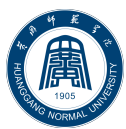
二分答案与三分法

把“直接算最优解”转化成“猜答案并 $O(n)$ 判可行”

HGNU-ACM 实验室算法组

承接第 3 章《高精度运算》，迈入值域砍半查找与单峰极值搜索





本章目录（1/2）

单调性判别与整数二分边界

二分答案与 check 函数建模

实数二分：固定迭代次数 vs eps

前三节打通核心原理：整数二分防死循环边界、二分答案与 check 建模、实数二分固定百轮收敛。

本章目录（2/2）

单峰与单谷函数的三分法

四大二分与三分经典真题精讲

常见错误、检查清单与练习安排

后三节掌握单峰函数三分法，精讲《砍树》《愤怒的牛》《河中跳房子》《曲线》与章末排错清单。

从猜数字开始：写二分前先回答三个问题

生活小例子：在递增数组 $[2, 5, 8, 12, 19]$ 中找第一个 ≥ 7 的位置

第 3 章学过怎么算超大整数；这一章我们面对另一类“大”：答案范围很大。
就像猜价格游戏，范围 $1 \sim 1000$，从 1 往上一块块猜太慢；每次猜正中间，根据“偏高/偏低”直接扔掉一半！

- 写二分前先回答三个问题：
  1. 待查区间 $[l, r]$ 的起点和终点是什么？（如数组下标 $[0, 4]$）
  2. 看到中点 mid 后，怎么判断它“够不够格”？（如 $a[mid] \geq 7$ 吗？）
  3. 判断结果出来后，留左半边还是留右半边？

本页要点

核心直觉：不是一个一个试，而是每一刀都把不可能的那一半直接扔掉。

为什么暴力从头扫会超时（TLE）？

如果只在 5 个数里找，循环一遍无所谓；但当范围达到 10^9 甚至 10^{18} 时：

问题规模 V	暴力逐一试	二分砍半次数	1 秒约 10^8 次操作结论
100	100 次	7 次	都能秒过
10^5	10^5 次	17 次	暴力勉强，若内层再乘 n 就危险
10^9	10^9 次	**30** 次	暴力约 10 秒必 **TLE**，二分稳过
10^{18}	10^{18} 次	**60** 次	暴力跑几百年，二分不到一毫秒

- 每次把区间长度除以 2，k 轮后只剩 $V/2^k$；当 $2^{30} \approx 1.07 \times 10^9$ 时只需 30 轮。

本页要点

看到 10^9 级别的值域或有序长序列，逐个枚举必 TLE，立刻想到折半查找。

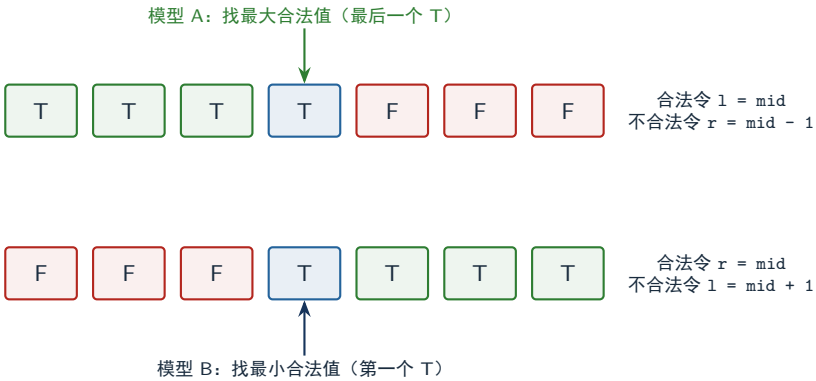
三个新术语大白话

- 单调性（**Monotonicity**）：条件是否成立像排队站成两截——要么前面全成立、后面全不成立（TTTT...FFFF），要么前面全不成立、后面全成立（FFFF...TTTT），绝不会出现“成立-不成立-又成立”反复横跳。
- 可行性分界点：把区间切成一真一假两半的那条边界——我们要找的要么是最后一个真（最大合法值），要么是第一个真（最小合法值）。
- 闭区间 $[l, r]$：左端点 l 和右端点 r 都包含在候选答案里；只要 $l < r$ 就说明候选至少还有 2 个，继续折半；当 $l == r$ 时只剩唯一候选，循环结束。

本页要点

没有单调性（不能切出清晰的 T/F 分界线），就不能直接用二分。

TikZ 图解：分清“最大合法值”与“最小合法值”



本页要点

先判断答案落在 T 的右端点（模型 A）还是左端点（模型 B），再选对应更新方式。

手算跟踪（一）：在 $[2,5,8,12,19]$ 找第一个 ≥ 7 的下标

这是模型 B（F F T T T），初始 $l=0,\ r=4$，取 $mid=l+(r-l)/2$，若 $a[mid]\geq 7$ 令 $r=mid$ 否则 $l=mid+1$：

轮次	$l<r$?	$[l,r]$	mid	$a[mid]\geq 7$?	更新动作	说明
1	真	$[0,4]$	2	$a[2]=8\geq 7$（真）	$r\leftarrow 2$	2 号位合法，答案 ≤ 2
2	真	$[0,2]$	1	$a[1]=5\geq 7$（假）	$l\leftarrow 2$	1 号位太小，必舍弃
3	假	$[2,2]$	—	退出循环	答案下标 **2**	$l==r==2$，对应数值 8

- 注意第 3 步：$l=2,r=2$ 时 $l<r$ 为假，不再进循环，直接输出 $l=2$。

本页要点

每轮先看 $l<r$ 是否成立；只有为真才算 mid，最后条件为假时 l 和 r 重合。

手算跟踪（二）：为什么 `l = mid` 漏掉 +1 会死循环？

假设区间只剩 $[3,4]$，其中 3 合法（T）、4 不合法（F），我们要找最大合法值 3：

写法对比	$[l,r]$	mid	条件真假	更新后 $[l,r]$	结局
错：下取整 `(l+r)/2`	$[3,4]$	3	`ok(3)` 为真	$l \leftarrow 3 \Rightarrow [3,4]$	原地踏步，死循环 **TLE**！
错：下取整（第 2 轮）	$[3,4]$	3	`ok(3)` 为真	$l \leftarrow 3 \Rightarrow [3,4]$	永远出不去
对：上取整 `(l+r+1)/2`	$[3,4]$	4	`ok(4)` 为假	$r \leftarrow 3 \Rightarrow [3,3]$	下一轮 $l<r$ 为假，退出！

- 只有 $r=l+1$ 时整数除法 $\lfloor (l+r)/2 \rfloor = l$。若分支又写 `l = mid`，区间长度还是 2！
- 加上 +1 变成上取整，$mid=4=r$：真则缩成 $[4,4]$，假则缩成 $[3,3]$，必减半。

本页要点

铁律：分支里有 `l = mid` 就必须用上取整 +1；分支里有 `r = mid` 就用下取整！

正确性证明：闭区间循环不变量

- 断言（循环不变量）：只要保证输入存在至少一个合法解，每轮开始判断 $l < r$ 时，真正答案 x^* 始终落在闭区间 $[l, r]$ 内。

阶段	验证内容（以模型 A 找最大合法值为例）
初始化	初始 l 取最小可能值、r 取最大可能值，覆盖全部候选，$x^* \in [l, r]$ 成立
保持	若 $ok(mid)$ 为真，则 mid 及右侧可能有更优解，舍弃 $< mid$，令 $l = mid$ 后 $x^* \in [l, r]$；若为假，单调性保证 $\geq mid$ 全假，令 $r = mid - 1$ 后 $x^* \in [l, r]$
终止	因有上取整，每轮长度严格减半，必在有限步到达 $l == r$；此时 $[l, l]$ 只剩一个元素，即 x^*

本页要点

既保证“不漏掉真答案”，又保证“区间长度每轮严格缩小”，程序就既不会 WA 也不会 TLE。

两套万能整数二分模板（背下这两段就够了）

```
// 模板 A：求【最大】可行答案（分布 TTTT...FFFF）
while (l < r) {
    long long mid = l + (r - l + 1) / 2; // 防 l+r 溢出，且上取整 +1 防死循环
    if (ok(mid)) l = mid;                // mid 合法，保留它并试探更大的右半边
    else         r = mid - 1;            // mid 不合法，彻底扔掉 mid 及右边
}
// 模板 B：求【最小】可行答案（分布 FFFF...TTTT）
while (l < r) {
    long long mid = l + (r - l) / 2;     // 下取整即可，r = mid 不会卡死在 [l, l+1]
    if (ok(mid)) r = mid;                // mid 合法，保留它并试探更小的左半边
    else         l = mid + 1;            // mid 不合法，必须严格右移一格
}
```

