Zig-Zag Array

Given an array of integers, change it in such a way that it follows a zig-zag pattern. A zig-zag array is one where for each integer, its adjacent integers are both greater than or less than itself. In other words, using L to mean a lower value and H to mean higher, the array follows either the pattern [L,H,L,H...] or [H,L,H,L...]. To make the array a zig-zag array, you can replace any element with any other integer (positive, negative, or zero). What is the minimum number of replacements required to accomplish this?

****Exapmle****

*arr = [1, 2, 3, 4, 5]*

Original: [1, 2, 3, 4, 5]

LHLHL: [1, 2, -, 4, -]

HLHLH: [+, 2, 3, -, 5]

To achieve an array starting with a low value, both the 3 and the 5 need to be reduced to any value less than 2 and 4 respectively.

To achieve an array starting with a high value, the 1 needs to be increased (any value > 2) and the 4 needs to be decreased (any value < 3)

In this case, creating either form of zig-zag array takes a minimum of 2 replacements, the final answer.

**Function Description**

Complete the function *minOperations* in the editor below.

minOperations has the following parameter:

    int *arr[n]:*  an array of integers

****Returns****

*int:* the minimum number of operations required to turn *arr* into a zig-zag array

**Constraints**

1 ≤ *n* ≤ 105

1 ≤ *arr[i]* ≤ 109

Input Format For Custom Testing

The first line contains an integer, *n*, the number of elements in *arr*.

Each line *i* of the *n* subsequent lines (where *0 ≤ i < n*) contains an integer, *arr[i]*.

Sample Case 0

**Sample Input For Custom Testing**

STDIN     Function

-----     --------

8     → n = 8

2     → arr = [2, 1, 2, 3, 4, 5, 2, 9]

1

2

3

4

5

2

9

**Sample Output**

2

**Explanation**

Original: [2, 1, 2, 3, 4, 5, 2, 9]

  L H L H L H L H

LHLHLHLH: [2, +, 2, 3, -, 5, 2, 9]

           H L H L H L H H

HLHLHLHL: [2, 1, 2, -, 4, -, 2, -]

For the LHLH... pattern, replace the second value (1) with a number greater than 2 and the fifth value (4) with a number less than 3.

For the HLHL... pattern, replace the fourth value (3) with a number less than 2, and the sixth value (5) and the eighth value (9) with a number less than 2.

The LHLH... pattern only requires two replacements.

Sample Case 1

**Sample Input For Custom Testing**

STDIN     Function

-----     --------

6    →    arr[] size n = 6

1    →    arr = [1, 2, 4, 4, 5, 6]

2

4

4

5

6

**Sample Output**

2

**Explanation**

Original: [1, 2, 4, 4, 5, 6]

  L H L H L H

LHLHLHLH: [1, 2, -, 4, -, 6]

           H L H L H L

HLHLHLHL: [+, 2, 4, -, 5, -]

Starting with a low value takes 2 replacements, while starting with a high value takes 3. Return 2.

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'minOperations' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts INTEGER\_ARRAY arr as parameter.

\*/

public static int minOperations(List<Integer> arr) {

// Write your code here

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int arrCount = Integer.parseInt(bufferedReader.readLine().trim());

List<Integer> arr = IntStream.range(0, arrCount).mapToObj(i -> {

try {

return bufferedReader.readLine().replaceAll("\\s+$", "");

} catch (IOException ex) {

throw new RuntimeException(ex);

}

})

.map(String::trim)

.map(Integer::parseInt)

.collect(toList());

int result = Result.minOperations(arr);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*