并查集及其应用

宁波市镇海蛟川书院 杨明天

目录

1.基本概念

1.1.定义

- 一堆没有交集的集合。
- 简单的例子：
- A = {1, 3, 7, 8}
- B = {4, 5}
- C = {2}
- A、B、C构成并查集。
- D = {1, 2, 3}
- A、B、C、D不构成并查集。

1.2.基本操作

- Union：将两个集合做并集，合并成一个集合。
- Find：找找看一个元素是在哪个集合里面。
- IsConnected：询问两个元素是否属于同一集合。
- Count：统计集合总数。
- Cardinality：统计某一集合所包含的元素数。
- Singleton：判断集合是否被合并过。

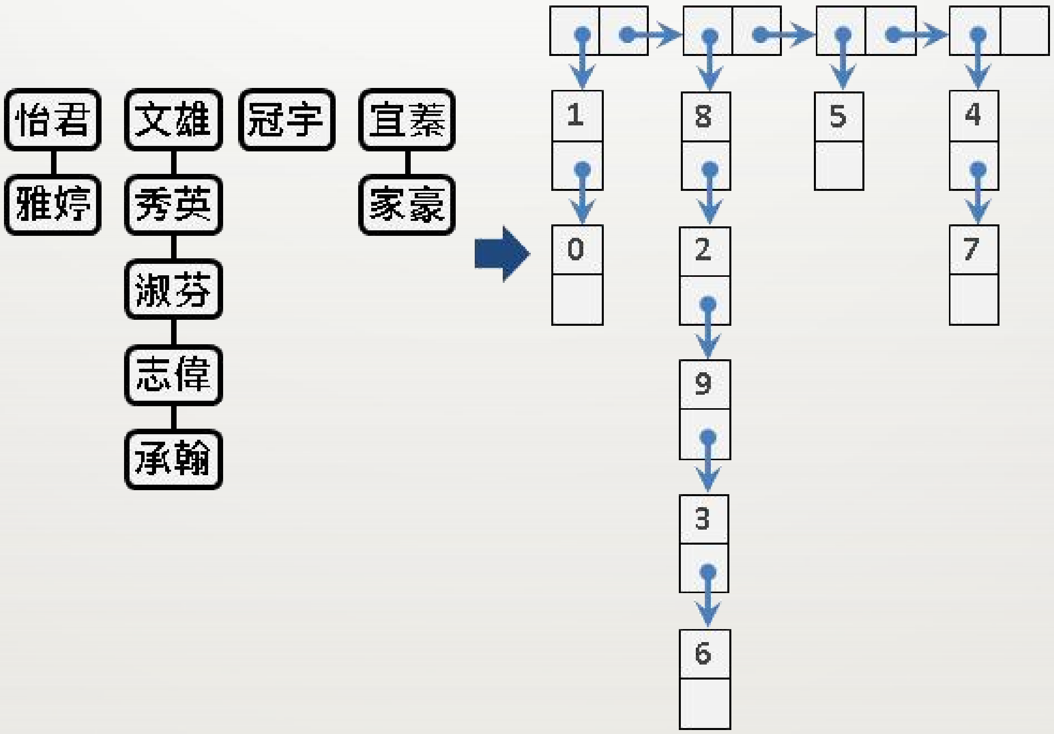
1.3.常见应用

- 维护不相交集合；
- 计算无向图的连通分量；
- 最近公共祖先(LCA)；
- Kruskal 求最小生成树；
- 带限制的作业排序。

1.4.实现方式

- 并查集常用的实现方式有 List 、 Array 、 Forest 三种。其中 Array 效率最低，List次之，Forest效率最高，同时也是最常用的实现方式。

