Practice > Algorithms > Constructive Algorithms > New Year Chaos

New Year Chaos ☆

You have successfully solved New Year Chaos Share Tweet

Try the next challenge | Try a Random Challenge

Problem Submissions Leaderboard Discussions Editorial

Author	Shafaet
Difficulty	Medium
Max Score	40
Submitted By	17133

NEED HELP?

View discussions

View editorial

View top submissions

RATE THIS CHALLENGE

☆ ☆ ☆ ☆ ☆

MORE DETAILS

Download problem statement

Download sample test cases

Suggest Edits

It's New Year's Day and everyone's in line for the Wonderland rollercoaster ride!

There are a number of people queued up, and each person wears a sticker indicating their initial position in the queue. Initial positions increment by 1 from 1 at the front of the line to n at the back.

Any person in the queue can bribe the person directly in front of them to swap positions. If two people swap positions, they still wear the same sticker denoting their original places in line. One person can bribe at most two others.

For example, if $n = 8$ and $Person\ 5$ bribes $Person\ 4$, the queue will look like this: $1, 2, 3, 5, 4, 6, 7, 8$.

Fascinated by this chaotic queue, you decide you must know the minimum number of bribes that took place to get the queue into its current state!

Function Description

Complete the function minimumBribes in the editor below. It must print an integer representing the minimum number of bribes necessary, or **Too chaotic** if the line configuration is not possible.

minimumBribes has the following parameter(s):

- q: an array of integers

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

- $1 \le t \le 10$
- $1 \le n \le 10^5$

Subtasks

For 60% score $1 \le n \le 10^3$
For 100% score $1 \le n \le 10^5$

Output Format

Print an integer denoting the minimum number of bribes needed to get the queue into its final state. Print **Too chaotic** if the state is invalid, i.e. it requires a person to have bribed more than 2 people.

Sample Input

```
2
5
2 1 5 3 4
5
2 5 1 3 4
```

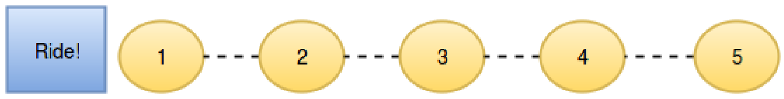
Sample Output

```
3
Too chaotic
```

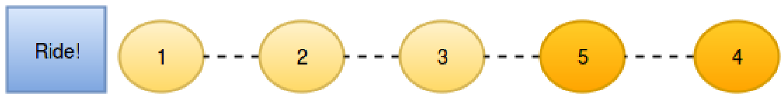
Explanation

Test Case 1

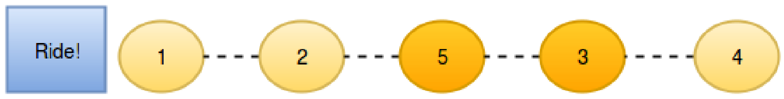
The initial state:



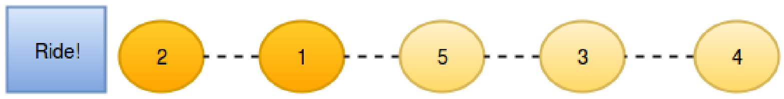
After person 5 moves one position ahead by bribing person 4:



Now person 5 moves another position ahead by bribing person 3:



And person 2 moves one position ahead by bribing person 1:



So the final state is $2, 1, 5, 3, 4$ after three bribing operations.

Test Case 2

No person can bribe more than two people, so its not possible to achieve the input state.

Current Buffer (saved locally, editable) Java 7

```java
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        int numberOfTestCases = reader.nextInt();

        for (int i = 0; i < numberOfTestCases; i++) {
            int numberOfPeopleInQueue = reader.nextInt();
            int[] finalStateOfQueue = new int[numberOfPeopleInQueue];
            for (int j = 0; j < numberOfPeopleInQueue; j++) {
                finalStateOfQueue[j] = reader.nextInt();
            }
            minimumBribes(finalStateOfQueue);
        }
    }

    static void minimumBribes(int[] finalStateOfQueue) {

        int minimumBribes = 0;
        boolean moreThanTwoBribesPerPerson = false;

        for (int i = finalStateOfQueue.length - 1; i >= 0; i--) {
            int bribesPerPerson = finalStateOfQueue[i] - (i + 1);

            if (bribesPerPerson > 2) {
                moreThanTwoBribesPerPerson = true;
                break;
            } else if (bribesPerPerson == 1) {
                minimumBribes += bribesPerPerson;
                int tempStore = finalStateOfQueue[i + 1];
                finalStateOfQueue[i + 1] = finalStateOfQueue[i];
                finalStateOfQueue[i] = tempStore;
            } else if (bribesPerPerson == 2) {
                minimumBribes += bribesPerPerson;
                int tempStore_01 = finalStateOfQueue[i + 1];
                int tempStore_02 = finalStateOfQueue[i + 2];
                finalStateOfQueue[i + 2] = finalStateOfQueue[i];
                finalStateOfQueue[i + 1] = tempStore_02;
                finalStateOfQueue[i] = tempStore_01;
                i++;
            }
        }

        if (moreThanTwoBribesPerPerson) {
            System.out.println("Too chaotic");
        } else {
            System.out.println(minimumBribes);
        }
    }
}

```

Line: 1 Col: 20

Upload Code as File ☐ Test against custom input Run Code Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

✓ Testcase 0 ✓ Testcase 1 ✓ Testcase 2 ✓ Testcase 3

11 Testcases

Input (stdin) Download

```
2
5
2 1 5 3 4
```

Expected Output Download

```
3
Too chaotic
```

Compiler Message

```
Success
```