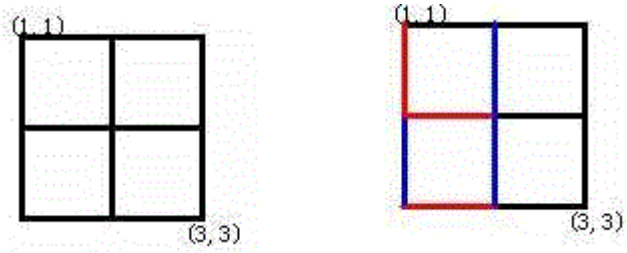
高级数据结构-并查集

一、AcWing 1250. 格子游戏

【题目描述】

Alice和Bob玩了一个古老的游戏：首先画一个$n \times n$的点阵（下图$n = 3$）。

接着，他们两个轮流在相邻的点之间画上红边和蓝边：



直到围成一个封闭的圈（面积不必为1）为止，“封圈”的那个人就是赢家。因为棋盘实在是太大了，他们的游戏实在是太长了！

他们甚至在游戏中都不知道谁赢得了游戏。

于是请你写一个程序，帮助他们计算他们是否结束了游戏？

【输入格式】

输入数据第一行为两个整数n和m。n表示点阵的大小，m表示一共画了m条线。

以后m行，每行首先有两个数字(x, y)，代表了画线的起点坐标，接着用空格隔开一个字符，假如字符是`D`，则是向下连一条边，如果是`R`就是向右连一条边。

输入数据不会有重复的边且保证正确。

【输出格式】

输出一行：在第几步的时候结束。

假如m步之后也没有结束，则输出一行`draw`。

【数据范围】

$1 \leq n \leq 200$

$1 \leq m \leq 24000$

【输入样例】

```
3 5
1 1 D
1 1 R
1 2 D
2 1 R
2 2 D
```

【输出样例】

```
4
```

【分析】

简单的并查集裸题，先将$n \times n$二维数组的各个坐标从左至右从上至下依次映射为$0,1,2,\ldots,n^2-1$，对于每步操作，若两个点已在一个集合内，说明将两点连接后形成了闭环，游戏结束，否则将两点相连，继续判断下一步，若最后一步结束后还没形成闭环，说明平局。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 40010;
int pre[N];
int n, m;

int find(int k)
{
    if (pre[k] == k) return k;
    return pre[k] = find(pre[k]);
}

int main()
{
    cin >> n >> m;
    for (int i = 0; i < n * n; i++) pre[i] = i;
    for (int i = 1; i <= m; i++)

    {
```

```
        char op;
        int x, y, a, b;
        cin >> x >> y >> op;
        a = (x - 1) * n + (y - 1);
        if (op == 'D') b = x * n + (y - 1);
        else b = (x - 1) * n + y;
        int pa = find(a), pb = find(b);
        if (pa == pb) { cout << i << endl; return 0; }
        pre[pa] = pb;
    }
    cout << "draw\n";
    return 0;
}
```

二、AcWing 1252. 搭配购买

【题目描述】

Joe觉得云朵很美，决定去山上的商店买一些云朵。

商店里有n朵云，云朵被编号为$1,2,\ldots,n$，并且每朵云都有一个价值。

但是商店老板跟他说，一些云朵要搭配来买才好，所以买一朵云则与这朵云有搭配的云都要买。

但是Joe的钱有限，所以他希望买的价值越多越好。

【输入格式】

第1行包含三个整数n,m,w，表示有n朵云，m个搭配，Joe有w的钱。

第$2\sim n+1$行，每行两个整数c_i,d_i表示i朵云的价钱和价值。

第$n+2\sim n+1+m$行，每行两个整数u_i,v_i，表示买u_i就必须买v_i，同理，如果买v_i就必须买u_i。

【输出格式】

一行，表示可以获得的最大价值。

【数据范围】

$1\le n\le 10000$

$0\le m\le 5000$

$1\le w\le 10000$

$1 \le c_i \le 5000$

$1 \le d_i \le 100$

$1 \le u_i, v_i \le n$

【输入样例】

