Searching and Sorting – (5)

- Count sort → Stable
- Radix sort
- Broken economy
- First and Last Index
- Pivot in Rotated Sorted array

★ Sort 0 1 2 : Variations: ② Sort Red, Green & Blue
①
③ %3 == {0, 1, 2}
④ Dual Pivot Partitioning <p1, [p1, p2], >p2
⑤ Three way Partitioning <pivot, = pivot, >pivot

↓ Improves Quick Sort when applied on Arrays with many duplicate elements

9 6 3 5 3 4 3 9 6 4 6 5 8 2 9

① Count Sort with Stability

① first traversal
find min & max el O(n)
2 9

② freq. array

freq	1	3	2	2	3	0	1	3
num	2	3	4	5	6	7	8	9

③ prefix Sum array

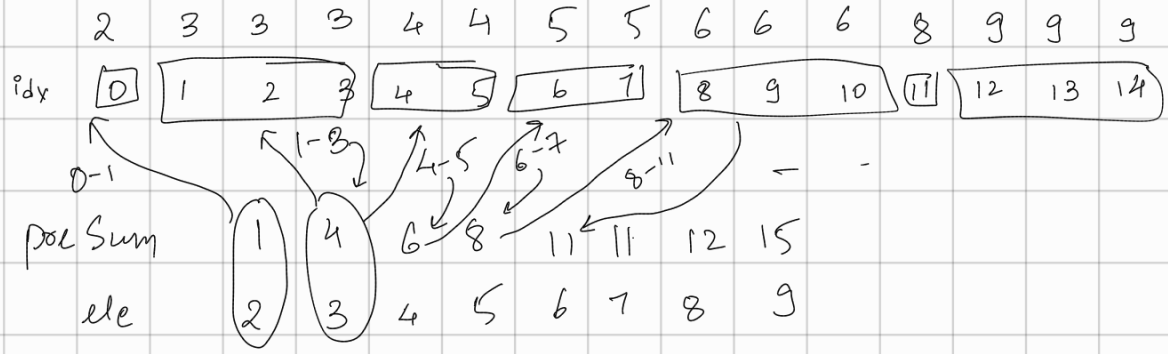
preSum	1	4	6	8	11	11	12	15
num	2	3	4	5	6	7	8	9

preSum

$$preSum[i] = \sum_{j=0}^{i} freq[j] = \begin{cases} freq[i] + preSum[i-1], & \forall i > 0 \\ freq[i], & i = 0 \end{cases}$$

Prefix Sum is a Dynamic Programming Concept

Sorted array



preSum[i] = last index of i in sorted array

Now check for stability,

original array

9' 6' 3' 5' 3'' 4' 3''' 9'' 6'' 4'' 6''' 5'' 8 2 9''''

4th Traversal for sorting, right to left

freq arr

1	4	6	8	11	11	12	15
2	3	4	5	6	7	8	9

On encountering 9'''' → reduce preSum of 9 by 1 → 15 - 1 = 14
& place 9th at (15-1) idx.

On encountering 2 → reduce preSum of 2 by 1 → 1 - 1 = 0
& place 2 at 0 idx

On encountering 8, place at 11

On encountering 5''' → reduce preSum of 5 by 1 → 8 - 1 = 7
place 5''' at 7

→ In this way count sort can be made stable because input order is maintained

first decrease posSum[ele] & then add value on new value of posSum[ele]

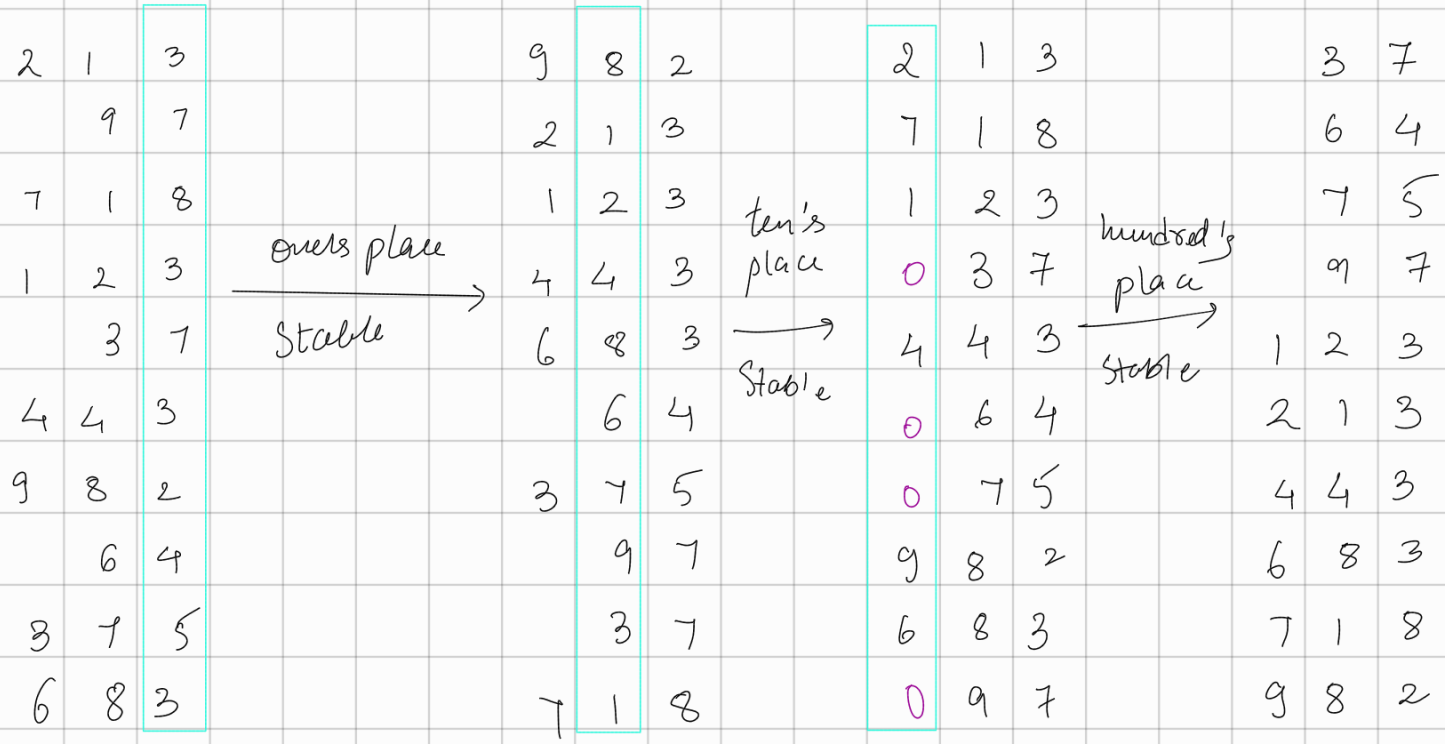
TC: O(n) , SC: O(max - min + 1)

② Radix Sort: {Count Sort}

* - Elements can be in any range

NO restriction on range of elements like count sort

Approach: apply count sort on places - ie - one's place, ten's place ...



Sorted

No. of digits in a num = $1 + \log_{10}(num)$

∴ TC of Radix Sort = $O(\log_{10}(max\ val) \times O(N))$

☆ Sort Dates — HW

D D M M Y Y Y Y

Similar to Radix Sort, Sort first on D D
then on M M
then on Y Y Y Y