1.4.1.实现方式1：List

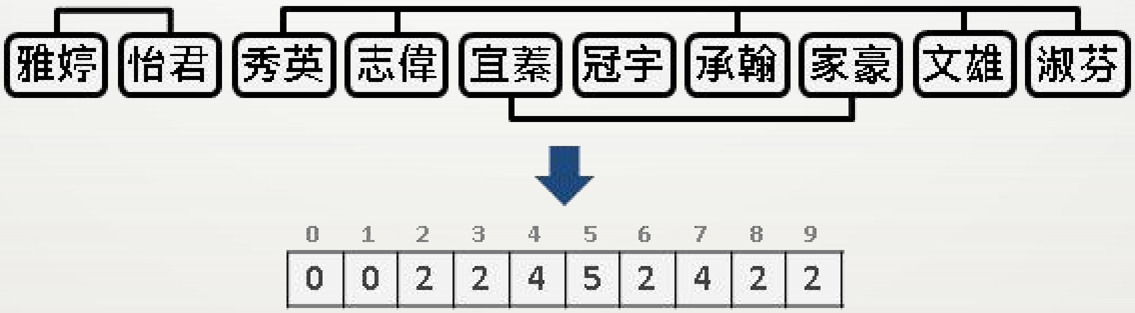


9

1.4.1.实现方式1：List 复杂度

- 时间复杂度：
  - build、find：O(1)
  - union：O(n)
- 空间复杂度：O(n)

1.4.2.实现方式2：Array



- 让一个数组的第x格代表第x人，格子里填上这个人所属的家族编号。若两个人在同一家族，他们的格子里就会有相同的家族编号。

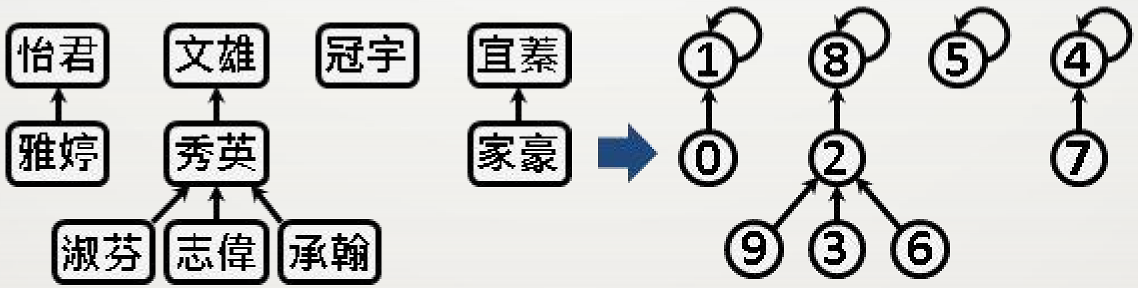
11

1.4.2.实现方式2：Array 复杂度

- 时间复杂度：
  - union: O(n)
  - find、equivalence、cardinality、singleton: O(1)
  - 如果全部的人都union一遍，每次要花O(n)的时间，总共花O(n^2)时间。
- 空间复杂度：O(n)

1.4.3.实现方式3：Forest

- 其原理正是图论的“有向森林”。



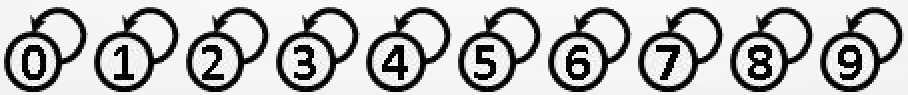
- 让一个数组的第x格代表第x人所属家族的祖先：

0	1	2	3	4	5	6	7	8	9
1	1	8	2	4	5	2	4	8	2

1.4.3.实现方式3：Forest

- 家族的所有成员，他们往上追溯之后，都是同一个祖先。一个家族中，只会有一个祖先。因此，可以用这个祖先来作为家族的代表。
- 一个家族之中，每个人都有一个祖先，那么祖先的祖先是谁呢？可以姑且设定成自己。
- 一个家族就像一棵树。同一个人不会属于两个家族，即两个家族不会公用同一个成员。因此这些树构成了一个不相交集合森林。

