Agenda :

1) Reverse Pairs
2) Maximum Unsorted Array
3) B closest points to Origin
4) Max & Min Magic

Question : Given 2 arrays A, B. Find the reverse pairs (i, j) such that $\underline{A[i] > 2 \cdot B[j]}$

Reverse Pair : (i, j)

i → Array A, j → Array B

	0	1	2	3
A =	15	13	2	25
B =	1	12	6	2

(15, 1) (15, 6) (15, 2) → 3

(13, 1) (13, 6) (13, 2) → 3

(25, 1) (25, 12) (25, 6) (25, 2) → 4

pairs = 3 + 3 + 4 = 10

Brute force :

```
count = 0
for(i = 0;  i < N;  i++) {
    for(j = 0;  j < M;  j++) {
        if( A[i] > 2·A[j]) {
            count++;
        }
    }
}
```

T.C: $O(N*M)$

S.C: $O(1)$

Approach2:

A : 15 13 2 25

B = 1 12 6 2

Sort both the array.

Sort(B) = [1 2 6 12]

$[l, r] = r - l + 1$

$[p1, N-1] \Rightarrow N - \cancel{1} - p1 + \cancel{1} = N - p1$

(2, 1) x

3 + 3 + 4

Count = 3 + 3 + 3 + 1 = $\boxed{10}$

p1

A = 2 13 15 25 ⇒ N = 4

0 1 2 3

B = 1 2 6 12 ⇒ M = 4

0 1 2 3

p2 p2

consider

(2, 1) x

2 ≯ 2*1

(13, 1)

13 > 2*1

15 > 2*1

25 > 2*1

((3,1) ((5,1) (25,1)

((3,2) ((5,2)(25,2)

((3,12)

(13,6)

N − P1 = 4 − 1 = 3

(N − P1)

4 − 3 = 1

(15,12)

(25,12)

(13,1)

(13,1)

n − 1

p1 p1 p1

A = 2 13 15 25

p2 p2 p2 p2

B = 1 2 6 12

Count = 0 + 3 + 3 + 3 + 1 = $\boxed{10}$

0 n − 1

A

0 m − 1

B

N = 6

0 1 2 3 4 5

A → $\boxed{A[2]}$

A[p1] && A[p2]

P_1 [0, N − 1]

P_2 [0, M − 1]

N = 5, P_2 = 10

$\boxed{P_1 = 4, \ P_2 = 12}$

(p1 < N || P2 < M) {

4 < 5

T

p1 = 0, p2 = 0;

while $\boxed{(p1 < N \ \&\& \ p2 < M)}$ {

if (A[p1] > 2 · B[p2]) {

count += N − P1;

p2++;

} else {

p1++;

}

}

}

i → $\boxed{0 \to N-1}$ → N

j → 0 → M − 1 → M

(N + M) iterations

O(N+M)

Question:

Given an array A, we have to find no. of reverse pair

Reverse Pair: A pair (i, j) such $\boxed{i < j}$ and $A[i] > 2 * A[j]$

$(3 > 2 \times 1)$

A = 1 3 2 3 1

0 1 2 3 4

(3, 1) (3, 1)

1 4 3 4

(i, j)

$A[i]$

$(i < j)$

Ans = $\boxed{2}$

A[i], A[j] $i < j$ $A[i] > 2 * A[j]$

Exe:

0	1	2	3	4	5	6	7
15	13	2	25	1	12	6	2

(15, 2) (15, 1) (15, 6) (15, 2) => 4

(13, 2) (13, 1) (13, 6) (13, 2) => 4

A[i] = 2

(2, 25) (2, 1) (2, 12) (2, 6) (2, 2)

$2 \times 2 > 25$

2 > 1 × 2

2 > 24

(25, 1) (25, 12) (25, 6) (25, 2) 4

1

(12, 6) ×

(12, 2) ✓

(6, 2) → 1

pairs = 4 + 4 + 4 + 1 + 1 = 14

Brute Force :