```
5 3 10
3 10
3 10
3 10
5 100
10 1
1 3
3 2
4 2
```

【输出样例】

```
1
```

【分析】

---

1. 将所有连通块找出，连通块的祖先结点记录该连通块中所有物品的价钱与价值之和；
2. 将每个连通块都看成是一个物品，做一遍01背包即可。

【代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 10010;
int pre[N], f[N];
int v[N], w[N];
int n, m, vol;

int find(int k)
{
    if (pre[k] == k) return k;
    return pre[k] = find(pre[k]);
```

```cpp
}

int main()
{
    cin >> n >> m >> vol;
    for (int i = 1; i <= n; i++)
    {
        pre[i] = i;
        cin >> v[i] >> w[i];
    }
    while (m--)
    {
        int a, b;
        cin >> a >> b;
        int pa = find(a), pb = find(b);
        if (pa != pb)
        {
            v[pb] += v[pa];
            w[pb] += w[pa];
            pre[pa] = pb;
        }
    }
    for (int i = 1; i <= n; i++)
        if (pre[i] == i)//当该点为祖先结点时计算
            for (int j = vol; j >= v[i]; j--)
                f[j] = max(f[j], f[j - v[i]] + w[i]);
    cout << f[vol] << endl;
    return 0;
}
```

三、AcWing 237. 程序自动分析

【题目描述】

在实现程序自动分析的过程中，常常需要判定一些约束条件是否能被同时满足。

考虑一个约束满足问题的简化版本：假设$x_1,x_2,x_3,...$代表程序中出现的变量，给定n个形如$x_i = x_j$或$x_i \neq x_j$的变量相等/不等的约束条件，请判定是否可以分别为每一个变量赋予恰当的值，使得上述所有约束条件同时被满足。

例如，一个问题中的约束条件为：$x_1 = x_2, x_2 = x_3, x_3 = x_4, x_1 \neq x_4$，这些约束条件显然是不可能同时被满足的，因此这个问题应判定为不可被满足。

现在给出一些约束满足问题，请分别对它们进行判定。

【输入格式】

输入文件的第1行包含1个正整数t，表示需要判定的问题个数，注意这些问题之间是相互独立的。

对于每个问题，包含若干行：

第1行包含1个正整数n，表示该问题中需要被满足的约束条件个数。

接下来n行，每行包括3个整数i,j,e，描述1个相等/不等的约束条件，相邻整数之间用单个空格隔开。若$e=1$，则该约束条件为$x_i=x_j$；若$e=0$，则该约束条件为$x_i\neq x_j$。

【输出格式】

输出文件包括t行。

输出文件的第k行输出一个字符串`YES`或者`NO`，`YES`表示输入中的第k个问题判定为可以被满足，`NO`表示不可被满足。

【数据范围】

$1\leq n\leq 10^5$

$1\leq i,j\leq 10^9$

【输入样例】

```
2
2
1 2 1
1 2 0
2
1 2 1
2 1 1
```

【输出样例】

```
NO
YES
```

【分析】

这道题目明显就有两个关系：变量相等的约束条件和不相等的约束条件。所以说，我们就将相等的约束条件，转化为将两个变量合并到一个集合中。

当相等约束条件全部执行完后，记住是全部执行完毕后，我们再去看不相等约束条件，如果说发现不相等的两个变量在同一个集合，那么就是不满足问题输出`NO`；如果都满足了，那么就是`YES`。

【代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
#include <unordered_map>
using namespace std;

typedef pair<int, int> PII;
const int N = 200010;
int pre[N];
int n, t, idx;
unordered_map<int, int> ids;

int find(int k)
{
    if (pre[k] == k) return k;
    return pre[k] = find(pre[k]);
}

int main()
{
    scanf("%d", &t);
    while (t--)
    {
        scanf("%d", &n);
        idx = 0;//重置离散化下标
        ids.clear();//将离散化哈希表清空
        vector<PII> st;//记录所有不等条件
        for (int i = 1; i <= n << 1; i++) pre[i] = i;//n个约束条件,最多有2n个元素
        while (n--)
        {
            int a, b, c;
            scanf("%d%d%d", &a, &b, &c);
            if (!ids.count(a)) ids[a] = ++idx;

