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

木材厂有 n 根原木，现在想把这些木头切割成 k 段长度**均**为 l 的小段木头（木头有可能有剩余）。

当然，我们希望得到的小段木头越长越好，请求出 l 的最大值。

木头长度的单位是 cm，原木的长度都是正整数，我们要求切割得到的小段木头的长度也是正整数。

例如有两根原木长度分别为 11 和 21，要求切割成等长的 6 段，很明显能切割出来的小段木头长度最长为 5。

输入格式

第一行是两个正整数 n, k，分别表示原木的数量，需要得到的小段的数量。

接下来 n 行，每行一个正整数 L_i，表示一根原木的长度。

输出格式

仅一行，即 l 的最大值。

如果连 $1\mathrm{cm}$ 长的小段都切不出来，输出 `0`。

题目描述

输入输出样例

输入 #1 复制

```
3 7
232
124
456
```

输出 #1 复制

```
114
```

说明/提示

数据规模与约定

对于 100% 的数据，有 $1 \leq n \leq 10^5$，$1 \leq k \leq 10^8$，$1 \leq L_i \leq 10^8 (i \in [1, n])$。

02

知识回归

TEACHING
COURSEWARE

TEACH

单调性：如果值X可行，则所有≤X（或≥X）的值都可行

念

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
“猜测” 答案，并验证该答案是否符合条件，逐步逼
最优解。

心逻辑

定答案的可能范围（左边界left为最小值，右边界right
最大值）；

算中点mid作为 “候选答案”，设计验证函数判断mid是
满足条件；

据验证结果调整范围：

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

03

解题思路

TEACHING
COURSEWARE

TEACH

解题思路

1. 确定搜索范围：

最小值 left：最小可能为 1最大值 right：最大可能为最长原木的长度单调性分析：

如果长度 l 是可行的（能切出至少 k 段），那么所有小于 l 的长度也一定可行

如果长度 l 不可行，那么所有大于 l 的长度也一定不可行

单调性

2. 判断可行性：

对于当前中点 mid，计算所有原木能切割出的小段总数（每根原木贡献 l_i / mid段

如果总数 ≥ k，说明 mid 可行，我们可以尝试更大的长度

如果总数 < k，说明 mid 不可行，需要尝试更小的长度

3. 记录最优解：

使用一个变量 ans 记录当前找到的最大可行长度

当找到可行解时，更新 ans 并继续搜索更大的可能值

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序


```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long  // 定义int为long long
const int N = 1e5 + 5;  // 最大木材数量
int a[N];  // 静态数组存储木材长度
int check(int mid)
{
    int count = 0;
    // 计算当前长度下能得到的总段数
    for (int i = 0; i < n; ++i) {
        count += a[i] / mid;
        if (count >= k) {
            return 1;
        }
    }
    return 0;
}
```

标准程序


```
signed main() {
    int n, k;
    cin >> n >> k;
    int total = 0;
    int mx = 0;
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        total += a[i];
        mx = max(mx, a[i]);
    }

    if (total < k) {
        cout << 0 << endl;
        return 0;
    }
```

标准程序

```
    int l = 1;
    int r = mx;
    int ans = 0;
    // 二分主循环
    while (l <= r) {
        int mid = l + (r - l) / 2;
        if (check(mid)) {
            // 可行，尝试更大的长度
            ans = mid;
            l = mid + 1;
        } else {
            // 不可行，尝试更小的长度
            r = mid - 1;
        }
    }

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