# 15-295 Fall 2021 #12 Selection Round 1

## A. Horrid Sorting

3 s., 1024 MB

You are given an array $[p_1, p_2, \ldots, p_n]$ which contains a permutation of the numbers $1, 2, \ldots n$. Your goal is to sort the array by repeatedly applying the following operation:

- Pick two indices $i$ and $j$ such that $|p_i - p_j| = 1$, and then swap the values of $p_i$ and $p_j$.

For example, we can sort the array $[2, 3, 1]$ in two operations:

1. Swap $p_1$ and $p_3$. The array becomes $[1, 3, 2]$.
2. Swap $p_2$ and $p_3$. The array becomes $[1, 2, 3]$ which is sorted.

Your goal is to compute the minimum number of operations to sort a given array in ascending order.

### Input

The first line is $n$. $1 < n \leq 2 \times 10^5$. The second line is the permutation $p_1, p_2, \ldots, p_n$.

### Output

Output the minimum number of operations required to sort the given array.

**input**

```
3
1 3 2
```

**output**

```
1
```

## B. Everything must be Monitored

2 s., 1024 MB

You're given a tree of $n$ nodes. The goal is to ensure that the entire tree – including all of its vertices and edges – is *monitored*. To do so you place measurement devices (MDs) at a subset of nodes. The MDs induce a cascade of monitoring, as explained below.

So given a tree with some placement of MDs on the nodes (and no nodes or edges initially monitored), apply the following rules, labeling more and more nodes and edges as monitored, until the rules no longer apply anywhere.

- A node with an MD is monitored.
- All edges incident to a node with an MD are monitored.
- If an edge is monitored, so are both of its nodes.
- If both nodes at the ends of an edge are monitored, then so is that edge.
- If a node of degree $k \geq 2$ is monitored, and $k - 1$ of its edges are monitored, then the last of its edges is also monitored.

This process, because it is Church-Rosser, is guaranteed to converge to a final and unique state for any tree and any placement of the MDs.

The goal is to compute the minimum number of MDs that must be placed on the nodes of the tree to ensure that the entire tree (all nodes and edges) is monitored.

## Input

The first line of input contains $n$ ($2 \leq n \leq 10^5$) the number of nodes in the tree. Each of the next $n - 1$ lines contain two numbers $a_i$ and $b_i$, the nodes at the ends of an edge of the tree ($1 \leq a_i, b_i \leq n$). These edges are guaranteed to form a tree of nodes $1, \ldots, n$.

## Output

Output just one number – the minimum number of measurement devices that must be placed in the tree to ensure that it's fully monitored.

**input**

```
10
8 9
8 10
1 2
4 5
4 6
6 7
6 8
2 4
2 3
```

**output**

```
2
```

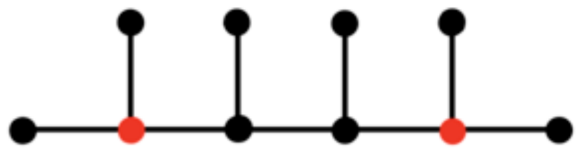
**input**

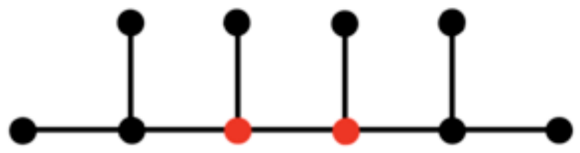
```
5
1 5
1 4
1 2
1 3
```

**output**

```
1
```

The first example corresponds to the figures below. The nodes with measurement devices are shown in red. The first figure is a configuration where everything is monitored with two measurement devices. In the second figure, the placement of measurement devices is not sufficient to monitor the entire tree.





## C. Alternation is the Key

3 s., 1024 MB

You're given a sequence of $n$ bits, denoted $a_1, a_2, \ldots, a_n$. The subsequence from $l$ to $r$ (with $1 \le l \le r \le n$) is $a_l, \ldots, a_r$. Such a subsequence is said to be *alternating* if $a_l \ne a_{l+1} \ne \cdots \ne a_r$. For example $1, 0, 1$ is an alternating subsequence of $1, 1, 0, 1, 1$, with $l = 2$ and $r = 4$.

In this problem, two types of operations will be applied to the given array:

- $1\ l\ r$: for every $i \in [l, r]$, change $a_i$ into $1 - a_i$.
- $2\ l\ r$: report the total number of pairs $(x, y)$ such that $l \le x \le y \le r$ where subsequence $a_x, a_{x+1}, \ldots, a_y$ is an alternating subsequence.

Your program will implement these operations.

### Input

The first line contains two integers $n$ and $q$ (with $1 \le n, q \le 2 \times 10^5$) indicating the length of the given sequence and the number of operations. The second line contains $a_1, a_2, \ldots, a_n$, the initial contents of the array of bits. Then $q$ lines follow, and the $i$-th of them contains $3$ integers $t_i, l_i, r_i$ where the $i$-th operation is $t_i\ l_i\ r_i$. ($1 \le t_i \le 2$ and $1 \le l_i \le r_i \le n$)

### Output

For each operation of the second type, output the required number on one line.

**input**

```
4 4
0 0 0 0
2 1 4
1 2 2
2 1 4
2 2 3
```

**output**

```
4
7
3
```

## D. Line, Meet Squares

2 s., 512 MB

You're given a collection of $n$ squares in the plane, along with a line. The goal is to count the number of intersections between the line and the boundaries of the squares.

More specifically, for each $i$ in $[1, n]$ a square with these four corners is generated: $(i, 0), (0, i), (-i, 0), (0, -i)$. And for that set of squares, a set of $q$ query lines is specified, and for each you must compute the intersection count.

