并查集讲义 2019-05-25

1. 并查集基本操作

①合并两个点所在集合

②查询某个点所在集合的代表点 -> 进而可以判断两个点是否属于同一集合

```
int fa[maxn];

void init() // 初始化：每个点构成一个集合，代表点为自己
{
    for(int i = 1; i <= n; i++)
        fa[i] = i;
}

int get(int x)
{
    if(fa[x] == x) return x;
    return get(fa[x]);

    //Or below ↓
    //while(fa[x] != x) x = fa[x];
    //return x;
}

void merge(int x, int y)
{
    fa[get(x)] = get(y);//把x的根并到y的根上
}
```

1.1. 例题

P3367 【模板】并查集 >> 参考代码

然而以上代码是过不了这个题的，有3个数据过不了，哪些环节可能时间开销较多?

思考1分钟

2. 优化方案：路径压缩

每次查询一个点的代表点，讲从该点向上直到找到根，经过一条链，特别构造的数据会把链设计的非常长，以至于多次get查询就能超时。那么，我们能不能不经过链，而是一步找到根呢？

回顾：并查集的本质

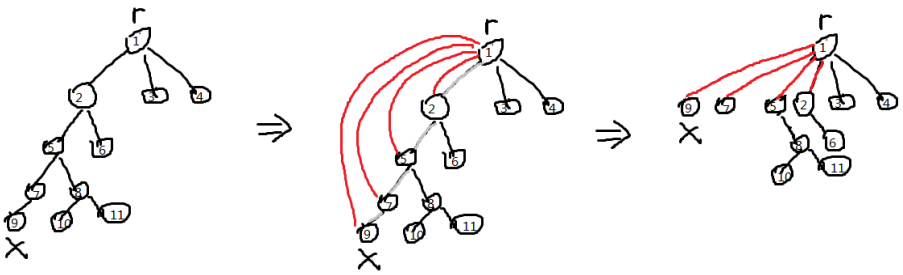
按照基本操作得到的并查集，会是一个深度>=2的森林，但是树上链的信息对我们并没什么用。这样我们就想到，把fa[x]设置为集合的根如何。

思考，按照上述操作是不是可以按下面方式写？

```
void merge(int x, int y)
{
    //fa[get(x)] = get(y);
    fa[x] = get(y);//错误写法
}
```

显然，上述写法逻辑上是错误的，这样只能把x并过去，x所在集合其他元素并没有跟着变。

既然在merge中不好写优化，我们不妨在get中来写 >> 通过get操作，把x到根路径上的点都挂到根节点上



以递归方式来写，①从x出发向上走直到根；②获得根节点，从上往下把路径上的点挂到根上

当然，也可以 非递归来写：①向上找根的过程中，记录下路径上所有节点；②把路径上的点都挂到根上

```
int get(int x)
{
    if(fa[x] == x) return x;
    int r = get(fa[x]);
    fa[x] = r;
    return r;
}
```

```
int lst[maxn];
```

```
int get(int x)// 非递归写法
{
    int k = 0;
    while(fa[x] != x)
    {
        lst[++k] = x;
        x = fa[x];
    }
    int r = x;
    for(int i = 1; i <= k; i++)
        fa[lst[i]] = r;
    return r;
}
```

在get中进行路径压缩还有一个好处：不需要查根的点不处理，均摊下来节省了时间。

这也给我们展示了一种“**懒操作**”的思想，一些操作可前可后（操作本身也有时间开销），那么我们只在万不得已的时候来做就好。

3. 集合大小

如果我们不仅有合并集合及判断是否属于同一集合，还要查看集合的大小（元素个数），我们仍然可以用并查集实现。只需要开一个size数组即可。

注意到一点：每个集合有一个代表点，只有代表点的size才有意义。

再注意到一点：只有merge操作才会改变集合大小。

于是我们在原来基础上，稍加改动即可

```
int sz[maxn];
init()
{
    for(int i = 1; i <= n; i++)
    {
        fa[i] = i;
        sz[i] = 1;
    }
}
void merge(int x, int y)
```

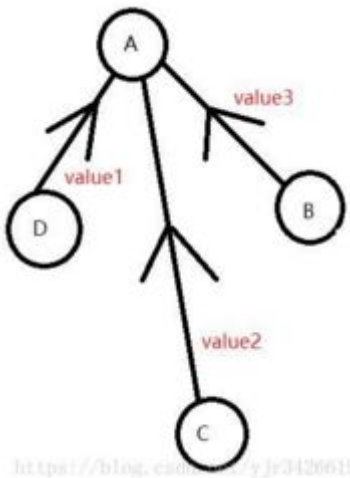
```
{
    int r1 = get(x);
    int r2 = get(y);
    if(r1==r2) return;//这条不写的话，sz可能会多算
    fa[r1] = r2;
    sz[r2] += sz[r1];
}
```