本页要点

不写 (l+r)>>1 防 2×10^9 相加溢出；记住：l=mid 配 +1，r=mid 不加 +1。

上机验证与反事实追问

把下面短程序跑通，测 3 组输入验证（在有序数组 a = {2, 5, 8, 12, 19} 找第一个 $\geq x$ 的下标）：

```
int a[5] = {2, 5, 8, 12, 19}, x;
cin >> x;
int l = 0, r = 4;
while (l < r) {
    int mid = l + (r - l) / 2;
    if (a[mid] >= x) r = mid;
    else l = mid + 1;
}
cout << (a[l] >= x ? l : -1) << '\n';    // 退出后需特判“全都不合法”的情况
```

- **3** 组验证：输入 7 输出 2；最小边界 1 输出 0；特殊无解 99 输出 -1。
- 反事实追问：如果把循环条件 `while (l < r)` 改成 `while (l <= r)`，且分支仍保留 `r = mid`，输入 7 会发生什么？答：当缩到 $l = 2, r = 2$ 时会一直执行 `r = 2` 死循环 TLE！

整数二分边界易错对照（WA / TLE / RE）

场景	错误写法	正确做法	评测结果
模型 A 算中点	写了 `l = mid` 却用 `(l+r)/2`	必须写成 `l + (r - l + 1) / 2`	**TLE**
模型 B 算中点	写了 `r = mid` 却加了 `+1`	不加 `+1`，写成 `l + (r - l) / 2`	**TLE**
循环条件错用等号	`while (l <= r)` 配 `r = mid`	留住边界必须写严格 `while (l < r)`	**TLE**
中点相加溢出	`int mid = (l + r) / 2;`	l, r 达 2×10^9 变负数，改 `l + (r - l) / 2`	**WA/RE**
全无解未特判	退出后直接输出 `l`	退出后需再检查一次 `ok(l)` 是否真合法	**WA**

本页要点

二分遇 TLE 查取整方向和 <；遇 WA 查溢出和无解特判。

生活小例子：不知道锯片设多高，能先“猜一个”试切吗？

具体例子：面前有 4 棵树高 $[20, 15, 10, 17]$，要切出至少 7 米木头，锯片最高设多少？

如果让你正面反推出锯片高度 H，需要按高度排序还要算前缀和，很容易写乱；
但如果我们倒过来猜一个高度，事情瞬间变简单：

- 猜 $H = 15$：高于 15 的截下来，共 $(20 - 15) + 0 + 0 + (17 - 15) = 7 \geq 7$ 米 $\Longrightarrow$ 可行！
- 猜 $H = 16$：高于 16 的截下来，共 $4 + 0 + 0 + 1 = 5 < 7$ 米 $\Longrightarrow$ 不可行！

- 题目常在题面里直接藏着提示词：“最小值最大”或 “最大值最小”。

本页要点

正面算不出来时，先问一句：如果告诉我一个答案 x，我能不能 $O(n)$ 验证它行不行？

为什么暴力枚举答案会 TLE？

在“砍树”问题中，$n \le 5 \times 10^5$，单棵树高度 $h_i \le 10^9$：

- 暴力做法：从 $H = 10^9$ 往下每次减 1，对每个 H 扫描 n 棵树求和判断 $\ge M$。
- 操作次数：最多试 10^9 个高度 $\times$ 每次扫 5×10^5 棵树 $= \mathbf{5 \times 10^{14}}$ 次计算！按 10^8/秒，要跑 **58 天 **，必 **TLE**。
- 二分答案做法：高度范围 $[0, 10^9]$ 只需二分 $\log_2(10^9) \approx 30$ 轮；总操作次数只有 $30 \times 5 \times 10^5 = \mathbf{1.5 \times 10^7}$ 次，约 0.05 秒 AC！

方案	计算式	耗时估计
暴力试所有高度	$H \times n \approx 5 \times 10^{14}$	必 **TLE**
二分答案 + $O(n)$ 校验	$\log_2(\max h) \times n \approx 1.5 \times 10^7$	< 0.1 秒 **AC**

本页要点

二分答案把“不可承受的 $O(V \cdot n)$”降维成了“秒过的 $O(n \log V)$”。

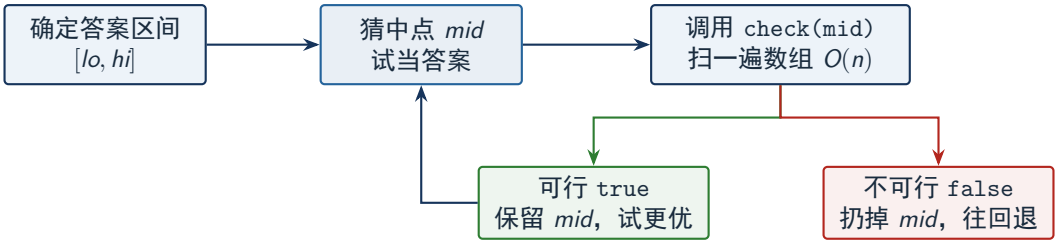
新术语大白话：二分答案、check 与对偶问法

- 二分答案：二分的不是题目给的数组下标，而是最终要输出的答案本身（如锯片高度、最短距离、最大分段和）。
- `check(mid)` 检验函数：给定一个猜出来的候选答案 `mid`，在 $O(n)$ 时间内返回 `true`（可行）或 `false`（不可行）的布尔函数。
- 最大化最小值：例如《愤怒的牛》《河中跳房子》，让最差的那段尽可能大。猜间距 d，越大越难放下（`TTT...FFF`），用模板 **A**。
- 最小化最大值：例如把数组分 m 段让“最大段和”尽量小。猜上限 S，上限越大越容易分下（`FFF...TTT`），用模板 **B**。

本页要点

“最大化的最小”选模板 A（最后一个 T）；“最小化的最大”选模板 B（第一个 T）。

TikZ 图解：二分答案的“猜—验—缩”工作流



- 在 `check(mid)` 里把待求最值当成已知常数，原题往往立刻退化成一步步贪心模拟！

本页要点

写二分答案只有两步：先写一个只干一件事的 `check(mid)`，再把它套进模板 A 或 B。

手算跟踪表：砍树 $[20, 15, 10, 17]$，$M = 7$ 全过程

初始 $lo = 0$，$hi = 20$，找最大合法高度（模板 A：`mid = lo + (hi - lo + 1) / 2`）：

轮次	$lo < hi$?	$[lo, hi]$	mid	切下木头 $\sum \max(0, h_i - mid)$	≥ 7?	动作
1	真	$[0, 20]$	10	$10 + 5 + 0 + 7 = 22$	真	$lo \leftarrow 10$
2	真	$[10, 20]$	15	$5 + 0 + 0 + 2 = 7$	真	$lo \leftarrow 15$
3	真	$[15, 20]$	18	$2 + 0 + 0 + 0 = 2$	假	$hi \leftarrow 17$
4	真	$[15, 17]$	16	$4 + 0 + 0 + 1 = 5$	假	$hi \leftarrow 15$
5	假	$[15, 15]$	—	条件为假，退出循环	—	输出 $lo = \mathbf{15}$

本页要点

区间仅用 4 次校验就从 21 个整数精确锁定到 15，第 5 轮条件为假直接退出。

为什么这样建模一定对？单调性证明三问

在套用二分答案之前，必须口头验证下面三条（以“砍树”和“数列分段”为例）：

检验问题	砍树（最大合法高度 H）	数列分段（最小化最大段和 S）
1. 为什么单调？	H 越大，每棵树截掉越少，总木头单调不增	上限 S 越大，每段能装下更多数，所需段数单调不增
2. check 怎么算？	直接逐棵树累加 $\max(0, h_i - H)$，与 M 比较	贪心往当前段塞数，超 S 就新开一段，看段数 $\le m$
3. 区间覆盖答案吗？	锯片最低 0（全砍），最高 $\max(h_i)$（不砍）	下界 $\max(a_i)$（单元素），上界 $\sum a_i$（全合一段）