$i \le 8$ $\quad i = 9 \quad i \le n-2$

i :

j : $i+$ to $n-1$

$(n-1)$

```
for (i=0; i<n; i++) {
    for (j = i+1; j<n; j++) {
        if( A[i] > 2·A[j] ) {
            count++;
        }
    }
}
return count
```

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

$N = 10$

$i = 9$

$j = 9+1 = 10$

Approach :

| # pairs = 14 |

A = 15 13 2 25 | 1 12 6 2

A_1 = 15 13 2 25 | A_2 = 1 12 6 2

N = 4 $\qquad$ M = 4

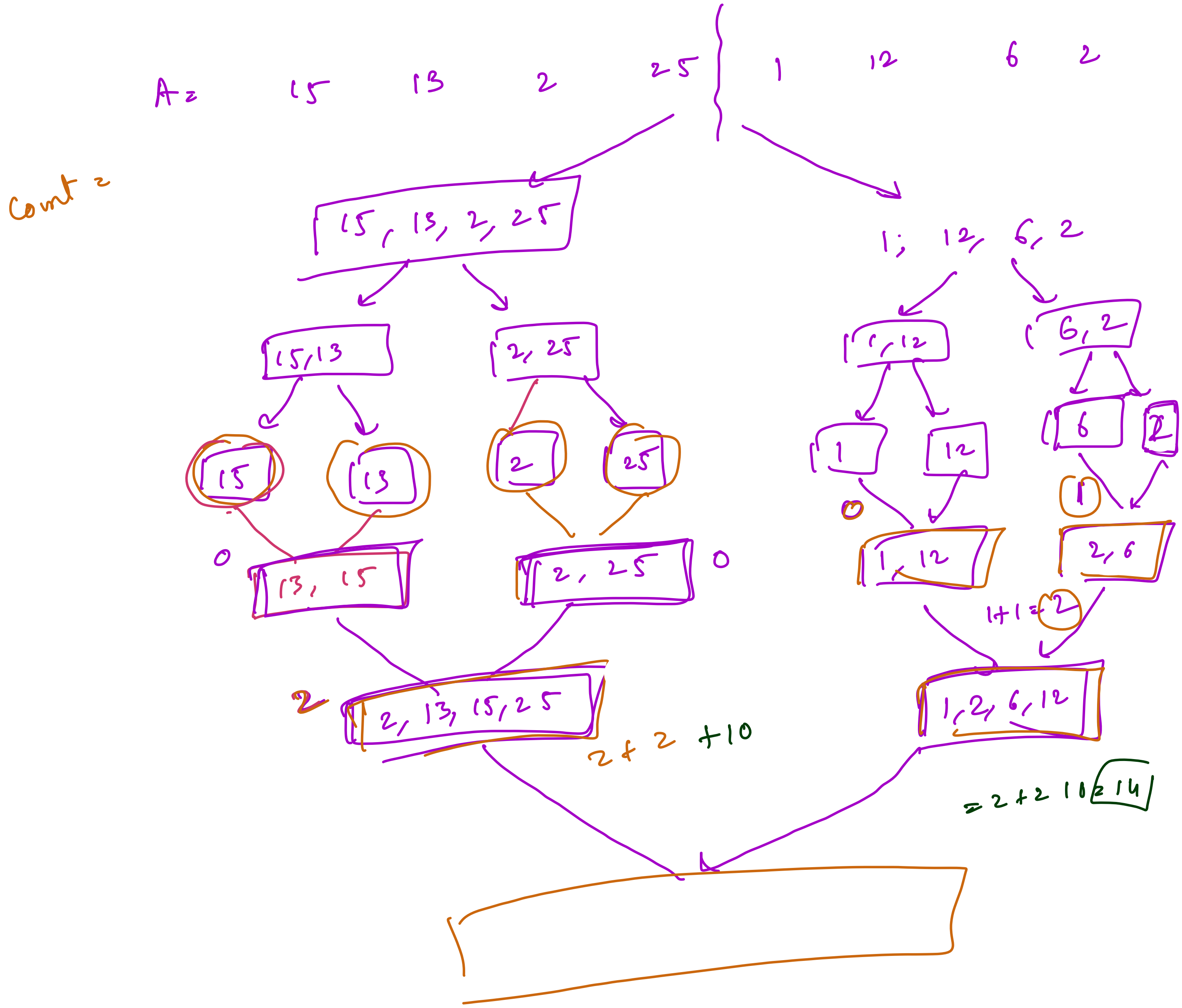
A_1 = 2 13 15 25 | A_2 = 1 2 6 12

P_1 $\quad P_2$

count = 3 + 3 + 3 + 1 = $\boxed{10}$

total pairs = | # Pairs in Left | + | # Pair in Right |

+ | # Pairs from 2 array |



Assumption : revPairs (l, r, A)
→ Sort the array & also return the count of reverse pairs

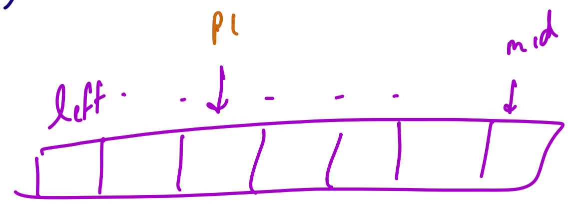
```
int mergeSort( int A[], int left, int right){
    if( right <= left) return 0;

    mid = (left + right)/2;
    count = 0
    count+= mergeSort( A, left, mid);
    count+= mergeSort( A, mid+1, right);
    p1= left,   p2= mid;
    while ( p1 <= mid && p2 <= right){
        if( A[p1] > 2*A[p2] ){
            count += mid - p1 + 1;
            p2++;
        }
        else{
            p1++;
        }
    }
    merge( A, left, mid, right);
    return count;
}
```