3.1. 例题

网络交友（YCOJ）>> 参考代码

4. 带权并查集

基于路径压缩，带权并查集就是：

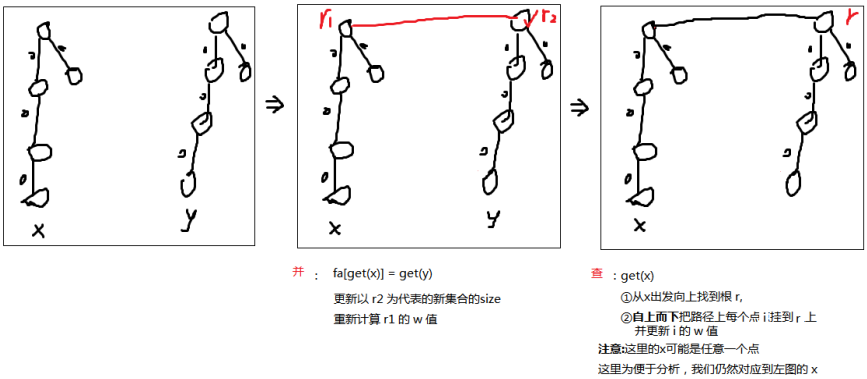


可以看到它的每一条边都记录了每个节点到根节点的一个权值，这个权值设为什么由**具体的问题而定**，一般都是两个节点之间的某一种相对的关系，但是考虑到权值就会有两个问题：

1.每个节点都记录的是与根节点之间的权值，那么在get的路径压缩过程中，权值也应该做相应的更新，因为在路径压缩之前，每个节点都是挂在其父节点下面的，权值自然也是与其父节点之间的权值

2.两个集合合并的时候，权值也要做相应的更新，因为两个集合的根节点不同。

（以上引自https://blog.csdn.net/yjr3426619/article/details/82315133 ）



并 ： fa[get(x)] = get(y)

更新以 r2 为代表的新集合的size

重新计算 r1 的 w 值

查 ：get(x)

①从x出发向上找到根 r,

②**自上而下**把路径上每个点 i 挂到r 上并更新 i 的 w 值

注意:这里的x可能是任意一个点

这里为便于分析，我们仍然对应到左图的 x

4.1. 例题

接龙 (YCOJ)

4.1.1. 问题化简：如果只有查询，没有合并，你会怎么做?

4.1.2. 前缀和与差分 思想

```
c(i,j) = abs(i-j) - 1
```

4.1.3. 现在考虑到会有合并

那么原来问题的特性就被打破了——每个纸牌前面到底有多少纸牌？不清楚！

然而我们可不可以**维护**这个特性呢？可以

我们使用并查集来维护

定义 w[x] 代表编号为x的纸牌前面有多少张纸牌。这样问题不就和上面的一样简单了?

```
c(i,j) = abs(w[i]-w[j]) - 1
```

4.1.4. 怎么维护?

先把与集合大小有关的代码打完

①get操作中，在把路径上每个点 i 挂到根节点之前，通过父亲更新其w值

```
...
w[i] += w[fa[i]]
fa[i] = r
...
```

②集合合并时，集合A的根r1挂到 集合B的根r2上

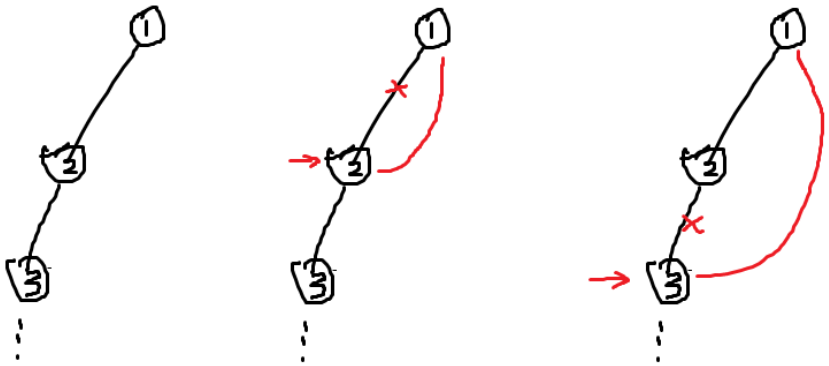
此时 r1前面的纸牌数量发生变化，重新计算：

```
w[r1] = sz[r2]
```

