// Comparator Basics → In Java we return 1, 0, -1

```
bool comp( A, B){
    If A should before B return True
    If B should befor A return False
}
```

check syntax

Sort(A, A+N, comp)

// 1(2) Given N Array, sort them in decreasing order

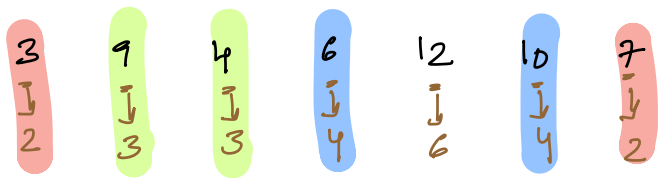
→ If func return True A come first
False B come first

```
bool comp( A, B){
    If( A > B) return True
    else return False
}
```

28) Given N Array elements sort them based on inc no. of factors. If 2 Elements have same factors one with greater value should come first.

Ass: fact(a) → Number of factors of A

Ex:



order: 7 3 9 4 10 6 12

Ex1: 8 12

fa 4 6

8 come first

Ex2: 10 9

4 3

9 come first

Ex3: 8 10

4 4

10 come first

```
bool comp(A, B){
    int ca = fact(A)
    int cb = fact(B)
    if(ca < cb){return True}
    else if(ca > cb){return False}
    else{
        if(A > B){return True}
        else{return False}
    }
}
```

28) Given N strings we can concatenate in any order get the maximum number.

Note: Numbers are given in string format

Idea: Sort them in decreasing order

Ex1: 33 90 39 → 90 + 39 + 33 = 903933

ans: 90 39 33

Ex2: 38 31 35 → 38+35+31 = 383531

ans: 38 35 31

Ex3: 9 91 89 → 91+89+9 = 91989 → 91989

9+91+89 = 99189 *

Ex4: 9 90 908 → 908+90+9 = 908909

9+90+908 = 990908 ✓

Ex5:

A	B
9	90

A	B
94	945

→ 94 945

→ 94 594

A	B
23	234

23 234

23423

A	B
90	908

90 908

90890

A	B
34	973

AB: 34973

BA: 97343

```
bool comp(String A, String B) {
    String x = A + B
    String y = B + A
    if( x > y) {return True, if A conc first value is more}
    else return False
}
```

String comparison
How to compare 2 strings

```
sort( data, comp)
// concatenate them
```

Ex: { 49 9 94 }

{ 9, 94, 49 }

// Concatenate them = 9 + 94 + 49 = 99449

-

18) Given 2 arrays A[N] & B[M] calculate no of pairs i, j such that A[i] > B[j]

Ex1: A[3] = 7 3 5 (0 1 2) B[3] = 2 0 6 (0 1 2)

{7, 2}
{7, 0}
{7, 6}
{3, 2}
{3, 0}
{5, 2}
{5, 0}

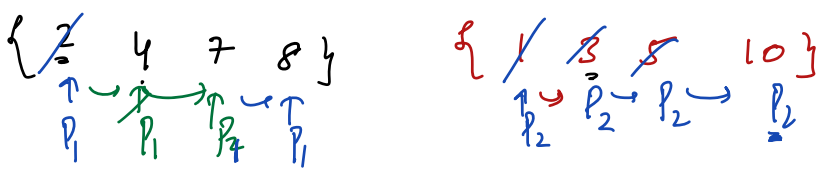
Idea: Check all pairs. → TC: O(N*M)

SC: O(1)

```
c = 0
i = 0; i < N; i++){
    j = 0; j < M; j++){
        if( A[i] > B[j]) {c++}
    }
}
```

