并查集初步

Chaigidel

November 6, 2019

效实中学

乱七八糟

乱七八糟

我真的不会并查集

基础

基础

并查集维护多个元素的不交集（每个元素只能属于一个集合）

本质上说并查集是一个森林

```
int f[N];
int getf(int x) { return (x == f[x] ? x : (f[x] = getf(f[x]))); }
void merge(int x, int y) { f[getf(x)] = getf(y); }
```

路径压缩最坏复杂度 $O(\log n)$

「NOI 2015」程序自动分析

有 n 个变量，$x_1, x_2, x_3, ...$

给定 m 个形如 $x_i = x_j$ 或 $x_i \neq x_j$ 的约束条件

判定是否存在一种取值方式满足约束

$m \leq 10^6,\ n \leq 10^9$

离散化 $i,\ j$

$x_i = x_j$ 能够用并查集轻易地维护

最后再去判断不等条件的满足情况

传递性

杂题选讲

「NOI 2002」银河英雄传说

懒得复制了

定义 d_x 为 x 到 f_x 的距离

getf 后，$f_x = f_y$，答案即为 $|d_x - d_y| - 1$

```
int getf(int x){
    if (x == fa[x]) return x;
    int rt = getf(fa[x]); d[x] += d[fa[x]];
    return fa[x] = rt;
}
void merge(int x, int y){
    x = getf(x), y = getf(y);
    if (x != y){
        d[y] = siz[x]; // 并到尾部
        siz[x] += siz[y], fa[y] = x;
    }
}
```

「NOIP 2010」关押罪犯

S 城现有两座监狱，一共关押着 N 名罪犯，编号分别为 $1\sim N$ 。他们之间的关系自然也极不和谐。很多罪犯之间甚至积怨已久，如果客观条件具备则随时可能爆发冲突。我们用「怨气值」（一个正整数值）来表示某两名罪犯之间的仇恨程度，怨气值越大，则这两名罪犯之间的积怨越多。如果两名怨气值为 c 的罪犯被关押在同一监狱，他们俩之间会发生摩擦，并造成影响力为 c 的冲突事件。

每年年末，警察局会将本年内监狱中的所有冲突事件按影响力从大到小排成一个列表，然后上报到 S 城 Z 市长那里。公务繁忙的 Z 市长只会去看列表中的第一个事件的影响力，如果影响很坏，他就会考虑撤换警察局长。

在详细考察了 N 名罪犯间的矛盾关系后，警察局长觉得压力巨大。他准备将罪犯们在两座监狱内重新分配，以求产生的冲突事件影响力都较小，从而保住自己的乌纱帽。假设只要处于同一监狱内的某两个罪犯间有仇恨，那么他们一定会在每年的某个时候发生摩擦。那么，应如何分配罪犯，才能使 Z 市长看到的那个冲突事件的影响力最小？这个最小值是多少？

Sol 1 扩展域

贪心，优先满足仇恨度大的罪犯

考虑正在让 x, y 不在同一个集合

x, y 不在同一个集合的必要条件是

- x 的敌人和 y 没有冲突
- y 的敌人和 x 没有冲突

Sol 1 扩展域

贪心，优先满足仇恨度大的罪犯

考虑正在让 x, y 不在同一个集合

x, y 不在同一个集合的必要条件是

- x 的敌人和 y 没有冲突
- y 的敌人和 x 没有冲突

如何维护呢?

对于一个罪犯 x 维护不能和他在一个监狱的罪犯

用一个虚点 $x + n$

满足 $x,\ y$ 可以不在同一个集合的条件就是 `getf(x) != getf(y)`

`merge(x + n, y), merge(y + n, x)`

Sol 2 边带权

类似一个 $\mod 2$ 的操作吧

- 若 d_x 为 0，则 x 与 f_x 在一个监狱
- 若 d_x 为 1，则 x 与 f_x 不在一个监狱

进而推广到奇数和偶数

```
void merge(int x, int y) {
    int rx = getf(x);
    int ry = getf(y);
    if (getf(x) == ry) return;
    if (d[y] & 1) d[ry] = 0;
    else  d[ry] = 1;
    f[ry] = x;
}
```

```cpp
void merge(int x, int y) {
    int rx = getf(x);
    int ry = getf(y);
    if (d[y] & 1) d[ry] = d[x];
    else d[ry] = d[x] + 1;
    // 等价写法
    // if ((d[y] ^ d[x]) & 1) d[ry] = 0;
    // else d[ry] = 1;
    f[ry] = rx;
}
```

「NOI 2001」食物链

三类动物 A,B,C

A 吃 B，B 吃 C，C 吃 A。

第一种说法是 “1 X Y”，表示 X 和 Y 是同类。

第二种说法是 “2 X Y”，表示 X 吃 Y 。

K 句话，这 K 句话有的是真

当前的话与前面的某些真的话冲突，就是假话

当前的话表示 X 吃 X，就是假话

当前的话中 X 或 Y 比 N 大，就是假话

Sol 1 扩展域

建三个域，$self$, $enemy$, eat 表示自身，天敌，食物

与“x 和 y 是同类”矛盾

- x 吃 y
- y 吃 x

（如果你觉得不够充分就再想想，因为你发现就三种情况）

与“x 吃 y”矛盾

- x 是 y
- y 吃 x

Sol 2 边带权

类似一个 $\mod 3$ 的操作吧

比较套路

感觉环的性质比较重要