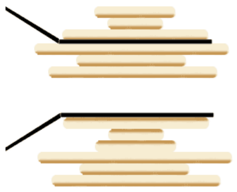
Burnt Pancakes

Statement

Fluffy the Hamster is cooking a stack of pancakes. However, Fluffy has managed to burn *every* single pancake. Fortunately, each pancake only has 1 side burnt. To hide the mess and make it presentable, he has decided to stack them from smallest on top, to largest below, with all burnt sides facing down and hidden.



The only way he can flip the stack of pancakes, is by inserting the spatula anywhere in the middle of the stack, then flipping all pancakes above the spatula together. As to not waste any more time, he wants to do the minimum number of flips possible.

Hint: At most 10 flips are needed to sort a stack of at most 5 burnt pancakes.

Input

The first input gives the number of pancakes N.

The second line contains N integers, corresponding to the pancakes, from top to bottom. The absolute value of the i-th integer P_i is the diameter of the i-th pancake.

The sign of P_i indicates the orientation of the i-th pancake - burnt side **down** if **positive**, burnt side **up** if **negative**.

For reference, the sorted stack of 5 pancakes corresponds to `1 2 3 4 5`. From there, flipping the top 3 pancakes yields the stack `-3 -2 -1 4 5`.

Constraints

- $1 \leq N \leq 5$. (There are at most 5 pancakes in the stack.)
- For each pancake i, $1 \leq |P_i| \leq N$. (Pancake sizes are integers from 1 to N).
- For distinct pancakes i and j, $|P_i| \neq |P_j|$. (All pancakes have different diameters.)

Output

Print the **minimum** number of flips needed to sort the stack of pancakes.

Examples

Sample Input	Expected Output

`2`
`-2 -1`
	`1`

`3`
`2 3 -1`
	`2`

In the first case, flipping all 2 pancakes once solves it.

In the second case, flipping the first 2 pancakes gives us `-3 -2 -1`, which we can then flip all 3 pancakes to get `1 2 3`.

Notes

1. A skeleton file has been given to help you. You should not create a new file or rename the file provided. You should develop your program using this skeleton file.
2. You are free to define your own helper methods and classes (or remove existing ones) if it is suitable but you must put all the new classes, if any, in the same skeleton file provided.

Skeleton File

You are given the skeleton file `Pancake.java`. You should see the following contents when you open the file:

```java
/**
 * Name         :
 * Matric. No   :
*/

import java.util.*;

public class Pancake {
    private void run() {
        // implement your "main" method here
    }

    public static void main(String args[]) {
        Pancake runner = new Pancake();
        runner.run();
    }
}
```