

CLC - Radix Sort

Duc Vo & Spencer Meren

VIDEO LINK

<https://www.loom.com/share/646d902c572e4c1084fc7adc7a802703>

Definition

- _ Sorting is one of the most important techniques being used in almost every single algorithm.
- _ One of the fastest, and the most efficient sort out there is radix sort
- _ Radix sort by sorting the elements starting with the least significant digit, especially under binary where this is when the radix sort shines the most

How does it work

_ Radix sort work by sorting all of the elements based on the least significant digit first, and then work their ways up the higher one.

_ For example, let's sort: 123, 321, 5, 235, 748, 40, 92, 14, 349, 100

123

321

5

235

748

40

92

14

349

100

How does it work (cont.)

_ First, find the largest number in the elements, in our case 748 is the largest with 3 numbers, so fill every other elements to 3 numbers

_ Color is added accordingly for ease of demonstration

123

321

005

235

748

040

092

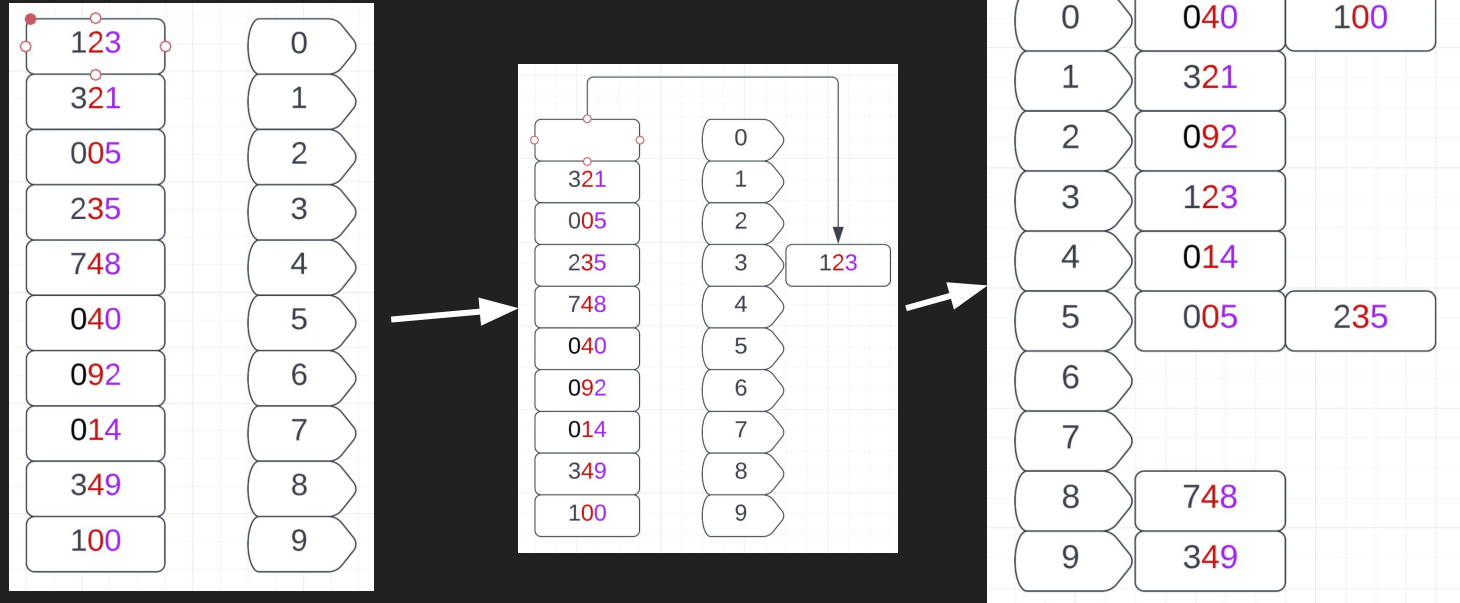
014

349

100

How does it work (cont.)

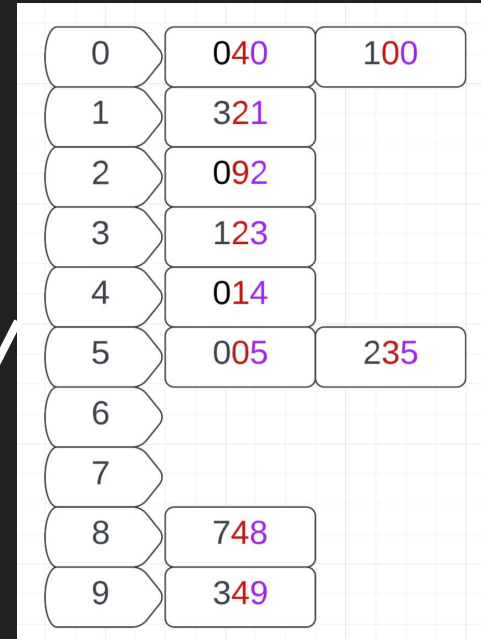
_ Create the bins from 0, 1, 2,... to 9, then sort them accordingly based on the digit of ones



How does it work (cont.)

