Q.1 Given 2 Arrays A & B of sizes N & M. Count the no. of pairs i,j such that

A[i] > B[j].

A: $\{7, 3, 5\}$ $(A[i], B[j])$

B: $\{2, 0, 6\}$

(7,2)   (3,2)   (5,2)
(7,0)   (3,0)   (5,0) $\Rightarrow$ ⑦
(7,6)

Quiz

A: $\{3, 1, 6\}$

B: $\{2, 4, 9\}$

(3,2)
(6,2) $\Rightarrow$ ③ pairs.
(6,4)

Quiz

A: $\{\overset{0}{2}, \overset{1}{4}, \overset{2}{4}, \overset{3}{5}\}$

B: $\{\overset{0}{3}, \overset{1}{2}, \overset{2}{9}\}$

(4,3) | (4,2) | (5,3) $\Rightarrow$ 6
(4,2) | (4,3) | (5,2)

Brute force

$$TC: O(N \times M)$$

A: $\{7, 3, 5\}$ $\xrightarrow{\text{Sort}}$ $\{3, 5, 7\}$ ↑i

B: $\{2, 0, 6\}$ $\xrightarrow{\text{So}=}$ $\{0, 2, 6\}$ ↑j

B[j]	count
0	3
2	3
6	1

⑦

Ex

A: $\{7, 8, 2, 4\}$ $\xrightarrow{\text{SORT}}$ $\{2, 4, 7, 8\}$ N=4
                                   0 1 2 3
                                         ↑i   N-1

B: $\{3, 5, 1\}$ $\xrightarrow{\text{SORT}}$ $\{1, 3, 5\}$ M=3.
                                          ↑j

A[i] > B[j]

B[j]	Count
1	4
3	3
5	2

9

[i, N-1]

N-X-i+X

N-i

Code

```
Sort (A)  →  N log N
Sort (B)  →  M log M
i = 0
j = 0
count = 0
while ( i < N && j < M ) {
        if ( A[i] > B[j] ) {
                count += (N-i);
                j++;
        }
        else {
                i++;
        }
}
return count;
```

TC : $O(N \log N + M \log M)$

SC : $O(N + M)$

Q. ** Inversion Count

Given an Array of size N. Count the no. of pairs i, j s.t

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

A: { 10, 3, 8, 15, 6, 12, 2, 18, 7, 1 }
     0   1 2 3   4   5   6   7   8 9

(10,3)	(3,2)	(8,6)	(15,6)	(6,2)	(12,2)	(2,1)	(18,7)	(7,1)
(10,8)	(3,1)	(8,2)	(15,12)	(6,1)	(12,7)		(18,1)	
(10,6)		(8,7)	(15,2)		(12,1)			
(10,2)		(8,1)	(15,7)					
(10,7)			(15,1)					
(10,1)								

⇒ 26

Quiz A: [3, 1, 2]

(3,1)
(3,2)   ⇒ ②

Quiz   A: [8, 4, 2, 1]

(8,4)   (4,2)   (2,1)   ⇒ 6
(8,2)   (4,1)
(8,1)

Brute force

$$TC: O(N^2)$$
$$SC: O(1)$$

Observations

```
  0    1   2    3    4    5    6    7    8   9
{ 10,  3,  8,  15,  6,  12,  2,  18,  7,  1 }
```

A: { 10, 3, 8, 15, 6 }   B: { 12, 2, 18, 7, 1 }
     0   1 2   3 4   5   6   7   8 9

↓ Sort   ↓ Sort

```
  0  1   2  3   4              5  6  7   8   9
{ 3, 6,  8, 10, 15 }         { 1, 2, 7, 12, 18 }
                  ↑                          ↑
                  i                          j
```

count = 5 + 5 + 3 + 1 = (14)

Total pairs = Pairs in (A) + Pairs in (B) +

Pairs b/w (A) & (B)

Subproblem of size N/2

⇒ Recursion

Assumption :- Our fun^ will return the count of inversions in a subarray from s to e & also it will sort the subarray from s to e.

