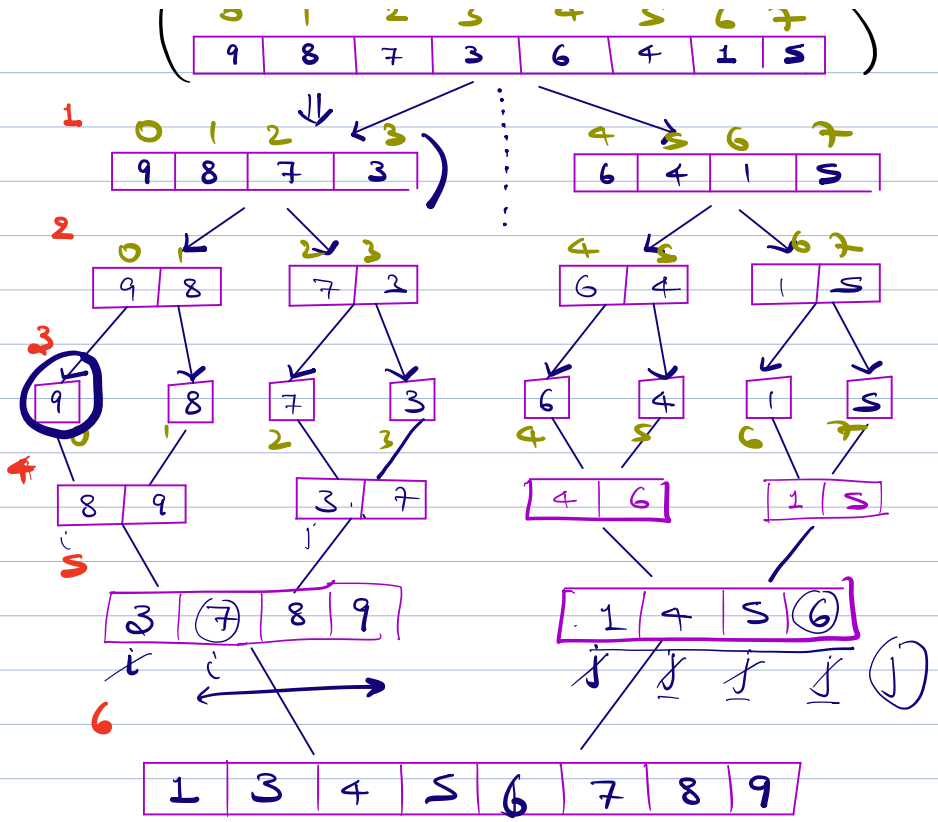
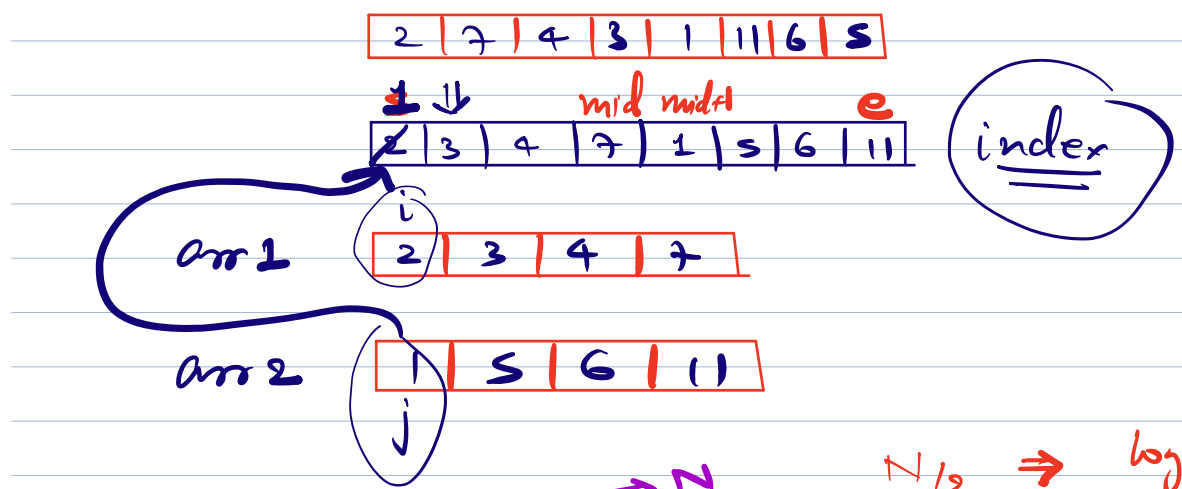


