北京大学暑期课《ACM/ICPC竞赛训练》

北京大学信息学院 郭炜
guo_wei@PKU.EDU.CN
http://weibo.com/guoweiofpku

课程网页：
http://acm.pku.edu.cn/summerschool/pku_acm_train.htm

Disjoint-Set

并查集

北京大学信息学院

Zhou Hang/ 郭炜

July 15, 2014

Disjoint-Set(Union-Find Set) 并查集

- N 个不同的元素分布在若干个互不相交集合中，需要进行以下3个操作：
- 1. 合并两个集合
- 2. 查询一个元素在哪个集合
- 3. 查询两个元素是否属于同一集合

并查集操作示例

Operation	Disjoint sets					
初始状态	{a}	{b}	{c}	{d}	{e}	{f}
Merge(a,b)	{a,b}		{c}	{d}	{e}	{f}
Query(a,c)	False					
Query(a,b)	True					
Merge(b,e)	{a,b,e}		{c}	{d}		{f}
Merge(c,f)	{a,b,e}		{c,f}	{d}		
Query(a,e)	True					
Query(c,b)	False					
Merge(b,f)	{a,b,c,e,f}			{d}		
Query(a,e)	True					
Query(d,e)	False					

土算法

- 给集合编号

Op \ *Element*	{a}	{b}	{c}	{d}	{e}	{f}
	1	2	3	4	5	6
Merge(a,b)	1	1	3	4	5	6
Merge(b,e)	1	1	3	4	1	6
Merge(c,f)	1	1	3	4	1	3
Merge(b,f)	1	1	1	4	1	1

Query(a,e)

土算法

- 给集合编号

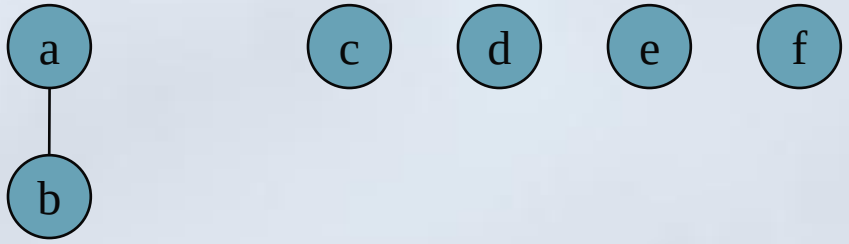
Op \ *Element*	{a}	{b}	{c}	{d}	{e}	{f}
	1	2	3	4	5	6
Merge(a,b)	1	1	3	4	5	6
Merge(b,e)	1	1	3	4	1	6
Merge(c,f)	1	1	3	4	1	3
Merge(b,f)	1	1	1	4	1	1

- Query – O(1); Merge – O(N)

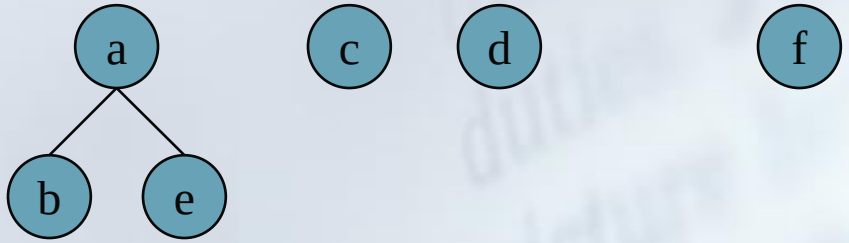
用树结构表示集合

- Init:

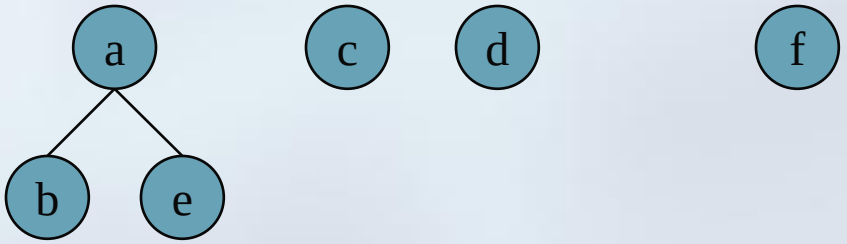
- *Merge(a,b)*



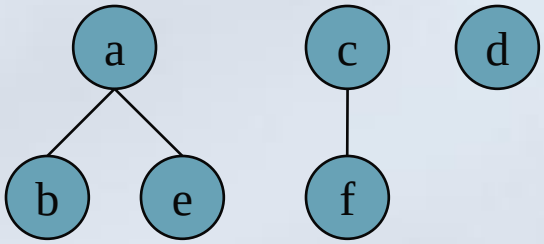
- *Merge(b,e)*



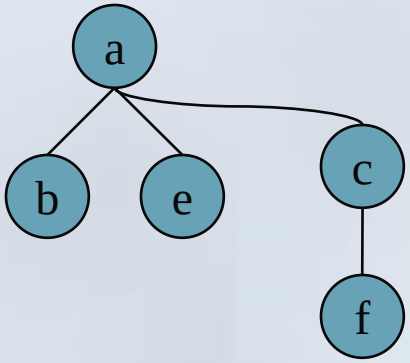
用树结构表示集合



- *Merge(c,f)*

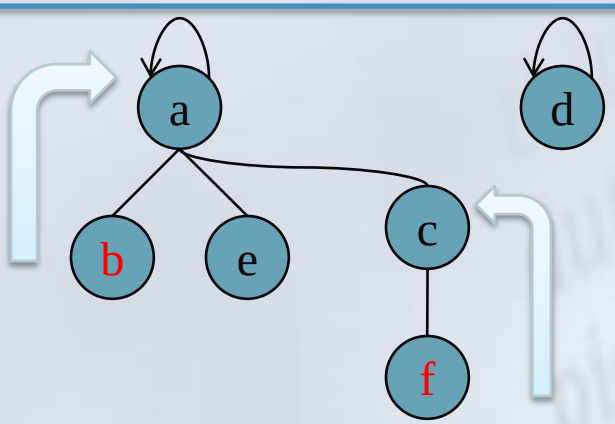


- *Merge(b,f)*



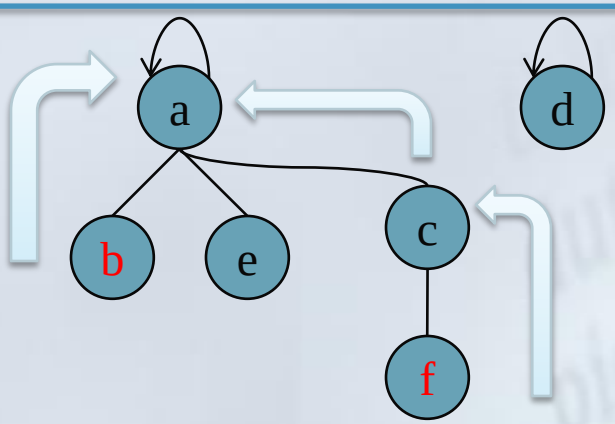
用树结构表示集合

- *Merge(b,f)*
  - 将f所在树挂为b所在树的直接子树
  - 开设父节点指示数组Par,Par[i]代表第i个元素的父亲。若元素i是树根，则Par[i] = i



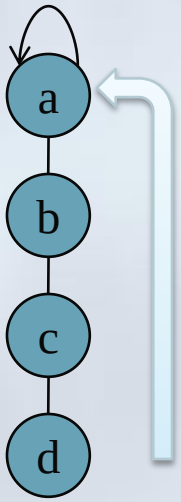
用树结构表示集合

- *Query(b,f)*
  - 简单比较b和f所在树的根节点是否相同



用树结构表示集合

- *缺点:树可能层次太深，以至于查树根太慢*
- *Merge(d,c), Merge(c,b), Merge(b,a) …*



O(N)!

改进方法一 – 根据树的层次进行合并

- 每个节点（元素）维护一个*Rank*表示子树最大可能高度
- 较小*Rank*的树连到较大*Rank*树的根部。

改进方法一 – 根据树的层次进行合并

New One

- LINK(x, y)
  - If Rank[x]>Rank[y]
    - par[y] ← x
  - Else
    - Par[x] ← y
    - If Rank[x]=Rank[y]
      - Rank[y]++