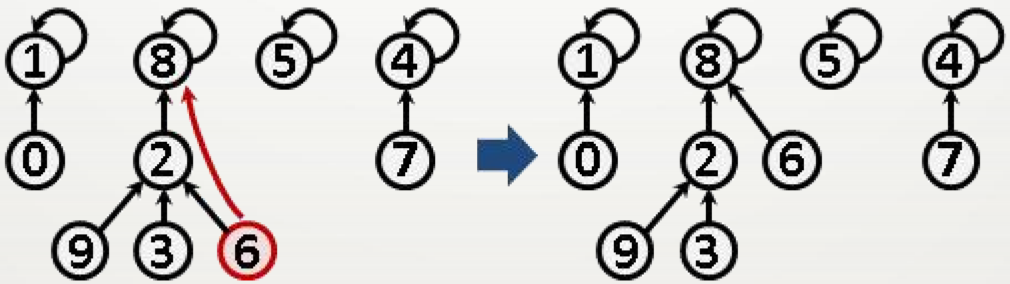
1.4.3.实现方式3：Forest
初始化



- 每个人都属于不同家族，每个人都是自己一个人一个家族，家族的祖先就是自己。将第x格的值设成x，这样每个人都是不同家族的祖先了。

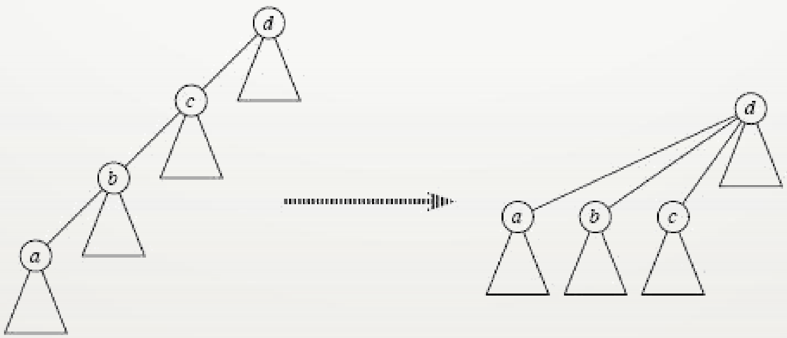
```
void Reset(int n) {
    for(int i=1;i<=n;i++) anc[i]=i;
}
```

1.4.3.实现方式3：Forest
Find: 找出一个人在哪一个家族？



- 一个家族只会有一个祖先，而一个祖先只会属于一个家族，故可以用代表的编号作为家族的编号。

1.4.3.实现方式3：Forest
启发式策略：路径压缩



- find的时候，可以把途中遇到的所有人，将其指向的父亲节点改为祖先。这样，下次find的时候就会变快了。

```
int Find(int x) {
    return x==anc[x]?x:(anc[x]=Find(anc[x]));
}
```

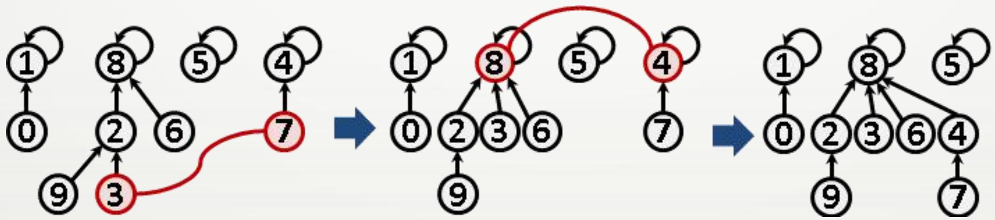
17

1.4.3.实现方式3：Forest IsConnected: 两个人是否同一个家族？

- 同一个家族中的成员，必定有同一个祖先。要看两个人是不是同一个家族，看看他们的祖先是否相同就行了。

```
bool isConnected(int x,int y) {
    return Find(x)==Find(y);
}
```

1.4.3.实现方式3：Forest
Union: 两个人想合并自己所属家族



- 让x家族的祖先带着家族中所有成员，投靠y家族的代表，如此一来两个家族就拥有共同的祖先了，而且可以使树的深度增加较少，下次find的时候就会变快了。

```
void Union(int x,int y) {
    anc[Find(x)]=Find(y);
}
```