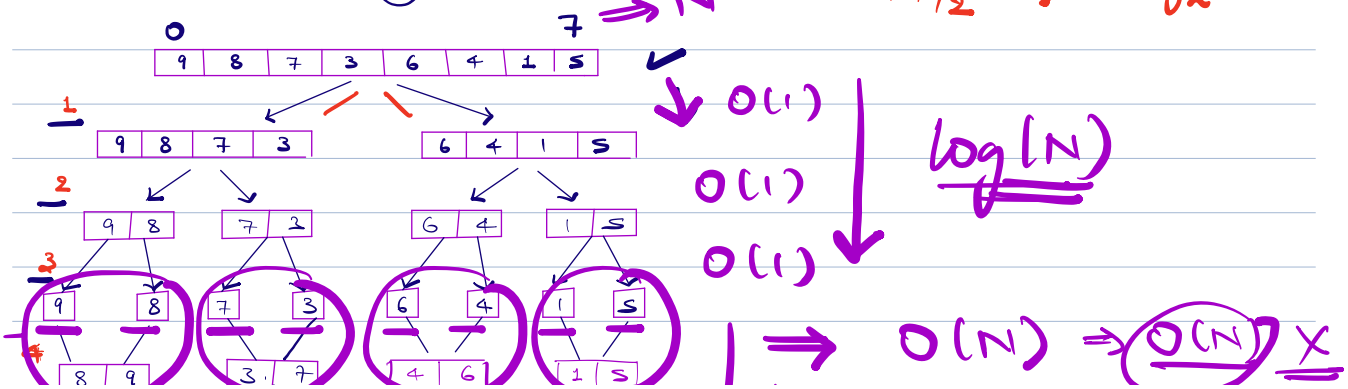
Code

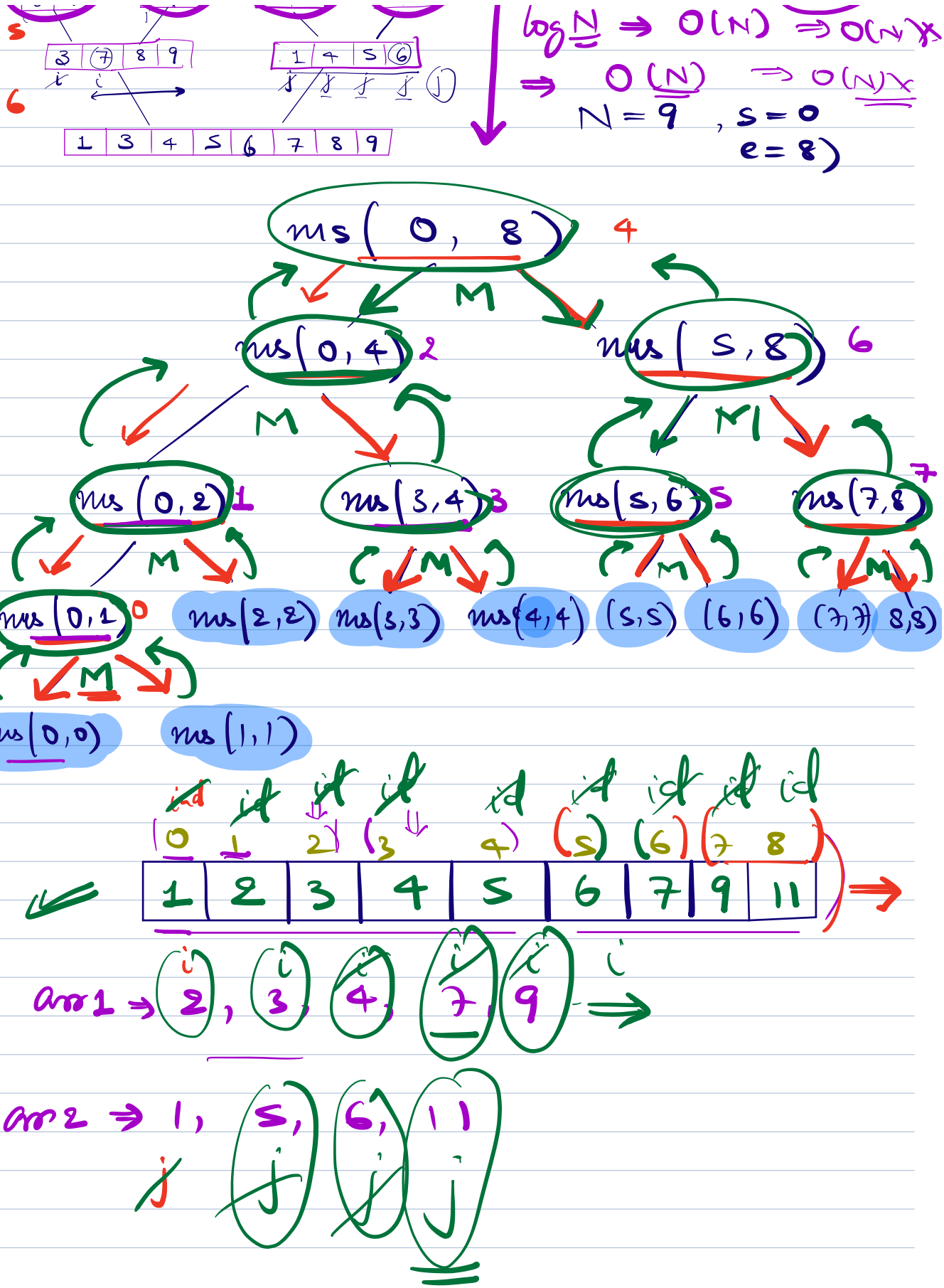
```
void MergeSort ( arr, s, e )          // s: 0, e: N-1, N
{
    if (s == e) { return; }
    mid = (s+e)/2;
    MergeSort ( arr, s, mid );        // N/2
    MergeSort ( arr, mid+1, e);       // N/2
    merge ( arr, s, mid, e);          => DFS
}
```

```
merge ( arr, s, mid, e) {

// arr1 ⇒ s to mid
   n1 ⇒ mid - s + 1 ;
  ⇒ arr1 [n1];
// arr2 ⇒ mid+1, e
   n2 ⇒ e - (mid+1) + 1 ;
   arr2 [n2];
  index = 0;
  for (i = s; i <= mid ; i++)

      arr1 [index] = arr[i];
       index ++;
}
 index = 0;
 for (i = mid+1; i <= e; i++) {

      arr2[index] = arr [i];
       index ++;
}

int i = 0, j = 0;
index = s ⇒ starting of the segment
while (i < n1 && j < n2) {

   if (arr1 [i] <= arr2[j]) {

      arr[index] = arr1 [i];
       index ++;
       i++;

