**Day 35**

**Radix Sort**

The [lower bound for the Comparison based sorting algorithm](https://www.geeksforgeeks.org/lower-bound-on-comparison-based-sorting-algorithms/) (Merge Sort, Heap Sort, Quick-Sort .. etc) is Ω(nLogn), i.e., they cannot do better than *nLogn*. [Counting sort](https://www.geeksforgeeks.org/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 n2?

We can’t use counting sort because counting sort will take O(n2) which is worse than comparison-based sorting algorithms. Can we sort such an array in linear time?

[Radix Sort](http://en.wikipedia.org/wiki/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.

The Radix Sort Algorithm

Do the following for each digit I where I varies from the least significant digit to the most significant digit. Here we will be sorting the input array using counting sort (or any stable sort) according to the i’th digit.

Example:

*Original, unsorted list: 170, 45, 75, 90, 802, 24, 2, 66 Sorting by least significant digit (1s place) gives: [\*Notice that we keep 802 before 2, because 802 occurred before 2 in the original list, and similarly for pairs 170 & 90 and 45 & 75.] 170, 90, 802, 2, 24, 45, 75, 66 Sorting by next digit (10s place) gives: [\*Notice that 802 again comes before 2 as 802 comes before 2 in the previous list.] 802, 2, 24, 45, 66, 170, 75, 90 Sorting by the most significant digit (100s place) gives: 2, 24, 45, 66, 75, 90, 170, 802*

What is the 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(logb(k)). So overall time complexity is O((n+b) \* logb(k)). Which looks more than the time complexity of comparison-based sorting algorithms for a large k. Let us first limit k. Let k <= nc where c is a constant. In that case, the complexity becomes O(nLogb(n)). But it still doesn’t beat comparison-based sorting algorithms.   
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 nc if the numbers are represented in base n (or every digit takes log2(n) bits).

Applications of Radix Sort:

* In a typical computer, which is a sequential random-access machine, where the records are keyed by multiple fields radix sort is used. For eg., you want to sort on three keys month, day and year. You could compare two records on year, then on a tie on month and finally on the date. Alternatively, sorting the data three times using Radix sort first on the date, then on month, and finally on year could be used.
* It was used in card sorting machines with 80 columns, and in each column, the machine could punch a hole only in 12 places. The sorter was then programmed to sort the cards, depending upon which place the card had been punched. This was then used by the operator to collect the cards which had the 1st row punched, followed by the 2nd row, and so on.

Is Radix Sort preferable to Comparison based sorting algorithms like Quick-Sort?

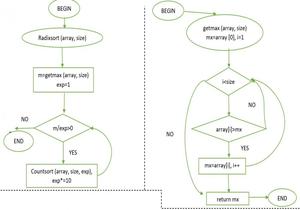
If we have log2n bits for every digit, the running time of Radix appears to be better than Quick Sort for a wide range of input numbers. The constant factors hidden in asymptotic notation are higher for Radix Sort and Quick-Sort uses hardware caches more effectively. Also, Radix sort uses counting sort as a subroutine and counting sort takes extra space to sort numbers.

Key points about Radix Sort:

Some key points about radix sort are given here

1. It makes assumptions about the data like the data must be between a range of elements.
2. Input array must have the elements with the same radix and width.
3. Radix sort works on sorting based on an individual digit or letter position.
4. We must start sorting from the rightmost position and use a stable algorithm at each position.
5. Radix sort is not an in-place algorithm as it uses a temporary count array.

FLowcharts



*Flowchart radix sort, getmax*

