta(N^2)$ on [1, 2, 3, 4, 5]

1C Quicksort

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

1C Quicksort

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

1. Randomly choose pivots.

2. Shuffle the list before running Quicksort.

2A Radix Sorts Sort [30395, 30326, 43092, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1				
2				
3				
4				
5				

2A Radix Sorts Sort [30395, 30326, 43092, 30315] using LSD Radix sort.

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

2A Radix Sorts Sort [30395, 30326, 43092, 30315] using LSD Radix sort.

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

2A Radix Sorts Sort [30395, 30326, 43092, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	4309**2**	3039**5**	3031**5**	3032**6**
2	3031**5**	3032**6**	4309**2**	3039**5**
3	43**0**92	30**3**15	30**3**26	30**3**95
4				
5				

2A Radix Sorts Sort [30395, 30326, 43092, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	4309**2**	3039**5**	3031**5**	3032**6**
2	3031**5**	3032**6**	4309**2**	3039**5**
3	4**3**092	3**0**315	3**0**326	3**0**395
4	**3**0315	**3**0326	**3**0395	**4**3092
5				

2A Radix Sorts Sort [30395, 30326, 43092, 30315] using LSD Radix sort.

	30395	30326	43092	30315
1	4309<u>2</u>	3039<u>5</u>	3031<u>5</u>	3032<u>6</u>
2	303<u>1</u>5	303<u>2</u>6	430<u>9</u>2	303<u>9</u>5
3	43<u>0</u>92	30<u>3</u>15	30<u>3</u>26	30<u>3</u>95
4	3<u>0</u>315	3<u>0</u>326	3<u>0</u>395	4<u>3</u>092
5	<u>3</u>0315	<u>3</u>0326	<u>3</u>0395	<u>4</u>3092

2B Radix Sorts Sort [21295, 22316, 30753, 21248, 30751] using MSD Radix sort.

	21295	22316	30753	21248	30751
1					
2					
3					
4					
5					

2B Radix Sorts Sort [21295, 22316, 30753, 21248, 30751] using MSD Radix sort.

	21295	22316	30753	21248	30751
1	21295	22316	21248	30753	30751
2					
3					
4					
5					

2B Radix Sorts Sort [21295, 22316, 30753, 21248, 30751] using MSD Radix sort.

	21295	22316	30753	21248	30751
1	21295	22316	21248	30753	30751
2	21295	21248	22316	30753	30751
3					
4					
5					

2B Radix Sorts Sort [21295, 22316, 30753, 21248, 30751] using MSD Radix sort.

	21295	22316	30753	21248	30751
1	21295	22316	21248	30753	30751
2	21295	21248	22316	30753	30751
3	21295	21248	22316	30753	30751
4					
5					

2B Radix Sorts Sort [21295, 22316, 30753, 21248, 30751] using MSD Radix sort.

	21295	22316	30753	21248	30751
1	21295	22316	21248	30753	30751
2	21295	21248	22316	30753	30751
3	21295	21248	22316	30753	30751
4	21248	21295	22316	30753	30751
5					

2B Radix Sorts Sort [21295, 22316, 30753, 21248, 30751] using MSD Radix sort.

	21295	22316	30753	21248	30751
1	21295	22316	21248	30753	30751
2	21295	21248	22316	30753	30751
3	21295	21248	22316	30753	30751
4	21248	21295	22316	30753	30751
5	21248	21295	22316	30751	30753

2C Radix Sorts **Find the best case runtime, worst case runtime, and stability of MSD and LSD Radix Sort. Assume that we have N elements, a radix R, and a maximum number W of digits in an element.**

	Best Time Complexity	Worst Time Complexity	Stable?
LSD Radix Sort			
MSD Radix Sort			

2C Radix Sorts **Find the best case runtime, worst case runtime, and stability of MSD and LSD Radix Sort. Assume that we have N elements, a radix R, and a maximum number W of digits in an element.**

	Best Time Complexity	Worst Time Complexity	Stable?
LSD Radix Sort	Θ(W(N + R))	Θ(W(N + R))	yes
MSD Radix Sort	Θ(N + R)	Θ(W(N + R))	yes

2D Radix Sorts

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

2D Radix Sorts

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

No: Though radix sort runs linear with respect to the number of elements in the list, the runtime also depends on the size of the radix R and the length of the longest "word" W (or the number of digits in a number). Additionally, it is not always possible to use radix sort, because not all objects can be split up into digits.

3A Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
12, 7, 8, 4, 10, 2, 5, 34, 14

7, 8, 4, 10, 2, 5, 12, 34, 14

4, 2, 5, 7, 8, 10, 12, 14, 34
```


3A Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
12, 7, 8, 4, 10, 2, 5, 34, 14

7, 8, 4, 10, 2, 5, 12, 34, 14

4, 2, 5, 7, 8, 10, 12, 14, 34
```

Quicksort, using the first element as a pivot.

3B Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
23, 45, 12, 4, 65, 34, 20, 43
```

```
4, 12, 23, 45, 65, 34, 20, 43
```


3B Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
23, 45, 12, 4, 65, 34, 20, 43
```

```
4, 12, 23, 45, 65, 34, 20, 43
```

Insertion Sort.

3C Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
12, 32, 14, 11, 17, 38, 23, 34
```

```
12, 14, 11, 17, 23, 32, 38, 34
```


3C Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
12, 32, 14, 11, 17, 38, 23, 34
```

```
12, 14, 11, 17, 23, 32, 38, 34
```

MSD Radix Sort.

3D Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
45, 23, 5, 65, 34, 3, 76, 25

23, 45, 5, 65, 3, 34, 25, 76

5, 23, 45, 65, 3, 25, 34, 76
```


3D Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
45, 23, 5, 65, 34, 3, 76, 25

23, 45, 5, 65, 3, 34, 25, 76

5, 23, 45, 65, 3, 25, 34, 76
```

Merge Sort.

3E Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
23, 44, 12, 11, 54, 33, 1, 41
```

```
54, 44, 33, 41, 23, 12, 1, 11
```

```
44, 41, 33, 11, 23, 12, 1, 54
```


3E Sorting: Identification

Determine what sorting algorithm results in these intermediate steps.

```
23, 44, 12, 11, 54, 33, 1, 41

54, 44, 33, 41, 23, 12, 1, 11

44, 41, 33, 11, 23, 12, 1, 54
```

Heapsort.

4A Conceptual Sorts

Give a 5 integer array that elicits the worst case runtime for insertion sort.

4A Conceptual Sorts

Give a 5 integer array that elicits the worst case runtime for insertion sort.

5 4 3 2 1

4B Conceptual Sorts

Why would someone choose mergesort over quicksort?

4B Conceptual Sorts

Why would someone choose mergesort over quicksort?

The worst case runtime of mergesort is Θ(NlogN), while quicksort is Θ(N^2). Additionally, mergesort is stable while quicksort is not.

4C Conceptual Sorts

Which sorts do each of the following statements describe?

Bounded by Ω(NlogN) lower bound:

Worst case runtime is asymptotically better than Quicksort's worst case runtime:

In the worst case, performs Θ(N) pairwise swaps:

Never compares the same two elements twice:

Runs in best case Θ(logN) for certain inputs:

4C Conceptual Sorts

Which sorts do each of the following statements describe?

Bounded by Ω(NlogN) lower bound: A, B, C

Worst case runtime is asymptotically better than Quicksort's worst case runtime: B, E

In the worst case, performs Θ(N) pairwise swaps: C

Never compares the same two elements twice: A, B, D

Runs in best case Θ(logN) for certain inputs: F