本页要点

单调性成立 $+$ check 贪心不劣 $+$ 初始 $[lo, hi]$ 包住极端解 $\Longrightarrow$ 二分答案绝对正确。

二分答案通用骨架与“最小化最大值”示例

```cpp
auto ok = [&](long long lim) {
    int seg = 1; long long cur = 0;       // 至少 1 段，当前段和 cur
    for (long long x : a) {
        if (cur + x > lim) { ++seg; cur = x; } // 放不下就新开一段
        else cur += x;
    }
    return seg <= m;                      // 段数不超过 m 即合法
};
long long lo = *max_element(a.begin(), a.end()); // 下界：最大单元素
long long hi = accumulate(a.begin(), a.end(), 0LL); // 上界：全部和（0LL 防溢出）
while (lo < hi) {
    long long mid = lo + (hi - lo) / 2;  // 模板 B：求最小合法上限
    if (ok(mid)) hi = mid; else lo = mid + 1;
}
```

本页要点

accumulate 第 3 个参数必须写 0LL，否则会按 int 累加直接溢出！

上机验证与反事实追问

用下面“砍树最小测试程序”（$n = 3$）验证 3 组数据：

```
int n; long long m; cin >> n >> m;
vector<long long> h(n); long long lo = 0, hi = 0;
for (int i = 0; i < n; ++i) { cin >> h[i]; hi = max(hi, h[i]); }
while (lo < hi) {
    long long mid = lo + (hi - lo + 1) / 2, wood = 0;
    for (long long x : h) if (x > mid) wood += x - mid;
    if (wood >= m) lo = mid; else hi = mid - 1;
}
cout << lo << '\n';
```

- **3** 组输入：(1) 普通 `3 5 / 4 6 8` $\to 4$；(2) 极限总和 `3 18 / 4 6 8` $\to 0$；(3) 全相等 `3 3 / 5 5 5` $\to 4$。
- 反事实追问：如果第 2 行错写成 `long long lo = 1`，输入第 (2) 组 `3 18 / 4 6 8` 会怎样？答：正确答案是 0，但区间从 $[1, 8]$ 缩到 $[1, 1]$ 就退出，错误输出 1（WA）！

二分答案建模易错对照（WA / TLE）

易错环节	错误写法	正确做法与原因	评测结果
check 累加类型	总木材 `int wood = 0;`	$5\times10^5\times10^9$ 超 `int`，必用 `long long`	**WA**
下界设窄了	砍树设 `lo = 1`	全砍完时锯片高度为 0，应设 `lo = 0`	**WA**
分段下界太小	分段设 `lo = 0`	某段单个元素可能大于 `mid`，应设 $\max(a_i)$	**WA**
贪心前忘排序	像《愤怒的牛》直接扫描原坐标	隔间无序时贪心判定失效，必须先 `sort`	**WA**
模板选反方向	“求最小合法”套了模板 A	写前先画 `TTT...FFF` 还是 `FFF...TTT`	**WA/TLE**

本页要点

二分答案写挂 80% 都在 check：先查 `long long`，再查初始 $[lo, hi]$ 和是否需要 `sort`。

生活小例子：要切出小数厘米的绳子，整数“加一减一”跳过去了！

具体小例子：求三次方程 $f(x) = x^3 + ax + b = 0$ 在 $[-10^5, 10^5]$ 内的唯一实根

例如 $x^3 - 2 = 0$，真正的根是 $\sqrt[3]{2} \approx 1.259921$。
在整数二分里，我们写 `l = mid + 1` 或 `r = mid - 1`；但到了小数世界，1 和 2 之间还有无穷多个小数！一旦写成 `+ 1`，答案就被直接跳过。

- 写实数二分的三个改动：
  1. 变量类型全部换成 `double`（或 `long double`）。
  2. 更新区间直接写 `l = mid;` 和 `r = mid;`，不要加减 **1**！
  3. 循环结束不能靠 $l == r$，而要靠区间缩到足够小。

本页要点

实数域里没有“下一格整数”，缩小区间时两边一律直接令 `l = mid` 或 `r = mid`。

为什么暴力扫步长会 TLE，而 while(r - l > eps) 也会 TLE?

- 暴力做法为什么 **TLE**？题目要求保留 6 位小数，若从 -10^5 到 10^5 每次走 10^{-6} 检查 $f(x)$ 符号，一共要试 $2\times10^5/10^{-6}=\mathbf{2\times10^{11}}$ 次，跑半小时必 **TLE**！
- 为什么经典的 `while (r - l > eps)` 也会死循环 **TLE**？
  - 计算机用 `double` 存小数，一共只有约 $15\sim17$ 位有效数字（53 位二进制尾数）。
  - 当数值大到 10^5 时，相邻两个能表示的 `double` 之间差距约为 $10^5\times2^{-52}\approx2.2\times10^{-11}$。
  - 如果你随手写了 `eps = 1e-12`，当 $r-l=2.2\times10^{-11}>10^{-12}$ 时，中点 `(l+r)/2` 算出来要么等于 `l` 要么等于 `r`——区间再也缩不动，死循环 **TLE**！

本页要点

把“靠 `r - l > eps` 退出”换成“固定跑 100 次 `for` 循环”，永远不会死循环！

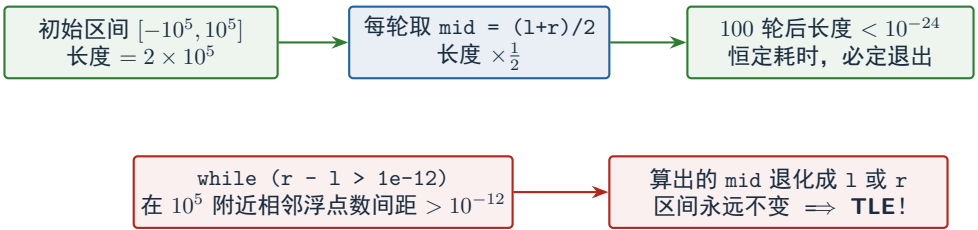
三个新术语大白话

- 精度容限 `eps`：题目允许的误差范围（比如保留 6 位小数时，通常要求误差 $< 10^{-8}$）。
- 固定次数迭代：不管区间多大，直接写 `for (int it = 0; it < 100; ++it)`。每轮区间长度变成原来的 $\frac{1}{2}$，100 轮后区间长度只有原来的 $\frac{1}{2^{100}} \approx 7.8 \times 10^{-31}$，精度早就碾压 `double` 极限！
- 负零陷阱 `-0.000000`：当根非常接近 0（如 -10^{-16}）时，`printf` 或 `cout` 按 6 位小数四舍五入可能印出 `-0.000000`，导致 OJ 判 **WA**！

本页要点

固定 100 次循环既快（恒定 100 步）又准（缩到 2^{-100}），还能彻底免除精度死循环。

TikZ 图解：固定 100 轮收敛 vs eps 卡死对比



本页要点

实数二分写 `for (int it = 0; it < 100; ++it)`，把“不确定停不停得下来”变成确定 100 步。

手算跟踪表：实数二分求 $x^3 - 8 = 0$（在 $[0, 8]$ 上前 5 轮）

由于 $f(x) = x^3 - 8$ 单调递增，若 $f(mid) < 0$ 令 $l = mid$，否则令 $r = mid$：

轮次 it	it < 100?	$[l, r]$	$mid = (l + r)/2$	$f(mid) = mid^3 - 8$	动作
0	真	$[0.000,\ 8.000]$	4.000	$56.000 \geq 0$	$r \leftarrow 4.000$
1	真	$[0.000,\ 4.000]$	2.000	$0.000 \geq 0$	$r \leftarrow 2.000$
2	真	$[0.000,\ 2.000]$	1.000	$-7.000 < 0$	$l \leftarrow 1.000$
3	真	$[1.000,\ 2.000]$	1.500	$-4.625 < 0$	$l \leftarrow 1.500$
4	真	$[1.500,\ 2.000]$	1.750	$-2.641 < 0$	$l \leftarrow 1.750$
⋮	⋮	⋮	⋮	⋮	⋮
100	假	$[2.000,\ 2.000]$	—	退出循环	输出 `2.000000`

