A = [5, 8, 1, 15, 7, 6, 2]

Q → Find max element → TC = O(N) SC = O(1)

```
ans = A[0]
for i → 1 to (N-1)
    if (A[i] > ans)
        ans = A[i]
return ans
```

Q → Find second max element → TC = O(2N) ≈ O(N) SC = O(2) = O(1)

Q → Find third max element → TC = O(3N) = O(N) SC = O(3) = O(1)

⋮

Q → Find K^{th} max element → TC = O(KN) SC = O(K) → O(1)

A = [5, 8, 1, 15, 7, 6, 2] K = 4

	0	1	2	3	4	5	6
K=1			~~2~~			15	
K=2		6			8		
K=3							
K=4	5	2	~~1~~	6	7	8	15
K=5	1		5				
K=6	1	2	5	6	7	8	15

K = N-1

Selection Sort

TC = O(N²) SC = O(1)

```
for i → N-1 to 1
    maxId = 0
    for j → 1 to i
        if (A[j] > A[maxId])
            maxId = j
    swap(A[i], A[maxId])
```

A = [5, 8, 1, 15, 7, 6, 2]
0 1 2 3 4 5 6

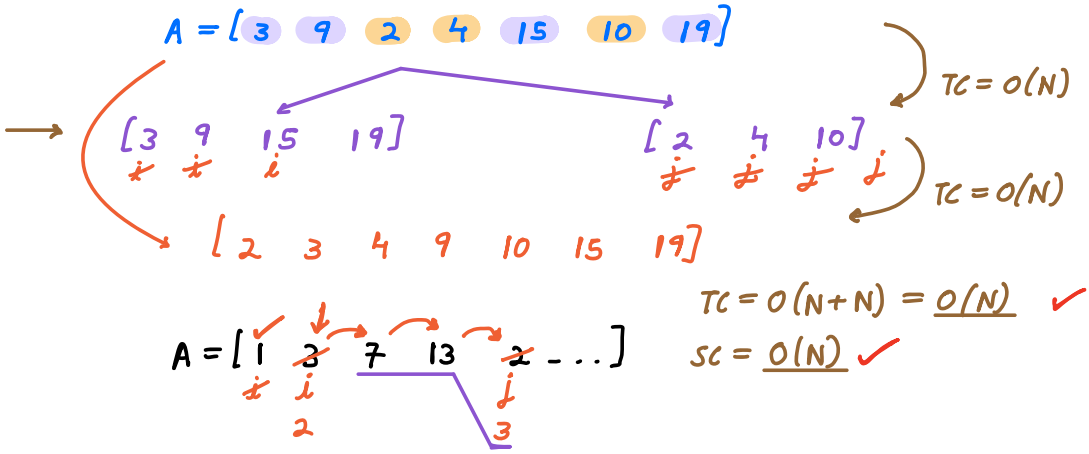
i = N-1

maxId = ~~0~~ 3 j → 1 to i

Q → Given an integer array where all odd elements are sorted & all even elements are sorted. Sort the array.

A = [3 9 2 4 15 10 19]

[2 3 4 9 10 15 19]



A = [1 3 7 13 2 ...]

