Todays Content

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

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

Constraints:

$1 <= N <= 10^5$
$1 <= arr[i] <= 10^9$
} Add in script

Ex: arr[] = { 7 3 10 8 9 10 6 } cnt = 5

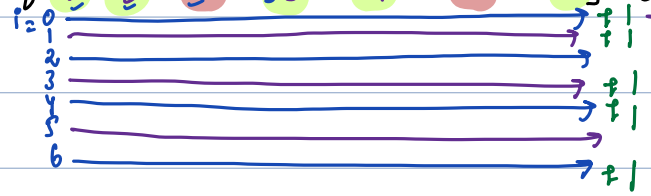
Ex: arr[] = { 9 6 4 7 9 4 } cnt = 4

Ideal: Iterate in arr[]:

For every arr[i]: Iterate & check if there an element > arr[i].

arr[] = { 7 3 10 8 9 10 6 } cnt = 5

Dry Run: c = 0




```
int greaterItself(int[] arr, int N){ TC: O(N^2)
    int c=0;                          ↳ N <= 10^5 => N^2 => (10^5)^2 = 10^10 > 10^8
    for(int i=0; i<N; i++){
        // For arr[i]: Check if there exists an elemet > arr[i]
        bool isgre = false;
        for(int j=0; j<N; j++){
            if(arr[j] > arr[i]){
                isgre = true;
            }
        }
        if(isgre == true){ // there is an ele > arr[i].
            c++;
        }
    }
    return c;
}
```

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

ar[] = { 7 3 10 8 9 10 6 3 10 }

obs1: Max element will not have an element greater than itself, Other than max every element will have element > itself

Idea2: Step1: Iterate & calculate max

Step2: Iterate & calculate nm-max elements counts & return.

```
int greaterItself(int[] ar, int N){ TC: O(2N) = O(N)  SC: O(1)
    int max = ar[0] or INT_MIN;
    for(int i=0; i<N; i++){
        if(ar[i] > max){
            max = ar[i];
        }
    }
    int c=0
    for(int j=0; j<N; j++){
        if(ar[j] < max){
            c++;
        }
    }
    return c;
}
```

TODO: Try to do it with 1 iteration.

28

Given ar[N] elements & k

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

Note: i & j are indices.

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

Note2: (i ≠ j)

Constraints: → Add in script

$1 <= N <= 10^3$ ⟶ TC: $O(N^2) = (10^3)^2 = 10^6 < 10^8$ ✓

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

Eg1:

	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) (5 2) (6 7) (6 6) (i == j)

Idea: Generate all pairs & calculate sum

if sum == k: Inc count

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

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

pairs:

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

Note: When we generate all pairs, valid pairs are counted twice

Before returning final ans/2;

↳ Script

```
int pairSum(int[] ar, int N, int k){   TC: O(N^2)   SC: O(1)
    int c=0;                            ↳ N^2 = 10^3 : (10^3)^2 = 10^6 < 10^8 ✓
    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 only on half of pairs. { Upper or lower triangle } TODO.

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

(0 1 2 3 4)

pairs:

i j=	0	1	2	3	4	
i=0	(0 0)	(0 1)	(0 2)	(0 3)	(0 4)	i=0 j=1 2 .. N-1
i=1	(1 0)	(1 1)	(1 2)	(1 3)	(1 4)	i=1 j=2 ... N-1
i=2	(2 0)	(2 1)	(2 2)	(2 3)	(2 4)	i=2 j=3 ... N-1
i=3	(3 0)	(3 1)	(3 2)	(3 3)	(3 4)	:
i=4	(4 0)	(4 1)	(4 2)	(4 3)	(4 4)	i j=i+1 ... N-1

```
int pairSum(int[] arr, int N, int k){ TC: O((N^2+N)/2) = 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(arr[i]+arr[j]==k){
                c++;
            }
        }
    }
    return c;
}
```

Issues in Arrays:

```
int ar[5];
```

0	1	2	3	4
10	20	30	40	50

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

Size is not dynamic.

Dynamic Arrays: Size is dynamic

In C++:	In Java	In Pythm
Vector	ArrayList	List

Create a Vector:

Way1: `vector<datatype> vname1;` // vname = []

```
vector<int> v1; // v1: [] : Can only int
vector<float> v2; // v2: [] : Can only float
```

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

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

0	1	2	3	4
10	14	9	19	10

Acess: `vname[index]`

v[2]=9; v[1]=14 v[3]=19

Insert into a vector:

`vname.push_back(val);` // Adds new element after last index

```
v.push_back(60);
v.push_back(3);
```

// v:

0	1	2	3	4	5	6
10	14	9	19	10	60	3

Size of Vector

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

```
int n = v.size(); // n = 7;
```

Remove from vector

`vname.pop_back();` // It will delete last index ele

`v.pop_back();`

// v:

0	1	2	3	4	5	6
10	14	9	19	10	60	3

Note: Check inserting & deleting from middle

Iterate on Vector

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

v:

0	1	2	3	4	5
10	14	9	19	10	60

Sort a vector

```
sort(vname.begin(), vname.end());
sort(v.begin(), v.end());
```

v:

0	1	2	3	4	5
9	10	10	14	19	60

TC: Vector:	TC: For 1 single call
1. push_back(x)	O(1)
2. pop_back()	O(1)
3. size()	O(1)
4. sort(v.begin(), v.end())	O(NlogN)

↳ N = size of Vector.

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

```
vector<int> modify(vector<int> &v){
|
3
```