本页要点

第 1 轮恰好命中 2.0 后，后续几十轮会把左端点 l 不断逼近 2.0，最终第 100 轮退出。

正确性与收敛证明：介值不变量

- 当 $a \geq 0$ 时，$f'(x) = 3x^2 + a \geq 0$，$f(x) = x^3 + ax + b$ 在全实数轴上单调不减。

阶段	验证内容（目标：保持 $f(l) < 0 \leq f(r)$）
初始化	题目保证唯一实根落在 $[-10^5, 10^5]$ 内，故 $f(-10^5) \leq 0 \leq f(10^5)$
保持	若 $f(mid) < 0$，因单调不减，根必在 $[mid, r]$，令 $l = mid$；否则根在 $[l, mid]$，令 $r = mid$
终止	执行 100 轮后循环必终止，此时 $r - l = \frac{2\times10^5}{2^{100}} < 10^{-24}$，中点 $(l+r)/2$ 与真根误差不到 10^{-24}

本页要点

100 轮二分能把 2×10^5 的区间压缩到 10^{-24} 以内，远超 6 位小数的精度需求。

实数二分模板代码（与 code/root_bisect.cpp 一致）

```
double a, b;
if (!(cin >> a >> b)) return 0;
double l = -1e5, r = 1e5;
for (int it = 0; it < 100; ++it) {       // 固定迭代 100 次，精度远超 1e-18
    double mid = (l + r) / 2.0;          // 浮点运算必须写 2.0
    if (mid * mid * mid + a * mid + b < 0) l = mid;
    else r = mid;
}
double ans = (l + r) / 2.0;
if (fabs(ans) < 5e-7) ans = 0.0;         // 避免输出 "-0.000000" 导致 WA
cout << fixed << setprecision(6) << ans << '\n';
```

- 为什么 5e-7？保留 6 位小数时，绝对值小于 0.5×10^{-6} 的负数四舍五入会变成 -0.000000，强行赋成正 0.0 即可消掉负号。

本页要点

固定 100 轮 + / 2.0 + 负零修正，构成实数二分三件套。

上机验证与反事实追问

运行 `code/root_bisect.cpp` 验证以下 3 组输入：

- 普通样例：输入 `0 -8`（即 $x^3 - 8 = 0$）$\Rightarrow$ 预期输出 `2.000000`。
- 零根特殊边界：输入 `1 0`（即 $x^3 + x = 0$，唯一实根 0）$\Rightarrow$ 预期输出 `0.000000`。
- 负根测试：输入 `0 1`（即 $x^3 + 1 = 0$，唯一实根 -1）$\Rightarrow$ 预期输出 `-1.000000`。

两个反事实追问

1. 如果把循环次数从 100 改成 10，输入 `0 -8` 会输出什么？
   答：区间只缩成 $2 \times 10^5 / 2^{10} \approx 195.3$，答案误差接近 100，完全错误！
2. 如果删掉 `if (fabs(ans) < 5e-7) ans = 0.0;`，输入 `1 0` 可能输出什么？
   答：当左端残留极小负数（如 -10^{-16}）时可能印出 `-0.000000`，直接判 **WA**！

实数二分易错对照（WA / TLE）

易错点	错误写法	正确做法与原因	评测结果
eps 过小卡死	`while (r - l > 1e-12)`	大实数尾数间隔大于 1e-12，改跑固定 100 次	**TLE**
只跑太少次	`for (it = 0; it < 30; ++it)`	初始长 2×10^5 缩 2^{30} 仅 2×10^{-4} 不够 6 位精度，跑 100 次	**WA**
输出负零	直接打印极小负数 ans	绝对值 $<0.5\times10^{-k}$ 时强行置 0.0	**WA**
把整数当浮点除	(1 + 2) / 2 忘了 .0	变量或字面量写 2.0，防整除截断小数	**WA**
多根区间直接二分	如一元三次方程有 3 个根直接扫 $[-100,100]$	先按步长 1 划出只含 1 个单调根的小区间再二分	**WA**

本页要点

延伸提醒：如《一元三次方程求解》（1238）有三个根，要先按间距 1 枚举小区间，再对每个含根小区间实数二分！

生活小例子：只看山腰一个点，能知道谷底在左还是右吗？

具体小例子：求开口向上的抛物线 $F(x) = (x-3)^2 + 1$ 在 $[0, 9]$ 的最小值

把 $[0, 9]$ 切一刀看正中点 $mid = 4.5$，算得 $F(4.5) = 3.25$。
只给你一个数字 3.25，你完全不知道最低点在 4.5 左边还是右边！因为函数先降后升，不再具有整体单调性，普通二分当场失效。

- 怎么破局？切一刀看不出走势，那就同时切两刀！把区间 $[l, r]$ 三等分成两处试探点 m_1 和 m_2，比较它俩的高低就能看出往哪边走！

本页要点

单调函数切一刀（二分）；先降后升或先升后降的拐弯函数切两刀（三分）。

为什么暴力枚举和普通二分都用不了？

以例题《曲线》（1435）为例：有 $n \le 10^4$ 条二次函数，求 $F(x) = \max_i S_i(x)$ 在 $[0, 1000]$ 的最小值（精确到 10^{-4}）：

- 暴力做法：每隔 10^{-5} 取一个 x，共取 10^8 个点；每个点还要扫一遍 10^4 条抛物线求最大值，总计算量 10^{12} 次，必 **TLE**！
- 普通二分：函数 $F(x)$ 先下降到谷底再上升，既不是全递增也不是全递减，单凭一个 $F(mid)$ 无法判定谷底方向。
- 三分法：每轮只算 $F(m_1)$ 和 $F(m_2)$ 两次函数值，区间长度立刻缩成原来的 $\frac{2}{3}$；固定跑 100 轮只需 $200 \times 10^4 = 2 \times 10^6$ 次，稳稳 **AC**！

本页要点

三分法把 $O(10^8 \cdot n)$ 的枚举压缩到仅仅 100 轮（200 次求值）。

三个新术语大白话

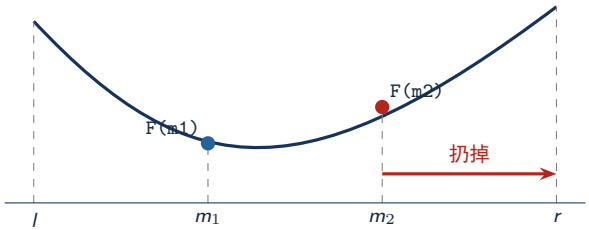
- 单谷函数（先减后增）：从左往右先一路下坡到唯一最低点（谷底），过谷底后一路上坡。比如单个开口向上抛物线，或者多条开口向上抛物线的最大值上包络 $F(x) = \max_i S_i(x)$。
- 单峰函数（先增后减）：从左往右先一路上坡到唯一最高点（山峰），过山峰后一路下坡。
- 三等分点 m_1, m_2：把闭区间 $[l, r]$ 平均切成三段的两刀：

$$m_1 = l + \frac{r-l}{3} = \frac{2l+r}{3}, \qquad m_2 = r - \frac{r-l}{3} = \frac{l+2r}{3}.$$

本页要点

求极小值选单谷，求极大值选单峰；三等分点公式记住“$2l + r$ 偏左，$l + 2r$ 偏右”。

TikZ 图解：比较 $F(m_1)$ 与 $F(m_2)$，安全扔掉外侧三分之一



若 $F(m_1) \le F(m_2)$
谷底绝不在 $[m_2, r]$！
安全令 r = m2

本页要点

口诀（求极小值）：谁高就扔谁那一侧的外段——$F(m_1) \le F(m_2)$ 扔右边 $[m_2, r]$，反之扔左边 $[l, m_1]$！

手算跟踪表：用三分法求 $F(x) = (x-3)^2 + 1$ 在 $[0, 9]$ 的极小值

每轮取 $m_1 = \frac{2l+r}{3}$，$m_2 = \frac{l+2r}{3}$，若 $F(m_1) \le F(m_2)$ 令 $r = m_2$，否则令 $l = m_1$：

轮次	$[l, r]$	m_1	m_2	$F(m_1)$ vs $F(m_2)$	扔掉外段	新区间
1	$[0.00,\ 9.00]$	3.00	6.00	$1.00 \le 10.00$	扔 $[6, 9]$	$[0.00,\ 6.00]$
2	$[0.00,\ 6.00]$	2.00	4.00	$2.00 \le 2.00$	扔 $[4, 6]$	$[0.00,\ 4.00]$
3	$[0.00,\ 4.00]$	1.33	2.67	$3.78 > 1.11$	扔 $[0, 1.33]$	$[1.33,\ 4.00]$
4	$[1.33,\ 4.00]$	2.22	3.11	$1.60 > 1.01$	扔 $[1.33, 2.22]$	$[2.22,\ 4.00]$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	$[3.00,\ 3.00]$	—	—	退出循环	—	最低点 $x = 3.00,\ F = 1.00$

