```
int partition ( arr , s, e) {
        pivot = arr[e]
        i = s;

        for (j= s; j<=e; j++ ){
                    if (arr[j] < pivot) {
                            temp = arr[i];
                            arr[i] = arr[j];
                            arr[j] = temp;
                            i++;
                    }
        }
        // swap elements on index i & e

          temp = arr[i];
         arr[i]= arr[e]
        arr[e] = temp;

        return i;

}
```

Q// Given an array of size N
Sort the array.

$A[i] \Rightarrow \{0, 1, 2\}$ ✓

	0	1	2	3	4	5	6
A:	1	0	0	2	1	0	2

→

#count

Sorted A : 0, 0, 0, | 1, 1, 2, 2

# count \# count

Sol^n

1) Use any sorting algorithm.

T.C. = O(N log N)

2) Count of 0, 1 & 2

⇒ Count 0 : 3

⇒ Count 1 : 2

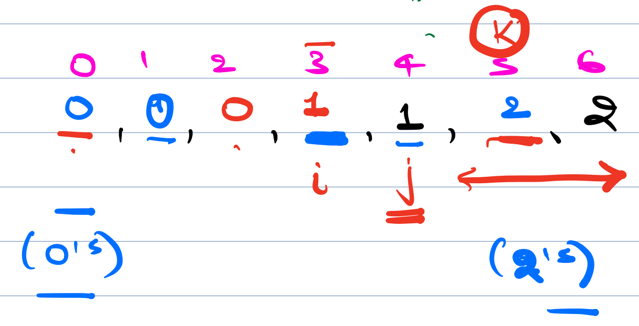
⇒ Count 2 : 2

0 0 0 1 1 2 2

T.C. = O(N + N) = O(N)

S.C. = ~~O(3)~~ O(1)

3)



Code

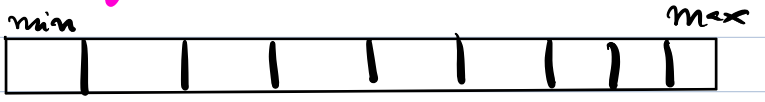
```
i = 0;
for (i = 0; i < N; i++) {
            if (A[i] != 0) {
                     break;
            }
}
```

```
int j = 0;
for ( j = N-1; j >= 0; j-- ) {
            if ( A[j] != 2) {
                    break;
            }
}

K = i;
while ( K <= j) {

    → if (A[K] == 0){
            swap( i, K);
            i++;
            K++;      continue;
      }
    → else if ( A[K] == 2){
            swap (j, K)
            j--;
            if (A[K] == 0) {
                    swap (i, K);
                    i++;
            }
            K++;        continue;
      }

      K++;
}
```

$0 \le A[i] \le 9$ (10)

→ traverse the array & create a

freq array.

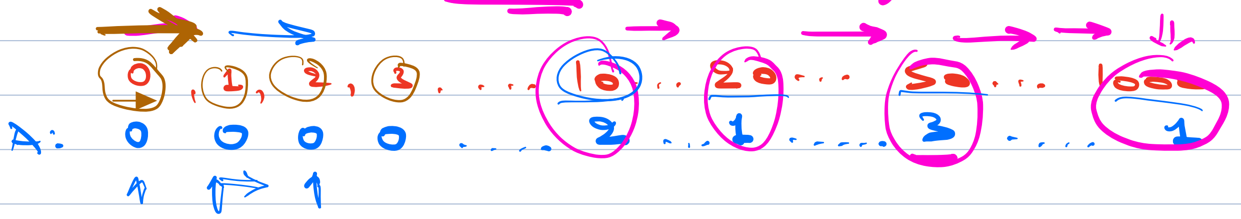


↑ O(1)   ⇑ O(Range)

O(N)

A : [50, 10, 50, 1000, 50, 20, 10]

O(1000)   O(Range)



10	10	20	50	50	150	1000
0	1	2	3	4	5	6

T.C. = O(N + Range)

HashMap →

10	3
20	1
50	2
1000	1

log N

TreeMap (Ordered Map) → log N

T.C. = O(N log N)

