

2023-01-16

蛟龙四班第五课节比赛题解

——石硯

目录

- A. 单调的代价
- B. 三堆石子
- C. 有趣的赛制
- D. 气球





A. 单调的代价

题目大意

题目描述

给定一个整数序列 a_1, \dots, a_n , Mas 需要修改其中一些数字,将它调整成一个上升且连续的整数序列

所谓上升且连续,就是指每一个数字恰好比前一个数大 1

若某个数字 a 被改成了 a' ,则定义它的修改工作量为 $|a - a'|$

请找到一种修改方法,使得修改工作量的总和达到最小,输出这个最小值

输入格式

第一行: 单个整数 n ,表示数列长度

第二行: n 个整数,表示 $a_1 \sim a_n$

输出格式

单个整数: 表示最小的修改工作量

数据范围

对于 10% 的数据, $1 \leq n \leq 20, 1 \leq a_i \leq 20$

对于 40% 的数据, $1 \leq n \leq 5000, 1 \leq a_i \leq 5000$

对于 100 的数据, $1 \leq n \leq 200000, -10^9 \leq a_i \leq 10^9$

解析

- 既然是 $a_{i+1} - a_i = 1, i \in [1, n]$ ，我们可以以 $a_1 - 1$ 作为基准数，则 $a_i (i \in [1, n])$ 可以记作 $a_i - i$ ，我们就要让 $a_1 - 1, a_2 - 2, a_3 - 3, \dots, a_n - n$ 相等，就可以转化成[#2141 货仓选址](#)。
- 当然，我们也可以用二分答案来做。每次的`check()`都是 $O(n)$ ，在乘上 $\log_2 10^9$ ，约等于 6×10^6 ，不会超时。

参考代码

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 long long ans; // 注意, ans要是long long
4 int n, a[200005];
5 int main()
6 {
7     cin >> n;
8     for (int i = 1; i <= n; ++ i)
9         cin >> a[i], a[i] -= i; // 最后相等的结果
10    sort(a + 1, a + n + 1);
11    for (int i = 1; i <= n; ++ i)
12        ans += abs(a[i] - a[(1 + n) / 2]); // 减去中位数的差的绝对值
13    cout << ans << '\n';
14    return 0;
15 }
```



B. 三堆石子

题目大意

题目描述

saM 给 Mas 设计了一个游戏:

saM 在地面上放置了三堆石子,数量分别为 a, b, c .

Mas 可进行若干轮下列操作

- Mas 可地面上三堆石子中选择任意非空的两堆
- Mas 从选出的两堆中分别取走一颗石子

当存在超过 1 堆石子为空时,游戏结束

请你计算 Mas 最多能进行多少轮游戏

输入格式

输入三个个整数 a, b, c

输出格式

输出一个整数表示答案

数据规模

对于 10% 的数据, $1 \leq a, b, c \leq 15$

对于 40% 的数据, $1 \leq a, b, c \leq 10^9$

对于 100% 的数据, $1 \leq a, b, c \leq 10^{18}$

解析

- 我们先来分析一下样例：

输入样例1

2 4 6

输出样例1

6

- 在第一组样例中， $a_1 + a_2 = a_3$ ，不难想到，将 a_1 与 a_3 抵消，则抵消完后 $a_2 = a_3$ ，在做一次抵消，得 $\{0,0,0\}$ ，全部用完。
- 所以，当 $a_1 + a_2 = a_3$ 时，结果等于 a_3 。

解析

- 我们再来分析一下样例2：

输入样例2

4 6 4

输出样例2

7

- 在第二组样例中，排完序后， $a_1 + a_2 > a_3$ ，不难想到，将 a_1 和 a_2 各自 $- 1$ ，得 $a_1 + a_2 = a_3$ ，根据上一页的推论可得。
- 不难发现，我们可以全部用完或只剩一个，则答案为 $[(a_1 + a_2 + a_3) \div 2]$ 。

解析

- 如果 $a_1 + a_2 < a_3$ ，则不难发现，至多取 $a_1 + a_2$ 次，就会把 a_1, a_2 取完，游戏结束。

参考代码