本页要点

每轮保留真极小值且长度严格乘 $2/3$，第 2 轮即使 $F(2) = F(4)$ 相等，谷底 3 仍在 $[0, 4]$ 内！

正确性证明（反证法）与适用边界禁区

- 为什么 $F(m_1) \le F(m_2)$ 时谷底一定不在 $(m_2, r]$ 内？
  - 假设（反证）谷底 x^* 居然落在 m_2 右边（即 $m_1 < m_2 < x^*$）。
  - 因为单谷函数在谷底 x^* 左侧必须严格递减，所以 $m_1 < m_2$ 必然推出 $F(m_1) > F(m_2)$。
  - 这与已知的 $F(m_1) \le F(m_2)$ 直接矛盾！因此 x^* 必在 $[l, m_2]$ 内。
- 两大禁区（不能直接用三分）：
  1. 双峰 / 多峰函数：存在多个波谷时，三分可能会错误地扔掉更深的另一个谷。
  2. 大段平顶 / 水平平台：若某一段函数值完全相等，$F(m_1) == F(m_2)$ 无法分辨谷底在左、中还是右。

本页要点

如果自变量是整数而不是实数，三分缩到 $r - l < 3$ 时必须停下，再用 `for` 暴力扫完这几个整数！

单谷函数三分法模板代码

```
long double lo = 0, hi = 1000;
for (int it = 0; it < 100; ++it) {      // 每轮缩成 2/3，100 轮后仅剩 (2/3)^100 ~=
    2e-18
    long double m1 = (2 * lo + hi) / 3.0L; // 左三等分点（注意用 3.0L 保高精度）
    long double m2 = (lo + 2 * hi) / 3.0L; // 右三等分点
    if (F(m1) <= F(m2)) hi = m2;        // 求极小值：左低，丢弃右外段 [m2, hi]
    else                lo = m1;        // 右低，丢弃左外段 [lo, m1]
}
long double ans = F((lo + hi) / 2.0L);  // 区间缩成一点后取中点函数值
```

- 为什么用 `long double`？比 `double` 多出十多位有效数字，做两步三等分和二次多项式求值时舍入误差更小。

本页要点

若是求“单峰函数最大值”，只需把 `F(m1) <= F(m2)` 的小于等于改成 `>=`（谁小扔谁那侧）！

上机验证与反事实追问

把 $F(x)$ 设为不同小函数，跑上面模板测 3 组情况：

- 谷底在内部：$F(x)=(x-3)^2+1$ 在 $[0,10]\Rightarrow$ 输出 `1.0000`（在 $x=3$）。
- 谷底在左端点：$F(x)=(x+2)^2$ 在 $[0,10]\Rightarrow$ 在 $[0,10]$ 单调递增，始终触发 $F(m_1)\le F(m_2)$ 缩到 0，输出 `4.0000`。
- 谷底在右端点：$F(x)=(x-15)^2$ 在 $[0,10]\Rightarrow$ 缩到 10，输出 `25.0000`。

反事实追问

1. 如果把 `m1 = (2 * lo + hi) / 3.0L` 错写成 `(lo + hi) / 3.0L` 会怎样？
   答：当 $lo=500, hi=1000$ 时算出 $m_1=500=lo$，每轮扔掉 0 长度，最后答案完全跑偏 **WA**！
2. 如果是整数域三分（如 $lo, hi\in\mathbb{Z}$），能直接写 `for (it=0; it<100; ++it)` 吗？
   答：不能只靠二分式除 3 缩到 1，因为 $hi-lo=2$ 时 $(2lo+hi)/3==lo$ 会卡死；需 `while (hi - lo >= 3)` 后再暴力枚举 $[lo, hi]$！

三分法易错对照（WA）

易错点	错误写法	正确做法与原因	评测结果
三等分点公式错	`m1 = (lo + hi) / 3.0`	应从 `lo` 加起：`(2*lo + hi) / 3.0L`	**WA**
极值方向写反	求最大值仍写 `if(F(m1)<=F(m2)) hi=m2`	求极大值谁小扔谁，条件改为 `F(m1) >= F(m2)`	**WA**
区间边界漏检查	认为端点不可能是答案	谷底在区间外时最值在端点，建议与 $F(0), F(1000)$ 取 min	**WA**
非单峰强行三分	对双峰或带水平平台函数三分	先画图或求导确认严格单峰/单谷性，否则改扫描/DP	**WA**

本页要点

三分前必确认：区间上只有唯一拐点；求最小扔更高侧，求最大扔更低侧。

例 1 | 砍树 [基础]：先读懂要求

大白话题意

有 n 棵树，第 i 棵高度为 h_i。选定一个整数锯口高度 H，把所有高于 H 的树梢锯下，能收集到的木材总量是 $\sum_{i=1}^{n} \max(0, h_i - H)$。
在保证木材总量至少为 M 的前提下，锯片最高能设多高？

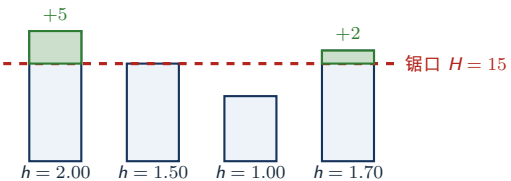
- 输入：第一行 n（$\le 5 \times 10^5$）和 M；第二行 n 个整数 h_i（$\le 10^9$）。题目保证 $M \le \sum h_i$。
- 处理：猜锯口高度 $H \in [0, \max h_i]$，校验所得木头是否 $\ge M$。
- 输出：一行一个整数，表示最大可行的 H。
- 原题样例：输入 4 7 与 20 15 10 17 $\Rightarrow$ 输出 15。

本页要点

目标关键词是“至少 M”且“高度 H 最大”，一眼锁定最大化合法值的模型 A。

例 1 | 砍树：怎么想到的与建模图解

- 暴力会怎样？H 最大到 10^9，每次扫描 5×10^5 棵树，总操作 5×10^{14} 次必 **TLE**。
- 关键性质：锯片抬得越高，截下的树梢越短，总木材量单调不增。一旦某个 H 不够 M，比它更高的 H' 一定更不够（呈 `TTTT...FFFF` 分布）。



本页要点

当 $H = 15$ 时只有 20 和 17 高出锯口，合计 $5 + 2 = 7 \geq 7$，合法！

例 1 | 砍树：原题样例手推表

输入 n=4，M=7，树高 [20，15，10，17]。初始 $lo = 0$，$hi = 20$，按模板 A 执行：

轮次	$[lo, hi]$	mid	高于 mid 的截取量	木材和 ≥ 7?	区间更新
1	$[0, 20]$	10	$10+5+0+7=22$	真（$22 \geq 7$）	$lo \leftarrow 10$
2	$[10, 20]$	15	$5+0+0+2=7$	真（$7 \geq 7$）	$lo \leftarrow 15$
3	$[15, 20]$	18	$2+0+0+0=2$	假（$2 < 7$）	$hi \leftarrow 17$
4	$[15, 17]$	16	$4+0+0+1=5$	假（$5 < 7$）	$hi \leftarrow 15$
5	$[15, 15]$	—	退出循环	$lo < hi$ 为假	输出 **15**

本页要点

第 4 轮试 16 不够后立刻压低上界到 15，第 5 轮 $lo == hi$ 停止并输出 15。

例 1 | 砍树：核心步骤与相对模板的变化

按顺序计算	纸上得到	核对问题
1. 确定范围 $[lo, hi]$	$lo = 0,\ hi = \max(h_i)$	若 $M = \sum h_i$，锯口 0 合法吗？合法！
2. $O(n)$ 累计木材和	遍历 h_i，若 $h_i > mid$ 加上 $h_i - mid$	总木材可达 5×10^{14}，变量开 `long long` 了吗？
3. 套用模板 A 更新	`wood >= M` 令 `lo = mid`，否则 `hi = mid - 1`	算 `mid` 时写了 `+1` 上取整吗？

相对模板改了哪里

在模板 A 的框架上，把 `ok(mid)` 写成“遍历树高数组求 $\sum \max(0, h_i - mid) \geq M$”；初值 `hi` 直接取输入树高的最大值，避免无谓扩大区间。