Ex2: A[4] = 7 8 2 4 B[4] = 3 5 1 10

↳ {2 4 7 8} {1 3 5 10}



1	2	3	4	5	7	8	10	
4	0	3	0	2	0	0	0	→ {9}

cnt =

all index from [P_1, N), N Excluded

A[4] 8 5 3 10 ↑ B[4] = 6 4 9 2

P_1 N

0 1 2 3

3 5 8 10

P_1 P_1 P_1

0 1 2 3

2 4 6 7

P_2 P_2 P_2

2	3	4	5	6	8	9	10

cut = +4 | 0 | +3 | +0 | +2 | +0 | +1 |

Number of Element on left = N - P_1 = 10

// When we take Element from right side

cut = cut + [Number of Elements of left side]

Else

cut won't ever change

TODO:

TC: (N log N + M log M + N + M)

Sort A Sort B Iterate on 2 Arrays

28) Given A[N], find no of pairs i, j such that

(Inversion Count) {sorting algorithms} $i < j$ && $A[i] > A[j]$

	0	1	2	3	4	5	6	7	8	9	Total
ar[10] =	10	3	8	15	6	12	2	18	7	1	26
cnt =	6	2	4	5	2	3	1	2	1	0	

Ideal:

Check all pairs.

```
c = 0
i = 0, i < N
    j = i+1 ... )
        |
        j
}
```

TC: $O(N^2)$

SC: $O(1)$

	0	1	2	3	4	5	6	7	8	9
ar[10] =	10	3	8	15	6	12	2	18	7	1

A[] = 10 3 8 15 6 (0 1 2 3 4)

B[] = 12 2 18 7 1 (5 6 7 8 9)

A[] sorted: 3 6 8 10 15 (p_1 p_1 p_1 p_1 p_1)

B[] sorted: 1 2 7 12 18 (p_2 p_2 p_2 p_2 p_2)

	1	2	3	6	7	8	10	12	15	18
cnt =	5	5	0	0	3	0	0	1	0	0

cnt = 14

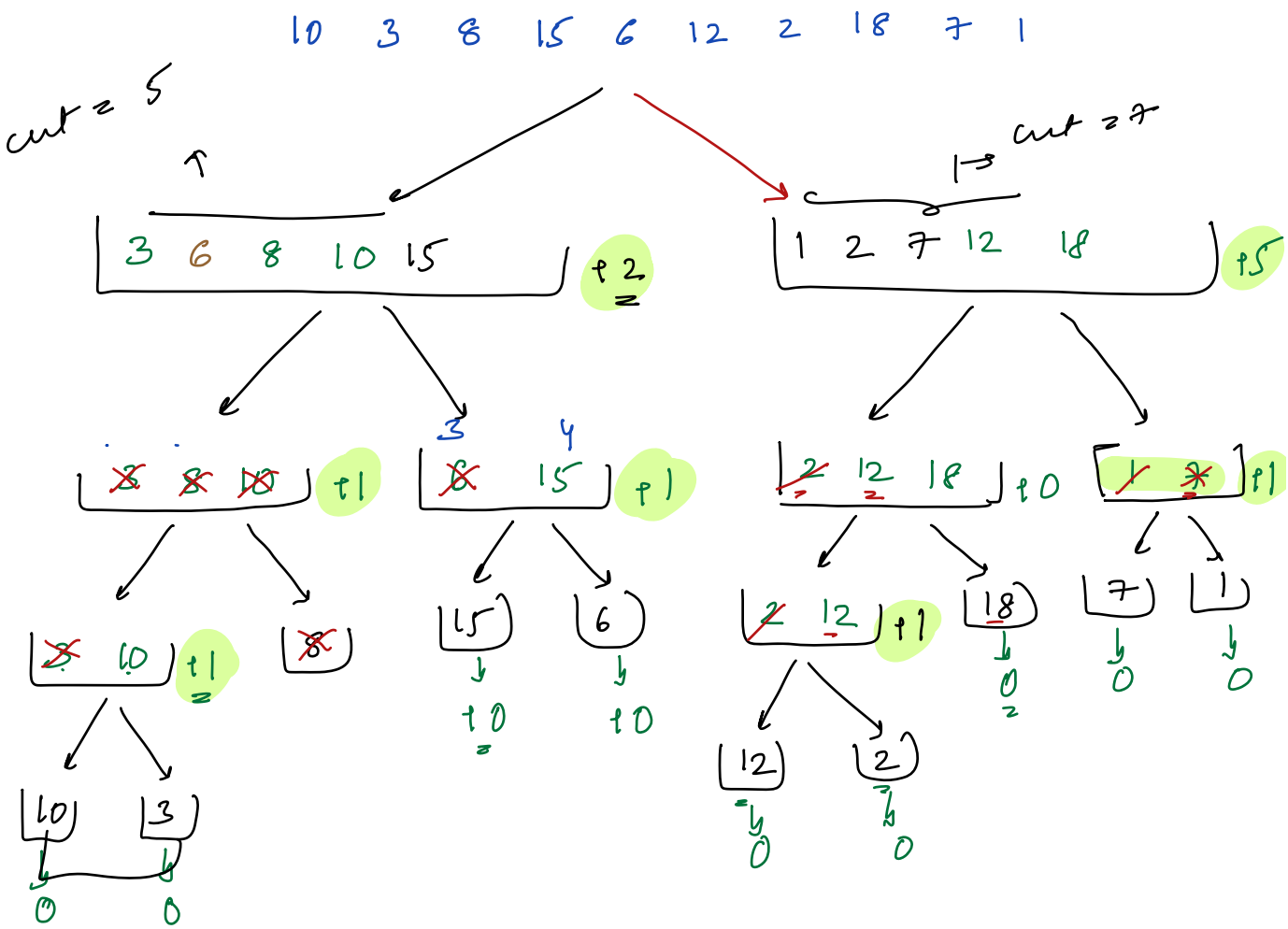
Total pairs = {pairs in A} + {pairs in B} + {pairs between A & B}