}
else {
```

```
        arr[index] = arr2[j];
        index++;
        j++;
    }
}
while (i < n1) {
    arr[index] = arr1[i];
    index++;
    i++;
}
while (j < n2) {
    arr[index] = arr2[j];
    index++;
    j++;
}
}
```






3 7 8 9
1 4 5 6
1 3 4 5 6 7 8 9
log N ⇒ O(N) ⇒ O(N)*
⇒ O(N) ⇒ O(N)*
N = 9 , s = 0
e = 8)
ms(0,8)
ms(0,4)
ms(5,8)
ms(0,2)
ms(3,4)
ms(5,6)
ms(7,8)
ms(0,1)
ms(2,2)
ms(3,3)
ms(4,4)
(5,5)
(6,6)
(7,7)
8,8)
ms(0,0)
ms(1,1)
M
0 1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 9 11
arr1 ⇒ 2, 3, 4, 7, 9
arr2 ⇒ 1, 5, 6, 11
j

$$T.C. = O(1 \times \log N + N \times \log N)$$

$$= O(N \log N)$$

$$S.C. = O(N)$$

$$T(N) = 2T\left(\frac{N}{2}\right) + N \Rightarrow$$

H.W. will splitting in 3. Improves T.C.

Given an array of size N.

Return the inversion count of the array.

(i, j) s.t. $i < j$ & $a[i] > a[j]$.