那么r1下面的点的w值怎么办？——在步骤①中计算！

4.1.5. 最后再来验证一下get操作

举个例子，如下



get操作：从下往上找根，再从上往下更新

> *对于节点1，由于1是根，不做操作*

> *对于节点2，进行操作*

```
w[i] += w[fa[i]]
fa[i] = r
```

由于fa[2] 是根，所以w[fa[2]] == 0，于是w并不受影响

fa[2] 再次挂到根上，fa也不手影响

> *对于节点3，进行操作*

```
w[i] += w[fa[i]]
fa[i] = r
```

w和fa均按照我们的意图进行了更新

> *对于3以下的节点，均会与节点3一样，依次得到更新*

综上所述，以上我们的分析与代码的写法是可行的，借此，我们所需的各组信息均得到了维护

5. 带权并查集解种类关系问题

5.1. 例题

P2024 NOI2001 食物链 --> 参考代码

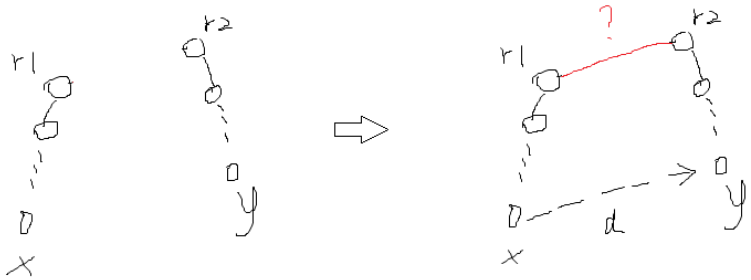
使用并查集，设置权值w表示点到根的距离

同一集合：w[x]%3与w[y]%3相同，则x与y为同一种类

w[x]%3与w[y]%3 差值为1 或-2，则x吃y (x->y)

不同集合的w[x],w[y] 没有比较意义 ,也就不存在谎言

两个集合合并时，以其中一个集合的w 值为基准 ，另一集合的w值需 重算



设w'为合并后的权值表示， d 为合并后x与y的差值

表示为 w'[x] - w'[y] = d

那么有：

w'[y] = w[y]

w'[x] = w'[y] + d

w[r1] = w[r2] = 0

w'[r2] = w[r2]

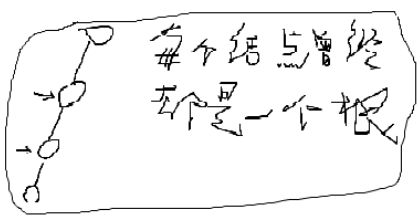
据此，你是否能推算出

w'[r1] = ？

这样，在合并的时候，只用重算r1的w值即可;

而r1下面的点可以在get时重算

w'[i] += w[fa[i]]



接下来，我们讨论怎么计算w'[r1]

注意到一点，x与r1的权值之差是恒定的

（其已有的种间关系不能改变），

那么问题就简单了，如下：

w'[x] - w[x] = w'[r1] - w[r1]

=> w'[r1] = w'[x] - w[x] + w[r1]

= w[y] + d - w[x]

注意，结果有可能为负，这时可以稍微修正一下：

w'[r1] = w'[r1]%3 + 3

6. 并查集与二分图

6.1. 例题

昆虫的生活 >> 参考代码

基本思路：

两只昆虫有互动关系，我们就尝试把它们安排到不同集合中

> Tips:
> 并查集擅长做的是合并，但是“保证不在一个”这个特性怎么实现?
>
> 技巧：对于每个点i，开一个虚点n+i，表示对i操作的**反面**
>
> 如果把x和y安排到不同集合，就等价于把x和y+n安排到一起，把y和x+n安排到一起

如果检测到这两只昆虫已被安排到一起了，则假设不正确

...

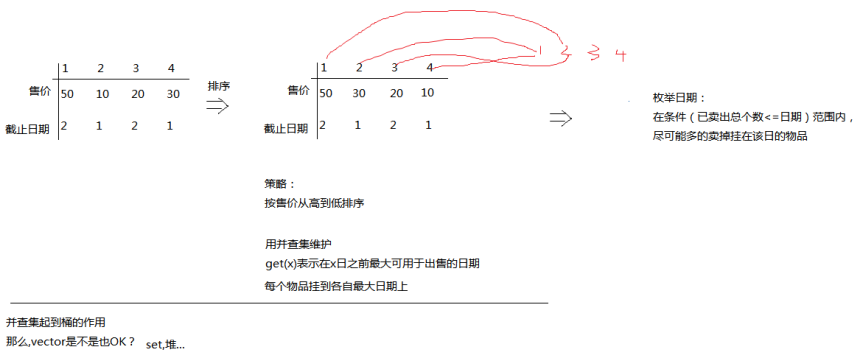
如果所有昆虫都安排上了，那么说明假设是正确的。

小结：二分图判断的两种思路：①dfs染色；②并查集

请尝试**思考**，两种思路有哪些相同之处?

7. 并查集的巧妙运用

poj1456 Supermarket



与之相似的：桶的思想

另 **贪心**：把产品按最后期限从大到小排，然后从最大的那个期限时间往前一天一天推，如果当前天的期限产品没有完全卖出，就放到前一天期限，和前一天最后期限的产品一走卖，利润大的就先卖。没有卖出的又放到前一天去卖，这样从大期限往小期限推，因为小期限的产品不可能在大期限时间去卖，而大期限的产品可以在小的期限去卖。

8. 本节必做习题

P1536 村村通

P1551 亲戚

P2921 [USACO08DEC]在农场万圣节Trick or Treat on the Farm

P1330 封锁阳光大学

[P1197 [JSOI2008]星球大战

P1892[BOI2003]团伙

P3958 奶酪

关押罪犯 (NOIP2010)