{pairs in A} = 5

{pairs in B} = 7

{pairs between A & B} = 14

⇒ 12 + 14 = 26


10 3 8 15 6 12 2 18 7 1
cut = 5
3 6 8 10 15
+2
1 2 7 12 18
+5
+1
+1
+0
+1
+1
+0
+0
0
0
0
0
0
0
0

// Recursion: Ass: Given a subarray of short & return count

```
// mergeSrt(A[], s, e) {
    if(s==e) return 0
    m = (s+e)/2
    l = mergeSrt(A, s, m)   // l
    r = mergeSrt(A, m+1, e) // r
    return l+r+merge(A, s, m, e)
}
// int merge(A[], s, m, e) {
    T[e-s+1], int c = 0
    p1 = s, p2 = m+1, p3 = 0
    while(p1<=m && p2<=e)
        if(A[p1] <= A[p2]) { T[p3] = A[p1]; p3++, p1++ }
        else { T[p3] = A[p2]; p3++; p2++, c = c + m - p1 + 1 }
    while(p1<=m) { T[p3] = A[p1]; p3++, p1++ }
    while(p2<=e) { T[p3] = A[p2]; p3++; p2++ }
    for(p=0; p<=(e-s); p++) {
        A[s+p] = T[p]
    }
    return c;
}
```

[s m] [m+1 e]

[p1 → m] m is also a pointer

2nd

no: of left or right side

[p1, m]

