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，不能步步不能少，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，**请自行解析题意获取。**

输出描述

正整数，表示最少的步数，如果不存在输出-1

用例

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，经过后续数字9 8 3的2步个成员到最后。

输入	1 2 3 7 1 6 9 3 2 1
输出	-1
说明	无

题目解析

首先，这题至少要走两步，因为第一步的步长是自选的，只要小于len/2，即可，也就意味第一步走完，应该处于1_len/2的范围中，我们假设处于A位置。

向第二步开始，每次走的步长都是前一步结果所在位置的值，比如第二步要走的步长就是A位置的值，因此有三种可能：

- 第二步要走的步长刚好到达arr.length-1位置，则直接输出数组最2
- 第二步要走的步长小于arr.length-1位置，则说明距离终点还不足，逻辑还是第二步的逻辑，进行下去
- 第二步要走的步长大于arr.length-1位置，则说明当前路径走不通

JavaScript算法源码

可以解开代码中注释log辅助理解

```
// 导入readline模块，用于处理输入和输出
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const arr = line.split(" ").map((ele) => parseInt(ele));

  console.log(getMinStep(arr));
});

// 解法一
function getMinStep(arr) {
  let res = [];
  for (let i = 1; i <= Math.floor(arr.length / 2); i++) {
    // console.log(i);
    res.push(loop(arr, i, 2));
    // console.log("一轮结束", res[res.length - 1], "次");
    // console.log("-----------------------");
  }

  let step = res.filter((ele) => ele > 0).sort((a, b) => a - b)[0];

  return step ? step : -1;
}

function loop(arr, i, count) {
  let j = i + arr[i];
  // console.log(i + "---" + j);

  if (j === arr.length - 1) {
    return count;
  } else if (j < arr.length - 1) {
    count++;
    return loop(arr, j, count);
  } else {
    return -1;
  }
}
```

Java算法源码

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

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    Integer[] arr =
        Arrays.stream(sc.nextLine().split(" ")).map(Integer::parseInt).toArray(Integer[]::new);

    System.out.println(getResult(arr));
  }

  public static int getResult(Integer[] arr) {
    ArrayList<Integer> res = new ArrayList<>();

    for (int i = 1; i <= arr.length / 2; i++) {
      res.add(loop(arr, i, 2));
    }

    return res.stream().filter(ele -> ele > 0).min((a, b) -> a - b).orElse(-1);
  }

  public static int loop(Integer[] arr, int i, int count) {
    int j = i + arr[i];

    if (j == arr.length - 1) {
      return count;
    } else if (j < arr.length - 1) {
      count++;
      return loop(arr, j, count);
    } else {
      return -1;
    }
  }
}
```

Python算法源码

```
# 输入处理
arr = list(map(int, input().split()))


def loop(arr, i, count):
    j = i + arr[i]

    if j == len(arr) - 1:
        return count
    elif j < len(arr) - 1:
        count += 1
        return loop(arr, j, count)
    else:
        return -1


# 解法函数
def getResult():
    res = []
    for i in range(1, len(arr)//2 + 1):
        res.append(loop(arr, i, 2))

    tmp = list(filter(lambda x: x>0, res))

    tmp.sort()

    if len(tmp) > 0:
        return tmp[0]
    else:
        return -1


# 解法调用
print(getResult())
```