- 注：“union”在C++中有特殊含义，必须换个写法。

1.4.3.实现方式3：Forest
启发式策略：按秩合并

- union的时候，让小的家族并入大的家族，可以让树的深度增加最少。这样下次find的时候就会变快了。

```
void Union(int x,int y) {
    int p=Find(x),q=Find(y);
    if(size[p]>size[q]) swap(p,q);
    size[q]+=size[p];
    anc[p]=q;
    groups--;
}
```

1.4.3.实现方式3：Forest
Number of Sets: 总共有几个家族？

- 两个家族合并成一个后，家族数就会减少一个。只需修改一下union的代码。

```
int groups;
void Reset(int n) {
    groups=n;
    for(int i=1;i<=n;i++) anc[i]=i;
}
void Union(int x,int y) {
    anc[Find(x)]=Find(y);
    groups--;
}
```

1.4.3.实现方式3：Forest
Cardinality of a Set: 一个家族总共几个人？

- 建立一个数组size去记录每个家族的人数。
- size[祖先]=家族中的人数。
- 两个家族合并成一个后，家族的规模就会扩大。新祖先吸收旧祖先的人数，旧祖先则不再是祖先，不须修改他家族的人数。只需修改一下union的代码，统计时返回size[祖先]。

```
void Union(int x,int y) {
    int p=Find(x),q=Find(y);
    size[q]+=size[p];
    anc[p]=q;
    groups--;
}
```

```
int Cardinality(int x) {
    return size[Find(x)];
}
```

1.4.3.实现方式3：Forest Singleton Set: 家族是否合并过？

- 自己一个人一个家族，没有合并过。

```
bool Singleton(int x) {
    return size[Find(x)]==1;
}
```

23

1.4.3.实现方式3：Forest 复杂度

- 时间复杂度：
  - union、find、equivalence、cardinality、singleton都是O(logN)。值得一提的是，均摊时间皆是O(α(N))，其中α(N)是阿克曼函数f(N,N)的反函数，故α(N)在N十分巨大时还是小于等于4。因此，平均运行时间是一个极小的常数。
  - 实际上，这是渐近最优算法：Fredman和Saks在1989年解释了Ω(α(N))的平均时间内可以获得任何并查集。
- 空间复杂度：
  - 如果有N个人，就需要一个N格的数组，为O(N)。

1.5.例题：亲戚

- OJ题号：洛谷1551
- 题目大意：
  - 总共有n个人，给出m对亲戚关系，询问p对人是否为亲戚。
- 数据范围：n,m,p≤5,000。

2.进阶操作

2.1.正反集

- 概念：对于同一个元素，维护两个集合，包括元素本身的集合s1与一定不会包括元素本身的集合s2。
- 应用：处理诸如“我朋友的朋友是我的朋友，我敌人的敌人也是我的朋友”的敌友关系。

2.1.1.例题：团伙

- OJ题号：洛谷1892
- 题目大意：
  - 有n个人参加了黑社会。按顺序给出下面的两种信息共m条：
    - 1.a和b属于朋友关系；
    - 2.a和b属于敌对关系。
  - 黑社会有一个不成文的规定：我朋友的朋友是我的朋友，我敌人的敌人也是我的朋友。
  - 两个人属于同一团伙当且仅当他们是朋友。问最多可能有多少团伙？
- 数据范围：$n \le 1{,}000; m \le 5{,}000$。

28

2.1.1.例题：团伙 Solution

- 对于不能确定是否是同一集合的元素，我们认为它们是不同集合的。不难发现这样下来最后的团体数最多，且符合题意。
- 开两个数组，一个表示同集关系，一个记录异集中的一个成员。
- 对于每次操作Union(x,y)：
- 如果是同集，直接合并；
- 如果是异集：
  - 1.如果x的异集为空，则y为x的异集中的成员。
  - 2.如果x的异集不为空，则合并y与x的异集。
- 反过来，对y进行同样的处理。