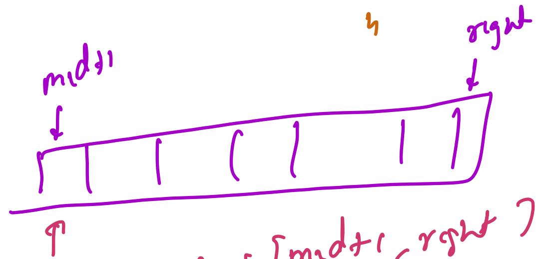
O(n)

O(n)

Max ( ) {

}





p1: [left, mid]

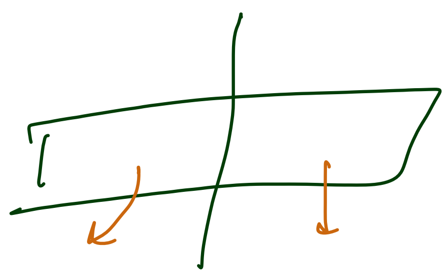
p2 : [mid+1, right]

[p1, mid]

[l, r] = r - l + 1

mid - p1+1

$\boxed{\text{T.C: } O(n \log n)}$



$\boxed{[0:23]}$

Question: Maximum Unsorted Subarray

Given an array of integers, Find the minimum length sub array [A_l A_{l+1} ..... A_r] such that, such that if this subarray is sorted, the entire array is sorted

Ex 1

A =

0	1	2	3	4	5
1	4	2	3	5	6

$\boxed{\text{Ans} = 3}$

1 2 3 4 5 6

Ex 2

A =

0	1	2	3	4
1	3	2	4	5

[1, 3] 1 2 3 4 5 ⟹ 3

[1, 2] 1 2 3 4 5 2, $\boxed{2}$

Ex 3

A =

0	1	2	3	4	5	6	7	8	9	10
10	12	20	30	25	40	32	31	35	50	60
10	12	20	25	30	31	32	35	40	50	60

Size = $\boxed{6}$

Exa

0	1	2	3	4	5	6	7	8	9	10
10	12	20	30	25	17	55	31	35	50	60

17 20 25 30 31 35 50 55

[2, 9]

Subarray size : $\boxed{8}$

9-2+1 =

First Last

Approach 1 :

	0	1	2	3	4	5
A =	1	4	3	2	5	6
A' =	1	2	3	4	5	6

Extra space

[first ..... last]

T.C:

1) Take a new array & sort (nlogn)
2) Iterate & get the L & R O(n)

$\boxed{\begin{array}{ll} T.C: & O(n\log n) \\ S.C: & O(n) \end{array}}$

$\boxed{A.C}$

Approach 2:

Lets consider $[A_L \dots A_R]$ is the minimum length subarray to be sorted to make the entire array sorted.