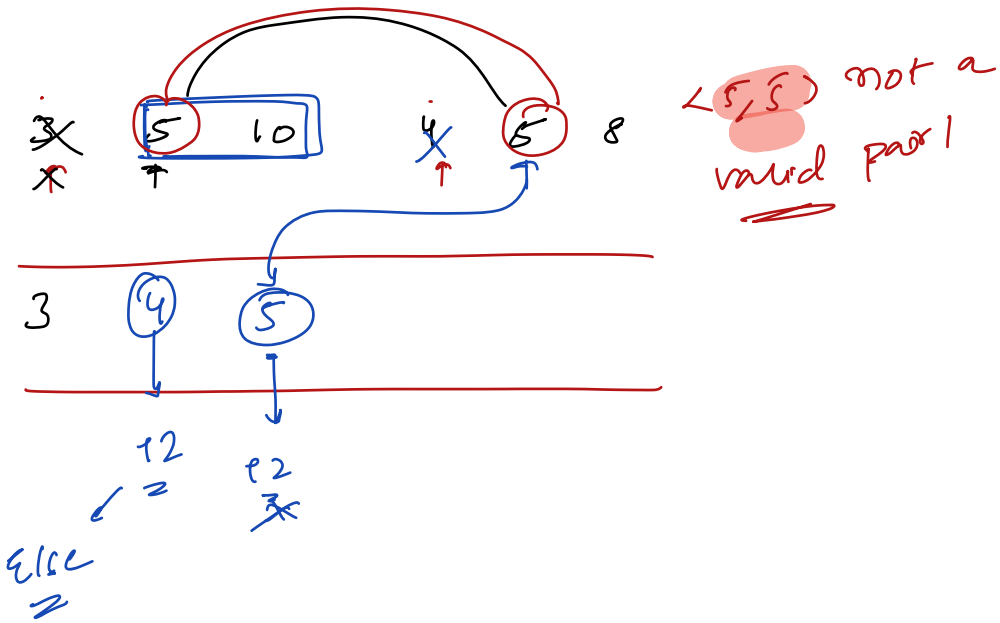
Break:

11 & 10 break

If a(P1) < a(P2)

else you increase

Else:



// Comp when ever returning false and cnt + 1 //

↳ This idea won't work

// put c = 0 ⟶ global variable

// void mergeSort(A[], s, e) {

```
    if(s==e) return;
    m = (s+e)/2
    mergeSort(A, s, m)
    mergeSort(A, m+1, e)
    merge(A, s, m, e)
}
```

// merge(A[], s, m, e) {

```
    T[e-s+1]
    P1 = s, P2 = m+1, P3 = 0
    while(P1<=m && P2<=e)
        if(A[P1]<=A[P2]) { T[P3] = A[P1]; P3++, P1++ }
        else { T[P3] = A[P2]; P3++; P2++, c = c + (m-P1+1) }
    }
    while(P1<=m) { T[P3] = A[P1]; P3++, P1++ }
    while(P2<=e) { T[P3] = A[P2]; P3++; P2++ }
    for(i=0; i<=(e-s); i++) {
        A[s+i] = T[i]
    }
}
```

2nd

Count Sort

```
// ar[10] = { 3 1 4 4 2 1 3 3 2 1 }
```

0 1 2 3 4 5 6 7 8 9

[1 - 4] → MergeSort : NlogN

cnt[5] = {0}

cnt[0] *

cnt[1] = 3

cnt[2] = 2

cnt[3] = 3

cnt[4] = 2

1 1 1 2 2 3 3 3 4 4

Array sorted

all Elements range : [1 k]

cnt[k+1] = 0

```
for(i=0; i< N; i++){
    cnt[ar[i]]++
}
```

Times → N

```
for(i=1; i<=k; i++){
    for(j=1; j<=cnt[i]; j++){
        print(i)
    }
}
```

for i

Total TC: O(N+k) SC: O(k)

