Todays Content

1. Count no:f elements with atleast 1 ele > itself
2. Pair sum = k
3. Vector Intro & pass by value & pass by reference.

1Q: Given ar[N] return count no: of elements with atleast 1 ele > itself.

Constraints:

$1 <= N <= 10^5$

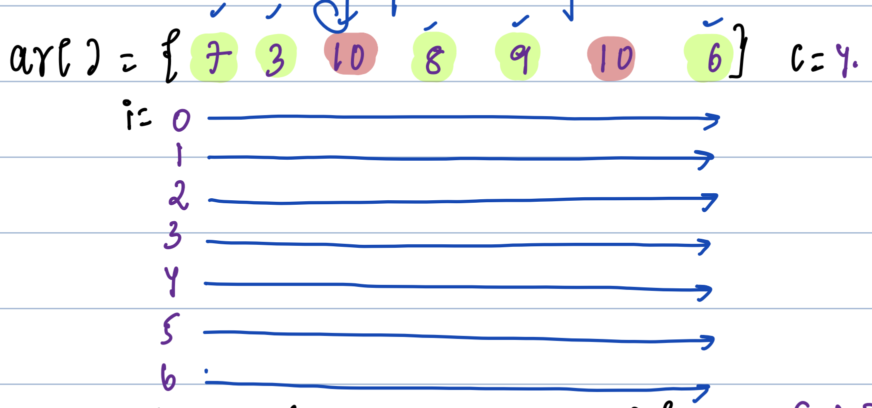
$1 <= ar[i] <= 10^9$

Ex: ar[] = { 7 3 10 8 9 10 6 } ans = 5

Ex: ar[] = { 9 6 4 7 9 4 } ans = 4.

Idea1: For every ar[i] element:

Iterate on array & check if there exists an element > ar[i]



```
int greaterItself( int[] ar, int N){ TC: O(N^2)   SC: O(1)
    int c=0;                          ↳ N<=10^5 => N^2 = 10^10 >>10^8 TLE.
    for( int i=0; i<N; i++){
        // ar[i]: Check if there exists an element > ar[i];
        bool isgreater = false;
        for(int j=0; j<N; j++){
            if( ar[j] > ar[i]){ // greater ele
                isgreater = true;
            }   break;
        }
        if( isgreater == true){
        }   c++;
    }
}   return c;
```

```
          0   1   2   3   4   5   6   7   8
arr[] = { 7   3   10  8   9   10  6   3   10 }
```

obs1: In arr[] max element won't have element > itself

Idea2: 1. Iterate & get max
2. Iterate & get count of non-max elements. } Estimated TC: O(N)

```
int greaterItself( int[] ar, int N){   Calculated TC: O(N)   SC: O(1)
                                        ↳ N<=10^5: 10^5 < 10^8 ✓
    int m = INT_MIN;
    for(int i=0; i<N; i++){
        if( ar[i] > m){
            m = ar[i];
        }
    }
    int c=0;
    for(int i=0; i<N; i++){
        if( ar[i] != m){
            c++;
        }
    }
    return c;
}
```

$\boxed{\text{TODO: Try to do it with 1 iteration.}}$

2Q

Given ar[N] elements & k

Count no: of pairs (i, j) are there such ar[i] + ar[j] == k

Note1: (i, j) pair same as (j, i)

Note2: (i ≠ j)

Constraints:

$1 <= N <= 10^3$

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

Ex1:

	0	1	2	3	4	5	6	7
ar[] = {	7	4	6	3	7	4	5	5 }

k=10

pairs = (0 3) (1 2) (2 5) (3 4) (2 1) (6 6) (6 7)

i = j   i ≠ j

Idea: Generate all pairs, check if their sum = k & Inc c.

Tracing: ar[] = { 7 4 6 3 7 } (indices 0 1 2 3 4)

pairs:

i j=	0	1	2	3	4
0	(0 0)	(0 1)	(0 2)	(0 3)	(0 4)
1	(1 0)	(1 1)	(1 2)	(1 3)	(1 4)
2	(2 0)	(2 1)	(2 2)	(2 3)	(2 4)
3	(3 0)	(3 1)	(3 2)	(3 3)	(3 4)
4	(4 0)	(4 1)	(4 2)	(4 3)	(4 4)

Note: In this idea (i, j) & (j, i) are considered, so for final ans/2 & return it.

