**Inversion of array**

[array](http://www.practice.geeksforgeeks.org/tag-page.php?tag=array&isCmp=0)

Given an array, find inversion count of array.

***Inversion Count***for an array indicates – how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.   
Formally speaking, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.  
The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3).

**Input:**

The first line of input contains an integer T denoting the number of test cases.  
The first line of each test case is N,N is the size of array.  
The second line of each test case contains N input C[i].  
  
**Output:**

Print the inversion count of array.  
  
**Constraints:**

1 ≤ T ≤ 100  
1 ≤ N ≤ 100  
1 ≤ C ≤ 500  
  
**Example:**

**Input:**  
1  
5  
2 4 1 3 5

**Output:**  
3

\*\*For More Examples Use Expected Output\*\*

<http://www.practice.geeksforgeeks.org/problem-page.php?pid=558>

#include <iostream>

#include <stdio.h>

#include <string> // std::string

using namespace std;

int main() {

int t;

scanf("%d", &t);

while(t--) {

int n;

scanf("%d", &n);

int C[n];

for(int i =0; i < n; i++)

scanf("%d", &C[i]);

int inv=0;

for(int i =0; i < n ; i++) {

for(int j =i+1; j<n; j++) {

if(C[i] > C[j]){

inv++;

}

}

}

printf("%d\n", inv );

}

return 0;

}