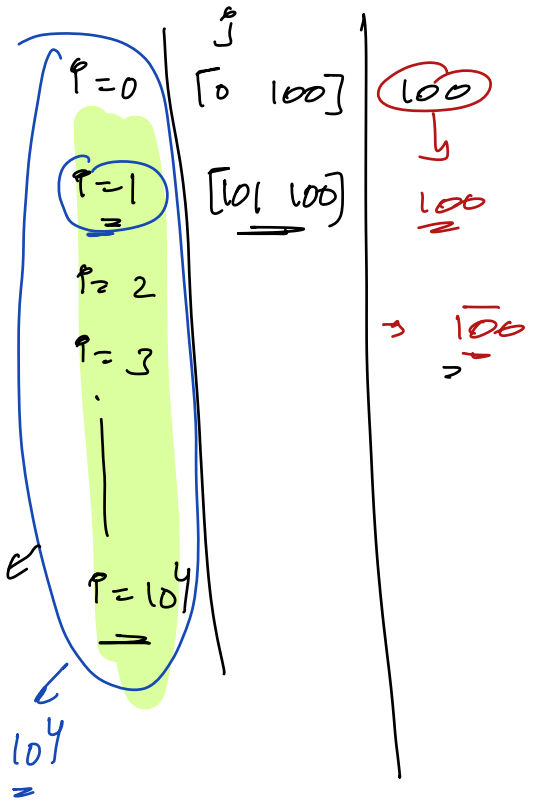
Table

i	j : [1, cnt[i]]	Total j
1	cnt[1]	cnt(1)
2	cnt[2]	cnt(2)
3	cnt[3]	.
4	cnt[4]	.
.		.
k	cnt[k]	cnt[k]

k times

N

```
j = 0;
// i = 0; i < 10^4; i++
    |
    |   j < 100; j++) {
    |
    3
    |
    |
    y
```



// Count Sort :

↳ TC : $O(N+k)$ SC ↓ $O(k)$

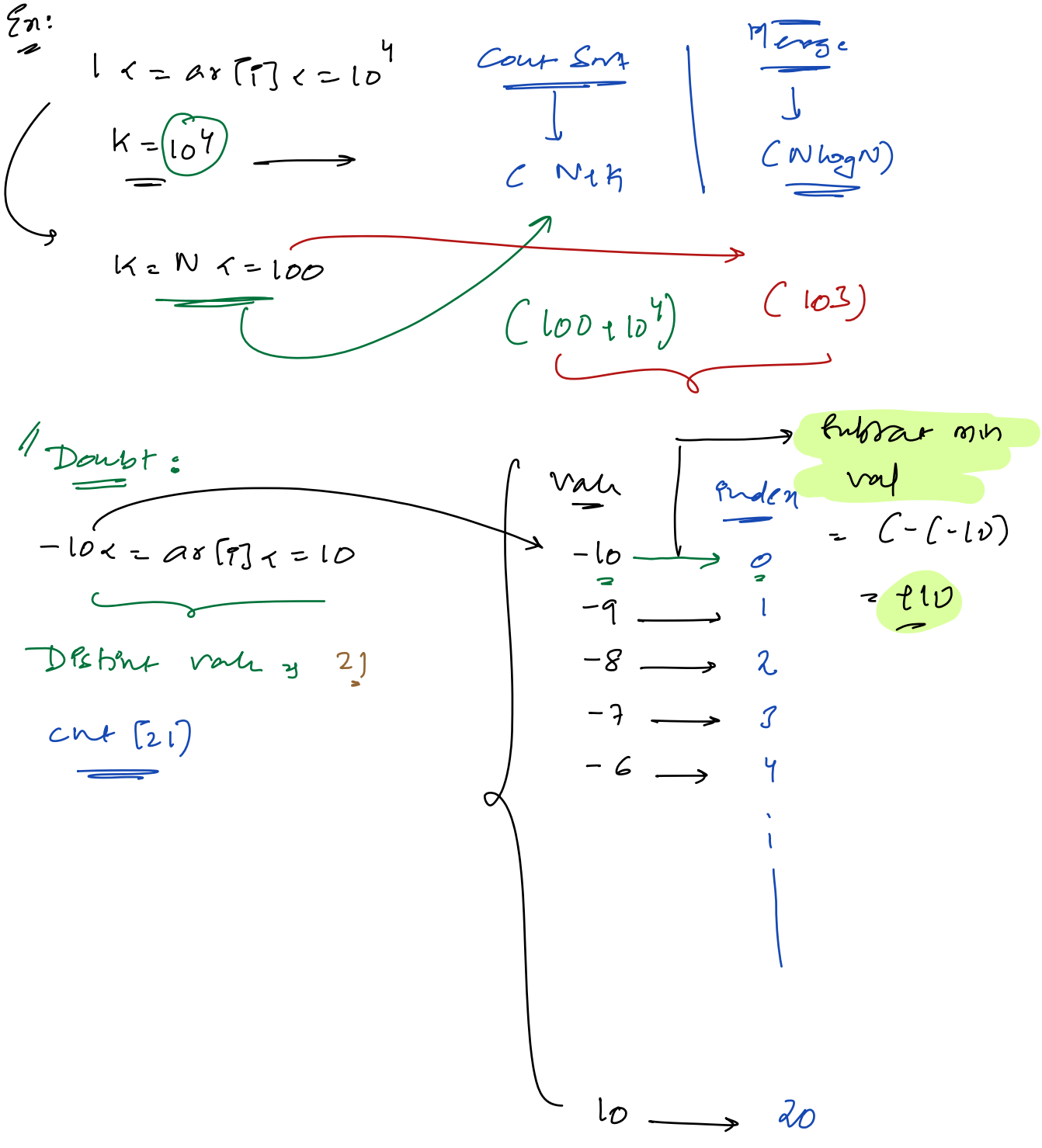
k : Number of elements in range.