Old One

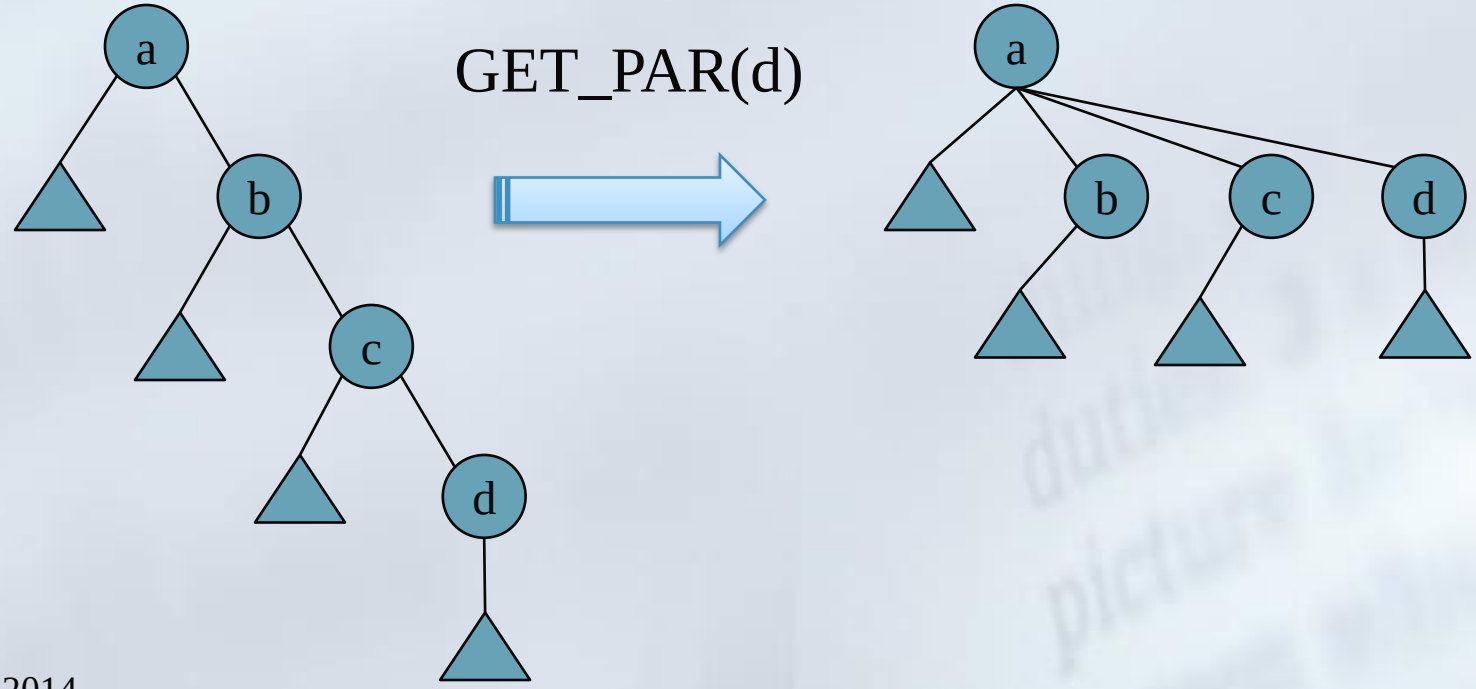
- LINK(x, y)
  - par[y] ← x

改进方法一 – 根据树的层次进行合并

- GET_PAR(a) //求a的根节点
  - If Par[a]=a
    - Return a
  - Else
    - Return GET_PAR(par[a])
- Query(a,b) – $O(\log_2 N)$
  - Return GET_PAR(a)==GET_PAR(b)
- Merge(a,b) – $O(\log_2 N)$
  - LINK(GET_PAR(a), GET_PAR(b))

改进方法2 – 路径压缩

- 将GET_PAR中查找路径上的节点直接指向根



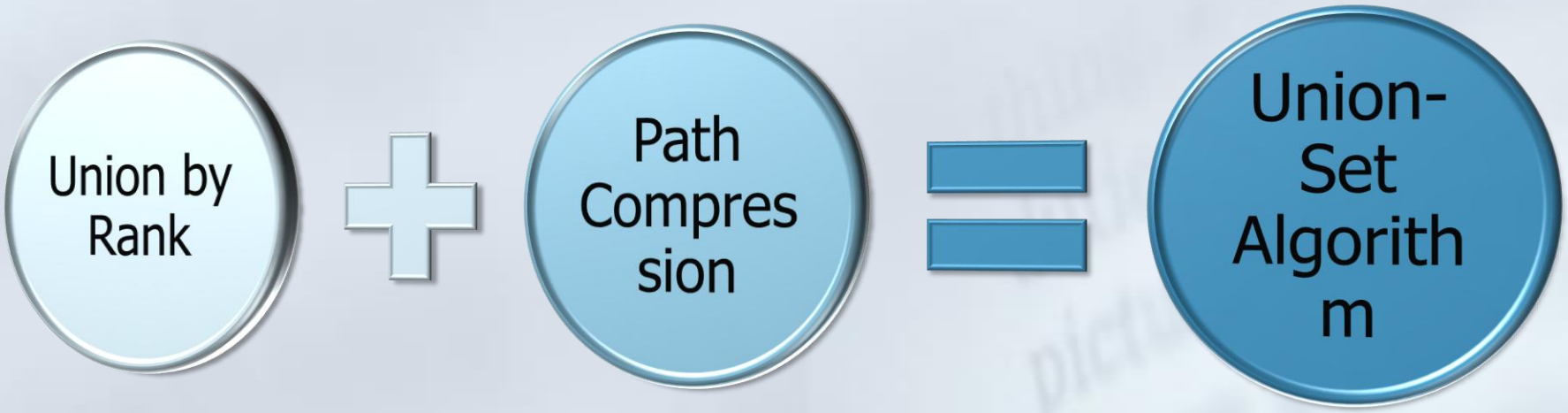
改进方法2 – 路径压缩

New Code

- GET_PAR(a)
  - If Par[a]!=a
    - Par[a] =GET_PAR(par[a])
  - Return par[a]