TC = O(N+N) = O(N) ✓

SC = O(N) ✓

✓ Merge two sorted array into one. ✓

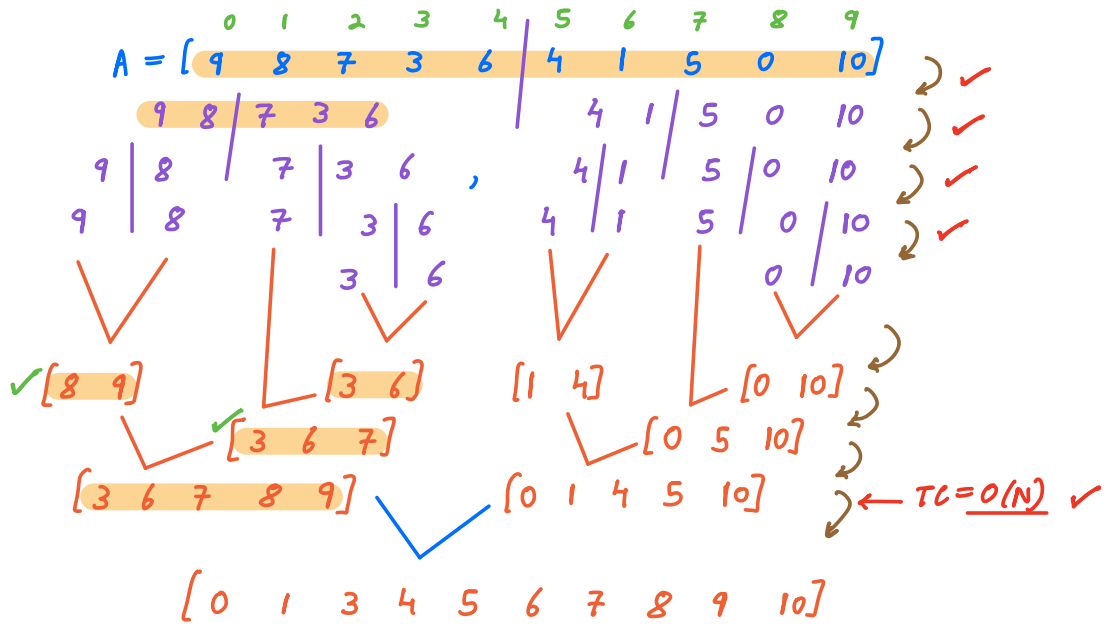
// A[N+M] ← I/P → B[N] ✓ (st — mid) C[M] ✓ (mid+1 — end)

```
i = 0 // B      j = 0 // C
for k → 0 to (N+M-1)
    if (i == N)
        A[k] = C[j] ✓
        j += 1 ✓
    else if (j == M || B[i] <= C[j])
        A[k] = B[i]
        i += 1
    else
        A[k] = C[j]
        j += 1      // ans += (N-i)
```

B = [0 2 5] (i = N) C = [4 6 10]

A = [0 2 4 5 6 10]

Merge Sort → Divide & Conquer




```
void sort (A, st, end) {          // st: 0, end: N-1
    if (st == end)
        return
    mid = (st + end) / 2
    sort (A, st, mid)
    sort (A, mid+1, end)
    merge (A, st, mid, end)       // TC = O(N)  SC = O(N)
}
```

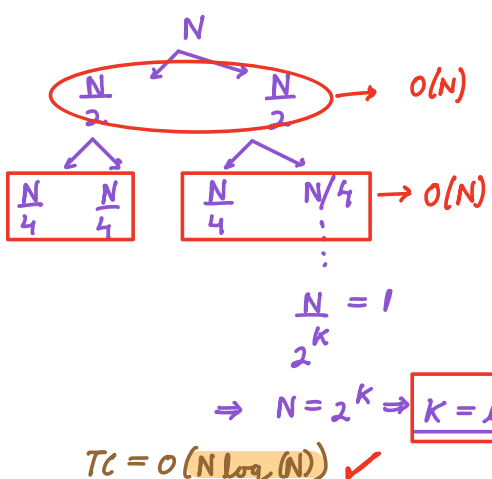
$T(N) = 2T(N/2) + O(N)$

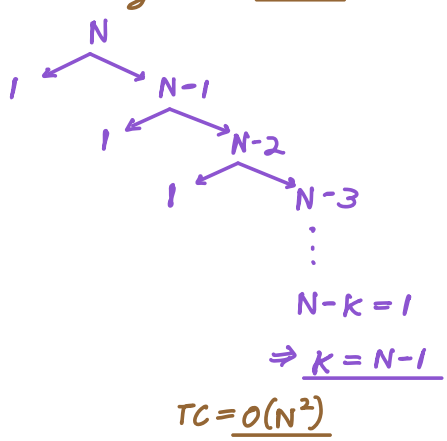
$a = 2 \quad b = 2 \quad d = 1$

$b^d = 2^1 = 2 = a$

$TC = O(N^d \log_b(N)) = O(N \log_2(N))$

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





10:40 PM

Q→ Given an integer array, count the number of inversion pairs in the array.

Inversion pair → (i, j)

$i < j$ and $A[i] > A[j]$

A = [8 3 4] (0,1) (0,2) Ans = 2

A = [4 5 1 2 6 3] (0,2) (0,3) (0,5) (1,2) (1,3) (1,5), (4,5)

Ans = 7 ✓

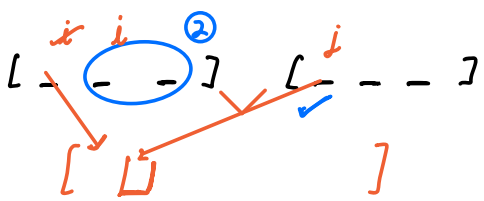
A = [4 4 4 4 4]

