数据结构作业一、带权并查集

并查集

基本介绍

- 并查集是一种树形的数据结构，它用于处理一些不交集的合并及查询问题
- 基本概念：
    - 我们将树的根节点称之为一些列衍生节点的祖先

> 例如，下图中，我们将a称为h,i,j等节点的祖先，将k称为q,r,t等节点的父亲

    - 我们将节点的上层节点称之为那个节点的父亲

> 例如，下图中，我们将l称之为p的父亲，值得注意的是，祖先节点的父亲节点就是它自己

并查集基本操作

并查支持两种操作：查找（Find）和合并（Union）

查找

查找：确定某个元素处于哪个子集（某个节点属于哪个祖先）

- 一般算法：

```
int fa[MAXN];       // 这个数组用来记录一个节点的父亲节点
int find(int x)     // 寻找 x 的祖先
{
    if (fa[x] == x) return x;// 如果 x 是祖先节点则返回
    else return find(fa[x]);  // 如果不是祖先节点，就向上继续查找祖先节点
}
```

- 路径压缩优化

```
int find(int x)
{
    if (x != fa[x])  // x 不是祖先，即 x 不是该集合的代表
        fa[x] = find(fa[x]);  // 在查找的过程中同时进行路径压缩
    else return fa[x];  // 如果是祖先节点则直接返回
}
```

> 路径压缩的过程可以用下面一张图进行表示

合并

合并：将两个子集合并成一个集合

- 事实上，合并操作并不会创造一个新的并查集，而是将一个并查集的祖先节点作为另一个并查集的祖先节点，这样就完成了两个并查集的合并

```
void unionSet(int x, int y)
{
    x = find(x);   // 找到 x 所在并查集的祖先节点
    y = find(y);   // 找到 y 所在并查集的祖先节点
    fa[x] = y;  // 把 y 的祖先节点变成 x 的祖先节点的父节点
}
```

- 时空复杂度分析
    - 并查集的时间复杂度证明不作为本次作业的研究内容，但是我们可以给出同时使用**路径压缩**之后得到的并查集每个操作的平均时间为$O(mlog(n))$，其增长极其缓慢，也就是说其单次操作的平均运行时间可以认为是一个很小的常数
    - 采用数组，空间复杂度显然为$O(n)$

带权并查集

- 在实际应用上，在并查集的边上定义某种权值、以及这种权值在路径压缩时产生的运算，可以解决更多的问题
- 其中，权值代表着**当前节点与父节点的某种关系**

带权并查集的实现和应用

洛谷P2014[NOI2001] 食物链

动物王国中有三类动物 A,B,C，这三类动物的食物链构成了有趣的环形。A 吃 B，B 吃 C，C 吃 A。

现有 N 个动物，以 1 - N 编号。每个动物都是 A,B,C 中的一种，但是我们并不知道它到底是哪一种。

有人用两种说法对这 N 个动物所构成的食物链关系进行描述：

- 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 。

此人对 N 个动物，用上述两种说法，一句接一句地说出 K 句话，这 K 句话有的是真的，有的是假的。当一句话满足下列三条之一时，这句话就是假话，否则就是真话。

- 当前的话与前面的某些真的话冲突，就是假话
- 当前的话中 X 或 Y 比 N 大，就是假话
- 当前的话表示 X 吃 X，就是假话

你的任务是根据给定的 N 和 K 句话，输出假话的总数。

> 输入格式

第一行两个整数，N，K，表示有 N 个动物，K 句话。

第二行开始每行一句话（按照题目要求，见样例）

> 输出格式

一行，一个整数，表示假话的总数。

> 样例 #1

- 样例输入 #1

```
100 7
1 101 1
2 1 2
2 2 3
2 3 3
1 1 3
2 3 1
1 5 5
```

- 样例输出 #1

```
3
```

> 提示

1 ≤ N ≤ 5 * 10^4，1 ≤ K ≤ 10^5

实现思路和程序清单

基本定义

- `enum _relations = {same, eating, eaten}`，分别代表同种生物，捕食关系，被捕食关系
- `_relations re[x]`数组表示一种生物和父亲节点之间的关系，这个值应该初始化为0，表示自己和自己属于同种生物

基本操作

- 权值的叠加
- 权值的相减
- 合并 对于我们规定的这三种操作，应该满足下面的关系

合并时：`re[fa1] = _relations((3 - re[a] + re[b]) % 3);`或`re[fa1] = _relations((3 - re[a] + re[b] + 1) % 3);`

查找时：`re[fa1] = _relations((3 - re[a] + re[b]) % 3);`

程序清单：`coding/code_1.cpp`