↑ 26

{10, 3, 8, 15, 6, 12, 2, 18, 7, 1}
  0   1 2   3   4   5   6   7   8 9

{1, 2, 3, 6, 7, 8, 10, 12, 15, 18}

26

{10, 3, 8, 15, 6} (5+7+14)   {12, 2, 18, 7, 1}
  0   1 2   3   4   5   6   7   8 9

C {3, 6, 8, 10, 15}   {1, 2, 7, 12, 18}

⑤   ⑦

{10, 3, 8}   {15, 6}   {12, 2, 18}   {7, 1}
  0   1 2   3   4   5   6   7   8 9
X {3, 8, 10}   r {6, 15}   L {2, 12, 18} R{1, 7}

    ②   ①   ①   ①

{10, 3} {8}   {15} {6}   {12, 2} {18}   {7}   {1}
 0   1   2   3   4   5   6   7   8   9
{3, 10} 0   0   0   {2, 12}   0   0   0

   ①   ①

A {10} {3} B   {12} {2}
  0   1   5   6
  0   0   0   0
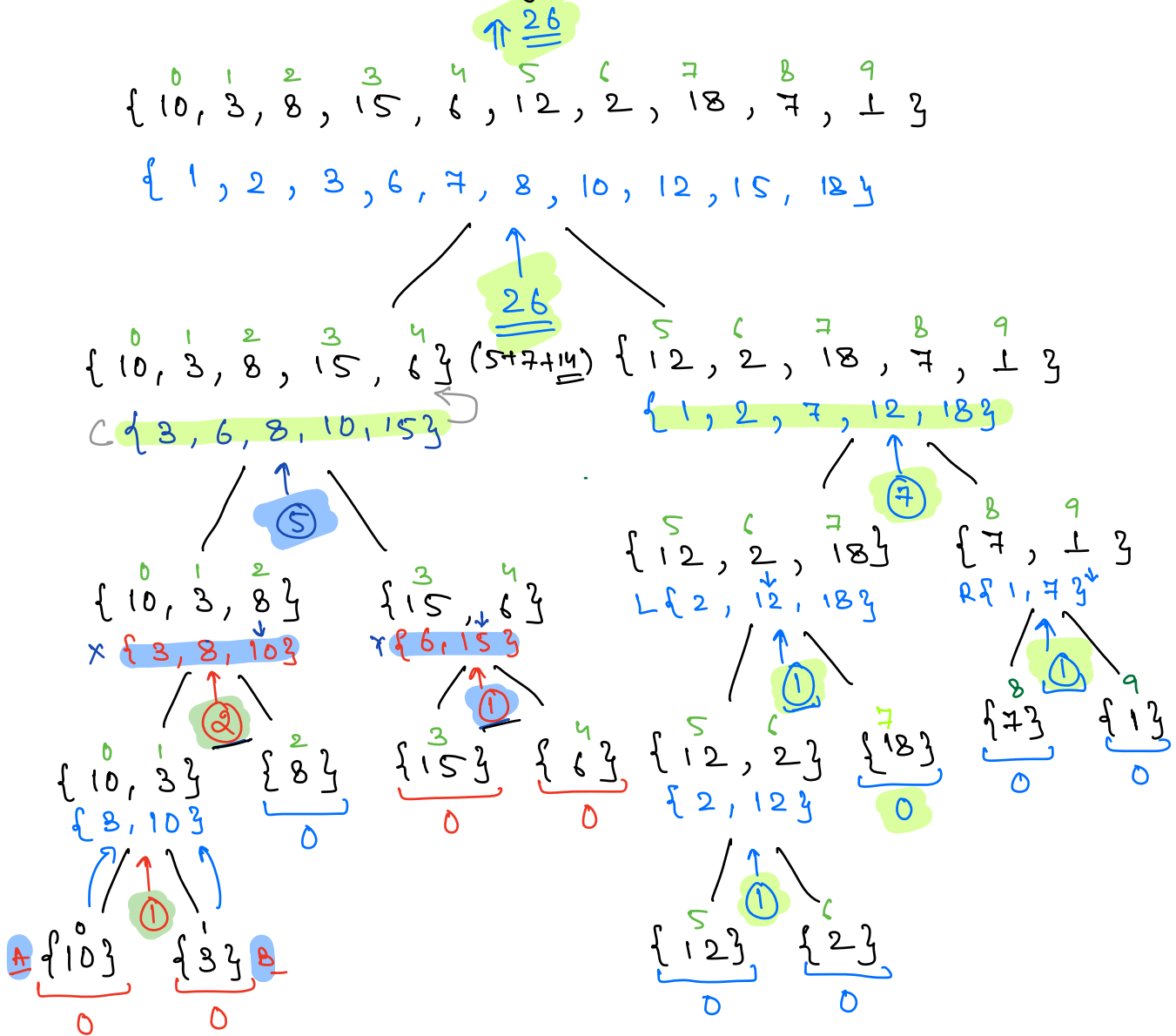
[i, mid]

```
int mergeSort (int A[], int s, int e) {
    if (s == e) return 0;
    mid = (s+e)/2;
    l = merge Sort (A, s, mid)
    r = merge Sort (A, mid+1, e);
    return l + r + merge(A, s, mid, e);
}

int merge (A[], s, m, e) {
    int C[e-s+1];
    count = 0
    i = s, j = m+1;
    k = 0;
    while (i <= m && j <= e) {
        if (A[i] > A[j]) {
            count += (m - i + 1);
            C[k] = A[j]
            j++, k++;
        }
        else {
            C[k] = A[i]
            i++, k++;
        }
    }
    while (i <= m) { C[k] = A[i], k++, i++ }
    while (j <= e) { C[k] = A[j], k++, j++; }
    for (i = 0; i <= e - s; i++) {
        A[s+i] = C[i]
    }
}
```