Old Code

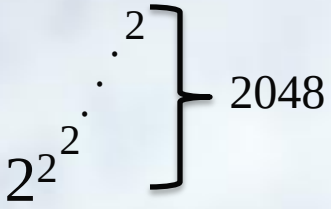
- GET_PAR(a)
  - If Par[a]=a
    - Return a
  - Else
    - Return GET_PAR(par[a])



复杂度

Amortized cost of GET_PAR operation O(α(n))

GET_PAR函数的平摊复杂度为O(α(n))

- α(n) =0, if 0<=n<=2
- =1, if n=3
- =2, if 4<=n<=7
- =3, if 8<=n<=2047
- =4, if 2048<=n<=$A_4(1)\approx$ 

- 实际上可以说GET_PAR的复杂度<= 4

实际应用

实际应用: 路径压缩足矣，不要什么rank了

实际应用

```
int get_par(int u) {
  if (par[a]!=a)
    par[a] = get_par(par[a]);
  return par[a];
}
```

int par[];

```
int query(int a,int b) {
  return get_par(a)==get_par(b);
}
```

```
void merge(int a,int b) {
  par[get_par(a)] = get_par(b);
}
```

POJ1611 The Suspects

- n个学生分属m个团体，(0 < n <= 30000 , 0 <= m <= 500) 一个学生可以属于多个团体。一个学生疑似患病，则它所属的整个团体都疑似患病。已知0号学生疑似患病，以及每个团体都由哪些学生构成，求一共多少个学生疑似患病。

POJ1611 The Suspects

- n个学生分属m个团体，(0 < n <= 30000 , 0 <= m <= 500) 一个学生可以属于多个团体。一个学生疑似患病，则它所属的整个团体都疑似患病。已知0号学生疑似患病，以及每个团体都由哪些学生构成，求一共多少个学生疑似患病。

- 解法：最基础的并查集,把所有可疑的都并一块。

POJ1611 The Suspects

- Sample Input
- 100 4
- 2 1 2
- 5 10 13 11 12 14
- 2 0 1
- 2 99 2
- 200 2
- 1 5
- 5 1 2 3 4 5
- 1 0
- 0 0

- Sample Output
- 4
- 1
- 1

POJ1611 The Suspects

```
#include <iostream>
using namespace std;
const int MAX = 30000;
int n,m,k;
int parent[MAX+10];
int total[MAX+10];
//total[GetParent(a)]是a所在的group的人数
int GetParent(int a)
{//获取a的根,并把a的父节点改为根
        if( parent[a]!= a)
                parent[a] = GetParent(parent[a]);
        return parent[a];
}
```

```
void Merge(int a,int b)
{
        int p1 = GetParent(a);
        int p2 = GetParent(b);
        if( p1 == p2 )
                return;
        total[p1] += total[p2];
        parent[p2] = p1;
}
```

POJ1611 The Suspects

```
int main()  {
        while(true) {
                scanf("%d%d",&n,&m);
                if( n == 0 && m == 0)      break;
                for(int i = 0;i < n; ++i) {
                        parent[i] = i;       total[i] = 1;
                }
                for(int i = 0;i < m; ++i) {    int h,s;
                        scanf("%d",&k);   scanf("%d",&h);
                        for( int j = 1; j < k; ++j) {
                                scanf("%d",&s);
                                Merge(h,s);
                        }
                }
                printf("%d\n",total[GetParent(0)]); //此处写 parent[0]可否？
        }  return 0;
}
```

POJ1611 The Suspects

```
int main()  {
        while(true) {
                scanf("%d%d",&n,&m);
                if( n == 0 && m == 0)      break;
                for(int i = 0;i < n; ++i) {
                        parent[i] = i;       total[i] = 1;
                }
                for(int i = 0;i < m; ++i) {    int h,s;
                        scanf("%d",&k);   scanf("%d",&h);
                        for( int j = 1; j < k; ++j) {
                                scanf("%d",&s);
                                Merge(h,s);
                        }
                }
                printf("%d\n",total[GetParent(0)]); //此处写 parent[0]可否？
        }  return 0;
}
```

不行，因为可能还没进行过路径压缩，parent[0]的值不正确

POJ 1988 Cube stacking

- 有N(N<=30,000)堆方块，开始每堆都是一个方块。方块编号1 – N. 有两种操作：
- M x y： 表示把方块x所在的堆，拿起来叠放到y所在的堆上。
- C x：问方块x下面有多少个方块。
- 操作最多有 P (P<=100,000)次。对每次C操作，输出结果。

