HZOJ-72 猜拳

宋星霖

2024 年 2 月 25 日

1 题目描述

1.1 题目描述

在一次聚会中，每人拿着一张印有石头、剪刀、布的卡片，每个人具体拿得是哪种卡片不得而知。

现在告诉你某些人之间的胜负关系，并会询问某两个人之间的对战结果，人按照从 1 到 n 编号。

对于每个询问，请给出正确的回答：`Win(胜)`、`Loss(负)`、`Tie(平)`

1.2 输入

第一行输入两个整数 $n, m(1 \leqslant n \leqslant 10000, \quad 3 \leqslant m \leqslant 10000)$

接下来 m 行，每行三个整数 a, b, c ($a \in [1, 2]$, $1 \leqslant b, c \leqslant n$)

1. 当 $a = 1$ 时，代表新增一条已知信息，表示 b, c 对战中 b 胜
2. 当 $a = 2$ 时，代表根据以上信息，询问 b, c 对战中 b 的结果

如果出现某条新增的信息与之前的信息发生冲突，就忽略此条信息。

1.3 输出

对于每个 $a = 2$ 的操作，输出`Win`、`Loss`、`Tie`或`Unknown`代表对战双方的结果。

2 算法解析

对于这道题目，需要用到带权并查集。

2.1 第 1 步：构建带权并查集

带权并查集就是每一条边上面也有对应的数字。在这里，笔者与读者约定：0 代表 Tie,1 代表 Loss,2 代表 Win。

带权并查集的实现方法也很简单。只需要多开一个 val 数组，表示第 i 个节点和父节点的结果。

对于这个题目，如果两个节点没有联通，就可以输出 `Unknown` 。

2.2 第 2 步：设计权值

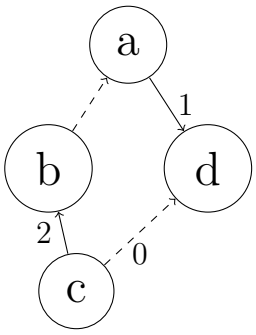
因为 0 代表 Tie,1 代表 Loss,2 代表 Win，可以得到权值以 3 为周期重复结果。所以，计算两个点的权值可以统计从 A 点走到 B 点的权值总和，再对 3 取余就是 A 点到 B 点的权值。

因为 A 点到 B 点是 2，B 点到 A 点是 1，又因为-2 和 1 是同余数系，所以 B 点到 A 点也可以是-2。类似的，如果 A 点到 B 点是 1，B 点到 A 点可以是-1。

所以，统计 A 点到 B 点权值总和的方法是：正着走加权值，倒着走减权值。

2.3 第三步：设计合并方法

对于下面的带权并查集，应该怎么合并呢?



其实，b 到 a 的权值是 b 到 c,c 到 d,d 到 a 的权值之和对 3 取余的值。也就是 $-2+0+1=-1$。不过，结果是负数，我们可以先加 1 个 3，再取余。也就是 $(-1+3) \mod 3=2$。

3 代码演示

Listing 1: HZOJ-72

```
#include <bits/stdc++.h>
using namespace std;
class UnionSet {
    public:
    UnionSet (int n) : fa (n + 1), code (n + 1, 0) {
        for (int i = 0; i <= n; i++) {
            fa[i] = i;
        }
```

```
    }
    int find (int x) {
        if (fa[x] == x)
        return x;
        int xx = find (fa[x]);
        code[x] = (code[x] + code[fa[x]] + 3) % 3;
        return fa[x] = xx;
    }
    void merge (int a, int b) {
        int aa = find (a), bb = find (b);
        if (aa == bb)
        return;
        code[aa] = (2 - code[a] + code[b] + 3) % 3;
        fa[aa] = bb;
    }
    vector<int> fa, code;
};
int main () {
    int n, m;
    scanf ("%d%d", &n, &m);
    UnionSet u (n);
    int a, b, c;
    for (int i = 0; i < m; i++) {
        scanf ("%d%d%d", &a, &b, &c);
        if (a == 1) {
            u.merge (b, c);
        } else {
            if (u.find (b) != u.find (c)) {
                printf ("Unknown\n");
                continue;
            }
            switch ((u.code[b] - u.code[c] + 3) % 3) {
                case 0:
                printf ("Tie\n");
                break;
                case 1:
                printf ("Loss\n");
                break;
                case 2:
                printf ("Win\n");
                break;
            }
        }
    }
    return 0;
}
```