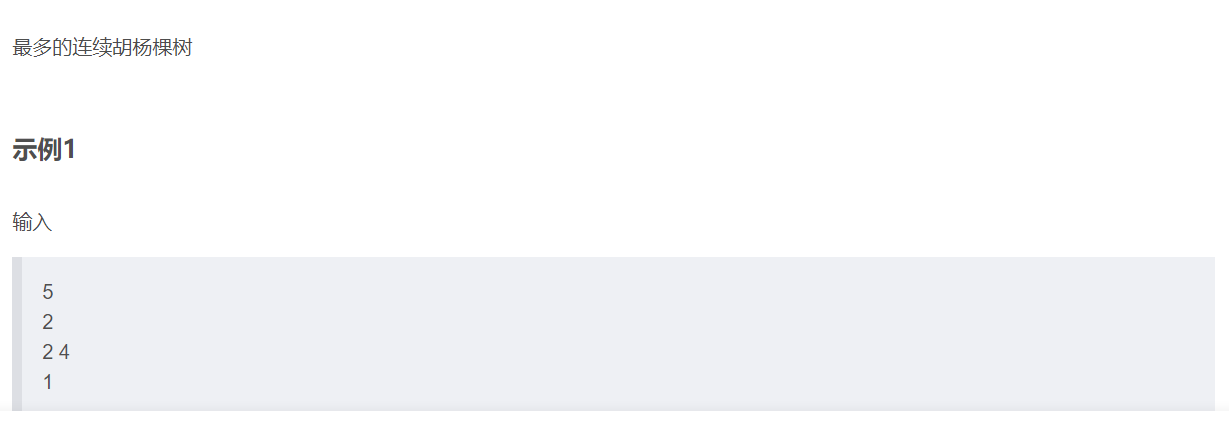
# **E卷-补种未成活胡杨[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**











import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in); // 创建Scanner对象以便读取输入

int n = sc.nextInt(); // 读入总种植数量N

int[] tree = new int[n]; // 创建一个长度为N的数组用于存储每棵树的状态

int m = sc.nextInt(); // 读入未成活胡杨数量M

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

int x = sc.nextInt(); // 读入每棵未成活胡杨的编号

tree[x - 1] = 1; // 将对应位置的值设为1，表示未成活

}

int cnt = sc.nextInt(); // 读入最多可以补种的数量K

int tot = 0, ans = 0, j = 0; // 初始化变量：总未成活树数、最大连续胡杨数、滑动窗口起始位置

for (int i = 0; i < n; i++) { // 遍历所有树的位置

tot += tree[i]; // 每次遍历都将当前树的状态加到tot，如果未成活则+1，成活则+0

while (tot > cnt) { // 如果未成活树的总数超过了补种限制

tot -= tree[j]; // 减去窗口起始位置的树的状态

j += 1; // 滑动窗口的起始位置右移

}

if (i - j + 1 > ans) { // 计算当前滑动窗口的大小，如果比之前的最大连续数大则更新

ans = i - j + 1;

}

}

System.out.println(ans); // 输出最大连续胡杨数

}

}



# 输入和输出操作

n = int(input()) # 读入总种植数量N

tree = [0] \* n # 创建一个长度为N的列表用于存储每棵树的状态

# 读入未成活胡杨数量M

m = int(input())

unlived = list(map(int, input().split())) # 读入未成活胡杨的编号

for x in unlived:

tree[x - 1] = 1 # 将对应位置的值设为1，表示未成活

# 读入最多可以补种的数量K

k = int(input())

tot = 0 # 用于记录当前窗口未成活树的数量

ans = 0 # 用于记录最大连续胡杨数

j = 0 # 滑动窗口的起始位置

# 遍历所有树的位置

for i in range(n):

tot += tree[i] # 如果当前树未成活，tot加1

while tot > k: # 如果未成活树的总数超过了补种限制

tot -= tree[j] # 减去窗口起始位置的树的状态

j += 1 # 滑动窗口的起始位置右移

ans = max(ans, i - j + 1) # 更新最大连续胡杨数

print(ans) # 输出最大连续胡杨数



#include <iostream>

using namespace std;

int tree[100005]; // 创建一个长度为100005的数组用于存储树的状态

int main() {

int n;

cin >> n; // 读入总种植数量N

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

tree[i] = 0; // 初始化数组，所有树初始状态为成活（0）

}

int m;

cin >> m; // 读入未成活胡杨数量M

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

int x;

cin >> x; // 读入未成活胡杨的编号

tree[x - 1] = 1; // 将对应位置的值设为1，表示未成活

}

int cnt;