POJ 1988 Cube stacking

解法：

除了parent数组，还要开设

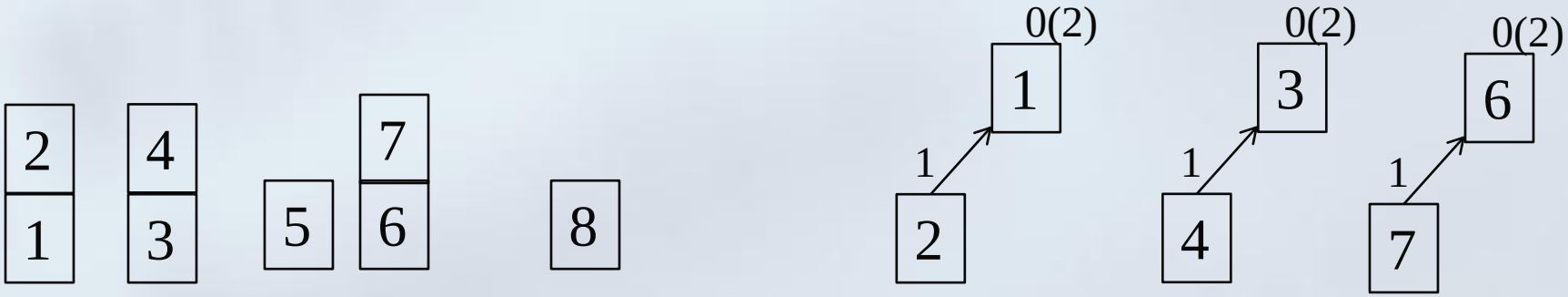
sum数组：记录每堆一共有多少方块。

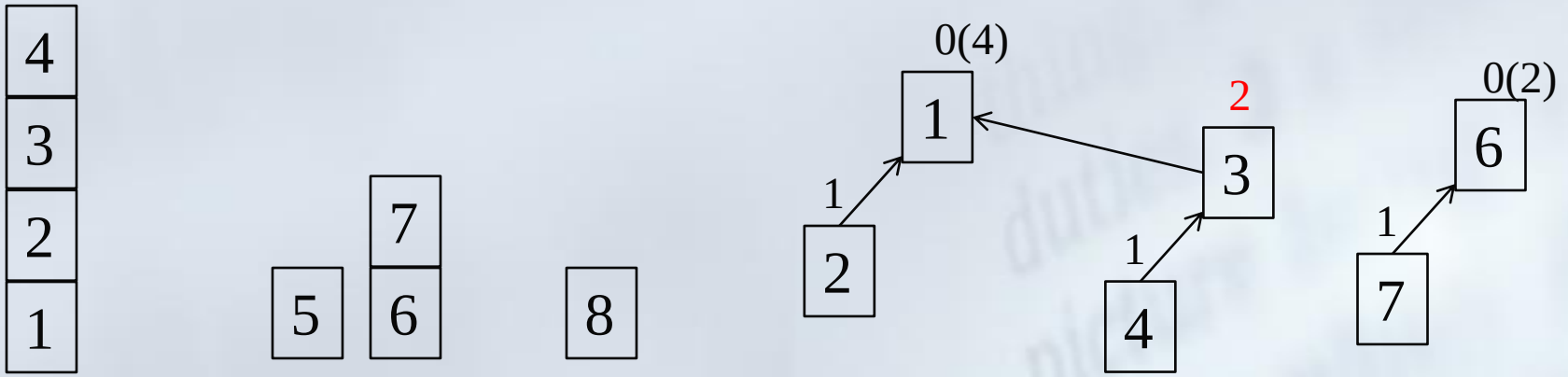
若parent[a] = a, 则sum[a]表示a所在的堆的方块数目。

under数组，under[i]表示第i个方块下面有多少个方块。

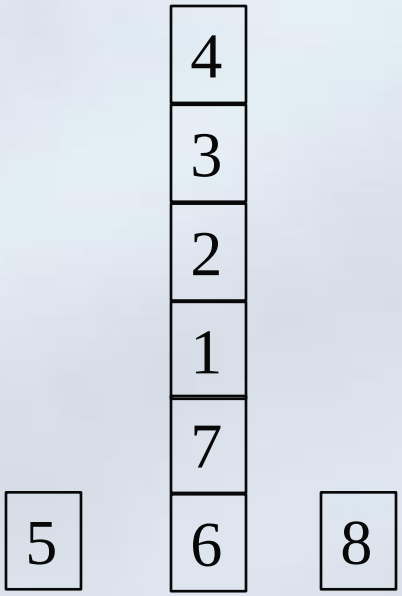
under数组在 堆合并和 路径压缩的时候都要更新。

POJ 1988 Cube stacking

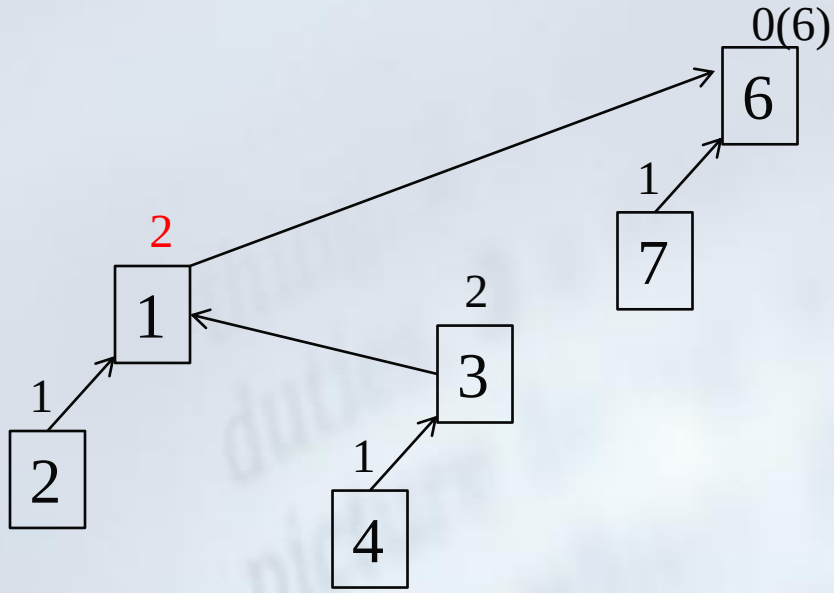




POJ 1988 Cube stacking

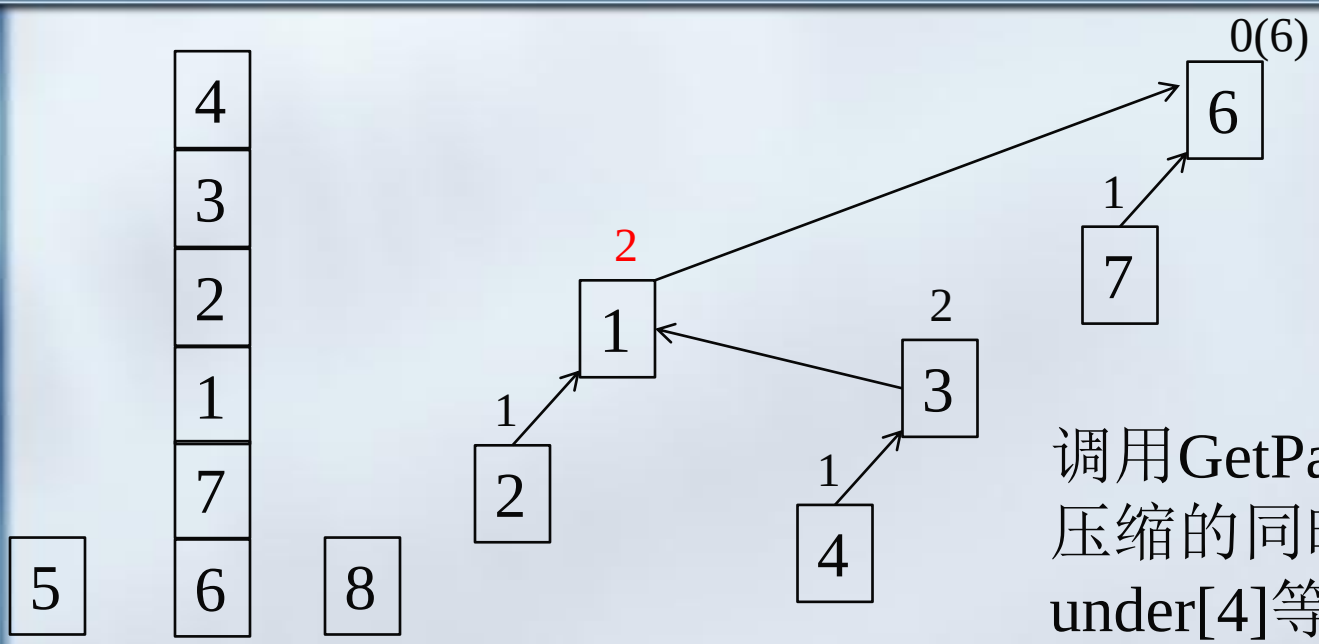


M 3 7



求： C 4

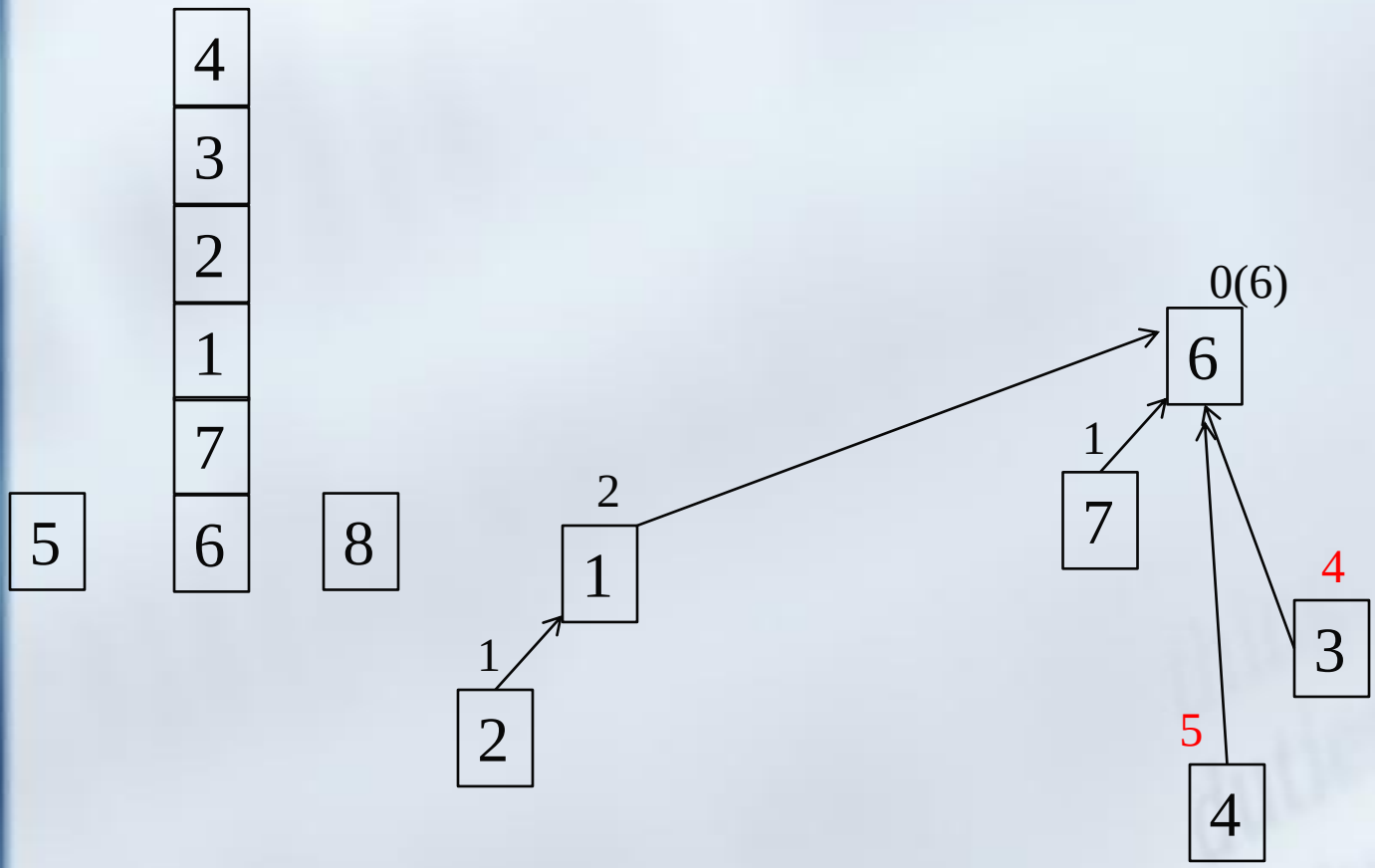
POJ 1988 Cube stacking



求： C 4

调用GetParent(4)进行路径压缩的同时，更新under[4]