2.1.1.例题：团伙
Solution

- 核心代码：

```
void Union(int op,int x,int y) {
    if(op==1) {
        if(Find(y)==Find(x)) return;
        anc[Find(y)]=Find(x);
    }
    if(op==2) {
        if(anti[x]) Union(1,anti[x],y);
        if(anti[y]) Union(1,anti[y],x);
        if(!anti[x]) anti[x]=Find(y);
        if(!anti[y]) anti[y]=Find(x);
    }
}
```

2.2.加权并查集

- 概念：每个元素拥有一个权值，保证合并时元素的权值保持不变。
- 应用：用于同时需要维护无向图中点的连通性和边权的问题。

2.2.1.例题：银河英雄传说

- 来源：NOI2002
- OJ题号：洛谷1196
- 题目大意：
  - 有30,000个链，每个链最初有1个点。按顺序给出下面的两种信息共T条：
    - 第一种：将x所在的链接至y所在的链的尾部；
    - 第二种：询问同一个链上点a,b之间的距离。
- 数据范围：$1 \le T \le 500{,}000$。

2.2.1.例题：银河英雄传说
Solution

- 对于每个点，分别记录所属链的头结点、该点到头结点的距离。
- 对于每条链，记录它的尾结点。
- 每次合并将y接在x的尾部，改变y头的权值和所属链的头结点，同时改变x的尾结点。
- 每次查询时计算两点的权值差。

2.2.1.例题：银河英雄传说 Solution

- 核心代码：

```
int Find(int x) {
    if(x==anc[x]) return x;
    int t=Find(anc[x]);
    w[x]+=w[anc[x]];
    return anc[x]=t;
}
```

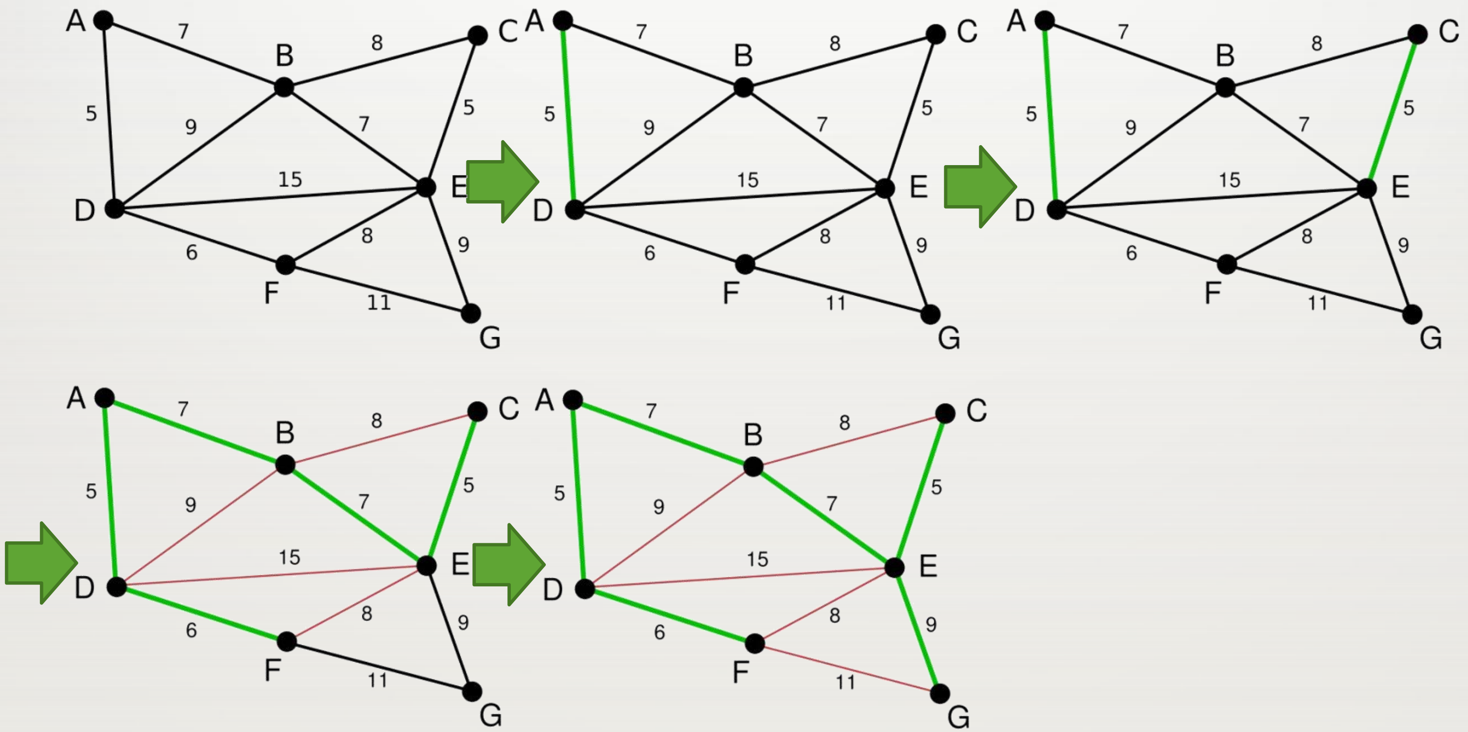
```
void Union(int x,int y) {
    int t=Find(x);
    w[t]=1;
    anc[t]=end[Find(y)];
    end[Find(y)]=end[t];
}
```

```
int Query(int x,int y) {
    return (Find(x)==Find(y))?std::abs(w[x]-w[y])-1:-1;
}
```