本页要点

所有变量（`m`，`h[i]`，`lo`，`hi`，`mid`，`wood`）全部统一开 `long long`，彻底杜绝溢出。

例 1 | 砍树：完整代码（1/2，code/woodcut.cpp）

```cpp
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    long long m;
    if (!(cin >> n >> m)) return 0;
    vector<long long> h(n);
    long long hi = 0;
    for (int i = 0; i < n; ++i) {
        cin >> h[i];
        hi = max(hi, h[i]);              // 右边界取最高那棵树的高度
    }
```

本页要点

先读入树高并顺手求出最高树 `hi`，作为二分右端点（完整头文件同前）。

例 1 | 砍树：完整代码（2/2）

```
    long long lo = 0;                       // 锯片最低可放到地面 0
    while (lo < hi) {
        long long mid = lo + (hi - lo + 1) / 2; // 求最大合法值必须上取整 +1
        long long wood = 0;                 // 累加和必开 long long 防溢出
        for (int i = 0; i < n; ++i)
            if (h[i] > mid) wood += h[i] - mid;
        if (wood >= m) lo = mid;            // 够用了，试着再抬高锯口
        else hi = mid - 1;                  // 不够用，必须降低锯口
    }
    cout << lo << '\n';
    return 0;
}
```

本页要点

内层只扫一遍数组，总时间复杂度 $O(n \log \max h_i)$，空间复杂度 $O(n)$。

例 1 | 砍树：易错对照与三组自测边界

- 常见错因 **1**（**WA**）：`wood` 开成 `int`。若 5×10^5 棵树每棵截 10^9，总量达 5×10^{14} 直接溢出成负数。
- 常见错因 **2**（**TLE**）：`mid` 漏写 `+1`，缩到 $[15, 16]$ 时死循环。
- 常见错因 **3**（**WA**）：`lo` 错设成 1，遇到“必须全部砍光（$M = \sum h_i$）”时答案 0 算不到。

三组自测边界数据

1. 最小树全砍：`1 10 / 10` $\Rightarrow$ 输出 `0`。
2. 树全等高：`3 6 / 5 5 5` $\Rightarrow$ 输出 `3`（每棵切 2 米共 6 米）。
3. 只需 1 米：`2 1 / 10 100` $\Rightarrow$ 输出 `99`（只切最高那棵的顶尖 1 米）。

例 2 | 愤怒的牛（一本通 1433）[基础]：先读懂要求

大白话题意

农夫 John 有 N 个畜栏隔间，坐标分别是 $x_1, \ldots, x_N$（输入顺序乱序，未必递增）。要把 C 头爱打架的牛放进不同的隔间里，希望任意两头牛之间距离的最小值尽量大。求这个“最大的最小距离”。

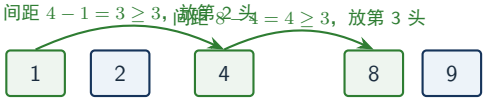
- 输入：第一行 N（$2 \le N \le 10^5$）与 C（$2 \le C \le N$）；接下来 N 个整数 x_i（$0 \le x_i \le 10^9$）。
- 处理：排序坐标后，猜一个最小间距 d，贪心检验能否放下至少 C 头牛。
- 输出：一个整数，表示最大的最小间距。
- 原题样例：输入 `5 3` 与 `1 2 8 4 9` $\Rightarrow$ 输出 `3`。

本页要点

看到“最大的最小值”，且 x_i 乱序，第一步永远是先 `sort` 再二分间距！

例 2 | 愤怒的牛：怎么想到的与建模图解

- 暴力会怎样？从 $N = 10^5$ 个隔间里选 C 个有 $\binom{N}{C}$ 种方案，指数级爆炸必 **TLE**。
- 单调性质：猜一个最小间距 d。若连间距 d 都放不下 C 头牛，那间距 $d+1$ 更放不下（TTT...FFFF）。
- 贪心校验：第 1 头牛放最左边的 $pos[0]$ 一定最划算（把右侧空间留到最大）；之后只要遇到离上一头距离 $\geq d$ 的第一个隔间就立刻放下一头。



本页要点

排序 + 猜间距 d + 从左到右贪心放牛，把组合爆炸问题变成 $O(N \log X)$。

例 2 | 愤怒的牛：原题样例手推表

先把 [1，2，8，4，9] 排序为 [1，2，4，8，9]，$lo = 0$，$hi = 9 - 1 = 8$，目标放 $C = 3$ 头牛：

轮次	$[lo, hi]$	mid	贪心选出的隔间	放下头数 ≥ 3?	动作
1	$[0, 8]$	4	$1 \to 8$（$9 - 8 = 1 < 4$，共 2 头）	假（$2 < 3$）	$hi \leftarrow 3$
2	$[0, 3]$	2	$1 \to 4 \to 8$（共 3 头）	真（$3 \geq 3$）	$lo \leftarrow 2$
3	$[2, 3]$	3	$1 \to 4 \to 8$（共 3 头）	真（$3 \geq 3$）	$lo \leftarrow 3$
4	$[3, 3]$	—	退出循环	$lo < hi$ 为假	输出 **3**

本页要点

第 1 轮试 $d = 4$ 只放下 $1, 8$ 两头不够 3 头；第 3 轮试 $d = 3$ 恰好放下 $1, 4, 8$，第 4 轮退出得 3。

例 2 | 愤怒的牛：核心步骤与相对模板的变化

按顺序计算	纸上得到	核对问题
1. 排序隔间坐标	把 pos 升序排好	输入未必有序，不排必 WA！
2. 初始化边界	$lo = 0,\ hi = pos[n-1] - pos[0]$	若有相同坐标隔间，间距可能为 0 吗？
3. 贪心校验放置	`cnt = 1, last = pos[0]` 往右扫	遇到 `pos[i] - last >= mid` 才更新 `last`

相对模板改了哪里

套用模板 A：(1) 二分前先 `sort(pos.begin(), pos.end())`；(2) `check(mid)` 从第 0 个隔间起贪心统计最多放置牛数 `cnt`，判 `cnt >= c`。

本页要点

第 1 头牛放 `pos[0]` 留给右侧空间最大，绝不吃亏！

例 2 | 愤怒的牛：完整代码（1/2，code/angry_cows.cpp）

```cpp
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, c;
    if (!(cin >> n >> c)) return 0;
    vector<long long> pos(n);
    for (auto &x : pos) cin >> x;
    sort(pos.begin(), pos.end());        // 关键：输入坐标未必有序，必须先排序！
    long long lo = 0, hi = pos.back() - pos.front();
```

本页要点

右边界直接设为最大坐标减去最小坐标，不会比这更大。

例 2 | 愤怒的牛：完整代码（2/2）

```
    while (lo < hi) {
        long long mid = lo + (hi - lo + 1) / 2;
        int cnt = 1; long long last = pos[0]; // 第一头贪心放最左隔间
        for (int i = 1; i < n; ++i) {
            if (pos[i] - last >= mid) {
                last = pos[i]; ++cnt;
            }
        }
        if (cnt >= c) lo = mid;          // 能放够 c 头，尝试扩大间距
        else hi = mid - 1;
    }
    cout << lo << '\n';
    return 0;
}
```

本页要点

注意 cnt 初值为 1（第 1 头已放在 pos[0]），循环从 i = 1 开始。

例 2 | 愤怒的牛：易错对照与三组自测边界

- 常见错因 **1**（**WA**）：忘写 `sort(pos.begin(), pos.end())`，样例本身就乱序（`1 2 8 4 9`），不排序直接 WA。
- 常见错因 **2**（**WA**）：`cnt` 初值误写成 0，导致少算最左边的第 1 头牛。
- 常见错因 **3**（**WA**）：若写 `pos[i] - last > mid`（少等号），刚好间隔等于 `mid` 时会被误判放不下。

三组自测边界数据

1. 只有两头牛：`4 2 / 10 1 5 2` $\Rightarrow$ 输出 `9`（放两端 1 和 10）。
2. 隔间全紧凑：`3 3 / 1 2 3` $\Rightarrow$ 输出 `1`。
3. 重复坐标隔间：`3 3 / 5 5 5` $\Rightarrow$ 输出 `0`（若 `lo` 误设 1 会错成 1）。

例 3 | 河中跳房子（一本通 1247）[提高]：先读懂要求

大白话题意

笔直河道起点在 0、终点在 L，中间有 N 块岩石位置严格递增（$0 < D_i < L$）。
可以移走中间至多 M 块岩石（起点 0 和终点 L 不能移走），问移走后相邻两次跳跃的最短距离最大能是多少？

