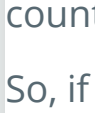
New Year Chaos



Editorial by rishi_07

Let's consider an example for n = 5. Initially, the array was A: [1, 2, 3, 4, 5].

Let's say after some number of bribes the array became A: [1, 5, 4, 2, 3].

The 5^{th} person moved from it's initial position to 2^{nd} position. For that, he must have bribed 3 people, which are 2, 3 and 4. Also, 4^{th} person moved from it's initial position to 3^{rd} position by bribing person 2 and person 3.

So the transformation goes something like this:

[1, 2, 3, 4, 5] -> [1, 2, 3, 5, 4] -> [1, 2, 5, 3, 4] -> [1, 5, 2, 3, 4] -> [1, 5, 2, 4, 3] -> [1, 5, 4, 2, 3]

Observation:

The number of people the i^{th} person (for $1 \leq i \leq n$) has bribed is equal to the number of people on the right of that person with a value less than A_i (where array A represents the given array or final state of the people).

To get to the current position, each person has to bribe all the people who are behind them and have a smaller number. This is the same as counting inversions of an array.

So, if that number is greater than 2 for any index i we print Too chaotic else print the total sum of bribes.

Brute Force Approach:

Run two loops, and for every integer i, we find the number of j's such that $A_i > A_j$, where $1 \leq i \leq n$ and $i < j \leq n$.

Time complexity: $O(n^2)$, but we can do better.

Optimised (Linear Time) Approach:

We start from the end of the array A. If A_i is not equal to i, where

STATISTICS

Difficulty: Medium

Time Complexity: O(n)

Required Knowledge: Inversions

Publish Date: Dec 13 2015

Originally featured in HourRank 4

NEED HELP?

View discussions

View top submissions

$1 \leq i \leq n$, then we know that the last element must have bribed and moved towards the left since it cannot move to the right being the last element. Also, we know that it will be present either in position $i - 1$ or $i - 2$. This is because if it is in the position left to $i - 2$, he must have bribed more than 2 people. In that case, we just print Too chaotic and terminate our program. Else if A_i is equal to A_{i-1} just swap the two elements and increment the counter by 1. Else shift A_{i-1} to A_{i-2}, A_i to A_{i-1} and put A_i equal to i and increment the counter by 2. Repeat the process until we reach the start of the array.

Note: For the answer to be a valid count, our condition that if A_i is not equal to i, it will be present either in position $i - 1$ or $i - 2$ holds for all the elements because at every step we reorganize the array and make A_i equal to i for $1 \leq i \leq n$. So, if we are at index i, we are sure that A_j is equal to j for $i < j \leq n$.

Time complexity: $O(n)$.

Please check the code below for more clarity:

C++

```
void minimumBribes(vector<int> A)
{
    int n = A.size();
    int cnt = 0;
    for(int i = n - 1; i >= 0; i--)
    {
        if(A[i] != (i + 1))
        {
            if(((i - 1) >= 0) && A[i - 1] == (i + 1))
            {
                cnt++;
                swap(A[i], A[i-1]);
            }
            else if(((i - 2) >= 0) && A[i - 2] == (i +
            {
                cnt += 2;
                A[i - 2] = A[i - 1];
                A[i - 1] = A[i];
                A[i] = i + 1;
            }
            else
            {
                printf("Too chaotic\n");
                return;
            }
        }
    }
```

```
    printf("%d\n",cnt);
    return;
}
```

Set by Shafaet_Ashraf

Problem Setter's code:

Python 2

```
t = int(raw_input())

for _ in range(t):
    n = int(raw_input())
    arr = map(int, raw_input().split())
    org = range(n+1)
    pos = range(n+1)
    cnt = [0]*(n + 1)
    ans = 0
    invalid = 0

    for i in xrange(n - 1, -1, -1):

            if invalid:
                break

        # Get position where arr[i] should have be

        oldp = pos[arr[i]]

        # Get the position where arr[i] currently

        newp = i + 1

        # oldp != newp indicates that even after t
        # counting the number of furthter bribes t

        while oldp != newp:

            ans = ans + 1

            # arr[i] is at the right of org[oldp +
            # that means org[oldp + 1] bribed arr[
            # so increasing its count by 1

            cnt[org[oldp + 1]] += 1


            if cnt[org[oldp + 1]] > 2:
                invalid = 1
```

```
                    break

                # updating the original array to match

                org[oldp], org[oldp+1] = org[oldp+1], 


                # update the positions also due to the
                # caused by bribe that took place so f

                pos[org[oldp]] = oldp
                pos[org[oldp + 1]] = oldp + 1

                oldp = oldp + 1

    if invalid:
        ans = "Too chaotic"

    print ans
```

Feedback

Was this editorial helpful?

Yes | No