2.3.Kruskal

- 应用：求解最小生成树问题。
- 首先将所有的边按照边权从小到大排序。
- 从小到大枚举每一条边，判断这条边所连接的两个点是否已经联通。
- 如果不是，则合并两个联通分量。
- 当所有点互相联通时，所有的边所构成的集合就是最小生成树。

2.3.Kruskal


A B C D E F G
7 8 5 9 7 15 5 6 8 9 11
A B C D E F G
7 8 5 9 7 15 5 6 8 9 11
A B C D E F G
7 8 5 9 7 15 5 6 8 9 11
A B C D E F G
7 8 5 9 7 15 5 6 8 9 11
A B C D E F G
7 8 5 9 7 15 5 6 8 9 11

2.3.Kruskal 复杂度

- 设边数为E，点数为V。
- 对图中所有边排序，时间复杂度为O(ElogE)。
- 用并查集合并两个联通分量，时间复杂度为O(Eα(E,V))。
- 总时间复杂度为O(ElogE)。
- 优化：如果两点之间有多条边，先遍历图中所有边，保留权值最小的边，使得两点之间只剩下一条边。边的总数至多V(V-1)/2条。时间复杂度O(ElogE)可以改写成O(ElogV)。

2.3.Kruskal
与其他MST算法复杂度的比较

- Prim：$O(V^2)$。
- Borůvka ：$O(E\log V)$。
- Edmonds：$O(V^3)$。
- 其中，Borůvka算法和Edmonds算法很少在算法竞赛中使用。而在稀疏图中，Kruskal算法明显优于Prim算法。

2.4.可持久化并查集

- 概念：支持修改、查询历史版本的并查集。

2.4.1.例题：可持久化并查集加强版

- OJ题号：BZOJ3674
- 题目大意：
  - 给定n个集合，按顺序给出下面的三种信息共m条。
    - 第一种：合并a,b所在集合；
    - 第二种：回到第k次操作之后的状态；
    - 第三种：询问a,b是否属于同一集合。
  - 强制在线。
- 数据范围：0<n,m≤200,000。

2.4.1.例题：可持久化并查集加强版 Solution

- 主席树+并查集。
- 用线段树代替数组anc存储每个元素所属集合的代表。每次合并时相当于单点修改。
- 也可以同时用线段树记录size，然后按秩合并。实际上只要前面常数够小，这并不是必须的。
- 当然用可持久化平衡树代替数组也可以。

