New Year Chaos

Isaac DeJager

March 15, 2023

Click here to view the HackerRank page

1 Problem Description

It is New Year's Day and people are in line for the Wonderland rollercoaster ride. Each person wears a sticker indicating their initial position in the queue from 1 to n. Any person can bribe the person directly in front of them to swap positions, but they still wear their original sticker. One person can bribe at most two others. Determine the minimum number of bribes that took place to get to a given queue order. Print the number of bribes, or, if anyone has bribed more than two people, print **Too chaotic**.

Example
$q = [1, 2, 3, 5, 4, 6, 7, 8]$
If person 5 bribes person 4,, the queue will look like this: $1, 2, 3, 5, 4, 6, 7, 8$. Only 1 bribe is required to reach q. Print **1**.

$q = [4, 1, 2, 3]$
Person 4 had to bribe 3 people to get to the current position. Print **Too chaotic**.

Function Description
Complete the function minimumBribes in the editor below.
minimumBribes has the following parameter(s):

- int q[n]: the positions of the people after all bribes

Returns

- No value is returned. Print the minimum number of bribes necessary or **Too chaotic** if someone has bribed more than 2 people.

Input Format
The first line contains an integer t, the number of test cases.
Each of the next t pairs of lines are as follows:

- The first line contains an integer t, the number of people in the queue
- The second line has n space-separated integers describing the final state of the queue.

Constraints

- $1 \leq 10$
- $1 \leq 10^5$

Sample Input 0
q = [2, 1, 5, 3, 4]

Sample Output 0
3
Too chaotic

Explanation 0
The initial state:
1 2 3 4 5
After person 5 moves one position ahead by bribing person 4:
1 2 3 5 4
After person 5 moves another position ahead by bribing person 3:
1 2 5 3 4
And person 2 moves one position ahead by bribing person 1:
2 1 5 3 4
So the final state is 2, 1, 5, 3, 4 after three bribing operations.

Sample Input 1
2 5 1 3 4

Sample Output 1
Too chaotic

Explanation 1
No person can bribe more than two people, yet it appears person 5 has done so. It is not possible to achieve the input state.

2 Solution

```
void minimumBribes(vector<int> q) {

    int p[3] = {1, 2, 3};
    int pSize = sizeof(p) / sizeof(p[0]);
    int x = 0;
    bool y;

    for (int i = 0; i < q.size(); i++) {

        y = true;
        for (int j = 0; j < pSize; j++) {

            if (q.at(i) == p[j]) {

                y = false;
                for (int k = j; k < pSize - 1; k++) {
                    p[k] = p[k + 1];
                }

                p[pSize - 1]++;
                x += j;
                break;

            }

        }

        if (y) {
            cout << ”Too chaotic” << endl;
            return;
        }

    }

    cout << x << endl;

}
```

3 Solution

The basic logic of this solution is that at any given index i, the value $q[i]$ can only be one of three possibilities. If it is not, then we know the queue is too chaotic. For example, consider $i = 0$. If $q[0] \neq 0$, 1, or 2, then whatever number it is must have jumped at least 0, 1, and 2 to get there. Meaning it bribed at least 3 times and we print "Too chaotic".

For every index i, we keep track of what the possible values can be while also counting the total number of bribes. The solution above was optimized to work for any possible number of bribes. For this problem, it states each number can only bribe up to 2 times. But this solution would work for any number.

We begin by creating an array p which stands for "possibles"; the contents of which are the possible values for $q[i]$. Since we'll start at $i = 0$, p is initialized to be $1, 2, 3$. Next we create an integer x initialized at 0 which will count the total number of bribes. Finally, a boolean y that will track whether or not $q[i]$ is one of the values in p.

Then we start to loop through q, one index at a time. We start by setting $y =$ **true**. Then we loop through p to check if $q[i]$ is any value in p. If we find a match, y is set to **false** and we run through the rest of p and change all the values.

For example, consider $i = 0$ and $p = 1, 2, 3$.

- If $q[0] = 1$, then $p = 2, 3, 4$ and x is unchanged
- If $q[0] = 2$, then $p = 1, 3, 4$ and $x = x + 1$
- If $q[0] = 3$, then $p = 1, 2, 4$ and $x = x + 2$

And if $q[0]$ is none of those, then we print "Too chaotic" and return. After that, we repeat on the next value of q. We keep track of the number of jumps by locating the index of p that matches $q[i]$, as that lets us know how many numbers $q[i]$ bribed.

The time complexity of this is minimal. For this problem, its simply $O(3n) = O(n)$ since we loop through the whole array and only check each value against 3 others. But supposing instead of each person being able to bribe 2 people, they could bribe as many as m people where $m \leq n$. Then, our time complexity jumps to:

$$O(m * n) = O(n^2)$$

And our space complexity required is $O(m) = O(n)$. But for this problem, its simply an array of size 3.