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

您正在参加一场团体越野比赛。您的队伍共有 n 名队员，其中第 i 名队员的速度为 v_i，体重为 w_i。

比赛允许每名队员独立行动，也允许一名队员背着另一名队员一起行动。当队员 i 背着队员 j 时，如果队员 i 的体重大于等于队员 j，则队员 i 的移动速度不会变化，仍然为 v_i；如果队员 i 的体重小于队员 j，则队员 i 的移动速度会减去两者的体重差值，即变为 $v_i - (w_j - w_i)$。如果队员 i 的移动速度将变为负数，则队员 i 无法背起队员 j。每名队员最多只能背负另一名队员，被背负的队员无法同时背负其他队员。

所有未被背负的队员中，最慢的队员的速度，即为整个队伍的速度。求整个队伍能达到的最大速度。

输入格式

有多组测试数据。第一行输入一个整数 T 表示测试数据组数，对于每组测试数据：

第一行输入一个整数 n（$1 \le n \le 10^5$）表示队员人数。

对于接下来 n 行，第 i 行输入两个整数 v_i 和 w_i（$1 \le v_i, w_i \le 10^9$）表示第 i 名队员的速度和体重。

保证所有数据中 n 之和不超过 10^5。

输出格式

每组数据输出一个整数，表示整个队伍可以达到的最大速度。

题目描述

【样例解释】

样例数据的最优策略如下：

- 队员 1 背起队员 4。因为 $w_1 > w_4$，因此队员 1 速度不变，仍然为 10。
- 队员 3 背起队员 2。因为 $w_3 < w_2$，因此队员 3 的速度减少 $w_2 - w_3 = 2$，即速度变为 $10 - 2 = 8$。
- 队员 5 独立行动，速度为 9。

因此答案为 8。

题意翻译

显示翻译

输入输出样例

输入 #1 复制

```
2
5
10 5
1 102
10 100
7 4
9 50
2
1 100
10 1
```

输出 #1 复制

```
8
1
```

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

本题的核心是寻找一个最大的期望速度值，使得所有速度不满足该值的人都能被他人背起，且背起后整体速度仍不低于期望速度。这一问题可通过二分答案结合贪心策略高效解决，具体思路如下：

若某个期望速度 x 是可行的（即所有速度小于 x 的人都能被合理背起），则所有小于 x 的速度也一定可行；

若 x 不可行，则所有大于 x 的速度更不可行。

分类与预处理
承载者筛选：从数组 A（已按v + w降序排序）中，筛选出所有速度v >= x的元素作为 “承载者”，计算其最大可承载重量：若承载者的速度为v_i、重量为w_i，则其能承载的最大重量为p[i] = v_i + w_i - x（含义是：承载者自身速度降至x时，可额外背负的重量上限）。由于 A 已按v + w降序排序，p数组自然按从大到小排列。
被承载者筛选：从数组 B（已按w降序排序，且 B 是 A ）中，筛选出所有速度v < x的元素作为 “被承载者”，记录其重量为q[i] = w_i。由于 B 已按w降序排序，q数组自然按从大到小排列。
匹配逻辑
若承载者的数量（p_cnt）少于被承载者的数量（q_cnt），直接返回不可行（无法全部背负）。
按排序后的顺序依次匹配：最大的p[i]（承载能力最强的承载者）对应最大的q[i]（重量最大的被承载者）。
若存在某对p[i] < q[i]，则当前x不可行（因最大的承载者都无法背负最大的被承载者，更小的承载者更不可能）；反之，若所有p[i] >= q[i]，则x可行。

上述逻辑通过预处理时的排序（A 按v + w降序、B 按w降序），避免了在check(x)中重复排序，直接利用有序数组的特性完成高效匹配，确保了二分查找的整体效率。

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序


```
#include<bits/stdc++.h>
using namespace std;
const int N = 100010;
const int INF = 1e9;
int T;int n;
struct node
{
    int v, w;
} A[N], B[N];
int p[N], q[N];
```

标准程序


```
// 排序函数：按w降序
bool cmp(node a, node b)
{
    return a.w > b.w;
}
// 排序函数：按v+w降序
bool cmp1(node a, node b)
{
    return (a.v + a.w) > (b.v + b.w);
}
```

```
int main()
{
    ios::sync_with_stdio(false);//这是什么这是什么这是什么这是什么这是什么这是什么
    cin.tie(0);//这是什么这是什么这是什么这是什么这是什么
    cout.tie(0);//这是什么这是什么这是什么这是什么这是什么
    cin >> T;
```

标准程序


```
while (T--)
{
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> A[i].v >> A[i].w;
        B[i].v = A[i].v;
        B[i].w = A[i].w;
    }
    // 按指定规则排序
    sort(A + 1, A + n + 1, cmp1);
    sort(B + 1, B + n + 1, cmp);
    int l = 0, r = INF;
    int ans = 0;  // 用于记录最优解
    // 使用l <= r的循环条件
```

标准程序

```
    while (l <= r)
    {
        int mid = (l + r) / 2;  // 普通中点计算
        if (check(mid))
        {
            // 当前mid可行，记录答案并尝试更大值
            ans = mid;
            l = mid + 1;
        }
        else
        {
            // 当前mid不可行，尝试更小值
            r = mid - 1;
        }
    }
    cout << ans << endl;
}
return 0;
```

标准程序

```
bool check(int x)
{
    int p_cnt=0, q_cnt=0;  // 重置计数器
    // 收集A中v >= x的元素，计算其可用值
    for (int i = 1; i <= n; i++)
        if (A[i].v >= x)
            p[++p_cnt] = A[i].v + A[i].w - x;
    // 收集B中v < x的元素，获取其w值
    for (int i = 1; i <= n; i++)
        if (B[i].v < x)
            q[++q_cnt] = B[i].w;
    // 如果可用的A元素数量少于需要的B元素数量，不可行
    if (p_cnt < q_cnt)
        return false;
    // 检查每个B元素是否能被对应的A元素覆盖
    for (int i = 1; i <= q_cnt; i++)
        if (p[i] < q[i])
            return false;
    return true;
}
```