3.习题选讲

3.1.星球大战

- 来源：JSOI2008
- OJ题号：洛谷1197、BZOJ1015
- 题目大意：
  - 一个有n个点，m条边的无向连通图。有k次操作，每次删去编号为x的点和与其相连的所有边。求每次操作后剩下的联通块个数。
- 数据范围：$1 \le n \le 400,000; 1 \le m \le 200,000$。

3.1.星球大战 Solution

- 将已经连接的点拆开并不容易，因此可以考虑一种离线做法。
- 由题意得总共有n个结点，其中k个结点最终被删除，因此最终剩下的结点为n-k。
- 首先对这n-k个结点进行合并操作。然后将删除的点逆序加入图中，同时进行合并操作。每进行一次合法的合并操作就意味着减少一个联通块，同时要注意加入结点时，该结点本身也算一个新的联通块。
- 最后按原顺序输出每次的联通块个数。

3.2.关押罪犯

- 来源：NOIp2010提高组
- OJ题号：洛谷1525
- 题目大意：
  - 有2座监狱，n名犯人。m对犯人之间有“怨气值”c，表示如果该两名犯人被关押在同一监狱，会产生严重程度为c的恶性事件。求合理安排犯人后最大的c最小值。
- 数据范围：n≤20,000;m≤100,000。

3.2.关押罪犯

- 贪心+正反集。
- 要使最大值最小，则应当尽量避免怨气值大的人在同一监狱。
- 将数据按怨气值大小排序。
- 优先考虑怨气值大的人，如果与现有情况不矛盾，则说明他们是反集，如果矛盾，则说明冲突不可避免地会出现，且是最大的冲突，输出答案即可。
- 当然也可以用二分答案二分图染色做。

3.3.Piggy Banks

- 来源：POI2005
- OJ题号：洛谷3420、BZOJ1529
- 题目大意：
  - 有n个储钱罐，每个储钱罐的钥匙都在其中一个储钱罐中，每个储钱罐对应的钥匙所在的储钱罐，问取出所有的钥匙至少要拿出几个储钱罐。
- 数据范围：$1 \le n \le 1{,}000{,}000$。

3.3.Piggy Banks Solution

- 把储钱罐抽象成点，把钥匙抽象成边。
- 每个钥匙相当于在其所在的储钱罐和能够打开的储钱罐之间连的边。
- 并查集求联通块个数即可。
- 当然也可以用Tarjan做，统计入度为0的连通分量即可。

3.4.狡猾的商人

- 来源：HNOI2005
- OJ题号：洛谷2294、BZOJ1202
- 题目大意：
  - 给定m条信息，描述一个长度为n的数组，每条信息描述从a[x]到a[y]的和为v，试判断这些信息是否互相矛盾。
- 数据范围：n<100;m<1,000。

3.4.狡猾的商人
Solution

- 加权并查集。
- 维护一个数组w，对于每个联通块，用w[i]表示从anc[i]到i的和。
- 每次将给出的区间[x-1,y]对应的v与w[y]-w[x]比较。
- 如果与已知条件冲突则说明条件互相矛盾。
- 如果条件未知，则将其加入并查集中。
- 当然也可以用差分约束系统。

3.5.The Door Problem

- OJ题号：CF776D
- 题目大意：
  - 有m个开关。每个开关控制了一些灯，而每个灯恰好被2个开关控制，当且仅当这2个开关中恰好有一个打开时灯才会亮。现在给定n个灯的亮灭情况，求是否可以控制开关达到这种情况。
- 数据范围：2≤n,m≤100,000。

3.5.The Door Problem Solution

- 并查集维护每个开关的状态on[i]和off[i] 。
- 假设灯L由开关S1和S2控制。
- 如果开关是亮的，则S1和S2的状态相反；
- 如果开关是灭的，则S1和S2的状态相同。
- 当一个开关状态已知时，可以得知另一个开关的状态，合并。
- 如果on[i]和off[i]在同一个集合就无解。
- 时间复杂度：$O((n+m)\alpha(n))$。

