常规题目解题报告

Problem1

题目链接：https://vjudge.net/contest/146774#problem/G

题目大意：给你 n 个人，他们之间相互欠钱 m 次，问你他们最小的负债率是多少

数据范围：$1 \le n \le 100$; $0 \le m \le 10000$，ai, bi, ci ($1 \le ai, bi \le n$; $ai \ne bi$; $1 \le ci \le 100$)（ai, bi, ci 分别表示欠钱者，被欠钱者，欠钱数目）

解题思路:一开始的考虑的是，因为存在 m 个关系，所以彼此之间可以用欠钱关系连接起来，然后做一个邻接矩阵表示，然后用一个人得到的钱去抵消他欠的钱，之后算 n 个人的负债率，但是感觉有些麻烦，就改了一个

因为 n 不超过 100，所以直接可以暴力遍历，所以假设 A 欠了 B 钱，B 已经拿到了这笔钱，然后 B 因为欠了别人的钱，再把这个钱还给了他欠的人，这样负债率其实减少很多，相当于 A 欠了其他人的钱，对于 B 来说，他的负债率减少，导致整个 n 个人的负债率减少了

所以总结起来就是：

1.用数组初始化每个人的状态为 0

2.用循环依次输入 m 个欠钱关系，对于欠钱者，他的现状态为-ci ，被欠钱者现状态是 ci

3.m 次循环结束后，把现状态为负的相加，然后为正，就是负债率

Problem2

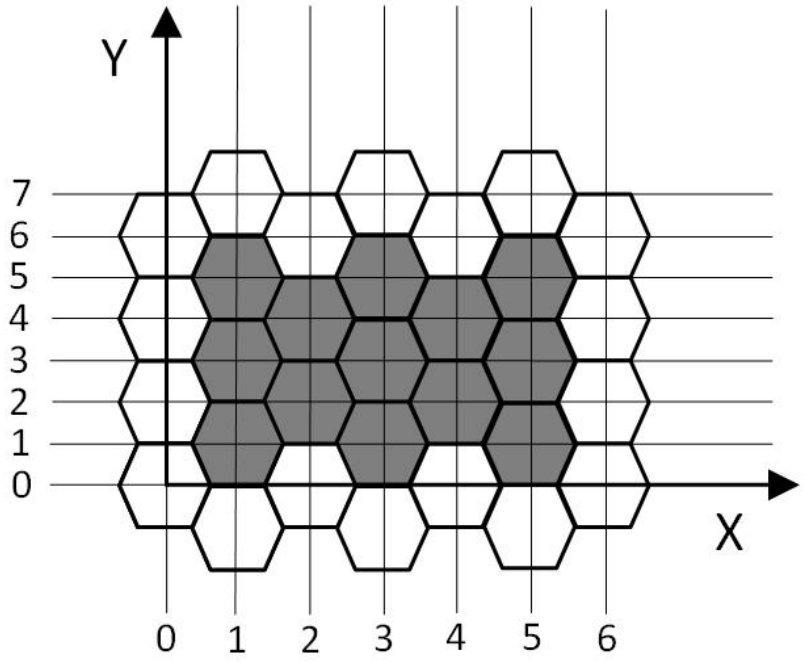
题目链接:https://vjudge.net/contest/146774#problem/C

题目大意:给你一个初始坐标和一个末坐标，然后求在这个范围内一共有

多少个正六边形

数据范围:(- 109 ≤ $x1$ ≤ $x2$ ≤ 10000000000, - 109 ≤ $y1$ ≤ $y2$ ≤ 10000000000)(x1,x2,y1,y2 分别是初末 X 轴坐标，初末 Y 轴坐标）

解题思路：



图形如图所示，这道题就是一个找规律的题.

一开始的时候，是想着每一个单位方格里面是一个正六边形的一半，然后看给的范围内一共有多少个正六边形的一半就可以了，但是发现不是，所以又发现 X 轴都是对应的一竖列正六边形的中心，Y 轴奇数对应一横排的正六边形的中心，偶数是正六边形的边，这种情况随着 X 轴的奇偶性发生变化，所以也是可以找到规律，x,y 奇偶性相同就可以有一个正六边形的中心

所以奇数一共是（x2-x1)/2*(y2-y1)/2, 偶数是((x2-x1)/2+1)*((y2-y1+1)/2+1)

两者相加即可

算法题目解题报告

Problem1

题目链接:https://vjudge.net/contest/146798#problem/E

题目大意:在 n 个学生中有一种病会传染，其中 0 号是已经患病的嫌疑人，然后有 m 个小组进行沟通，要是小组内有嫌疑人，那么这个小组的人都会患病，问一共有多少人患病