Eg → 4, 5, 1, 2, 6, 3

indices: 0, 1, 2, 3, 4, 5

i: 0, 1, 2

j: 2, 3, 5 ⇒ 3

2, 3, 5 ⇒ 3

× ⇒ 0

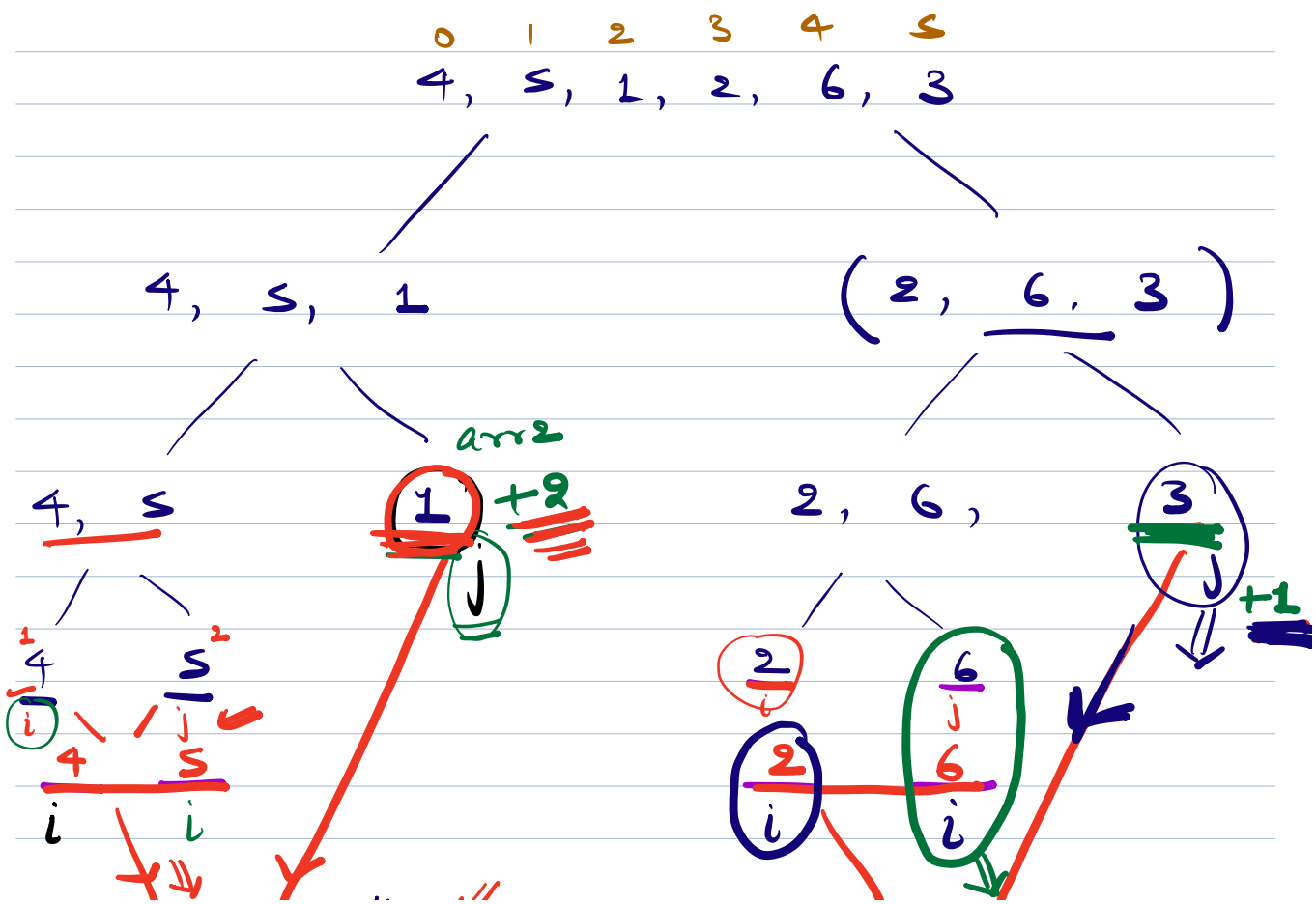
3 × 0
4 5 1
7

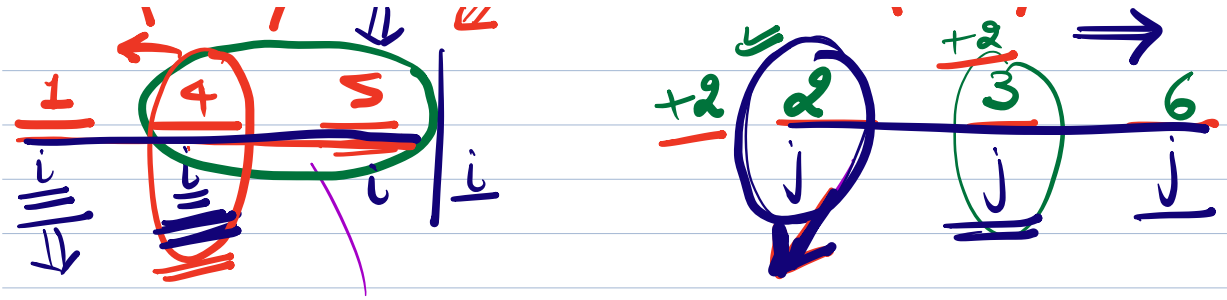
1) Brute force

T. C. = $O(n^2)$

(2) Optimised approach

Hint : Merge step of merge sort





1 2 3 4 5 6

i to (n1-1)

$(n1-1)-i+1 = n1-i$

i n1-1

1, 4, 5 → 3

Lock & Key Company.