3.6.find the most comfortable road

- OJ题号：HDU1598
- 题目大意：
  - 给定一个n个点，m条带权边的无向图，每条边分别有一个边权w。有q次询问，每次求一条从S到T的路径，使得路径上最大边权与最小边权的差最小。
- 数据范围：$1<n\le200;m\le1,000;w\le1,000,000;q\le10$。

3.6.find the most comfortable road Solution

- Kruskal。
- 考虑一个暴力：枚举最大的边权和最小的边权，然后将边权在这之间的边全拿出来构成一张无向图，剩下的就是判断是否存在一条从S到T的路径。
- 相当于判S和T是否连通，用并查集连一下即可。
- 时间复杂度：$O(m^3\alpha(n))$
- 考虑优化：枚举了最小边权之后，从小到大枚举最大边权，每次能新添加一条边。
- 因为并查集是支持动态加边的，所以复杂度就降到$O(m^2\alpha(n))$了。

3.7.Parity game

- 来源：CEOI1999
- OJ题号：POJ1733
- 题目大意：
    - 对于一个长度为L的未知的01串，提供N条信息。每条信息描述该串中从S到T的1的个数的奇偶性。求最后一个不与前面信息产生矛盾的信息是第几个信息？
- 数据范围：$L \le 10^9; N \le 5{,}000$。

3.7.Parity game Solution

- 加权并查集。
- 每条信息就相当于从S-1到T连了一条边。
- 把奇偶性作为边权。
- 每次加入条件时只要判断是否与之前条件矛盾。
- 路径压缩时就相当于对两条边取异或。
- $L\leq 10^9$?
- 离散化即可。

3.8.奶酪

- 来源：NOIp2017提高组
- 题目大意：
  - 一个实心的三维空间中有n个半径相等的球形空洞，告诉你这些球形空洞的坐标，问如果路线中只经过这些空洞，能否从平面z=0走到平面z=h？
- 数据范围：n≤1,000。

3.8.奶酪
Solution

- 首先想办法将这道题转化为一道图论题。
- 用$O(n^2)$的时间枚举每一对点，判断是否相交，如果相交就用并查集合并这个点对。
- 将两个平面分别抽象成点，用$O(n)$的时间枚举每个点，判断是否和平面$z=0$和平面$z=h$有相交，如果是，则合并。
- 最后查询代表两个平面的点是否在同一个集合中即可。
- 时间复杂度$O(n^2\alpha(n))$。
- 如果判断相交的时候直接给点对连边，然后用$O(n)$的BFS直接判断两个平面的连通性，可以做到$O(n^2)$。

3.9.食物链

- OJ题号：POJ1182
- 题目大意：
  - 有N只编号为1~N的动物。所有动物都属于A,B,C,中的其中一种。已知A吃B、B吃C、C吃A。按顺序给出下面的两种信息共K条。
    - 第一种：x和y属于同一种类。
    - 第二种：x吃y。
  - 有可能有的信息和之前给出的信息矛盾，也有的信息可能给出的x和y不在1~N的范围内。求共有多少条是不正确的。计算过程中，我们将忽视诸如此类的错误信息。
- 数据范围：1≤N≤50,000;0≤K≤100,000。

3.9.食物链 Solution

- 对于每只动物i创建3个元素i-A,i-B,i-C，并用这些元素建立并查集，维护如下信息：
  - i-x表示“i属于种类x”。
  - 并查集中每一个组表示组内元素代表的情况都同时发生或不发生。
- 因此，我们可以对每一条信息进行如下操作：
  - 第一种：x和y是同一种。合并x-A和y-A、x-B和y-B、x-C和y-C。
  - 第二种：x吃y。合并x-A和y-B、x-B和y-C、x-C和y-A。
- 每次合并前判断是否与已知条件矛盾，统计个数即可。