数据范围:0 < n <= 30000 and 0 <= m <= 500

涉及算法:并查集

并查集是将一些数据进行合并操作，一开始的时候每个人都是单独独立的，然后根据条件，进行合并，合并后会有些元素的根不再是自己了，同时多棵树也会根据要求变成一棵树，在完成这些条件后，就可以完成问题中的查询得到结果

关键点如下:

1.存在一个数组 per[]用来存储每个元素的根，切记每次要进行初始化，初始化的要求是每个元素的初始化的根是自己

```
//根和高度的初始化
for (i = 0; i < n; i++)
{
    p[i] = i;
    r[i] = 0;
}
//寻找该元素的跟
int find(int i)
```

```
{
    if (p[i] == i)
    {
        return i;
    }
    else
    {
        return find(p[i]);
    }
}
```

2.用一个数组 rank[]来存储该树的长度，当多棵树进行合并的时候，为了能够提高速度和效率，会把 rank[]小的转移到 rank[]大的上面，但是原来的 rank[]不改变

```
//两个根的合并
void unite(int x, int y)
{
    int u, v;
    u = find(x);
    v = find(y);
    if (r[u] < r[v])
    {
        p[u] = v;
    }
    else
    {
        p[v] = u;
        if (r[u] == r[v])
        {
            r[u]++;
        }
    }
}
```

3.查询

```
z = find(0);
for (i = 1; i < n; i++)
{
    if (z == find(i))
    {
        sum++;
    }
```

}

解题思路：

这道题是一道标准的并查集的题，我的思路是把每组的第一个当作这组的根，然后在 m 个小组循环之后，查找 per[0]是几，也就是 0 号嫌疑人的根是谁，查到后开始遍历整个数组，由于数组 n 不大于 30000，所以不会出现爆的可能，遍历每个数组的根是否跟 0 号嫌疑人的相同，相同的话算一个

总结起来就是:

1.初始化根，使得每个元素的初始根都是自己

2.随着循环更新 rank[]的值，也就是树的高度，以此在进行树与树的合并时，能够将小的往大的连边，提高效率

3.查找最初嫌疑人 0 号的根，然后遍历数组中与 0 号跟相同的个数

Problem2

题目链接:https://vjudge.net/contest/146798#problem/A

题目大意:一共有 N 头牛，每头牛的高度为 height，然后询问你 Q 次，问你 A 与 B 内，最高的那头牛与最低的那头牛相差多少

数据范围:$1 \le N \le 50,000$，$1 \le Q \le 200,000$，$1 \le height \le 1,000,000$，$1 \le A \le B \le N$

涉及算法:RMQ

*RMQ 是在给定的一个区间内最值查询的一个算法，主要包括处理和查询两个部分，在预处理中，用的是动态规划去处理，首先*设 A[i]是要求区间最值的数列，F[i, j]表示从第 i 个数起连续 2^j 个数中的最大值，且 F[i,0]就等于 A[i]，为了高效，把 F[i, j]分为两半，F[i，j]就是这两段各自最大值中的最大值，所以状态转移方程 F[i, j]=max（F[i，j-1], F[i + 2^(j-1)，j-1]）

```
void RMQ(int n)
{
    int i, j;
    for (j = 1; j < 20; j++)
    {
        for (i = 1; i <= n; i++)
        {
            if (i + (1 << j) - 1 <= n)
            {
                maxsum[i][j] = max(maxsum[i][j - 1], maxsum[i + (1 << (j
- 1))][j - 1]);
                minsum[i][j] = min(minsum[i][j - 1], minsum[i + (1 << (j
- 1))][j - 1]);
            }
        }
    }
```

切记 j 在外，i 在前

在查询部分，切记你要算出来对于给定的区间进行求解用 log2 求解，并且是要比较在区间中，两部分的最大值

```
for (i = 1; i <= m; i++)
    {
        scanf("%d %d", &x, &y);
        k = (int)log2(y - x + 1);
        ma = max(maxsum[x][k], maxsum[y - (1 << k) + 1][k]);
        mi = min(minsum[x][k], minsum[y - (1 << k) + 1][k]);
        printf("%d", ma - mi);

    }
```

解题思路：这是一道标准的 RMQ，题的正确思路都在上面，这道题 WA 了几发，主要是因为提交格式和超时，超时原因是用的 C++输入输出方式，这是蛮尴尬的一点，c++的比 c 会耗时一些，所以考虑到这种情况，以后这种擦边线球的卡时间的情况，还是优先考虑 c 的输入输出