Radix Sort

Radix Sort

- Counting sort is a linear time sorting algorithm that sort in O(n+k) time when elements are in the range from 1 to k.
- ***What if the elements are in* the *range from 1 to n^2?***
  We can't use counting sort because counting sort will take $O(n^2)$ which is worse than comparison-based sorting algorithms. Can we sort such an array in linear time?
- Radix Sort is the answer. The idea of Radix Sort is to do digit by digit sort starting from least significant digit to most significant digit. Radix sort uses counting sort as a subroutine to sort.

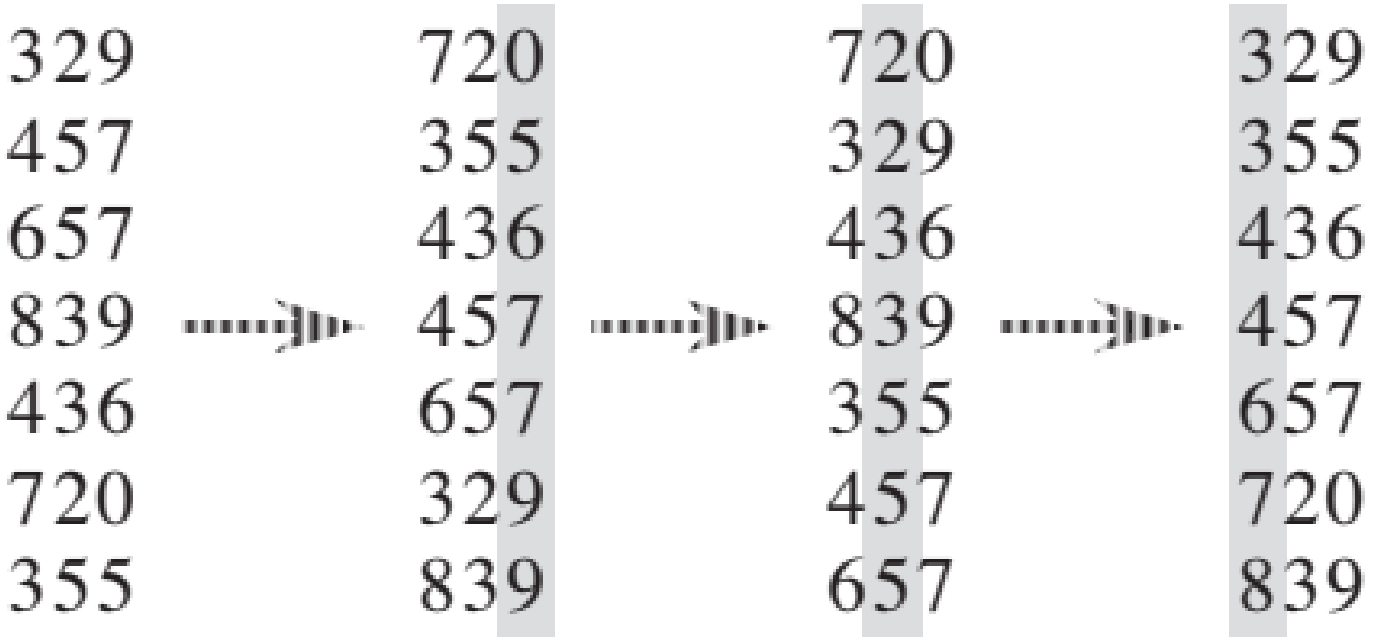
Radix Sort Algorithm

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      use a stable sort to sort array A on digit i
```

Radix Sort Dry Run

329
457
657
839
436
720
355

Radix Sort Dry Run



Radix Sort Dry Run

```
170, 45, 75, 90, 802, 24, 2, 66
```

Running Time of Radix Sort

$1 - n^2$

$2(n+n)\frac{1-n^2}{1-n^3}$

$\Theta(\log_b k(n+b))$

$k = n^c$

$cn\log_n n + cn\log_n n$

$cn(1) + n(1)$

$2cn$

$\Theta(n)$

$\frac{n\log_b k + b\log_b k}{n\log_b n^c + b\log_b n^c}$

$\frac{n\log_b k + b\log_b k}{n \cdot c\log_b n + cb\log_b n}$

$\frac{n\log_b k + b\log_b k}{n\log_b n + b\log_b n}$

$b = n$

$1 - n^2$

$n\log_b n^2$

$2n\log_b n$

$2n$

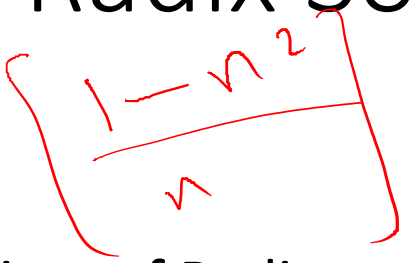
Running Time of Radix Sort

- Let there be d digits in input integers.
- Radix Sort takes O(d*(n+b)) time where b is the base for representing numbers
- For example, for the decimal system, b is 10. What is the value of d?
  - If k is the maximum possible value, then d would be $O(\log_b(k))$
  - So overall time complexity is $O((n+b) * \log_b(k))$.
  - Let us first limit k. Let $k <= n^c$ where c is a constant. In that case, the complexity becomes $O(n\text{Log}_b(n))$.

Running Time of Radix Sort

- So overall time complexity is $O((n+b) * \log_b(k))$.
- What if we make the value of b larger?
- What should be the value of b to make the time complexity linear?
- If we set b as n, we get the time complexity as O(n).
- In other words, we can sort an array of integers with a range from 1 to n^c if the numbers are represented in base n (or every digit takes $\log_2(n)$ bits).

Comparison of Quick Sort and Radix Sort



Pros of Radix Sort

- If we have $\log_2 n$ bits for every digit, the running time of Radix appears to be better than Quick Sort for a wide range of input numbers

- Cons of Radix Sort
  - The constant factors hidden in asymptotic notation are higher for Radix Sort
  - Quick-Sort uses hardware caches more effectively
  - Radix sort uses counting sort as a subroutine and counting sort takes extra space to sort numbers.