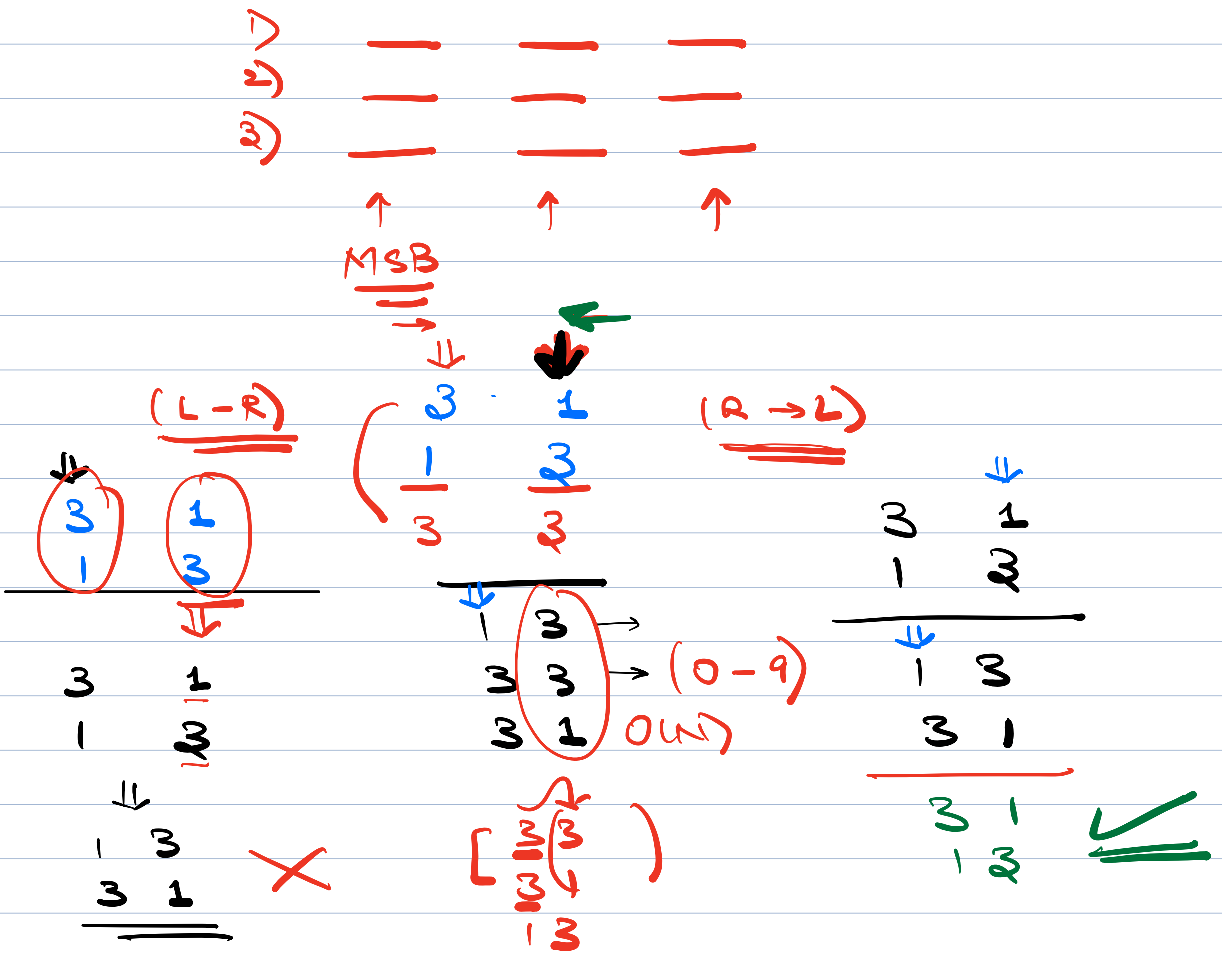
Count Sort

↳ Useful when range is small

(Range $<<$ N)