            if (!ids.count(b)) ids[b] = ++idx;
```

```
            a = ids[a], b = ids[b];
            if (!c) st.push_back({ a, b });
            else pre[find(a)] = find(b);//只考虑所有相等条件时不会起任何冲突
        }
        bool flag = true;
        for (auto x : st) if (find(x.first) == find(x.second)) { flag = false; break; }
        if (flag) puts("YES");
        else puts("NO");
    }
    return 0;
}
```

四、AcWing 239. 奇偶游戏（带边权，扩展域）

【题目描述】

小A和小B在玩一个游戏。

首先，小A写了一个由0和1组成的序列S，长度为N。

然后，小B向小A提出了M个问题。

在每个问题中，小B指定两个数l和r，小A回答$S[l \sim r]$中有奇数个1还是偶数个1。

机智的小B发现小A有可能在撒谎。

例如，小A曾经回答过$S[1 \sim 3]$中有奇数个1，$S[4 \sim 6]$中有偶数个1，现在又回答$S[1 \sim 6]$中有偶数个1，显然这是自相矛盾的。

请你帮助小B检查这M个答案，并指出在至少多少个回答之后可以确定小A一定在撒谎。

即求出一个最小的k，使得01序列S满足第$1 \sim k$个回答，但不满足第$1 \sim k+1$个回答。

【输入格式】

第一行包含一个整数N，表示01序列长度。

第二行包含一个整数M，表示问题数量。

接下来M行，每行包含一组问答：两个整数l和r，以及回答`even`或`odd`，用以描述$S[l \sim r]$中有偶数个1还是奇数个1。

【输出格式】

输出一个整数k，表示01序列满足第$1 \sim k$个回答，但不满足第$1 \sim k+1$个回答，如果01序列满足所有回答，则输出问题总数量。

【数据范围】

$N \leq 10^9, M \leq 5000$

【输入样例】

```
10
5
1 2 even
3 4 odd
5 6 even
1 6 even
7 10 odd
```

【输出样例】

```
3
```

【分析】

带边权并查集解法：

首先本题序列长度为10^9，但是最多只有5000个问题，因此最多只会出现10000个点的下标，需要进行离散化预处理。

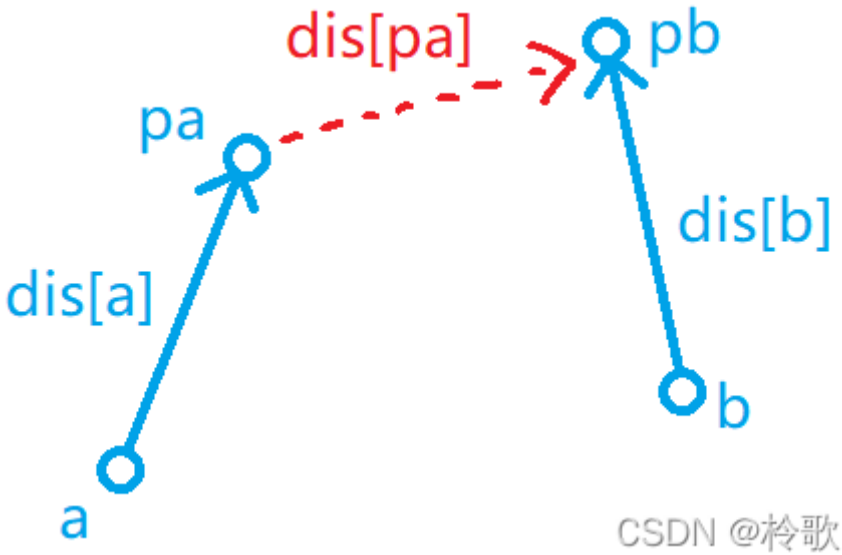
假设$s[i]$表示前i个数中1的个数（前缀和），因此题目所给信息可进行如下转换：

- $[l,r]$中1的个数为奇数→$s[r]-s[l-1]$为奇数→$s[r]$与$s[l-1]$奇偶性不同；
- $[l,r]$中1的个数为偶数→$s[r]-s[l-1]$为偶数→$s[r]$与$s[l-1]$奇偶性相同。

因此我们可以维护一个距离数组$dis[i]$表示i到集合的祖先节点的距离，为1表示与祖先节点奇偶性不同，为0表示相同。

给定两点a,b，若$pre[a]==pre[b]$，说明两点在同一个集合，即关系已经确定，如果题中所给描述为两点奇偶性不同，则两点中必是其中一点与祖先节点奇偶性相同，另一点与祖先节点奇偶性不同，即满足：$(dis[a]+dis[b])\%2==1$；反之，如果题中所给描述为两点奇偶性相同，则满足：$(dis[a]+dis[b])\%2==0$。若不满足以上两种情况则说明出现矛盾。

若$pre[a]!=pre[b]$，说明两点关系未知，因此需要将两点合并，如下图所示：



- 若a,b奇偶性不同，则满足：$dis[a]+dis[pa]+1==dis[b]$，即$dis[pa]=dis[b]-dis[a]-1$（注意在代码中要进行正数取模操作，防止运算结果为负数）；
- 若a,b奇偶性相同，则满足：$dis[a]+dis[pa]==dis[b]$，即$dis[pa]=dis[b]-dis[a]$（同理在代码中要进行正数取模操作）。

扩展域并查集解法：

令x为偶数域，$x+OS$为奇数域，其中OS为偏移量，一般为原始域的最大元素数量。本题原始元素最多为10000个，因此将pre扩展一倍即$N=20000$，将偏移量设置为$N/2$即可。

- 若a,b奇偶性相同，则a为偶数时b也为偶数，即$pre[find(a)]=find(b)$，a为奇数时b也为奇数，即$pre[find(a+OS)]=find(b+OS)$；
- 若a,b奇偶性不同，则a为偶数时b为奇数，即$pre[find(a)]=find(b+OS)$，a为奇数时b为偶数，即$pre[find(a+OS)]=find(b)$；
- 不为以上两种情况则出现矛盾。

【带边权解法代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <string>
#include <unordered_map>
using namespace std;

const int N = 10010;

int pre[N], dis[N];
```

```cpp
int n, m;
unordered_map<int, int> ids;

int find(int k)
{
    if (pre[k] != k)
    {
        int root = find(pre[k]);
        dis[k] = (dis[k] + dis[pre[k]]) % 2;
        pre[k] = root;
    }
    return pre[k];
}

int main()
{
    cin >> n >> m;
    n = 0;
    for (int i = 1; i < N; i++) pre[i] = i;
    for (int i = 0; i < m; i++)//由于是输出矛盾语句的上一句,因此从0开始计数
    {
        string op;
        int a, b, t = 0;
        cin >> a >> b >> op;
        if (!ids.count(--a)) ids[a] = ++n;//注意是s[r]与s[l - 1]
        if (!ids.count(b)) ids[b] = ++n;
        a = ids[a], b = ids[b];
        if (op == "odd") t = 1;//t为0表示ab奇偶性相同,1表示奇偶性不同
        int pa = find(a), pb = find(b);
        if (pa != pb)
        {
            dis[pa] = ((dis[b] - dis[a] - t) % 2 + 2) % 2;
            pre[pa] = pb;
        }
        else if ((dis[a] + dis[b]) % 2 != t)//ab在同一个集合但关系与描述不符
        {
            cout << i << endl;
            return 0;
        }
    }
    cout << m << endl;

    return 0;
```

```
}
```

