

课程内容

T E A C H I N G C O U R S E W A R E P O W P O I N T

授课时间：2025.08.04

目录

- PART-01 概念引入 TEACHING COURSEWARE
- PART-02 概念分析 TEACHING COURSEWARE
- PART-03 经典例题 TEACHING COURSEWARE
- PART-04 总结 TEACHING COURSEWARE

01

概念引入

TEACHING
COURSEWARE

TEACH



概念引入

水太冷 → 向右拧热水阀门

水太热 → 向左拧冷水阀门

目标：找到最舒适的"黄金温度"

02

概念分析

TEACHING
COURSEWARE

TEACH

概念分析

单调性：如果值 X 可行，则所有 $\leq X$ （或 $\geq X$ ）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

经典例题

TEACHING
COURSEWARE

TEACH

题目描述

[复制 Markdown](#) [展开](#) [进入 IDE 模式](#)

伐木工人 Mirko 需要砍 M 米长的木材。对 Mirko 来说这是很简单的工作，因为他有一个漂亮的新伐木机，可以如野火一般砍伐森林。不过，Mirko 只被允许砍伐一排树。

Mirko 的伐木机工作流程如下：Mirko 设置一个高度参数 H （米），伐木机升起一个巨大的锯片到高度 H ，并锯掉所有树比 H 高的部分（当然，树木不高于 H 米的部分保持不变）。Mirko 就得到树木被锯下的部分。例如，如果一排树的高度分别为 20, 15, 10 和 17，Mirko 把锯片升到 15 米的高度，切割后树木剩下的高度将是 15, 15, 10 和 15，而 Mirko 将从第 1 棵树得到 5 米，从第 4 棵树得到 2 米，共得到 7 米木材。

Mirko 非常关注生态保护，所以他不会砍掉过多的木材。这也是他尽可能高地设定伐木机锯片的原因。请帮助 Mirko 找到伐木机锯片的最大的整数高度 H ，使得他能得到的木材至少为 M 米。换句话说，如果再升高 1 米，他将得不到 M 米木材。

输入格式

第 1 行 2 个整数 N 和 M ， N 表示树木的数量， M 表示需要的木材总长度。

第 2 行 N 个整数表示每棵树的高度。

输出格式

1 个整数，表示锯片的最高高度。



经典例题

输入输出样例

输入 #1

复制

```
4 7
20 15 10 17
```

输出 #1

复制

```
15
```

输入 #2

复制

```
5 20
4 42 40 26 46
```

输出 #2

复制

```
36
```

说明/提示

对于 100% 的测试数据, $1 \leq N \leq 10^6$, $1 \leq M \leq 2 \times 10^9$, 树的高度 $\leq 4 \times 10^5$, 所有树的高度总和 $> M$ 。

核心函数check(mid):

功能: 判断当砍树高度为mid时, 能否获得至少m的木材

实现: 遍历所有树木, 计算每棵树能提供的木材 (仅保留高于mid的部分)

优化: 当累计木材达到m时提前返回, 提高效率

二分查找过程:

初始边界: $l=0$ (最低可砍到地面), r =最高树的高度 (最高只能砍到最高树的顶端)

循环条件: $l \leq r$ (当左右边界交叉时结束)

中间值计算: $mid = l + (r - l) / 2;$

调整策略:

若check(mid)为真: 当前高度有效, 尝试更高高度 ($l = mid + 1$), 同时**记录**当前高度

若check(mid)为假: 当前高度无效, 需要降低高度 ($r = mid - 1$)

时间复杂度:

二分查找次数: $O(\log(\max_height))$, 最多约 30 次

每次 check 函数: $O(n)$

总复杂度: $O(n \log(\max_height))$, 适合处理 $n=1e6$ 的规模

经典例题

```
#include<bits/stdc++.h>
using namespace std;
#define int long long // 定义int为long long类型
int a[1000005]; // 存储树的高度
int n;          // 树的数量
int m;          // 需要的木材总量
// 检查当前高度mid能否获得足够的木材
bool check(int mid) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] > mid) {
            sum += a[i] - mid;
        }
        // 提前退出，避免总和溢出
        if (sum >= m) {
            return true;
        }
    }
    return sum >= m;
}
```

经典例题

```
signed main() {
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    // 确定二分边界
    int l = 0;
    int r = 0;
    for (int i = 0; i < n; i++) {
        r = max(r, a[i]); // 最大可能高度为最高的树
    }
    int ans = 0; // 用ans记录答案
    // 二分查找过程 (l <= r版本)
    while (l <= r) {
        int mid = (l + r) / 2; // 中间高度
        if (check(mid)) {
            // 当前高度可以获得足够木材，尝试更高的高度
            ans = mid; // 记录当前有效高度
            l = mid + 1;
        } else {
            // 当前高度木材不足，需要降低高度
            r = mid - 1;
        }
    }

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

04

总结

TEACHING
COURSEWARE

TEACH

总结

二分过程逻辑：

确定答案的可能范围（左边界 l 为最小值，右边界 r 为最大值）；

计算中点 mid 作为候选答案，通过 $check(mid)$ 验证其是否符合条件；

根据验证结果调整范围：

若 $check(mid)$ 返回 $true$ （当前 mid 满足条件），说明可能存在更优的更大值，因此记录当前 mid 为有效答案，并将左边界右移以尝试更大值，即 $ans = mid$, $l = mid + 1$ ；

若 $check(mid)$ 返回 $false$ （当前 mid 不满足条件），说明需要寻找更小的可行值，因此将右边界左移，即 $r = mid - 1$ ；
重复步骤 2-3，直到 $l > r$ 时循环结束，此时记录的 ans 即为满足条件的最大值。

```
int ans = 0;
while (l <= r) {
    int mid = l + (r - l) / 2;
    if (check(mid)) {
        ans = mid;
        l = mid + 1;
    } else {
        r = mid - 1;
    }
}
cout<<ans<<endl;
```