↳

$1 < = ar[i] < = 10^5$ } $k = 10^5$

$1 < = ar[i] < = 10^9$ } $k = 10^9$

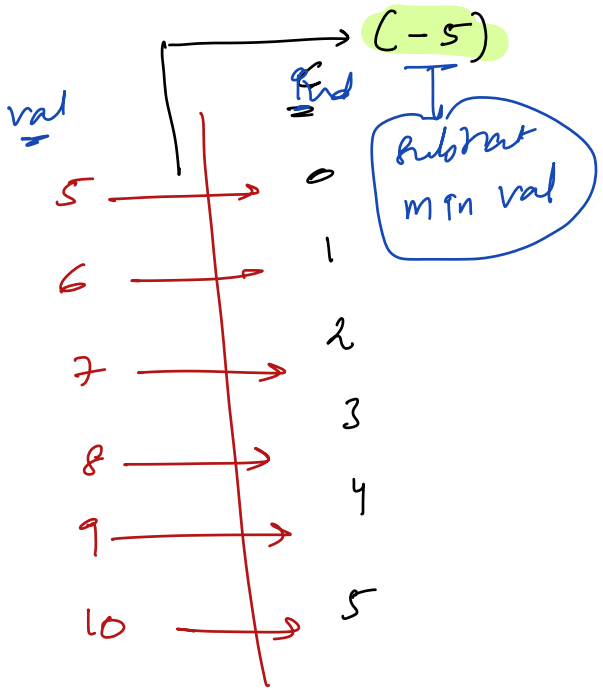
SC : 10^9 *


En:
1 <= arr[i] <= 10^4
K = 10^4
Count Sort
O(N+K)
Merge
O(NlogN)
K = N <= 100
(100 + 10^4)
(103)
// Doubt:
-10 <= arr[i] <= 10
Distinct value = 21
cnt[21]
val
index
-10 → 0
-9 → 1
-8 → 2
-7 → 3
-6 → 4
10 → 20
subtract min val
= 0 - (-10)
= +10

//

5 <= ar[i] <= 10

cnt[6]



// Dating algorithm → ur

	1	2	3	4	5	6
English :	Notebook	Titanic	skyfall	Avatar	Avengers	Inception
Grishma :	5	6	4	1	2	3

Inversion cnt :

Jatin : {5 4 2 3 1 6} → {1 3 5 6 4 2} : 6

Gaurav : 5 4 3 2 1 6 } → {1, 3, 6, 5, 4, 2} : 7

Rahul : 3 6 1 4 5 2 } → {6, 2, 4, 3, 1, 5} : 9

// N Elent

Min → 0

Max → (N)(N-1)/2

(N)(N-1)/2 → Mismatch → 100%

N=1

15 → 100%

6 → % of mismatch

→ 6*100/15 → 40 mismatch

→ 60 match

2D matrix → 1D : $O(N)$

→ 2D : $O(N^2 M)$