L1 - 5 L2 - 7 L3 - 3

K1 - 5 K2 - 7 K3 - 3

	0	1	2	3	4	5	6
L →	5,	3,	1,	2,	7,	6,	11
K →	1,	6,	5,	7,	11,	3,	2

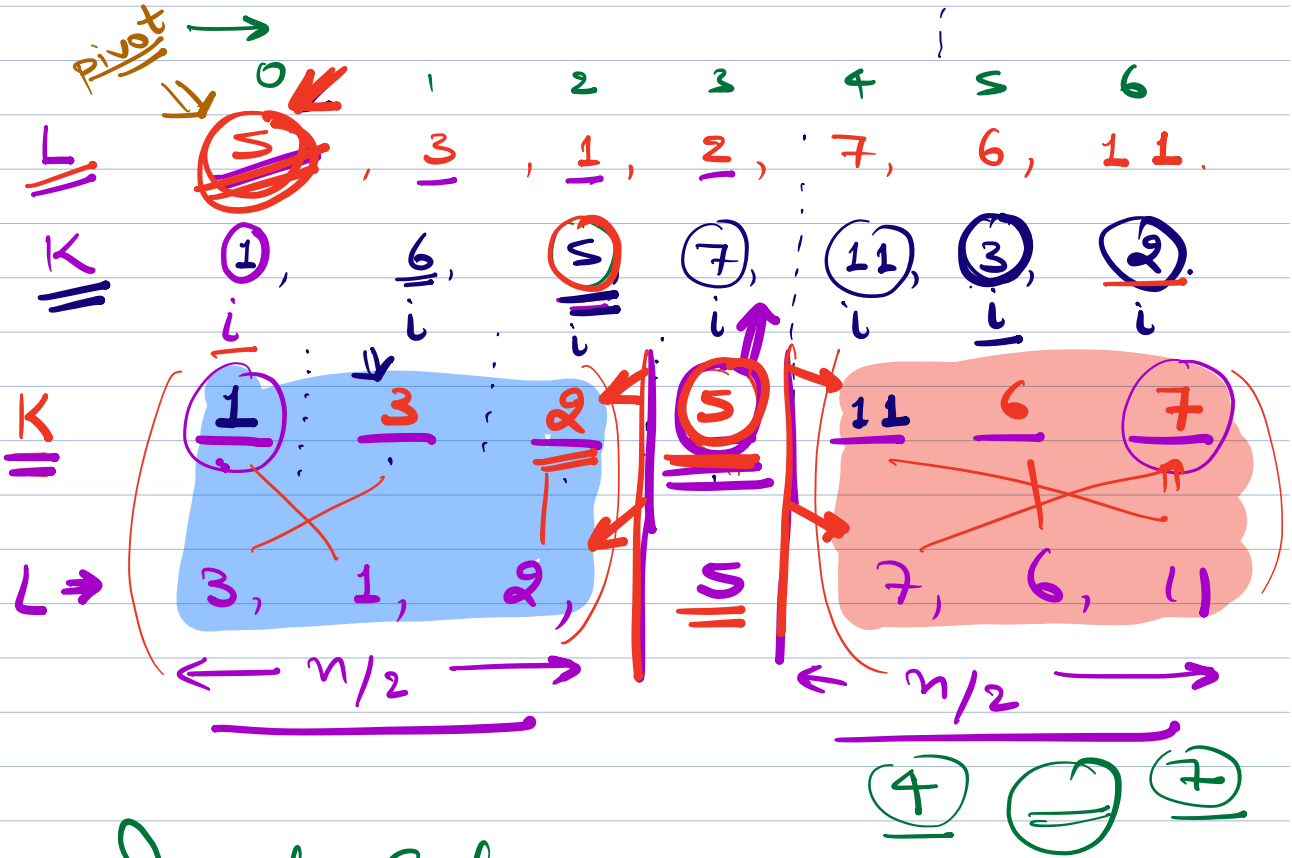
① Brute force Index Map

Shreya

T.C. = $O(N^2)$

L Ind	K Ind
0	2
1	5
2	0
3	6

Extra Space



⇒ Quick Sort

1) Select a pivot ⇒
2) Partition the array acc to pivot
3) Call quicksort recursively on both segments!

Pivot

1) 1st element.
2) Last element
3) Median ⇒
4) Random pivot

rand ( )

$T(N) = 2T(N/2) + N$

Best T.C. = $O(N \log N)$

Eg → 11 2, 3, 5, 6, 4, 7, 1