[0-9]

Raddix Sort



N Elements in the array &
Each element has D digits

T.C. = $O(N \times D)$

Integers ⇒ $\simeq \underline{10^9}$ (10 digits)

T.C. = $O(\underline{N \times 10}) = \underline{O(N)}$

$\underline{\underline{Eg}}$ ⇒

```
2  1  0
2  1  0
1  3  6
      7
   5  2
1  5  1
   5  7
```

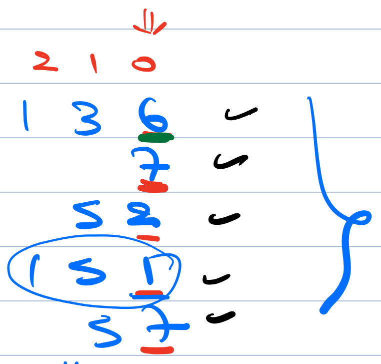
$$\begin{array}{r} 13 \\ 10\,)\,\overline{13\boxed{6}} \\ -130 \\ \hline \boxed{\underline{6}} \end{array}$$

$136 \rightarrow \underline{\underline{13}}$

(Ascending)

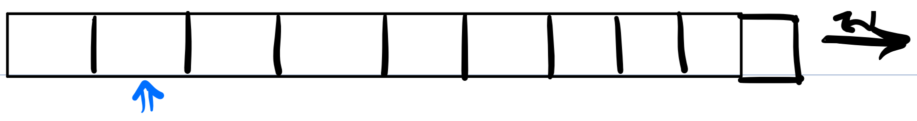
x ⇒ 1) last digit of x → $\underline{x \,\%\, 10}$
2) remaining no after removing last digit ⇒ $\underline{x/10}$

$(x \,\%\, \underline{10^1}) / \underline{10^0}$



0 1 2 3 4 5 6 7 8 9)