3   return count;

$$TC: O(N\log N) \quad \Big] \text{ Merge Sort}$$
$$SC: O(N)$$

Q: Given an Array of size N. Transform the
Google Array into Wave Pattern.
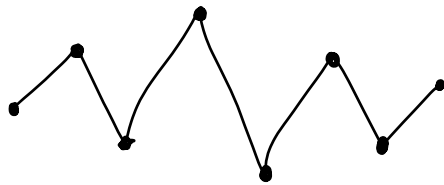
$A[0] > A[1]$: In wave form.

A : $\{1, 2, 3, 4\}$

o/p $\{1, 3, 2, 4\}$, $\{2, 1, 4, 3\}$, $\{1, 4, 2, 3\}$
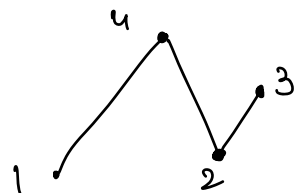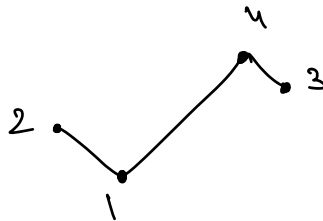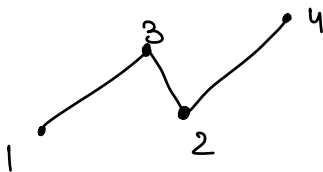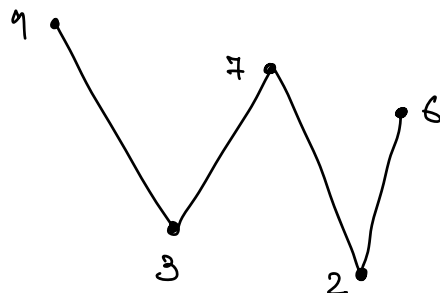

⇒ Return the lexiographically smallest ans.

Ex A : $\{3, 9, 7, 6, 2\}$

i) $\{9, 3, 7, 2, 6\}$
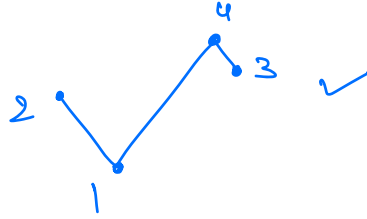
ii) $\{3, 2, 7, 6, 9\}$
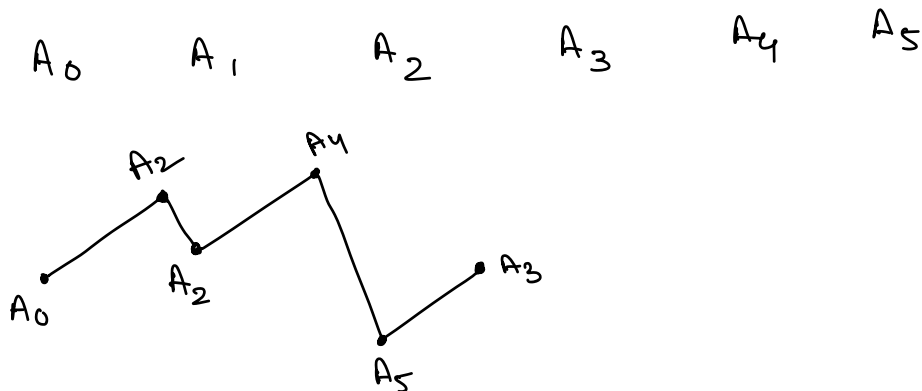
Steps

1) Sort (A)

A: { 1, 2, 3, 4 }
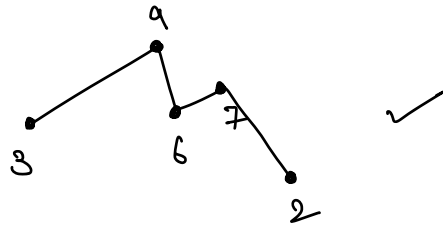
{ 2, 1, 4, 3 }

⇒ Lexicographically Smallest Ans.

