# Problem Subarray Sort

| | |
|---|---|
| C header | `subarray.h` |
| C++ header | `subarray.h` |

After spending winter break at his grandpa's, *Little Square* returned home. While he was away, his friend, *Little Triangle*, played with his toys, numbered from 1 to $N$. In order for *Little Square* not to be upset with him, *Little Triangle* has to put his toys back in order: $1, 2, \ldots, N$.

Initially, all the toys are lined up on a shelf in some order.

Knowing that *Little Triangle* can sort a continuous interval $[i, j]$ of toys in $\lfloor\sqrt{j-i+1}\rfloor$ seconds, help him find the minimum time he can order all the toys.

## Interaction protocol

The contestant must implement one function with the following signature:

```
int solve(int N, int P[]);
```

`solve` will be called exactly once.
The function will be supplied with $N$, the number of toys, and $P$, an array (indexed from 0) containing the initial order of the toys. It has to return an `int` representing the minimum time required by *Little Triangle* to sort all the toys.

The sample grader reads the input from standard input with the following format:

- on the first line, the number $N$
- on the second line, the permutation $P$

The sample grader prints the result returned by `solve` to the standard output.

**Attention! The contestant should not implement the main function.**

## Constraints

- $1 \leq N \leq 4 \cdot 10^6$
- $\lfloor x \rfloor$ denotes the greatest integer $k \leq x$.
- Each number between 1 and $N$ will appear exactly once in $P$.
- The grader given to the contestants is not necessarily the same with the grader used for scoring.

## Subtask 1 (7 points)

- $P$ is randomly generated.

## Subtask 2 (8 points)

- $1 \leq N \leq 9$

## Subtask 3 (35 points)

- $1 \leq N \leq 2000$

## Subtask 4 (25 points)

- $1 \leq N \leq 100000$

## Subtask 5 (25 points)

- No additional constraints.

## Example

| input | output |
|---|---|
| 5<br>3 1 4 2 5 | 2 |
| 3<br>1 2 3 | 0 |

## Explanation

In the first example, *Little Triangle* can sort the interval $[0, 1]$ in $\lfloor\sqrt{1-0+1}\rfloor = \lfloor\sqrt{2}\rfloor = \lfloor 1.41421\ldots\rfloor = 1$ second. The permutation becomes 1 3 4 2 5. He can now sort the interval $[1, 3]$ in $\lfloor\sqrt{3-1+1}\rfloor = \lfloor\sqrt{3}\rfloor = \lfloor 1.73205\ldots\rfloor = 1$ second. The permutation becomes 1 2 3 4 5. In total *Little Triangle* can sort all the toys in $1 + 1 = 2$ seconds, which is also the minimum possible time.

In the second example, the toys are already sorted.