freq

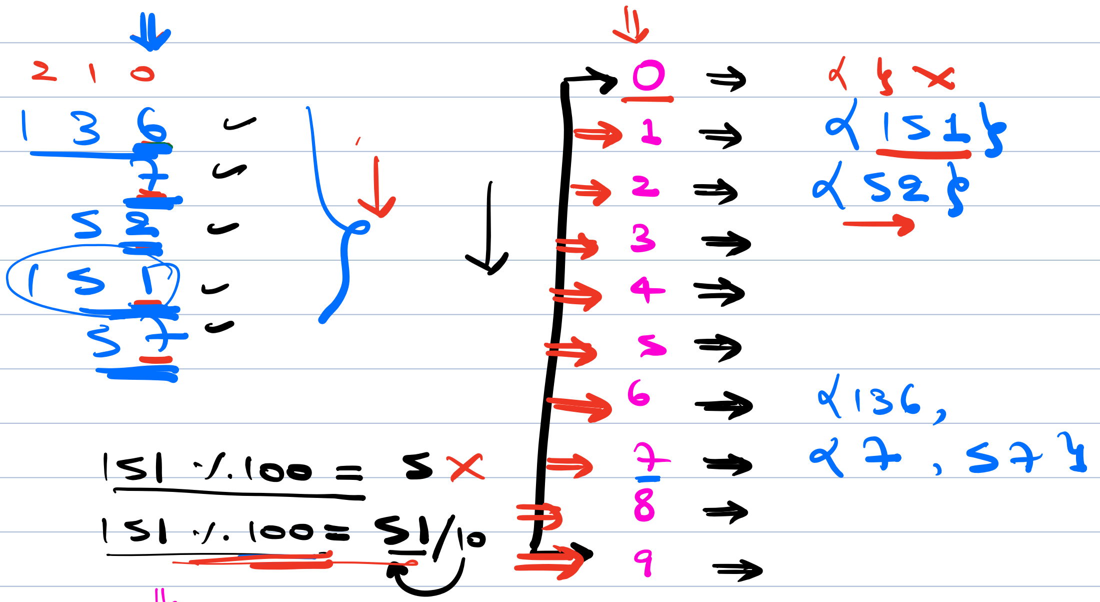


A : _ _ _ _ _

Array < Int > ×

Array < List >

$\Uparrow$ contain all element with a given last digit



$151 \% 100 = 5$ ×

$151 \% 100 = 51/10$

151 52 136 7 57

$7 \% 100 = 7/10 = 0$

$(x \% 10^2)/10^1$

0 ⇒ {7,

× 1

× 2

3 ⇒ {136

× 4

5 ⇒ {151, 52, 57}

x 6
x 7
x 8
x 9

$1136 \% 100 = \underline{\underline{36}}$

A: 7, 136, 151, 52, 57

$(x \% 1000) / 100$

$(x \% 10^3) / 10^2$

D^{th} digit $\Rightarrow (x \% 10^d) / 10^{d-1}$

0 → {7, 52, 57}
1 → {136, 151}
2
3
4
5
6
7
8
9

7, 52, 57, 136, 151

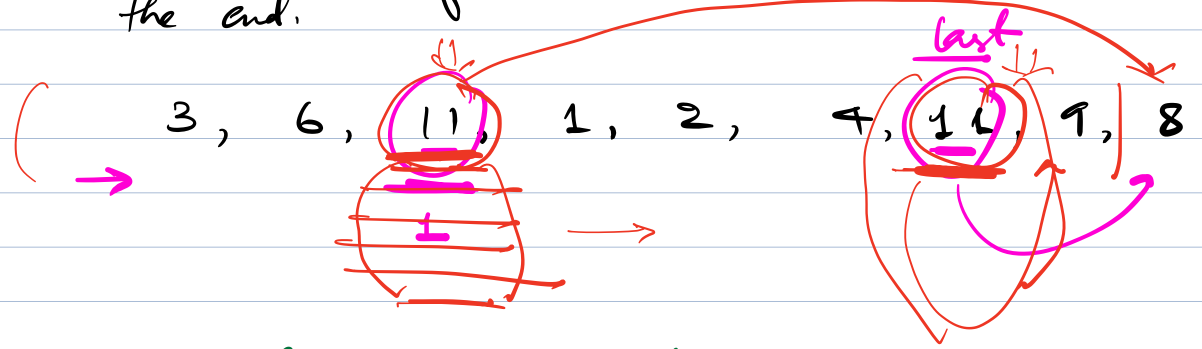
T.C. = $\underline{\underline{O(N)}}$

Code : $\boxed{\text{H.W.}}$ ✓

Stability

1) Selection Sort

Iterate, find max & send to the end.



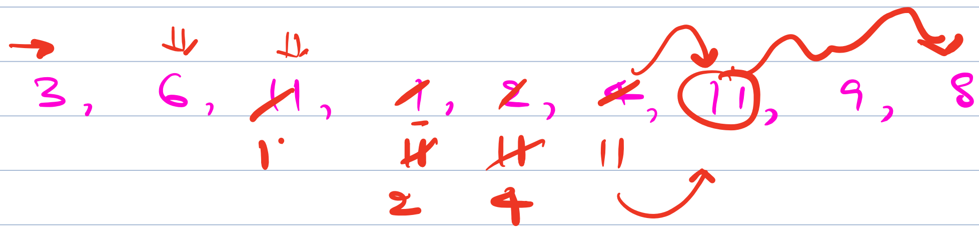
⇒ Selecting the last max every time

2) Bubble Sort

N-1 Iterations

if (A[i+1] < A[i]){ swap.