2) Swap (A[i], A[i+1])

i += 2;

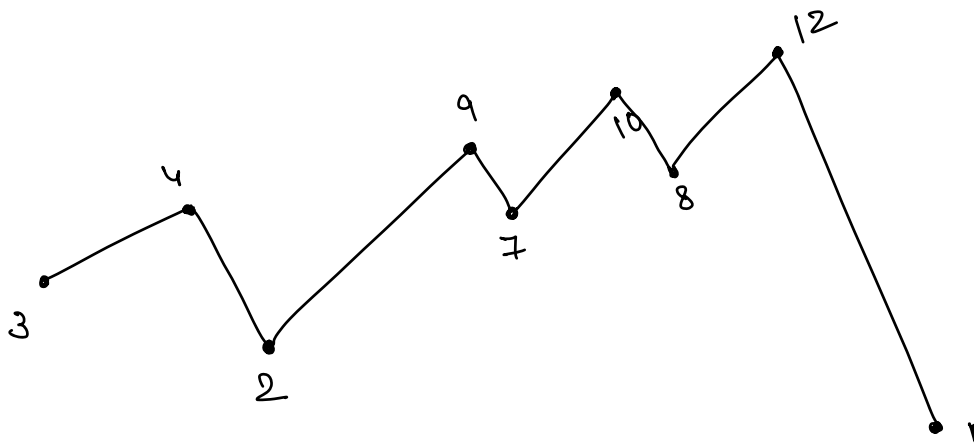TC : O(N log N)
SC : O(N)
] Merge Sort Algo.

* If we don't have to ans. the lexicographically Smallest ans.

A_0 A_1 A_2 A_3 A_4 A_5

A: { 3 , 9 , 6 , 7 , 2 }

9

7

3   6

2

A: { 3 , 4 , 2 , 9 , 8 , 7 , 10 , 12 , 1 }
                          7   8   8
                                   10

12

4   9   10

3   7   8

2   1

TC: O(N)

SC: O(1)

———————— * ————————

A :   { 3 , 9 , 7 , 6 , 2 }

{ 9 , 3 , ⑦ , ⑥ , 2 }

{ 9, 3 , 7 , 2 , 6 }
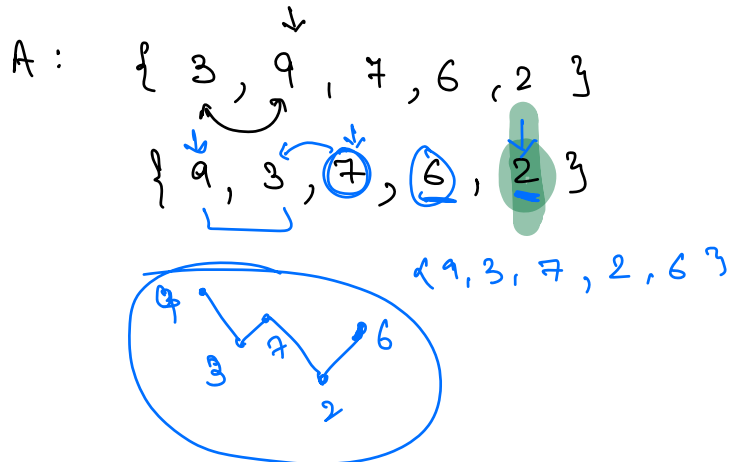
for (i = 0 ; i < N ; i += 2) {

A : { 3 , 9 , 7 , ⑥ , 8 } =) 3 , 9 , 6 , 7 , 8

3 , 9 , 6 8 7

A[0] >= A[1] <= A[2] >= A[3] - - - .

