# **E卷-找终点[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**

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，正好走到数织最后一个成员，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

示例1

输入

759426835439

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

1237159321

输出

－1

Java

import java.util.Scanner;

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

// 读取输入

Scanner scanner = new Scanner(System.in);

int[] a = new int[100];

int n = 0;

while (scanner.hasNextInt()) {

a[n++] = scanner.nextInt();

}

scanner.close();

// 初始化动态规划数组，dp[i]表示到达第i个位置所需的最少步骤数

int[] dp = new int[n];

Arrays.fill(dp, Integer.MAX\_VALUE);

// 第一个位置的步骤数为0

dp[0] = 0;

// 初始化从第一个位置开始的步长，步长范围为1到len/2

for (int i = 1; i < n / 2; i++) {

dp[i] = 1;

}

// 遍历数组，更新动态规划数组

for (int i = 0; i < n; i++) {

if (i + a[i] >= n) {

continue;

}

dp[i + a[i]] = Math.min(dp[i + a[i]], dp[i] + 1);

}

// 输出结果，如果最后一个位置的步数仍为无穷大，表示不可达，输出-1

System.out.println(dp[n - 1] == Integer.MAX\_VALUE ? -1 : dp[n - 1]);

}

}

Python3

# 读取输入的正整数数组

a = list(map(int, input().split()))

# 获取数组的长度

n = len(a)

# 初始化动态规划数组，dp[i]表示到达第i个位置所需的最少步骤数

dp = [float('inf')] \* n

# 第一个位置的步骤数为0，因为从这里开始

dp[0] = 0

# 初始化从第一个位置开始的步长，步长范围为1到len/2

for i in range(1, n // 2):

dp[i] = 1

# 遍历数组，更新动态规划数组

for i in range(n):

# 如果当前位置的步长超出数组范围，跳过

if i + a[i] >= n:

continue

# 更新到达当前位置的步数

dp[i + a[i]] = min(dp[i + a[i]], dp[i] + 1)

# 输出结果，如果最后一个位置的步数仍为无穷大，表示不可达，输出-1

print(-1 if dp[n - 1] == float('inf') else dp[n - 1])



#include <iostream>

#include <vector>

#include <climits>

using namespace std;

int main() {

// 读取输入

vector<int> a;

int num;

while (cin >> num) {

a.push\_back(num);

}

// 获取数组的长度

int n = a.size();

// 初始化动态规划数组，dp[i]表示到达第i个位置所需的最少步骤数

vector<int> dp(n, INT\_MAX);

// 第一个位置的步骤数为0

dp[0] = 0;

// 初始化从第一个位置开始的步长，步长范围为1到len/2

for (int i = 1; i < n / 2; ++i) {

dp[i] = 1;

}

// 遍历数组，更新动态规划数组

for (int i = 0; i < n; ++i) {

if (i + a[i] >= n) {

continue;

}

dp[i + a[i]] = min(dp[i + a[i]], dp[i] + 1);

}

// 输出结果，如果最后一个位置的步数仍为无穷大，表示不可达，输出-1

if (dp[n - 1] == INT\_MAX) {

cout << -1 << endl;

} else {

cout << dp[n - 1] << endl;

}

return 0;

}

C语言

#include <stdio.h>

#include <limits.h>

int main() {

// 读取输入

int a[100];

int n = 0;

while (scanf("%d", &a[n]) != EOF) {

n++;

}

// 初始化动态规划数组，dp[i]表示到达第i个位置所需的最少步骤数

int dp[100];

for (int i = 0; i < n; ++i) {

dp[i] = INT\_MAX;

}

// 第一个位置的步骤数为0

dp[0] = 0;

// 初始化从第一个位置开始的步长，步长范围为1到len/2

for (int i = 1; i < n / 2; ++i) {

dp[i] = 1;

}

// 遍历数组，更新动态规划数组

for (int i = 0; i < n; ++i) {

if (i + a[i] >= n) {

continue;

}

if (dp[i + a[i]] > dp[i] + 1) {

dp[i + a[i]] = dp[i] + 1;

}

}

// 输出结果，如果最后一个位置的步数仍为无穷大，表示不可达，输出-1

if (dp[n - 1] == INT\_MAX) {

printf("-1\n");

} else {

printf("%d\n", dp[n - 1]);

}

return 0;

}

JsNode

const readline = require("readline");

// 创建接口读取输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.on("line", (input) => {

// 将输入的字符串转换为数组

const a = input.split(" ").map(Number);

// 获取数组的长度

const n = a.length;

// 初始化动态规划数组，dp[i]表示到达第i个位置所需的最少步骤数

const dp = new Array(n).fill(Infinity);

// 第一个位置的步骤数为0，因为从这里开始

dp[0] = 0;

// 初始化从第一个位置开始的步长，步长范围为1到len/2

for (let i = 1; i < n / 2; i++) {

dp[i] = 1;

}

// 遍历数组，更新动态规划数组

for (let i = 0; i < n; i++) {

// 如果当前位置的步长超出数组范围，跳过

if (i + a[i] >= n) {

continue;

}

// 更新到达当前位置的步数

dp[i + a[i]] = Math.min(dp[i + a[i]], dp[i] + 1);

}

// 输出结果，如果最后一个位置的步数仍为无穷大，表示不可达，输出-1

console.log(dp[n - 1] === Infinity ? -1 : dp[n - 1]);

});

GO

package main

import (

"bufio"

"fmt"

"math"

"os"

"strconv"

"strings"

)

func main() {

// 读取输入

scanner := bufio.NewScanner(os.Stdin)

scanner.Scan()

input := scanner.Text()

parts := strings.Fields(input)

var a []int

for \_, part := range parts {

num, \_ := strconv.Atoi(part)

a = append(a, num)

}

// 获取数组的长度

n := len(a)

// 初始化动态规划数组，dp[i]表示到达第i个位置所需的最少步骤数

dp := make([]int, n)

for i := range dp {

dp[i] = math.MaxInt32

}

// 第一个位置的步骤数为0

dp[0] = 0

// 初始化从第一个位置开始的步长，步长范围为1到len/2

for i := 1; i < n/2; i++ {

dp[i] = 1

}

// 遍历数组，更新动态规划数组

for i := 0; i < n; i++ {

if i+a[i] >= n {

continue

}

if dp[i+a[i]] > dp[i]+1 {

dp[i+a[i]] = dp[i] + 1

}

}

// 输出结果，如果最后一个位置的步数仍为无穷大，表示不可达，输出-1

if dp[n-1] == math.MaxInt32 {

fmt.Println("-1")

} else {

fmt.Println(dp[n-1])

}

}