cin >> cnt; // 读入最多可以补种的数量K

int tot = 0, ans = 0, j =

0; // 初始化变量：总未成活树数、最大连续胡杨数、滑动窗口起始位置

for (int i = 0; i < n; ++i) { // 遍历所有树的位置

tot += tree[i]; // 如果当前树未成活，tot加1

while (tot >

cnt) { // 如果当前窗口未成活树的总数超过了补种限制

tot -= tree[j]; // 减去窗口起始位置的树的状态

j += 1; // 滑动窗口的起始位置右移

}

ans = max(ans, i - j + 1); // 更新最大连续胡杨数

}

cout << ans << endl; // 输出最大连续胡杨数

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int tree[100005]; // 用于存储每棵树的状态，0表示成活，1表示未成活

int main() {

int n;

scanf("%d", &n); // 读入总种植数量N

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

tree[i] = 0; // 初始化数组，所有树初始状态为成活（0）

}

int m;

scanf("%d", &m); // 读入未成活胡杨数量M

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

int x;

scanf("%d", &x); // 读入每棵未成活胡杨的编号

tree[x - 1] = 1; // 将对应位置的值设为1，表示未成活

}

int cnt;

scanf("%d", &cnt); // 读入最多可以补种的数量K

int tot = 0, ans = 0, j =

0; // 初始化变量：总未成活树数、最大连续胡杨数、滑动窗口起始位置

for (int i = 0; i < n; ++i) { // 遍历所有树的位置

tot += tree[i]; // 每次遍历都将当前树的状态加到tot，如果未成活则+1，成活则+0

while (tot > cnt) { // 如果未成活树的总数超过了补种限制

tot -= tree[j]; // 减去窗口起始位置的树的状态

j += 1; // 滑动窗口的起始位置右移

}

if (i - j + 1 >

ans) { // 计算当前滑动窗口的大小，如果比之前的最大连续数大则更新

ans = i - j + 1;

}

}

printf("%d", ans); // 输出最大连续胡杨数

}



const readline = require("readline");

// 创建接口以便读取输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

let input = []; // 用于存储输入

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

input.push(line.trim());

});

rl.on("close", () => {

let n = parseInt(input[0]); // 读取总种植数量N

let tree = new Array(n).fill(0); // 创建一个长度为N的数组用于存储树的状态

let m = parseInt(input[1]); // 读取未成活胡杨数量M

let unlived = input[2].split(" ").map(Number); // 读取未成活胡杨的编号

for (let x of unlived) {

tree[x - 1] = 1; // 将对应位置的值设为1，表示未成活

}

let k = parseInt(input[3]); // 读取最多可以补种的数量K

let tot = 0,

ans = 0,

j = 0; // 初始化变量：总未成活树数、最大连续胡杨数、滑动窗口起始位置

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

// 遍历所有树的位置

tot += tree[i]; // 如果当前树未成活，tot加1

while (tot > k) {

// 如果未成活树的总数超过了补种限制

tot -= tree[j]; // 减去窗口起始位置的树的状态

j += 1; // 滑动窗口的起始位置右移

}

ans = Math.max(ans, i - j + 1); // 更新最大连续胡杨数

}

console.log(ans); // 输出最大连续胡杨数

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

reader := bufio.NewReader(os.Stdin) // 创建一个读取器以从标准输入读取数据

// 读取第一行输入

line, \_ := reader.ReadString('\n') // 从标准输入读取一行数据

n, \_ := strconv.Atoi(strings.TrimSpace(line)) // 将读入的字符串转换为整数，表示总种植数量N

tree := make([]int, n) // 创建一个长度为N的切片用于存储每棵树的状态

// 读取第二行输入

line, \_ = reader.ReadString('\n') // 从标准输入读取第二行数据

// 读取第三行输入

line, \_ = reader.ReadString('\n') // 从标准输入读取第三行数据

unlived := strings.Fields(line)

for \_, v := range unlived {

x, \_ := strconv.Atoi(v) // 将每个未成活胡杨的编号字符串转换为整数

tree[x - 1] = 1 // 将对应位置的值设为1，表示未成活

}

// 读取第四行输入

line, \_ = reader.ReadString('\n') // 从标准输入读取第四行数据

k, \_ := strconv.Atoi(strings.TrimSpace(line)) // 将读入的字符串转换为整数，表示最多可以补种的数量K

tot := 0 // 用于记录当前窗口未成活树的数量

ans := 0 // 用于记录最大连续胡杨数

j := 0 // 滑动窗口的起始位置

// 遍历所有树的位置

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

tot += tree[i] // 如果当前树未成活，tot加1

for tot > k { // 如果未成活树的总数超过补种限制

tot -= tree[j] // 减去窗口起始位置的树的状态

j += 1 // 滑动窗口的起始位置右移

}

if (i - j + 1) > ans { // 如果当前窗口大小超过记录的最大连续数

ans = i - j + 1 // 更新最大连续胡杨数

}

}

fmt.Println(ans) // 输出最大连续胡杨数

}