

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  $1 \leq \text{第一步的步长} < \text{len}/2$ ；（`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`

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

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNext()) { // 注意，如果输入是多个测试用例，请通过 while 循环处理多个测试用例
            String s = in.nextLine();
            String[] sArr = s.split(" ");
            int len = sArr.length;
            int[] arr = new int[len];
            for(int i=0; i<len; i++){
                arr[i] = Integer.parseInt(sArr[i]);
            }
            boolean flag = false;
            int minStep = Integer.MAX_VALUE;
            for(int i=1; i<len/2; i++){
                int p = i;
                int step = 1;
                while(p < len-1) {
                    p = p + arr[p];
                    step++;
                }
                if(p == len-1){
                    minStep = Math.min(minStep, step);
                    flag = true;
                }
            }
            System.out.println(flag ? minStep : -1);
        }
    }
}
```