### Input

The first line contains $n$ and $q$, where $1 \le n \le 10^9$ and $1 \le q \le 10^5$. Each of the following $q$ lines contains four integers $x_1, y_1, x_2, y_2$, representing two points $(x_1, y_1)$ and $(x_2, y_2)$. The line through these two points is the one for which you must count intersections with the $n$ squares. All of these coordinates are in the range $[-10^9, 10^9]$. These two points are not equal, and also the line through the two points must not have slope $1$ or $-1$. (That is $|x_1 - x_2| \ne |y_1 - y_2|$.)

## Output

For each line indicated, compute the number of times it intersects the $n$ square boundaries. See the examples below.

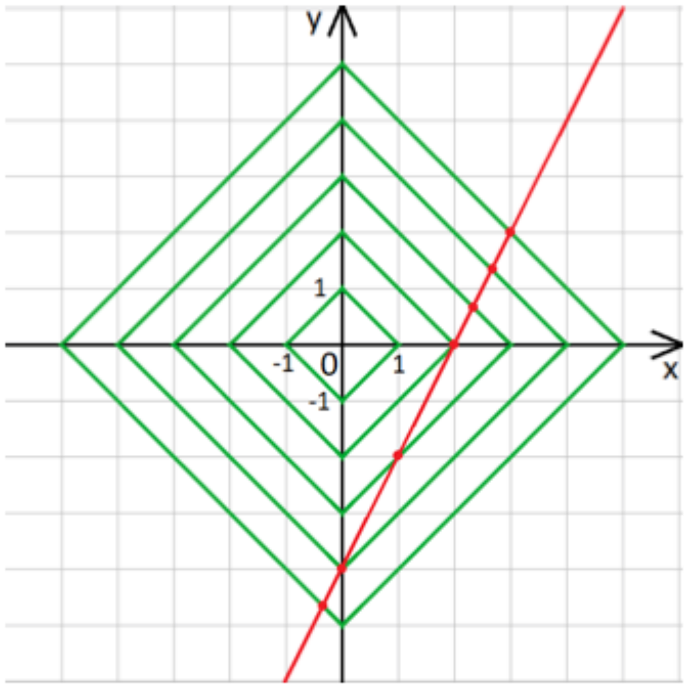
**input**

```
5 12
4 4 1 -2
0 0 -3 1
-1 0 1 0
7 0 0 6
5 1 -2 -1
-5 1 -4 1
4 -5 -3 -5
2 1000000000 0 -1000000000
2 999999999 0 -1000000000
2 1000000000 0 -999999999
1000000000 1000000000 -999999998 -1000000000
1000000000 1000000000 -999999997 -1000000000
```

**output**

```
7
10
10
0
10
9
1
9
8
10
9
8
```

The first example is shown below.



## E. Imperfect Numbers

0.5 s, 1024 MB

In number theory, a perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, $6$ has divisors $1, 2$ and $3$, and $1 + 2 + 3 = 6$, so $6$ is a perfect number. In this problem, numbers where the sum is too small are *deficient*, and numbers where the sum is too large are *abundant*.

You are given a list of positive integers your program should classify them accordingly.

### Input
The first line of the input contains one positive integer $T$ indicating the number of test cases. The second line of the input contains $T$ positive integers $n_1, \ldots, n_T$. We have $1 \leq n_i, T \leq 10^6$.

### Output
Output $T$ lines, one for each $n_i$ in the input. See the example.

**input**
```
3
28 12 1
```
**output**
```
perfect
abundant
deficient
```

## F. Flakey Ball Placement

1 s., 1024 MB

There are $n$ balls numbered $1, 2, \ldots, n$ about to be sequentially placed into bins numbered $1, \ldots, n$ by a machine. For ball $1$ the machine places it into a random bin that is *not* bin $1$. For each of balls $2, \ldots, n$ the machine tries to place the ball into the bin with its number. If that bin is occupied, it places the ball into a randomly chosen empty bin. (Whenever the machine makes a random choice, all of its viable options are equally probable.)

What's the probability that ball $n$ is placed into some bin other than $n$?

### Input
The input contains the integer $n$ with $2 \leq n \leq 300$.

### Output
Output the required probability with an absolute error of at less than $10^{-6}$.

**input**
```
2
```
**output**
```
1.00000000
```

**input**
```
3
```
**output**
```
0.75000000
```

**input**
```
4
```
**output**
```
0.66666667
```

# G. Monotonic Tree Paths

5 s., 1024 MB

You're given an unrooted tree $T$ of $n$ nodes and $n - 1$ bi-directional edges. Each edge is labeled with a (not necessarily unique) number. The problem is to compute the number of simple (non-self-intersecting) paths of at least one edge such that the sequence of labels on that path is strictly increasing. The paths under consideration may start and end at any pair of distinct vertices.

## Input

The first line contains $n$. The next $n - 1$ lines contain three integers $a_i, b_i, c_i$, where $a_i$ is one end of an edge and $b_i$ is the other end, and $c_i$ is the label on that edge. Here $1 \le n \le 2 \times 10^5$, $1 \le a_i, b_i \le n$, and $0 \le c_i \le 10^9$. The edges, of course, must form a tree.

## Output

Output the number of distinct simple paths with strictly increasing labels.

**input**

```
3
2 3 19
1 2 8
```

**output**

```
5
```

**input**

```
5
1 4 9
1 2 6
2 3 6
5 4 9
```

**output**

```
9
```

In the first example, the five paths are $[1, 2]$, $[2, 1]$, $[2, 3]$,$[3, 2]$, and $[1, 2, 3]$.