```
#include <bits/stdc++.h>
2 using namespace std;
3 long long a[3], ans; // long long
4 int main()
5 {
6     scanf ("%lld%lld%lld", &a[0], &a[1], &a[2]);
7     sort(a, a + 3);
8     if (a[2] >= a[0] + a[1])
9         ans = a[0] + a[1];
10    else
11        ans = (a[0] + a[1] + a[2]) / 2;
12    printf ("%lld", ans);
13    return 0;
14 }
```



C. 有趣的赛制

题目大意

题目描述

在某场时长为 S 的比赛中,有 n 道题,第 i 道题有一个最高得分 a_i 和一个递减因子 b_i

其中 a_i 是第 i 题的最高得分,比赛开始后每过一分钟,该题得分就会减少 b_i

当一道题通过后,选手该题的得分即为确定

选手的整场考试的总得分为选手每题的得分之和

假设 Mas 需要花费 t_i 分钟才能解决第 i 道问题

如何安排做题顺序,能让该场考试的得分最高?

数据规模

对于 10% 的数据, $1 \leq n \leq 10$

对于 60% 的数据, $1 \leq n \leq 10^3$

对于 100% 的数据, $1 \leq n \leq 10^5$

保证 $1 \leq S \leq 10^9, 1 \leq a_i \leq 10^{12}, 1 \leq b_i \leq 10^3, 1 \leq t_i \leq 10^4$

保证 $b_i \times S \leq a_i$, 即每题得分不会变成负数

输入格式

第一行输入两个整数 n, S , 表示该场考试题目数量和比赛时长

接下来 n 行,第 i 行三个正整数 a_i, b_i, t_i , 表示第 i 题的最高得分、递减因子以及通过该题需要的时间

保证 $\sum_{i=1}^n t_i \leq S$, 即 Mas 能在考试时间内解决所有问题

输出格式

输出一个整数能获得的最高分数

解析

- 我们先定义一个结构体，里面存放三个元素 `Score`, `Minus`, `Time`。
- 我们来看看怎样的排序规则是正确的。每次减的元素乘上另外的时间，与时间乘上另外减的元素，形成排序。
- 每次的时间，我们可以计在变量 `total` 里（准确的说是一个前缀和），**分数减去时间乘每一刻减的分数并累加**，就是最终的答案。

参考代码

```
#include <bits/stdc++.h>
using namespace std;
3 typedef long long ll;
4 ll n, s, ans, total;
5 struct node
6 {
7     ll a, b, t;
8 } aa[100005];
9 bool cmp(node x, node y)
10 {
11     return y.t * x.b > x.t * y.b;
12 }
13 int main()
14 {
15     scanf ("%lld%lld", &n, &s);
16     for (ll i = 1; i <= n; ++ i)
17         scanf ("%lld%lld%lld", &aa[i].a, &aa[i].b, &aa[i].t);
18     sort(aa + 1, aa + n + 1, cmp);
19     for (ll i = 1; i <= n; ++ i)
20     {
21         total += aa[i].t;
22         ans += aa[i].a - total * aa[i].b;
23     }
24     printf ("%lld", ans);
25     return 0;
26 }
```



D. 气球

题目大意

题目描述

在一个二维坐标系上,悬浮着 n 只静止不动的气球,第 i 个气球的坐标为 (x_i, h_i)

其中 x_i 表示它的横坐标, h_i 表示它的高度

保证在同一个坐标上,最多只有一只气球

Mas 打算用最少的弓箭射穿所有的气球

每只弓箭射出时需要确定一个高度,当弓箭没有遇到气球时,它会一直保持同样的高度沿 x 轴正方向运动

如果弓箭碰到了气球,气球就会被射穿,弓箭的高度会减少 1,然后继续沿水平方向运动,直到遇到下一个气球

请问, Mas 最少需要射出多少只箭,才能将所有的气球全部射穿?

数据范围

输入格式

第一行: 单个正整数表示 n

接下来 n 行,每行两个整数,表示一只气球的坐标

输出格式

单个正整数,表示最少需要多少只箭才能拿射穿所有的气球

对于 30% 的数据, $1 \leq n \leq 10^3$

对于 60% 的数据, $1 \leq n \leq 10^4$

对于 100% 的数据, $1 \leq n \leq 10^5$

对于全部的数据 $1 \leq x_i \leq n, 1 \leq h_i \leq 2 \times n$

解析

- 每一次枚举，需要 $O(n^2)$ ，只能拿60分。
- 我们还是定义一个结构体，存放高度(height)和x轴坐标。不难想到，要让箭每一次-1，必须要让高度作为第一关键字，坐标作为第二关键字。
- 我们将每一次没有射中的气球做一个标记，记为 vis_i ，并记录当前高度和坐标。每碰到一个未被戳穿的气球(即 $vis_i = 0$)，当前高度-1，当前的坐标改为此气球的坐标，继续往下搜。
- 理论时间复杂度 $O(n \log_2^n)$ 。

参考代码

```
struct node
4 {
5     int x, h;
6     bool operator < (const node t) const
7     {
8         if (h != t.h)
9             return h > t.h;
10        return x >= t.x;
11    }
12 } a[100005];
13 bool vis[100005];
14 int n, ans;
```

```
scanf ("%d", &n);
18 for (int i = 1; i <= n; ++ i)
19     scanf ("%d%d", &a[i].x, &a[i].h);
20 sort(a + 1, a + n + 1);
21 for (int i = 1; i <= n; ++ i)
22     if (! vis[i])
23     {
24         ++ ans;
25         vis[i] = true;
26         int X = a[i].x, H = a[i].h - 1;
27         for (int j = i + 1; j <= n; ++ j)
28             if (! vis[j] && a[j].h == H && a[j].x > X)
29                 vis[j] = true, -- H, X = a[j].x;
30     }
```



Thank you