习题课

TEACHING COURSEWARE POWPOINT

授课时间：2025.08.05

目录

01

题目描述

TEACHING
COURSEWARE

TEACH

题目描述

题目描述

一年一度的"跳石头"比赛又要开始了！

这项比赛将在一条笔直的河道中进行，河道中分布着一些巨大岩石。组委会已经选择好了两块岩石作为比赛起点和终点。在起点和终点之间，有 N 块岩石（不含起点和终点的岩石）。在比赛过程中，选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

为了提高比赛难度，组委会计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制，组委会至多从起点和终点之间移走 M 块岩石（不能移走起点和终点的岩石）。

输入格式

第一行包含三个整数 L, N, M，分别表示起点到终点的距离，起点和终点之间的岩石数，以及组委会至多移走的岩石数。保证 $L \geq 1$ 且 $N \geq M \geq 0$。

接下来 N 行，每行一个整数，第 i 行的整数 $D_i\ (0 < D_i < L)$，表示第 i 块岩石与起点的距离。这些岩石按与起点距离从小到大的顺序给出，且不会有两个岩石出现在同一个位置。

题目描述

输入输出样例

输入 #1 复制

```
25 5 2
2
11
14
17
21
```

输出 #1 复制

```
4
```

说明/提示

输入输出样例 1 说明

将与起点距离为 2 和 14 的两个岩石移走后，最短的跳跃距离为 4（从与起点距离 17 的岩石跳到距离 21 的岩石，或者从距离 21 的岩石跳到终点）。

数据规模与约定

对于 20%的数据，$0 \leq M \leq N \leq 10$。
对于 50% 的数据，$0 \leq M \leq N \leq 100$。
对于 100% 的数据，$0 \leq M \leq N \leq 50000, 1 \leq L \leq 10^9$。

02

知识回归

TEACHING
COURSEWARE

TEACH

概念分析

单调性：如果值X可行，则所有≤X（或≥X）的值都可行

概念

二分答案是一种解题思路，常用于求满足条件的最值问题（如最大值最小化、最小值最大化）。其核心是通过二分法“猜测”答案，并验证该答案是否符合条件，逐步逼近最优解。

核心逻辑

确定答案的可能范围（左边界left为最小值，右边界right为最大值）；

计算中点mid作为“候选答案”，设计验证函数判断mid是否满足条件；

根据验证结果调整范围：

若mid满足条件，说明最优解可能在mid或其右侧（求最大值时），更二分答案的本质：当问题符合单调性条件时，通过二分法快速找到最优解

03

解题思路

TEACHING
COURSEWARE

TEACH

解题思路

核心思路：采用二分答案法，寻找“最小跳跃距离的最大值”。即通过二分可能的距离值，检查该距离是否可行（即移走不超过 M 块岩石后，所有跳跃距离都不小于该值）。

check 函数：

模拟跳跃过程，从起点（0）开始，遍历所有岩石和终点（已将终点纳入数组统一处理）。

- 若当前位置（岩石或终点）与上一保留位置的距离小于 mid，说明需要移走该岩石（计数 + 1）。
- 若移走数量超过 M，返回不可行（该距离无法实现）。
- 若距离不小于 mid，则保留当前位置，更新上一保留位置。

遍历结束后，若移走数量未超过 M，说明该距离可行。

二分过程：

初始范围：left=1（最小可能距离），right=L（最大可能距离，即起点到终点的距离）。

用 ans 记录可行的最大距离。

- 当 check(mid) 可行时，说明当前距离可以实现，尝试更大的距离（更新 ans=mid，left=mid+1）。
- 当 check(mid) 不可行时，说明当前距离无法实现，尝试更小的距离（right=mid-1）。

最终 ans 即为“最小跳跃距离的最大值”。

解题思路

当「需要移走的岩石数 > 最多可移走的岩石数M」时，说明当前猜测的最短跳跃距离太大，无法通过移走M块岩石实现，因此需要缩小距离，进入左区间 right = mid - 1

当「需要移走的岩石数 = 最多可移走的岩石数M」时，此时刚好能通过移走M块岩石实现该距离，但可能存在更大的可行距离，因此需要尝试更大的距离，进入右区间 left = mid + 1

当「需要移走的岩石数 < 最多可移走的岩石数M」时，说明当前猜测的最短跳跃距离太小，即使移走更少的岩石也能实现，因此可以尝试更大的距离，进入右区间 left = mid + 1

综合三种情况就有：

当 需要移走的岩石数 > M 时，进入左区间

当 需要移走的岩石数 ≤ M 时，进入右区间

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序

```cpp
#include <iostream>
using namespace std;
const int MN = 50010;
int L, N, M;
int a[MN];  // 存储岩石位置，从1开始
// 检查距离mid是否可行：能否移走不超过M块岩石
int main() {
    cin >> L >> N >> M;
    for (int i = 1; i <= N; i++) {
        cin >> a[i];
    }
    a[N + 1] = L;  // 将终点加入数组，统一处理最后一段距离
    int left = 1;          // 最短可能的距离（至少为1）
    int right = L;         // 最长可能的距离（起点到终点）
    int ans = 0;           // 记录最优答案
    // 二分最大的可行距离
    while (left <= right) {
        int mid = left + (right - left) / 2;  // 避免溢出
        if (check(mid)) {
            // 当前距离可行，尝试更大的距离
            ans = mid;
            left = mid + 1;
        } else {
            // 当前距离不可行，尝试更小的距离
            right = mid - 1;
        }
    }

    cout << ans << endl;
    return 0;
}
```

标准程序


```
// 检查距离mid是否可行：能否移走不超过M块岩石，使最短跳跃距>=mid
bool check(int mid) {
    int cnt = 0;          // 已移走的岩石数量
    int last = 0;         // 上一个保留的位置（起点为0）
    // 遍历所有岩石和终点（a[N+1]是终点）
    for (int i = 1; i <= N + 1; i++) {
        // 当前位置与上一保留位置的距离小于mid，需要移走
        if (a[i] - last < mid) {
            cnt++;
            if (cnt > M) return false;  // 超过最大可移走数量，不可行
        } else {
            last = a[i];  // 保留当前位置，更新上一保留位置
        }
    }
    // 所有跳跃距均满足条件，且移走数量不超过M
    return true;
}
```