_ The sorting can be read from left to right, top to bottom

_ Now start the second round with sorting the tens digit



0	040	100
1	321	
2	092	
3	123	
4	014	
5	005	235
6		
7		
8	748	
9	349	



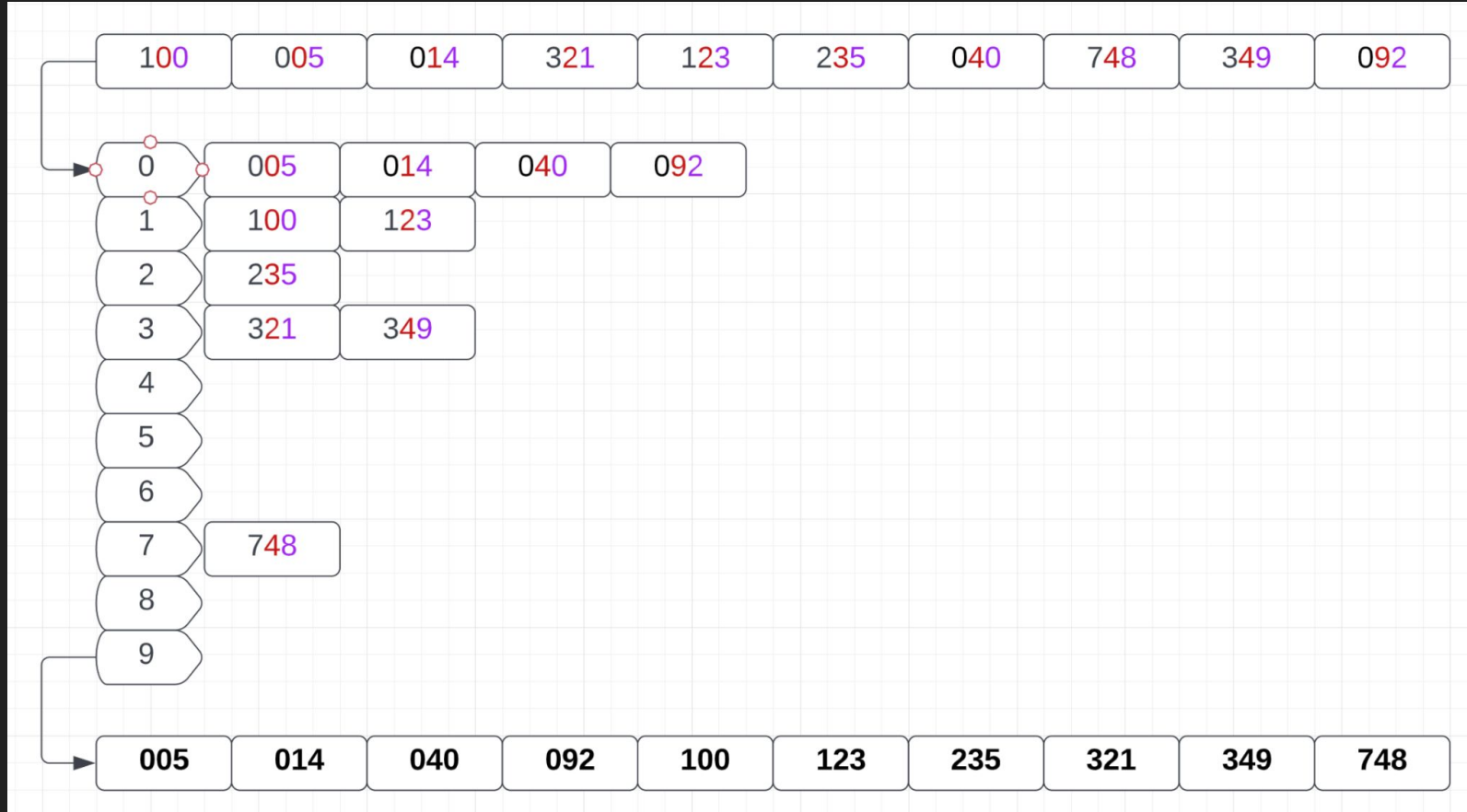
040	100	321	092	123	014	005	235	748	349
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

After the tens digits

0	100	005	
1	014		
2	321	123	
3	235		
4	040	748	349
5			
6			
7			
8			
9	092		

100	005	014	321	123	235	040	748	349	092
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

After the hundreds digits, we can have our sorted elements



Efficiency

_ Radix sort comes in to be one of the fastest sorting algorithm out there, with it being almost $O(nk)$, losing to only counting sort

_ It does take up spaces due to the bucket size

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
<u>Quicksort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
<u>Mergesort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Timsort</u>	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Heapsort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$
<u>Bubble Sort</u>	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
<u>Insertion Sort</u>	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
<u>Selection Sort</u>	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
<u>Tree Sort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$
<u>Shell Sort</u>	$\Omega(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
<u>Bucket Sort</u>	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
<u>Radix Sort</u>	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
<u>Counting Sort</u>	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$
<u>Cubesort</u>	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$

Work Cited

Know thy complexities! Big O Notation. (n.d.). Retrieved March 25, 2023, from <https://www.bigocheatsheet.com/>