Your Larry's Array submission got 40.00 points. Share Tweet

Try the Next Challenge | Try a Random Challenge

Larry's Array

by nikasvanidze

Problem Submissions Leaderboard Discussions Editorial

Larry has a permutation of N numbers, A, whose unique elements range from 1 to N (i.e.: $A = \{a_1, a_2, \ldots, a_{N-1}, a_N\}$). He wants A to be sorted, so he delegates the task of doing so to his robot. The robot can perform the following operation as many times as it wants:

- Choose any 3 consecutive indices and rotate their elements in such a way that ABC rotates to BCA, which rotates to CAB, which rotates back to ABC.

For example: if $A = \{1, 6, 5, 2, 4, 3\}$ and the robot rotates $(6, 5, 2)$, A becomes $\{1, 5, 2, 6, 4, 3\}$.

On a new line for each test case, print `YES` if the robot can fully sort A; otherwise, print `NO`.

Input Format

The first line contains an integer, T, the number of test cases.
The $2T$ subsequent lines each describe a test case over 2 lines:

1. An integer, N, denoting the size of A.
2. N space-separated integers describing A, where the i^{th} value describes element a_i.

Constraints

- $1 \leq T \leq 10$
- $3 \leq N \leq 1000$
- $1 \leq a_i \leq N$, where every element a_i is unique.

Output Format

On a new line for each test case, print `YES` if the robot can fully sort A; otherwise, print `NO`.

Sample Input

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

Sample Output

```
YES
YES
NO
```

Explanation

In the explanation below, the subscript of A denotes the number of operations performed.

Test Case 0:

$A_0 = \{3, 1, 2\} \rightarrow \text{rotate}(3, 1, 2) \rightarrow A_1 = \{1, 2, 3\}$

A is now sorted, so we print `YES` on a new line.

Test Case 1:

$A_0 = \{1, 3, 4, 2\} \rightarrow \text{rotate}(3, 4, 2) \rightarrow A_1 = \{1, 4, 2, 3\}$.

$A_1 = \{1, 4, 2, 3\} \rightarrow \text{rotate}(4, 2, 3) \rightarrow A_2 = \{1, 2, 3, 4\}$.

A is now sorted, so we print `YES` on a new line.

Test Case 2:

No sequence of rotations will result in a sorted A. Thus, we print `NO` on a new line.

Medium Submitted 14072 times
Max Score **40**

Need Help?

View Discussions

View Editorial Solution

View Top Submissions

Rate This Challenge:

Download problem statement

Download sample test cases

Suggest Edits

Current Buffer (saved locally, editable) C++14

```
/** Explanation
12  1 10 2 7 11 4 14 5 X 9 15 8 13 6 3
An inversion is when a tile precedes another tile with a lower number on it. The solution state has zero
inversions. For example, if, in a 4 x 4 grid, number 12 is top left, then there will be 11 inversions from this
tile, as numbers 1-11 come after it. To explain it another way, an inversion is a pair of tiles (a,b) such that a
appears before b, but a>b. Count the number of inversions in the grid. For example, on the grid

the 12 gives us 11 inversions
the 1 gives us none
the 10 gives us 8 inversions
the 2 gives us none
the 7 gives us 4 inversions
the 11 gives us 6 inversions
the 4 gives us one inversion
the  14 gives us 6
the 5 gives us one
the 9 gives us 3
the 15 gives us 4
the 8 gives us 2
2 from the 13
one from the 6

So there are 49 inversions in this example.

The formula says:
1. If the grid width is odd, then the number of inversions in a solvable situation is even.
2. If the grid width is even, and the blank is on an even row counting from the bottom (second-last, fourth-last
etc), then the number of inversions in a solvable situation is odd.
```

```
3. If the grid width is even, and the blank is on an odd row counting from the bottom (last, third-last, fifth-last etc) then the number of inversions in a solvable situation is even.

( (grid width odd) && (#inversions even) )  ||  ( (grid width even) && ((blank on odd row from bottom) == (#inversions even)) )
*/
#include <iostream>
#include <vector>
#include <algorithm>

int main()
{
   int T=0; std::cin >> T;

   while(T--)
   {
      int N; std::cin >> N;

      std::vector<int> vec(N);
      for(auto &it: vec)   std::cin >> it;

      int inversions = 0;
      for(int i = 0; i < N-1; ++i)
         for(int j = i+1; j < N; ++j)
            if(vec[i] > vec[j])
               ++inversions;

      (inversions & 1) ?
         std::cout << "NO" << "\n":
         std::cout << "YES" << "\n";
   }
   return 0;
}
```

Line: 56 Col: 1

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

Congrats, you solved this challenge!

Challenge your friends:

✔ Test Case #0	✔ Test Case #1	✔ Test Case #2
✔ Test Case #3	✔ Test Case #4	✔ Test Case #5
✔ Test Case #6	✔ Test Case #7	✔ Test Case #8
✔ Test Case #9	✔ Test Case #10	✔ Test Case #11
✔ Test Case #12	✔ Test Case #13	✔ Test Case #14
✔ Test Case #15	✔ Test Case #16	✔ Test Case #17
✔ Test Case #18	✔ Test Case #19	✔ Test Case #20

You've earned 40.00 points. Next Challenge