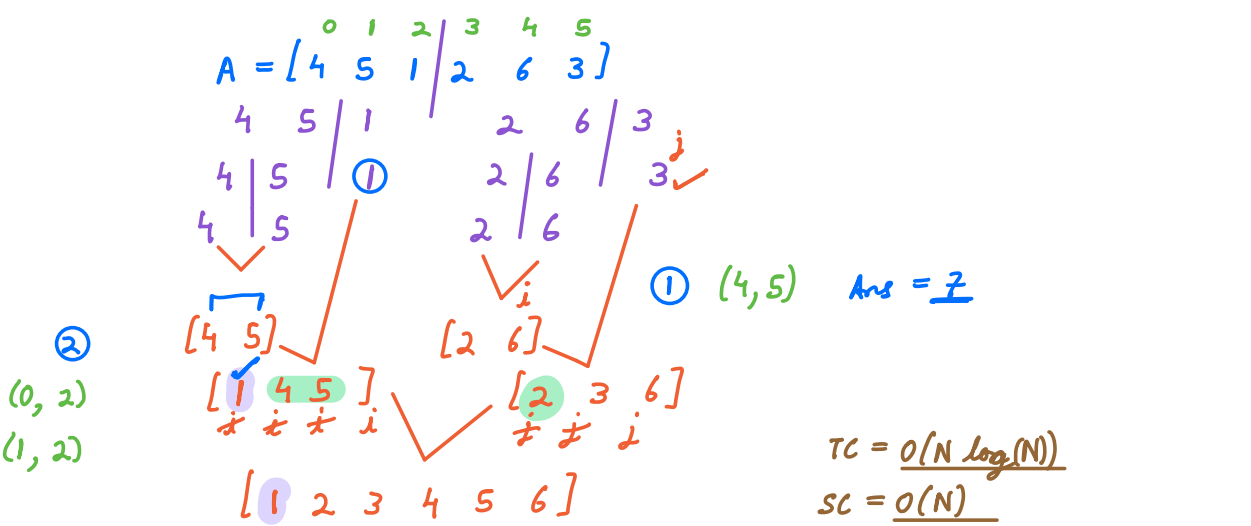
Ans = 0

Bruteforce → ∀i, j check for inversion.

TC = O(N²) SC = O(1)

Merge Sort





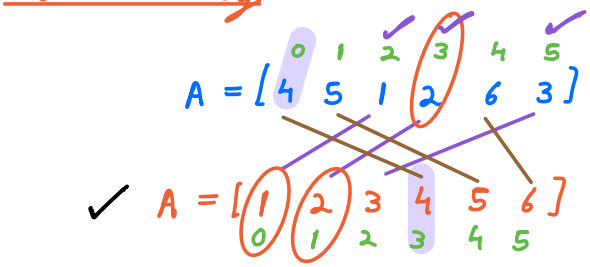
TC = O(N log(N))

SC = O(N)

② (0,3) (1,3) ② (0,5) (1,5)

Ans = Σ # remaining elements (N-i) in left part if an element from right part is selected. ✓

Sorted Array



sorted A — A

4 → 3 − 0 = 3
5 → 4 − 1 = 3
6 → 5 − 4 = 1
7

✓ # small on right.

(sorted index − original index)

A[] — sorted

1 → 2 − 0 = 2
2 → 3 − 1 = 2
3 → 5 − 2 = 3
7

large on left.

✓ (original index − sorted index)

4 → 0 − 3 = −3
5 → 1 − 4 = −3
6 → 4 − 5 = −1
−7

not recommended ✗

A = [✓ _ _ 1 _ _ _]

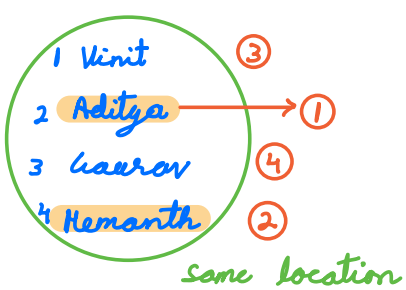
Sorted — 1

Stability → Relative order of equal elements should not change.

1	8
2	1
3	5
4	6
5	2
6	3
7	9
8	7
9	4
10	10

3, 4, 8 } equal

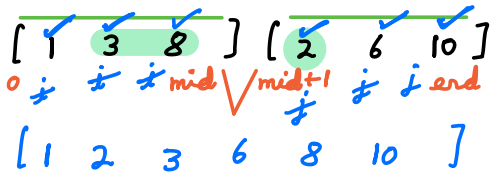
2, 5, 6, 9 } equal

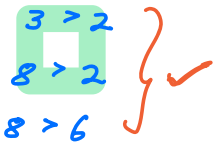


Stability in array → if elements are equal compare index.

---

left — right


[1 3 8] [2 6 10]
0 i i i mid mid+1 j i i end
[1 2 3 6 8 10]


3 > 2
8 > 2
8 > 6