under[4]等于从4到根路径上所有节点的under值之和,路径压缩的同时，该路径上所有节点的under值也都更新

POJ 1988 Cube stacking



POJ 1988 Cube stacking

```
#include <iostream>
#include <cstdio>
using namespace std;
const int MAX = 31000;
int parent[MAX];
int sum[MAX]; // 若parent[i]=i,sum[i]表示砖块i所在堆的砖块数目
int under[MAX]; // under[i]表示砖块i下面有多少砖块
int GetParent(int a)
{//获取a的根,并把a的父节点改为根
        if( parent[a] == a)
                return a;
        int t = GetParent(parent[a]);
        under[a] += under[parent[a]];
        parent[a] = t;
        return parent[a];
}
```

POJ 1988 Cube stacking

```
void Merge(int a,int b)
//把b所在的堆，叠放到a所在的堆。
{       int n;
        int pa = GetParent(a);
        int pb = GetParent(b);
        if( pa == pb )
                return ;
        parent[pb] = pa;
        under[pb] = sum[pa]; //under[pb] 赋值前一定是0，因为
parent[pb] = pb,pb一定是原b所在堆最底下的
        sum[pa] += sum[pb];
}
```

POJ 1988 Cube stacking

```
int main()
{
        int p;
        for(int i = 0; i < MAX; ++ i) {
                sum[i] = 1;
                under[i] = 0;
                parent[i] = i;
        }
        scanf("%d",&p);
```

