

习题课

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：2025.08.05



目录

- PART-01 题目描述 TEACHING COURSEWARE
- PART-02 知识回归 TEACHING COURSEWARE
- PART-03 解题思路 TEACHING COURSEWARE
- PART-04 标准程序 TEACHING COURSEWARE

01

题目描述

TEACHING
COURSEWARE

TEACH

题目描述

题目描述

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

对于给定的一个长度为 N 的正整数数列 $A_1 \sim A_N$ ，现要将其分成 M ($M \leq N$) 段，并要求每段连续，且每段和的最大值最小。

关于最大值最小：

例如一数列 4 2 4 5 1 要分成 3 段。

将其如下分段：

$[4\ 2][4\ 5][1]$

第一段和为 6，第 2 段和为 9，第 3 段和为 1，和最大值为 9。

将其如下分段：

$[4][2\ 4][5\ 1]$

第一段和为 4，第 2 段和为 6，第 3 段和为 6，和最大值为 6。

并且无论如何分段，最大值不会小于 6。

所以可以得到要将数列 4 2 4 5 1 要分成 3 段，每段和的最大值最小为 6。



题目描述

输入格式

第 1 行包含两个正整数 N, M 。

第 2 行包含 N 个空格隔开的非负整数 A_i ，含义如题目所述。

输出格式

一个正整数，即每段和最大值最小为多少。

输入输出样例

输入 #1

复制

```
5 3
4 2 4 5 1
```

输出 #1

复制

```
6
```

说明/提示

对于 20% 的数据， $N \leq 10$ 。

对于 40% 的数据， $N \leq 1000$ 。

对于 100% 的数据， $1 \leq N \leq 10^5$ ， $M \leq N$ ， $A_i < 10^8$ ，答案不超过 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 $\leq X$ （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

解题思路

TEACHING
COURSEWARE

TEACH

若某个值 mid 是可行解（即能分割成 $\leq m$ 个子数组），则所有大于 mid 的值都是可行解；若 mid 不可行，则所有小于 mid 的值也不可行。单调性。

分段数 $>$ 指定分段次数，说明指定的和太小，应该让和大点，才能让分段次数少点，因此进入右区间 $left = mid + 1$

分段数 = 指定分段次数，此时分段数刚刚好，若是从小到大遍历，可直接退出，但是倘若是二分查找，得继续往左区间看有没有更小的分段次数满足要求，因此 $right = mid - 1$

分段数 $<$ 指定分段次数，说明此时指定的和太大，导致分段数太少，因此要让和小一点，因此进入左区间， $right = mid - 1$

综合三种情况就有

当 分段数 $>$ 指定分段次数，进入右区间

当 分段数 $\leq$ 指定分段次数，进入左区间

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=1e5+10;
int a[N];
int n,m,ans; // 用ans记录答案
```

```
signed main()
{
    int maxa=0;
    int sums=0; // 初始化总和
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
        sums += a[i];
        maxa = max(maxa, a[i]);
    }
    int l = maxa; // 左边界: 最大元素 (最小可能的最大值)
    int r = sums; // 右边界: 总和 (最大可能的最大值)
    // 二分查找, 用l <= r条件
```

标准程序

```
// 二分, 用  $l \leq r$  条件
while(l <= r)
{
    int mid = (l + r) / 2;
    if(check(mid))
    {
        // 当前mid可行, 记录答案并尝试更小值
        ans = mid;
        r = mid - 1;
    }
    else
    {
        // 当前mid不可行, 需要更大的值
        l = mid + 1;
    }
}
cout << ans << endl; // 输出答案
return 0;
```

标准程序

```
bool check(int mid)
{
    int cnt=0; // 分割次数
    int sum=0; // 当前段的和
    for(int i=1;i<=n;i++ )
    {
        // 若当前元素加入后超过mid, 必须分割
        if(sum + a[i] > mid)
        {
            cnt++;
            sum = a[i]; // 新段从当前元素开始
        }
        else
        {
            sum += a[i]; // 加入当前段
        }
    }
    // 总段数为分割次数+1, 判断是否不超过m
    return cnt + 1 <= m;
}
```