$A = A_0 \; A_1 \dots A_{L-1} \; \boxed{A_L \dots A_R} \; A_{R+1} \dots A_{N-1}$

Left $\underbrace{A[L \dots R]}_{\text{centre}}$ Right

A =	10	12	20	30	25	17	55	31	35	50	60
	0	1	2	3	4	5	6	7	8	9	10
	Left		L							R	Right

Observations

1) $\max(A_0 \dots A_{L-1}) \le \underbrace{\min(A_L \dots A_R)}_{(17)}$

2) $\max(A_L \dots A_R) \le \min(A_{R+1} \dots A_{N-1})$

A =	10	12	20	30	25	17	55	31	35	50	60
	0	1	2	3	4	5	6	7	8	9	10
			L	L				R		R	

55

Left

$\boxed{17}$ 25 30 31 55

min = 17

max = 55

→ All element in Left subarray ≤ 17

L = 2   R = 9

A = 10 11 20 30 25 11 40 37 32 31 85 50 60

0 1 2 3 4 5 6 7 8 9 10 11 12

L R

→ max(L...R) = 40

min(L...R) = 11

$\boxed{\text{Left Array} \le 11}$ ✓

Right Array ≥ 40

[1,10]

Step1: Find candidates of L, R (O(n)

Step2: Find min & max O(n)

Step3: Find Actual answer

```
for(i = 0; i < points.size(); i++){
    d = (point.x)^2 + (points.y)^2;
    map[d] = ( Insert Point[i])
}
```

T.C: O(n)

S.C: O(1)

Question: B closest points to origin

map[4] = [(0,-4)]

Map: 13 → [(2,-3), (3,2)]

B = 4

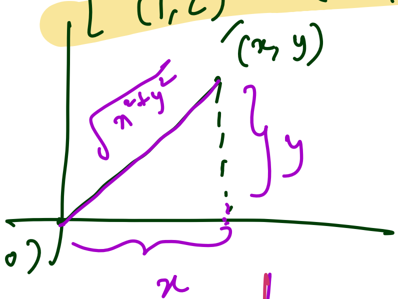
13

→ A: [(2,-3) (0,-4) (1,2) (3,1) (-2,2) (-3,-4)]

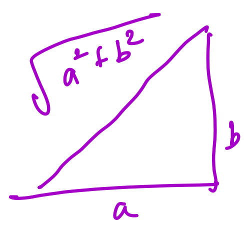
(3,2)

distance: [√13 √16 √5 √10 √8 √25]

Output: [(1,2) (-2,2) (3,1) (2,-3)]



x^2+y^2



$\boxed{\text{Sort(distance)}}$ = [5 8 10 13 16 25]

(4, 3) (4, -3)

(-4, 3) (-4, -3)

Map

5
10
1

13 : [(2, -3) . . .]
16 : [(0, -4) . . .]
5 : [(1, 2) . . .]
10 : [(3, 1) . . .]
8 : [(-2, 2) . . .]
25 : [(-3, -4) . . .]

[Key] → [Hashmap]

$\boxed{B=4}$

(1, 2) (-2, 2) (3, 1) (2, -3)

T.C : O(nlogn)

Step 1 : Create dist array & sort it O(nlogn)
Step 2 : Iterate the distance array & print the points [Map];

S.C : O(n) + O(n) ⇒ O(n)

O(n) → distance array

O(n) → Map

Approach 2 :

p1 (x1, y1)   p2 (x2, y2)

$\sqrt{x_1^2 + y_1^2} \leq \sqrt{x_2^2 + y_2^2}$

$\boxed{d_1 - d_2}$

$\boxed{\text{bool}}$ comp (Point p1, Point p2) {

$d_1 = (p_1.x)^2 + (p_1.y)^2$

$d_2 = (p_2.x)^2 + (p_2.y)^2$

if ($d_1 \leq d_2$){ return true; }

}

else return false; }

}

return $d_1 \leq d_2$

T.C: O(N log n)