POJ 1988 Cube stacking

```
        for( int i = 0;i < p; ++ i) {
                char s[20];         int a,b;
                scanf("%s",s);
                if( s[0] == 'M') {
                        scanf("%d%d",&a,&b);
                        Merge(b,a);
                }
                else {
                        scanf("%d",&a);
                        GetParent(a);
                        printf("%d\n",under[a]);
                }
        }
        return 0;
}
```

POJ 1182 食物链

- 三类动物A、B、C，A吃B，B吃C，C吃A。
- 给出K句话来描述N个动物（各属于A、B、C三类之一）之间的关系，格式及意义如下：
  - 1 X Y：表示X与Y是同类；
  - 2 X Y：表示X吃Y。
- K句话中有真话有假话，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。
  1） 当前的话与前面的某些真的话冲突，就是假话；
  2） 当前的话中X或Y比N大，就是假话；
  3） 当前的话表示X吃X，就是假话。
- 求假话的总数。

POJ 1182 食物链

- 输入：
  - 第一行是两个整数N和K，以一个空格分隔。以下K行每行是三个正整数 D，X，Y，两数之间用一个空格隔开，其中D表示说法的种类。
    若D=1，则表示X和Y是同类。
    若D=2，则表示X吃Y。
- 输出：
  - 只有一个整数，表示假话的数目。
- 约束条件：
  - 1 <= N <= 50000，0 <= K <= 100000。

POJ 1182 食物链

- 一个容易想到的思路：
  - 用二维数组s存放已知关系：
    - S[X][Y] = -1：表示X与Y关系未知；
    - S[X][Y] = 0：表示X与Y是同类；
    - S[X][Y] = 1：表示X吃Y；
    - S[X][Y] = 2：表示Y吃X。
  - 对每个读入的关系s(x,y)，检查S[x][y]：
    - 若S[x][y]=s，则继续处理下一条；
    - 若S[x][y] = -1，则令S[x][y]=s，并更新S[x][i]、S[i][x]、S[y][i]和S[i][y] (0<i<=n)。
    - 若S[x][y] != s且S[x][y] != -1，计数器加1。

- 复杂度：
  - 以上算法需要存储一个N×N的数组，空间复杂度为$O(N^2)$。
  - 对每一条语句
    - 进行关系判定时间为O(1)
    - 加入关系时间为O(N)
  - 总的时间复杂度为O(N*K)
- 0<=N<=50000，0<=K<=100000，复杂度太高。

进一步分析

- 对于任意$a \neq b$，a、b属于题中N个动物的集合S，当且仅当S中存在一个有限序列$(P_1, P_2, \ldots, P_m)$（$m \geq 0$）使得aP_1、P_1P_2、…、$P_{m-1}P_m$、P_mb（或$m=0$时的ab）之间的相对关系均已确定时，b对a的相对关系才可以确定。
- 由上面可知，我们不需要保留每对个体之间的关系，只需要为每对已知关系的个体保留一条路径a$P_1P_2\ldots P_mb$（$m \geq 0$）其中aP_1、P_1P_2、…、$P_{m-1}P_m$、P_mb之间的关系均为已知。两两关系已知的动物们，构成一个group

解决方案

- *使用并查集*

解决方案

- 用结点表示每个动物，边表示动物之间的关系。采用父结点表示法，在每个结点中存储该结点与父结点之间的关系。
- parent数组： parent[i]表示i的父节点
- relation数组：relation[i]表示i和父节点的关系

解决方案

- 初始状态下，每个结点单独构成一棵树。
- 读入a,b关系描述时的逻辑判断：
  - 分别找到两个结点a、b所在树的根结点ra、rb，并在此过程中计算a与ra、b与rb之间的相对关系。
    - 若ra!=rb，此句为真话，将a、b之间的关系加入；
    - 若ra=rb，则可计算出r(a,b)=f(r(a,ra) , r(b,rb))
      - 若读入的关系与r(a,b)矛盾，则此句为假话，计数器加1；
      - 若读入的关系与r(a,b)一致，则此句为真话。

Exercise 3 A Bug's Life

- POJ 2492 A Bug's Life
- 法一:深度优先遍历

  每次遍历记录下该点是男还是女，

  只有: 男->女，女->男 满足，

  否则，找到同性恋二分图匹配，结束程序
- 法二:并查集

Other Exercises

POJ 2524　　最基础的并查集

POJ 1182　　并查集的拓展

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。也就是说:只有三个group

POJ 1861　并查集+自定义排序+贪心求"最小生成树"

POJ 1703　并查集的拓展

Other Exercises

POJ 2236 并查集的应用

需要注意的地方：1、并查集；2、N的范围，可以等于1001；3、从N+1行开始，第一个输入的可以是字符串。

- POJ 2560 最小生成树
- 法一：Prim算法；法二：并查集实现Kruskar算法求最小生成树
- POJ 1456 带限制的作业排序问题（贪心+并查集）

Thank you!