}



3) Insertion Sort

3 j i-1

1, 2, 3, ~~7~~ 11, ~~9~~ 7, ~~3~~ 9

temp = 3

Already stable

4) Merge Sort

i

| 6, 11, 13, 21 |

j

| 6, 9, 16, 18 |

_ _ _ _ _ _ _ _

```
if (A1[i] <= A2[j]) {
        select i;
}
else {
        select j;
}
```

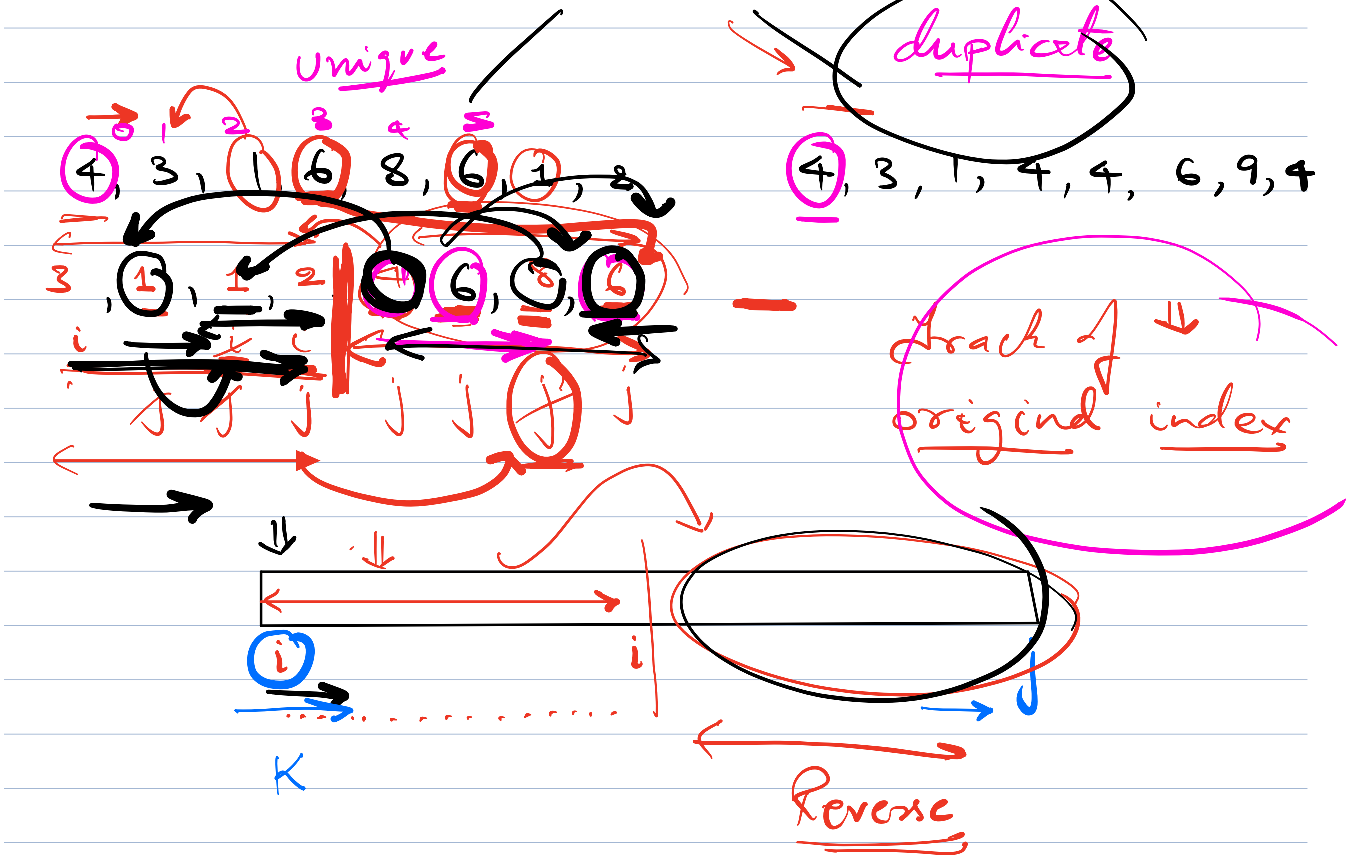
5) Count Sort

6) Radix Sort

⇒ Already stable

7) Quick Sort

H.W.



unique
duplicate
4, 3, 1, 6, 8, 6, 1, 2
3, 1, 1, 2, 4, 6, 8, 6
4, 3, 1, 4, 4, 6, 9, 4
Track of original index
i
j
K
Reverse