Larry's Array ☆

Editorial by nikasvanidze

Let's call the inversions in sequence A (size of N) the number of pairs (x, y), where $1 \leq x < y \leq N$ and $A_x > A_y$.

Let's prove that the parity of inversions will not change:
A, B, C -> B, C, A.
Order of A, C changed thats ± 1.
Order of A, B changed thats ± 1.
So the change is always -2, 0 or 2.

Prove that we can always get the permutation $[1, 2, 3, \ldots, n-2, n-1, n]$, Form 1 or $[1, 2, 3, \ldots, n-2, n, n-1]$, $*Form2*$. Note the reversal of the last two elements.

Using our operation we are moving B and C left by one. In this way we can move 1 into the first place, 2 into the second place and so on through $n-2$.

The sum of the inversions will be 0 when A is in Form 1 or 1 when A is in Form 2.

If the is 1, it's impossible to sort, but if it's 0, it's sorted.

Calculate parity using any BST. The solution will be $O(T * Nlog_2 N)$.

In this problem you can calculate the answer by comparing each pair. $O(T * N^2)$

Featured Code

Python 2

```
for i in range(input()):
    n = input()
    arr = map(int, raw_input().split())
    j = 1
    while j < n+ 1:
        ind = j - 1
        if arr[ind] == j:
            pass
        else:
            loc = arr.index(j)
            dist = loc - ind
            if dist == 1:
                if ind + 2 >= n:
                    print "NO"
                    break
                ele = arr.pop(ind)
                arr.insert(loc + 1, ele)
            elif dist % 2 == 0:
                ele = arr.pop(loc)
                arr.insert(ind, ele)
            elif dist % 2 == 1:
                if ind + 2 >= n:
                    print "NO"
                    break
                ele = arr.pop(loc)
                arr.insert(ind + 1, ele)
                j -= 1
        j += 1
    else:
        print "YES"
```

STATISTICS

Difficulty: Medium
Time Complexity: $O(T \times Nlog_2 N)$
Required Knowledge: invariant
Publish Date: Jan 27 2016
Originally featured in 101 Hack March 2016

NEED HELP?

View discussions
View top submissions

Set by nikasvanidze

Problem Setter's code:

C++

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1509;
int n;
int a[N];

void input() {
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
}

void sol() {
    int K = 1;
    for (int i = 1; i <= n; i++) {
        for (int j = i + 1; j <= n; j++) {
            K ^= (a[i] > a[j]);
        }
    }
    if (K) {
        printf("YES\n");
    }
    else {
        printf("NO\n");
    }
}

int main() {
    int test;
    scanf("%d", &test);
    while (test--){
        input();
        sol();
    }
    return 0;
}
```

Tested by gorbunovdv

Problem Tester's code:

Java 8

```
import java.io.OutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.util.Arrays;
import java.io.IOException;
import java.io.Reader;
import java.io.InputStreamReader;
import java.util.StringTokenizer;
import java.io.Writer;
import java.io.OutputStreamWriter;
import java.io.BufferedReader;
import java.io.InputStream;

/**
 * Built using CHelper plug-in
 * Actual solution is at the top
 */
public class Main {
    public static void main(String[] args) {
        InputStream inputStream = System.in;
        OutputStream outputStream = System.out;
        InputReader in = new InputReader(inputStream);
        OutputWriter out = new OutputWriter(outputStream);
        LarrysArray solver = new LarrysArray();
        solver.solve(1, in, out);
        out.close();
```

```
    }

    static class LarrysArray {
        public void solve(int testNumber, InputReader in, OutputWriter out) {
            int t = in.readInt();
            if (t <= 0 || t > 100) {
                throw new RuntimeException("t is out of range :(");
            }
            while (t-- > 0) {
                int n = in.readInt();
                if (n <= 0 || n > 1000) {
                    throw new RuntimeException("n is out of range :(");
                }
                int[] a = new int[n];
                for (int i = 0; i < n; i++) {
                    a[i] = in.readInt() - 1;
                }
                if (!isPermutation(a)) {
                    throw new RuntimeException("a is not a permutation :(");
                }
                int result = 0;
                boolean[] used = new boolean[n];
                for (int i = 0; i < n; i++) {
                    if (!used[i]) {
                        int cur = i, size = 1;
                        while (!used[cur]) {
                            used[cur] = true;
                            size ^= 1;
                            cur = a[cur];
                        }
                        result ^= size;
                    }
                }
                out.printLine(result == 0 ? "YES" : "NO");
            }
        }

        boolean isPermutation(int[] a) {
            int[] x = a.clone();
            Arrays.sort(x);
            for (int i = 0; i < x.length; i++) {
                if (x[i] != i) {
                    return false;
                }
            }
            return true;
        }

    }

    static class OutputWriter {
        private PrintWriter writer;

        public OutputWriter(Writer writer) {
            this.writer = new PrintWriter(writer);
        }

        public OutputWriter(OutputStream stream) {
            this(new OutputStreamWriter(stream));
        }

        public void print(Object... args) {
            for (Object arg : args) {
                writer.print(arg);
            }
        }

        public void printLine(Object... args) {
            print(args);
            writer.println();
        }

        void close() {
            writer.close();
        }

    }

    static class InputReader {
        private BufferedReader reader;
        private StringTokenizer tokenizer;
```

```
        public InputReader(Reader reader) {
            this.reader = new BufferedReader(reader);
        }

        public InputReader(InputStream stream) {
            this(new InputStreamReader(stream));
        }

        public String nextLine() {
            try {
                return reader.readLine();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }

        public String readWord() {
            while (tokenizer == null || !tokenizer.hasMoreTokens()) {
                tokenizer = new StringTokenizer(nextLine());
            }
            return tokenizer.nextToken();
        }

        public int readInt() {
            return Integer.parseInt(readWord());
        }

    }
}
```

Feedback

Was this editorial helpful?

Yes No