S.C: O(1)

Question! Max & Min Magic

N is even

A = [1 2 3 4 5 6 7 8 9 10] N

5 elements { } { }

S1: $\frac{n}{2}$ eles

S2: $\frac{n}{2}$ eles

Magic No: Sum of absolute difference of corresponding eles of subset

A = [1, 2, 11, 3, 9, 7]
N = 6

(arrangement)

S1 = {1, 7, 9}

S2 = {3, 2, 11}

1 way

Magic: |3 - 1| + |2 - 7| + |11 - 9|

= 2 + 5 + 2 = $\boxed{9}$

Case 1: Maximum Magic ✓

Case 2: Minimum Magic

N = 4

Ex: 5 1 3 7

A1: {5, 1} {3, 7} → |5-3| + |1-7| = 8

A2: {5, 1} {7, 3} → |5-7| + |1-3| = 4

{5, 3} {1, 7} → 8

{5, 3} {7, 1} → 4

{5, 7} {1, 3} → 8

{5, 7} {3, 1} → 8

A = [1, 2, 3, 4, 5, 6]

Min Magic No

S1 = [1, 3, 5]

S2 = [2, 4, 6]

} → 3

Solution

N els

N ele

S1 = {$a_1, a_2, \ldots\ldots a_{N/2}$}

S2 = {$b_1, b_2 \ldots\ldots b_{N/2}$}

Magic No = $|b_1 - a_1| + |b_2 - a_2| + |b_3 - a_3| \ldots - |b_{N/2} - a_{N/2}|$

A = 8 9 1 4 3 6

A = 1 3 4 6 8 9

Minimize the sum

S1 = [1, 4, 8]

S2 = [3, 6, 9]

A = [5, 1, 3, 7]

$|3-1| = 2$

s_1 = 1 4 8

s_2 = 9 6 3

$= |8+2+5| = \boxed{15}$

A = 8 9 ~~1~~ ~~4~~ ~~3~~ ~~6~~

S1 = [1, 4, 8]

S2 = [3, 6, 9]

Ans = 2 + 2 + 1 = $\boxed{5}$

A = 1, 3, 4, 6, 8, 9

$\underbrace{1, 3}_{2} + \underbrace{4, 6}_{2} + \underbrace{8, 9}_{1} = \boxed{5}$

Maximum Magic

A = [5, 1, 3, 7]

[1, 7] [5, 3]

$\left(\frac{n}{2}\right)$ $\left(\frac{n}{2}\right)$

~~8~~ ~~9~~ ~~1~~ 4 ~~3~~ 6

S1 = [1 3 4]

S2 = [9 8 6]

$(9+8+6) - (1+3+4)$

$(9-1) + (8-3) + (6-4)$

$8 + 5 + 2 = \boxed{15}$

S1 = [1, 3, 4]

S2 = [6, 8, 9]

$(6-1) + (8-3) + (9-4)$

$= \boxed{15}$

Sort(A) = 1 3 4 6 8 9

S1 = [1, 8, 4]

S2 = [6, 3, 9]

$|6-1| + |3-8| + |9-4|$

Step 1 : Sort the array : O(nlogn)

Pseudocode : Max Unsorted Array

```
maxUnsortedArray(int A[]){
     int n = A.size();
     int i = 0;
     // We have to stop when A[i] > A[i+1]
     while(i< n - 1 and A[i] <= A[i + 1] i++;

     int j = n-1;
     // We have to stop when A[j-1] > A[j]
     while(j > 0 and A[j] >= A[j - 1])j--;

     if(i == n - 1){
          "ALREADY SORTED";
     }
     int min = A[i], maxm = A[i];
     for(int k = i; k <= j; k++){
          maxm = max(maxm, A[k]);
          minm = min(minm, A[k]);
     }
     int l = 0;
     while(A[l] <= mn and l <= i){
          l++;
     }
     r = n - 1
     while(A[r] >= mx and r >= j){
          r--;
     }

     return [l, r];
}
```