Java-找终点-给定一个正整数数组

题目描述：

给定一个正整数数组，设为nums，最大为100个成员，求从第一个成员开始，正好走到数组最后一个成员，所使用的最少步骤数。

要求：

1、第一步必须从第一元素开始，且1<=第一步的步长<len/2;（len为数组的长度，需要自行解析）。

2、从第二步开始，只能以所在成员的数字走相应的步数，不能多也不能少，如果目标不可达返回-1，只输出最少的步骤数量。

3、只能向数组的尾部走，不能往回走。

输入描述：

由正整数组成的数组，以空格分隔，数组长度小于100，请自行解析数据数量。

输出描述：

正整数，表示最少的步数，如果不存在输出-1

补充说明：

示例1

输入：

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

输出：

```
2
```

说明：

第一步： 第一个可选步长选择2，从第一个成员7开始走2步，到达9；第二步： 从9开始，经过自身数字9对应的9个成员到最后。

示例2

输入：

```
1 2 3 7 1 5 9 3 2 1
```

输出：

```
-1
```

说明：

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        Integer[] array = Arrays.stream(scanner.nextLine().split(" ")).map(Integer::parseInt).toArray(Integer[]::new);
        System.out.println(getResult(array));

    }
    public static int getResult(Integer[] array){
        ArrayList<Integer> list = new ArrayList<>();
        for (int i = 0; i < array.length/2; i++) {
            list.add(method(array,i,2));
        }
        return list.stream().filter(ele -> ele>0).min((a,b) ->a-b ).orElse(-1);
    }
    public static int method(Integer[] array,int num,int count){
        int number = num + array[num];
        if (number==array.length -1){
            return  count;
        }else if (number < array.length-1){
            count++;
            return method(array,number,count);
        }else {
            return -1;
        }
    }
}
```