- 输入：第一行 L（$\leq 10^9$）、N（≤ 50000）、M（$0 \leq M \leq N$）；接下来 N 行递增整数 D_i。
- 处理：猜最短跳跃距离 d，贪心模拟看最少要移走几块石头才能保证步步 $\geq d$。
- 输出：一个整数，表示最大可能的最短跳跃距离。
- 原题样例：`L=25`，`N=5`，`M=2`，岩石 2，11，14，17，21 $\Rightarrow$ 输出 4。

本页要点

和《愤怒的牛》互为对偶：牛是“选留下 C 个位置”，跳房子是“至多删 M 块石头”。

例 3 | 河中跳房子：怎么想到的与建模图解

- 难点在哪里？起点 0 和终点 L 都必须踩到，最后一步跳到 L 也要 $\geq d$！所以必须把 $pos[0]=0$ 和 $pos[n+1]=L$ 一起塞进数组。
- 贪心校验：固定最短跳跃距离 d，记当前站立的石头为 $pos[last]$。往右看第 i 块石头：
  - 若 $pos[i]-pos[last]<d$：这一步太短！第 i 块不能留，只能把它移走（`++removed`），`last` 不动。
  - 若 $pos[i]-pos[last]\geq d$：这步合法！跳过去把落脚点更新为 `last = i`。



本页要点

千万记住：若连终点 L 这一步都 $<d$，相当于移走之前那块（计数仍 $+1$），同样归入 `removed`！

例 3 | 河中跳房子：原题样例手推表

补充首尾后 pos = [0, 2, 11, 14, 17, 21, 25]，$M = 2$，初始 $lo = 1$, $hi = 25$：

轮次	$[lo, hi]$	mid	逐石判定过程（须移走哪些石头）	移走数 ≤ 2?	动作
1	$[1, 25]$	13	移 $2, 11$，留 14；移 $17, 21, 25$ 共 5 次	假（$5 > 2$）	$hi \leftarrow 12$
2	$[1, 12]$	6	移 2，留 11；移 14，留 17；移 $21, 25$ 共 4 次	假（$4 > 2$）	$hi \leftarrow 5$
3	$[1, 5]$	3	移 2，留 $11, 14, 17, 21, 25$ 共 1 块	真（$1 \le 2$）	$lo \leftarrow 3$
4	$[3, 5]$	4	移 2 和 14，留 $11, 17, 21, 25$ 共 2 块	真（$2 \le 2$）	$lo \leftarrow 4$
5	$[4, 5]$	5	移 $2, 14$ 后，移 21 跳到 25 共 3 块	假（$3 > 2$）	$hi \leftarrow 4$
6	$[4, 4]$	—	退出循环	$lo < hi$ 为假	输出 **4**

本页要点

当试 $d = 5$ 走到最后一块 $21 \to 25$（距离 $4 < 5$）时，虽然终点不能移，但退一格移走 21 依然等价于 ++removed，所以计数完全准确！

例 3 | 河中跳房子：核心步骤与相对模板的变化

按顺序计算	纸上得到	核对问题
1. 构造完整序列	`pos[0]=0`，`pos[1..n]=D`，`pos[n+1]=L`	漏放终点 L 会错失最后一段跳跃！
2. `ok(d)` 移石贪心	间距 $< d$ 就 `++removed`，够就更新 `last`	超过 `m` 即可提前 `return false`
3. 套用模板 A 二分	在 $[1, L]$ 上找最大合法 `d`	$N = 0$ 时一刀不切直接跳 L，区间包住了吗？

相对模板改了哪里

套用模板 A，`pos` 首尾补入 0 和 L；校验时统计跳过岩石数 `removed`，判定 `removed <= m`。

本页要点

若 $N = 0$，只有 $0 \to L$ 一跳，$lo = 1, hi = L$ 时 `ok(L)` 为真，正确输出 L。

例 3 | 河中跳房子：完整代码（1/2，code/river.cpp）

```cpp
long long L; int n, m;
if (!(cin >> L >> n >> m)) return 0;
vector<long long> pos(n + 2, 0);
for (int i = 1; i <= n; ++i) cin >> pos[i];
pos[n + 1] = L;                          // 补充终点岩石位置 L
auto ok = [&](long long d) {
    int removed = 0, last = 0;
    for (int i = 1; i <= n + 1; ++i) {
        if (pos[i] - pos[last] < d) {
            if (++removed > m) return false; // 间距不够 d，移走
        } else last = i;                 // 够 d 就保留，更新落脚点
    }
    return true;
};
```

本页要点

循环必须跑到 `i <= n + 1`，把最后落到终点 L 那一跳也检查进去。

例 3 | 河中跳房子：完整代码（2/2）

```
    long long lo = 1, hi = L;            // 最短跳跃至少 1，至多一步到终点 L
    while (lo < hi) {
        long long mid = lo + (hi - lo + 1) / 2; // 模板 A：上取整防死循环
        if (ok(mid)) lo = mid;
        else hi = mid - 1;
    }
    cout << lo << '\n';
    return 0;
}
```

本页要点

若终点前一步 $< d$，等价于移走前一块石子；总复杂度 $O(N \log L)$。

例 3 | 河中跳房子：易错对照与三组自测边界

- 常见错因 **1**（**WA**）：循环只写到 `i <= n`，忘记把终点 L 放到 `pos[n+1]`，最后一段跨度再短也不会被发现。
- 常见错因 **2**（**WA**）：即使发现了 $pos[i] - pos[last] < d$，也顺手把 `last = i` 更新了——移走的石头怎么还能踩在脚下？
- 常见错因 **3**（**WA**）：右边界 `hi` 错设成 $pos[n]$ 而不是 L，当 $M = N$ 全移光时答案应该是 L。

三组自测边界数据

1. 中间无岩石：`10 0 0` $\Rightarrow$ 输出 `10`。
2. 允许全移走：`10 2 2 / 3 / 7` $\Rightarrow$ 输出 `10`。
3. 不允许移走：`10 2 0 / 3 / 7` $\Rightarrow$ 输出 `3`（各段 $3, 4, 3$ 中最短为 3）。

例 4 | 曲线（一本通 1435）[提高]：先读懂要求

大白话题意

给出 n 条二次函数 $S_i(x) = a_i x^2 + b_i x + c_i$（其中 $a_i \geq 0$，保证开口向上或退化成直线）。定义新函数 $F(x) = \max_{1 \leq i \leq n} S_i(x)$（每个 x 取所有曲线里的最高点）。求 $F(x)$ 在区间 $[0, 1000]$ 上的最小值，四舍五入保留 4 位小数。

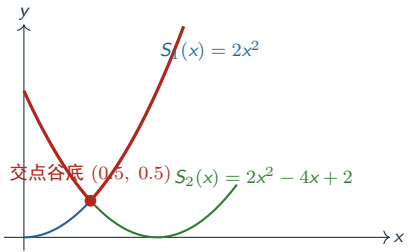
- 输入：$T < 10$ 组数据；每组先给 $n \leq 10^4$，再给 n 行系数 a, b, c。
- 处理：利用“凸函数取最大值仍为单谷函数”的性质，在 $[0, 1000]$ 上做固定 100 轮三分。
- 输出：每组输出一个保留 4 位小数的实数。
- 原题样例：两组输入（$2x^2$；以及 $2x^2$ 与 $2x^2 - 4x + 2$）$\Rightarrow$ 输出 `0.0000` 与 `0.5000`。

本页要点

单条开口向上抛物线是单谷的，多条取 max 形成的上包络依然是单谷的！

例 4 | 曲线：怎么想到的与建模图解

- 为什么能三分？每条 $S_i(x)$（$a_i \geq 0$）都是向下凸的函数；多个凸函数的逐点最大值 $F(x) = \max_i S_i(x)$ 必然先降后升（或单调），绝不会出现两个波谷！