A[i-1] =< A[i] >= A[i+1]

A : { 3 , 6 , 9 , 2 , 7 }
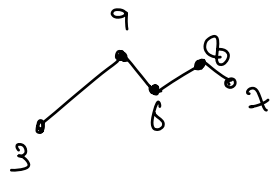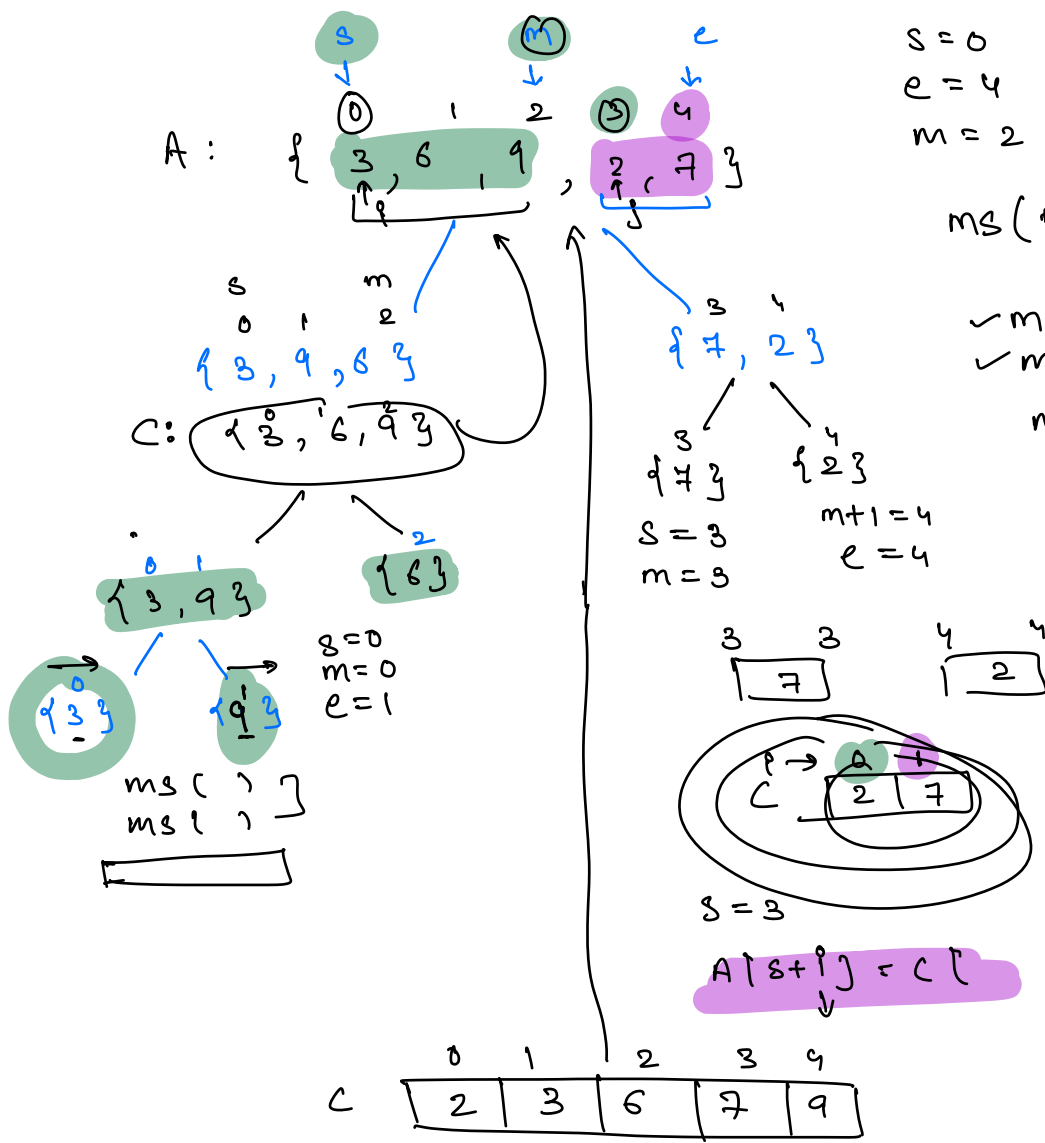
s = 0
e = 4
m = 2

ms(A, 3, 4) {

  ✓ ms(A, 3, 3)
  ✓ ms(A, 4, 4)
  merge(A, 3, 3, 4)

{ 3, 9, 6 }

C: { 3, 6, 9 }

{ 7, 2 }

{ 3, 9 }   { 6 }

{ 3 }   { 9 }

s = 0
m = 0
e = 1

ms ( )
ms ( )

{ 7 }   { 2 }

s = 3
m = 3

m+1 = 4
e = 4

| 3 3 |
| 7 |

| 4 4 |
| 2 |

s = 3

A[s + i] = C[

	0	1	2	3	4
C	2	3	6	7	9

A[i] = C[i]