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

$\hookrightarrow N \leq 10^3 : (10^3)^2 = 10^6 << 10^8$ ✓

```
int pairSum(int[] ar, int N, int k){
    int c=0;
    for(int i=0; i<N; i++){
        for(int j=0; j<N; j++){
            if( (ar[i]+ar[j]==k) && (i!=j) ){
                c++;
            }
        }
    }
    return c/2;
}
```

obs: Iterate in only upper part or lower part to avoid excess iterations

TODO

Tracing : ar[] = { 7 4 6 3 7 }

0	1	2	3	4
7	4	6	3	7

pairs:

i \ j:	0	1	2	3	4
0	(0 0)	(0 1)	(0 2)	(0 3)	(0 4)
1	(1 0)	(1 1)	(1 2)	(1 3)	(1 4)
2	(2 0)	(2 1)	(2 2)	(2 3)	(2 4)
3	(3 0)	(3 1)	(3 2)	(3 3)	(3 4)
4	(4 0)	(4 1)	(4 2)	(4 3)	(4 4)

obs: i, j = i+1 . . . N-1;

i=0, j=1

i=1, j=2

i=2, j=3

i=3, j=4

```
int pairSum(int[] ar, int N, int k){   TC: O(N^2)   SC: O(1)
    int c=0;
    for(int i=0; i<N; i++){
        for( int j=i+1; j<N; j++){
            if(ar[i]+ar[j]==k){
                c++;
            }
        }
    }
    return c;
}
```

Iterations: Construct Table

i	j : [i+1..N-1]
0	j : [1..N-1] = N-1
1	j: [2.. N-1] = N-2
⋮	
N-1	j: [N..N-1] = 0

Outer loop: N

Inner loop: $\underline{N-1+N-2+N-3+\dots 1}+0$

$: \frac{(N-1)(N)}{2}$

Total Iterations $= \frac{(N)(N+1)}{2}$

Issues in Arrays:

0	1	2	3	4
10	20	30	40	50

```
int ar[5];
ar[0]=10; ar[1]=20; ar[2]=30; ar[3]=40; ar[4]=50;
```

Issue: We cannot inc/dec size according to situation.

Dynamic Arrays: Size can be changed according to situation.

In C++	In Java	In Python
Vector	ArrayList<>	List

Create a Vector:

Way1: `Vector<datatype> vname1;`

Ex1: `vector<int> v1;`

Ex2: `vector<float> v2;`

Way2: `Vector<datatype> vname2(Initial_size, Initial_value);`

Ex1: `vector<int> v(5,10);` // v:

0	1	2	3	4
10	10	10	10	10

Insert into a vector:

`vname.push_back(val);` // Adds element at last

`v.push_back(15);`

// v:

0	1	2	3	4	5
10	10	10	10	10	15

Size of Vector

`vname.size();` // return size of vector

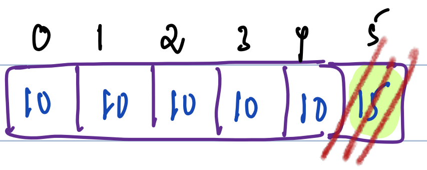
`int N = v.size();` // s=6.   `ar[i];`

Iterate on Vector: `vname[index];`

```
for(int i=0; i<N; i++){
    print(v[i]);
}
```

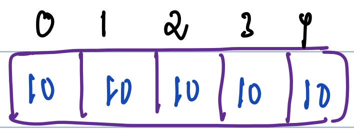
Remove from vector

```
vname.pop_back(); // It will delete last element
v.pop_back();
```

// v: 

Sort a vector

```
sort(vname.begin(), vname.end()); // Sort v from start → end
sort(v.begin(), v.end());
```

// v: 

Vector	TC for Single Call
1. push_back()	O(1)
2. pop_back()	O(1)
3. size()	O(1)
4. sort()	N log N ; // N: number of elements
5. v[i]	O(1)

Pass vector to a function.

1. Pass by value;

2. Pass by reference;

Q: Given vectr add all elements by +2 & return vectr.

```
vectr<int> modify(vectr<int> &v){
|
}
```