【扩展域解法代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <string>
#include <unordered_map>
using namespace std;

const int N = 20010, OS = N >> 1;//OS为扩展域的偏移量
int pre[N];//x表示偶数域,x+OS表示奇数域
int n, m;
unordered_map<int, int> ids;

int find(int k)
{
    if (pre[k] == k) return k;
    return pre[k] = find(pre[k]);
}

int main()
{
    cin >> n >> m;
    n = 0;
    for (int i = 1; i < N; i++) pre[i] = i;
    for (int i = 0; i < m; i++)//由于是输出矛盾语句的上一句,因此从0开始计数
    {
        string op;
        int a, b;
        cin >> a >> b >> op;
        if (!ids.count(--a)) ids[a] = ++n;//注意是s[r]与s[l - 1]
        if (!ids.count(b)) ids[b] = ++n;
        a = ids[a], b = ids[b];
        if (op == "odd")//ab奇偶性不同
        {
            if (find(a) == find(b))//如果ab奇偶性相同则矛盾
            {
                cout << i << endl;
                return 0;
            }

            pre[find(a)] = find(b + OS);//当a为偶数时b必为奇数
```

```
            pre[find(a + OS)] = find(b);//当a为奇数时b必为偶数
        }
        else//ab奇偶性相同
        {
            if (find(a) == find(b + OS))//如果ab奇偶性不同则矛盾
            {
                cout << i << endl;
                return 0;
            }
            pre[find(a)] = find(b);//当a为偶数时b必为偶数
            pre[find(a + OS)] = pre[find(b + OS)];//当a为奇数时b必为奇数
        }
    }
    cout << m << endl;
    return 0;
}
```

五、AcWing 238. 银河英雄传说（带边权）

【题目描述】

有一个划分为N列的星际战场，各列依次编号为$1, 2, \ldots, N$。

有N艘战舰，也依次编号为$1, 2, \ldots, N$，其中第i号战舰处于第i列。

有T条指令，每条指令格式为以下两种之一：

- `M i j`，表示让第i号战舰所在列的全部战舰保持原有顺序，接在第j号战舰所在列的尾部。
- `C i j`，表示询问第i号战舰与第j号战舰当前是否处于同一列中，如果在同一列中，它们之间间隔了多少艘战舰。

现在需要你编写一个程序，处理一系列的指令。

【输入格式】

第一行包含整数T，表示共有T条指令。

接下来T行，每行一个指令，指令有两种形式：`M i j`或`C i j`。

其中M和C为大写字母表示指令类型，i和j为整数，表示指令涉及的战舰编号。

【输出格式】

你的程序应当依次对输入的每一条指令进行分析和处理：

如果是`M i j`形式，则表示舰队排列发生了变化，你的程序要注意到这一点，但是不要输出任何信息；

如果是`C i j`形式，你的程序要输出一行，仅包含一个整数，表示在同一列上，第i号战舰与第j号战舰之间布置的战舰数目，如果第i号战舰与第j号战舰当前不在同一列上，则输出-1。

【数据范围】

$N \leq 30000, T \leq 500000$

【输入样例】