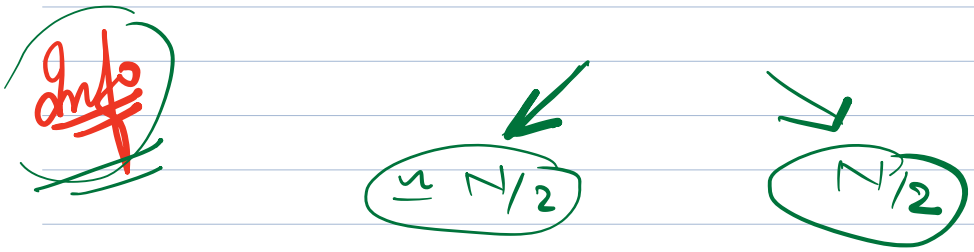
N-1

2, 3, 5, 6, 4, 7, 1, 11

N-1

$T(N) = T(N-1) + N$

T.C. = $O(N^2)$



Code

```
void quickSort(arr, s, e) {
    // Base Case
    pi = partition(arr, s, e);
    quickSort(arr, s, pi-1);
    quickSort(arr, pi+1, e);
}
```

H.W

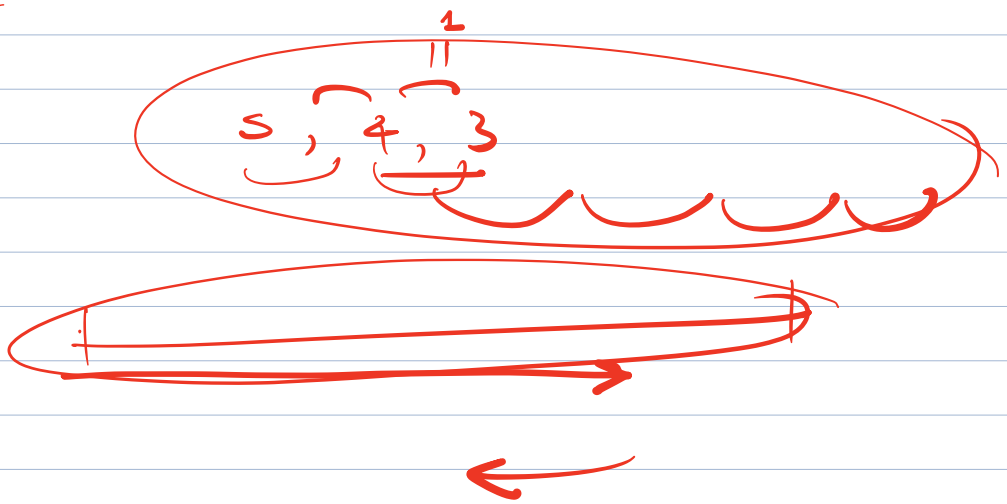
partition() ⇒ 1) divide the array into 2 parts.
2) Return the index of the pivot after partition.

1st element as pivot

0 1 2 3 4 5

4, 5, 1, 2, 6, 3

Doubt



$2^{31} - 1$

$\log_2 2^{31} = 31$

XOR $\simeq$ 32

N log N $\simeq$ 31 log 31