本页要点

图中红线即两条抛物线的上包络 $F(x)$，只在交点处出现唯一最低谷底！

例 4 | 曲线：原题样例手推表

针对第 2 组样例：$S_1(x) = 2x^2,\ S_2(x) = 2x^2 - 4x + 2$，若缩小到谷底附近 $[0, 1]$ 做三分：

轮次	$[lo, hi]$	m_1	m_2	$F(m_1)$	$F(m_2)$	决策
1	$[0.000, 1.000]$	0.333	0.667	0.889	0.889	$F(m_1) \le F(m_2) \Rightarrow hi = 0.667$
2	$[0.000, 0.667]$	0.222	0.444	1.210	0.617	$F(m_1) > F(m_2) \Rightarrow lo = 0.222$
3	$[0.222, 0.667]$	0.370	0.519	0.794	0.538	$F(m_1) > F(m_2) \Rightarrow lo = 0.370$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	收敛到 $x = 0.5000,\ F = 0.5000$

本页要点

即使第 1 轮左右两边算出的 $F(m_1)$ 和 $F(m_2)$ 恰好相等，真谷底 0.5 仍稳稳留在 $[0, 0.667]$ 内！

例 4 | 曲线：核心步骤与相对模板的变化

按顺序计算	纸上得到	核对问题
1. 实现上包络 `F(x)`	秦九韶 `(a*x + b)*x + c` 求 max	初值设成 `-1e30L`，勿设为 0（曲线可全在负半轴）
2. 固定 100 轮三分	在 $[0, 1000]$ 上取 m_1, m_2 比较	用 `long double` 提升精度
3. 端点兜底与格式化	`min({F(0), F(1000), F(mid)})`	`fabsl(ans) < 0.00005L` 防止 `-0.0000`

相对模板改了哪里

在三分法模板上，把 `F(x)` 写成了遍历 n 个二次三项式取最大值的 Lambda 函数；最后多加一次与区间端点 $F(0), F(1000)$ 取 min 以及保留 4 位小数的负零修正。

本页要点

`top` 的初值千万不能设为 0！若所有二次函数全在 x 轴下方（如 -10），初值 0 会直接算错！

例 4 | 曲线：完整代码（1/2，code/curves.cpp）

```
int T;
if (!(cin >> T)) return 0;
cout << fixed << setprecision(4);
while (T--) {
    int n; cin >> n;
    vector<array<int, 3>> coef(n);
    for (auto &[a, b, c] : coef) cin >> a >> b >> c;
    auto F = [&](long double x) {
        long double top = -1e30L;   // 极大负值作初值，防全在负半轴
        for (auto [a, b, c] : coef)
            top = max(top, (a * x + b) * x + c);
        return top;
    };
```

本页要点

用 `(a * x + b) * x + c` 计算二次式，少一次乘法且浮点误差更小。

例 4 | 曲线：完整代码（2/2）

```
    long double lo = 0, hi = 1000;
    for (int it = 0; it < 100; ++it) {
        long double m1 = (2 * lo + hi) / 3.0L;
        long double m2 = (lo + 2 * hi) / 3.0L;
        if (F(m1) <= F(m2)) hi = m2; // m1 更低，极小值必在 [lo, m2]
        else lo = m1;                // m2 更低，极小值必在 [m1, hi]
    }
    long double ans = min({F(0), F(1000), F((lo + hi) / 2.0L)});
    if (fabsl(ans) < 0.00005L) ans = 0; // 防止输出 -0.0000
    cout << ans << '\n';
}
```

本页要点

总复杂度 $O(T \cdot 100 \cdot n)$，当 $T = 10, n = 10^4$ 时约 2×10^7 次，轻松通过。

例 4 | 曲线：易错对照与三组自测边界

- 常见错因 **1**（**WA**）：`top` 初值设成了 0。当 $S(x)=x^2-100$ 时，最小值在负半轴 -100，错设 0 必错。
- 常见错因 **2**（**WA**）：误以为 $a=0$（退化为一次函数）不能三分——实际上直线也是凸函数，单调斜线和抛物线的 max 仍然单谷！
- 常见错因 **3**（**WA**）：忘写 `fabsl(ans) < 0.00005L` 判负零。

三组自测边界数据

1. 退化为常函数：`1 / 1 / 0 0 -5` $\Rightarrow$ 输出 `-5.0000`。
2. 两条负系数斜线交叉：`1 / 2 / 0 1 0 / 0 -1 10` $\Rightarrow$ 交点 $x=5$，输出 `5.0000`。
3. 单条曲线谷底在区间外：`1 / 1 / 1 -2002 0` $\Rightarrow$ 谷底 $x=1001$ 在右端外，在 $x=1000$ 取最小 `-1002000.0000`。

二分答案与三分法高频错误对照表

题号 / 场景	常见错误写法	正确做法与原因
整数模板 A（砍树等）	写了 `l = mid` 却写 `(l+r)/2`	区间缩到 $[l, l+1]$ 时死循环 TLE，改 `l+(r-l+1)/2`
浮点二分（方程的根）	用 `while(r - l > 1e-12)`	大实数尾数间隔不够导致死循环 TLE，改固定 100 次
愤怒的牛（1433）	读入后忘记 `sort` 直接贪心	题面不保证输入隔间有序，必须先升序排序
河中跳房子（1247）	漏掉终点 L 或移走后还更新 `last`	补充 `pos[n+1]=L`，移走石头时 `last` 保持不动
曲线（1435）	`top` 初值设 0 或输出 `-0.0000`	初值设为极小值 `-1e30L`，极小负数纠正为 0

本页要点

先查“有没有单调/单谷性”，再查“+1 取整、long long、初始边界”。

评测结果定位指南（CE / WA / RE / TLE）

评测结果	本章最常见元凶	第一时间排查步骤
CE	漏 `<iomanip>` / `<algorithm>`，或者使用了中文标点	检查 `setprecision`、`sort` 头文件与英文分号
TLE	模板 A 漏 `+1`；浮点用 `while(r-l>eps)`；内层暴力而非 $O(n)$ 校验	用长度为 2 的区间 $[0,1]$ 手动走一遍二分循环
WA	累加和开 `int` 溢出；忘排序；下界漏了 0；输出 `-0.0000`	造 $n=1$ 极小样例、全相等样例、极限大数据自测
RE	数组下标越界（如跳房子漏开 $n+2$ 大小）	检查 `pos` 大小是否开到 `n + 2` 并用 `at()` 调试

本页要点

二分代码极短，一旦 TLE 百分之九十九是取整方向或 eps 卡死，一旦 WA 优先查溢出与边界。

提交前检查清单（六必查）

1. 是否在写代码前，先确认目标函数真的具有“单调可行性”（二分）或“严格单峰/单谷性”（三分）？
2. 若分支写了 `l = mid;`，算 `mid` 时有没有加 `+ 1`？若写了 `r = mid;`，是不是没有加 `+ 1`？
3. 二分的初始区间 $[lo, hi]$ 是否包住了答案可能是 0、负数或极大值的全部极端情况？
4. 校验函数依赖有序坐标时（如《愤怒的牛》），读入后有没有先执行 `sort`？
5. 累加变量（如木材总量、分段和）与乘积是否开了 `long long` 防中间溢出？
6. 浮点二分/三分是否用了固定 100 轮循环，并把小于误差半位的负数纠正成 $+0.0$？

最后一步

提交前挑一组长度为 2（如 $[0, 1]$）的区间单步走一遍二分循环，确保一定能退出！

本章全部练习与完成标准

1. 整数二分与二分答案：砍树（code/woodcut.cpp）、愤怒的牛（一本通 1433，code/angry_cows.cpp）、河中跳房子（一本通 1247，code/river.cpp）、网线主管（一本通 1242，转厘米整数二分）。
2. 浮点二分：方程的根（code/root_bisect.cpp）、一元三次方程求解（一本通 1238）。
3. 三分法：曲线（一本通 1435，code/curves.cpp）。

完成标准

能一秒说出 `l = mid` 为什么要配上取整 `+1`；能独立写出任意题目的 $O(n)$ 贪心检验函数并与二分骨架拼装；独立 AC 上述精选习题。

课后安排与下一章预告

- 复习：合上讲义默写模板 A 和模板 B，各用 $[0,1]$ 区间手走两步证明不死循环。
- 练习：独立复写 `code/` 下 5 份源码并实测全部自测边界，额外挑战《网线主管》（1242）和《一元三次方程求解》（1238）。
- 下一章预告：第 5 章《贪心策略与邻项交换》——当不用再“猜答案”，每一步直接做出局部最优选择就能通吃全局时，怎么用“邻项交换不等式”彻底证明贪心绝不会 WA！

本章一句话

直接算最值很难时，不妨先“猜”一个答案，单调性会带你三十刀切出唯一真相。

谢谢！