```
4
M 2 3
C 1 2
M 2 4
C 4 2
```

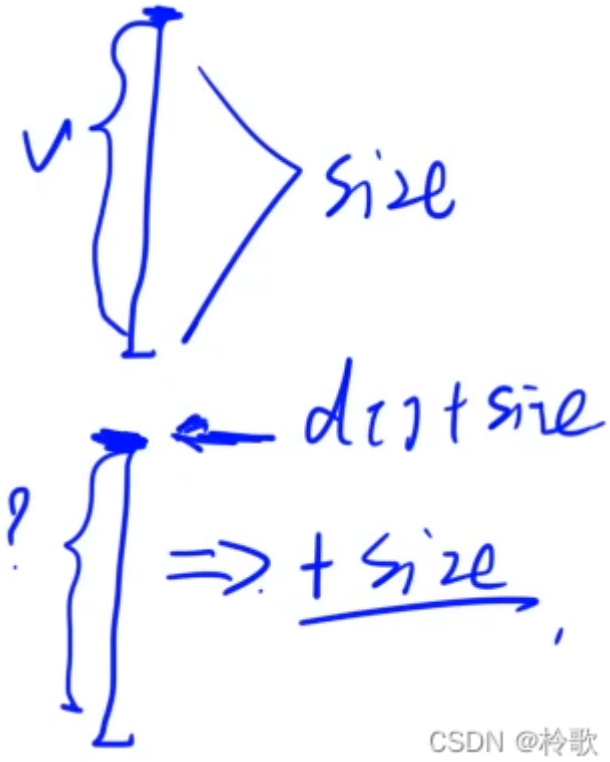
【输出样例】

```
-1
1
```

【分析】

这道题目要注意的就是，我们要边带权，也就是我们不能只处理集合的关系，而是要多一个附带的数组，这个数组就来记录这道题目中最特殊的间隔了多少战舰。听上去很高大上的边带权，实际上就是格外多了一个数组跟随着$merge$和$find$一起走而已。

假设$cnt[i]$表示祖先节点i表示的连通块中点的数量，$dis[i]$表示点i到其祖先节点的距离，则$abs(dis[i]-dis[j])-1$即为i,j之间的战舰数（若i,j为同一个点则距离也为0，因此需要对0取max，不然结果为-1），在进行合并时，将一列战舰a接到另一列战舰b后面，则a的祖先节点的距离将被更新为至b的祖先节点的距离，即b的战舰数量，$dis[pre[a]]=cnt[pre[b]]$，由于dis表示到祖先节点的距离，因此更新了a的祖先节点后其它节点在进行路径压缩（即进行$find$）时也会更新dis的值，如下图所示：



【代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 30010;
int pre[N], cnt[N], dis[N];
int t;

int find(int k)
{
    if (pre[k] != k)
    {
        int root = find(pre[k]);
        dis[k] += dis[pre[k]];
        pre[k] = root;
    }
    return pre[k];
}

int main()

{
```

```
    ios::sync_with_stdio(false);
    cin >> t;
    for (int i = 1; i < N; i++) pre[i] = i, cnt[i] = 1;
    while (t--)
    {
        char op;
        int a, b;
        cin >> op >> a >> b;
        int pa = find(a), pb = find(b);
        if (op == 'M')
        {
            dis[pa] = cnt[pb];
            cnt[pb] += cnt[pa];
            pre[pa] = pb;
        }
        else if (pa != pb) cout << -1 << endl;
        else cout << max(0, abs(dis[a] - dis[b]) - 1) << endl;
    }
    return 0;
}
```