[并查集小结 （参考birdfly+修改）](http://www.cnblogs.com/DreamUp/archive/2010/07/19/1780916.html)

    并查集的作用：并和查，即合并和查找，将一些集合合并，快速查找或判断某两个集合的关系，或某元素与集合的关系，或某两个元素的关系。

    并查集的结构：并查集主要操作对象是森林，树的结构赋予它独特的能力，对整个集合操作转换为对根节点(或称该集合的代表元素)的操作，一个集合里的元素关系不一定确定，但相对于根节点的关系很明了，这也是为了查找方便。

    并查集优化方法：按秩合并和路径压缩的配合使用，使得查找过程优化到极致。按秩合并，每次将深度小的树合并到深度大的树里面去，使得整棵树尽量矮；路径压缩，将当前节点到根节点路径上的所有点直接连到根节点上（有递归和非递归两种方法），使得每个点到根节点的距离更短，在下一次查找的时候更快。

如何快速确定偏移量公式：

例：现在要合并节点x,y, 找到根节点fx = Find(x); fy = Find(y);一般情况下，根节点的偏移量都保持为0, offset[foot] = 0;如果要使得x和y的偏移量为t，假设fx指向fy，则可以写出公式offset[x] + offset[fx] - offset[y] = t，则offset[fx] = (offset[y] + t - offset[x]) % n; 这个n即为总共有多少类，如：在poj1182 食物链中n = 3,，在poj2492 A Bug's Life中n = 2, 这样fx的偏移量就计算出来了，只需要改其中一个根节点的偏移量，这里是fx，因为假设是fx指向fy。

[Birdfly并查集小结](http://www.cnblogs.com/ylfdrib/archive/2010/07/16/1778919.html)

[孟孟起并查集转帖](http://apps.hi.baidu.com/share/detail/7943108)

[Slyar（并查集讲解）](http://www.slyar.com/blog/tag/%E5%B9%B6%E6%9F%A5%E9%9B%86)

[并查集学习小节(POJ版)）](http://www.cppblog.com/abilitytao/archive/2010/05/14/98899.html)

对应题目和代码

[PKU1611The Suspects](http://124.205.79.250/JudgeOnline/problem?id=1611)

**http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码**

**#include<stdio.h>**  
**#define N 3005**  
**int bin[N],cnt[N];  
int find(int x)  
{  
 int r=x;  
 while(bin[r]!=r)  
 r=bin[r];  
 int y=x;  
 while(bin[y]!=y){  
 y=bin[y];  
 bin[y]=r;  
 }  
 return r;  
}  
  
void merge(int x,int y)  
{  
 int fx=find(x);  
 int fy=find(y);  
 if(fx!=fy){  
 if(fx==0){  
 bin[fy]=fx;  
 cnt[fx]+=cnt[fy];  
 }  
 else if(fy==0){  
 bin[fx]=fy;  
 cnt[fy]+=cnt[fx];  
 }  
 else{  
 bin[fx]=fy;  
 cnt[fy]+=cnt[fx];  
 }  
 }  
}  
  
int main()  
{  
 int n,m,num,cur,next,i;  
 while(scanf("%d%d",&n,&m),n+m){  
 for(i=0;i<n;i++)  
 bin[i]=i, cnt[i]=1;  
 while(m--){  
 scanf("%d%d",&num,&cur);  
 while(--num){  
 scanf("%d",&next);  
 merge(cur,next);  
 cur=next;  
 }  
 }  
 printf("%d\n",cnt[0]);  
 }  
 return 0;  
}**

[**PKU2492 A Bug's Life**](http://124.205.79.250/JudgeOnline/problem?id=2492)

分析：很经典的一道并查集，这题关键在如何维护每一个点到集合顶点的偏移量

。第一次了解偏移量，还是挺有收获的！

对于两个点x，y，分别找到他们的根节点fx，fy。（fx = find(x); fy = find(y);）

如果fx == fy 则找到一对同性恋者，return，

否者：bin[fx] = fy; fx指向fy，为了保证x和y为异性，即相对根节点的偏移量不同，

fx的偏移量offset[fx] = (offset[x] + offset[y] + 1)%2; 是同或关系，

这样只对根节点操作，就不影响其它各点间的相互关系。

还有：在路径压缩的过程中，采用递归形式，由根节点反向改变每个叶子节点的值，

这样只用比较当前节点和他父节点之间的关系了：offset[r] = (offset[r] + offset[bin[r]]) % 2;是异或关系。

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include<stdio.h>  
#include<string.h>  
int bin[2002],offset[2002],ok;  
int find(int x)  
{  
 int r=x;  
 if(r==bin[r])  
 return r;  
 int t=find(bin[r]);  
 offset[r]=(offset[r]+offset[bin[r]])%2;  
 bin[r]=t;  
 return t;  
}  
void merge(int x,int y)  
{  
 int fx=find(x);  
 int fy=find(y);  
 if(fx==fy && offset[x]==offset[y])  
 ok=1;  
 else  
 {  
 bin[fx]=fy;  
 offset[fx]=(offset[x]+offset[y]+1)%2;  
 }  
}  
  
int main()  
{  
 int i,cas,ca,n,m,x,y;  
 scanf("%d",&cas);  
 for(ca=1;ca<=cas;ca++)  
 {  
 scanf("%d%d",&n,&m);  
 for(i=0;i<n;i++)  
 bin[i]=i, offset[i]=0;  
 ok=0;  
 while(m--)  
 {  
 scanf("%d%d",&x,&y);  
 if(ok==1)  
 continue;  
 merge(x,y);  
 }  
 printf("Scenario #%d:\n",ca);  
 if(ok==1)  
 puts("Suspicious bugs found!");  
 else  
 puts("No suspicious bugs found!");  
 puts("");   
 }  
 return 0;  
}

[HDU1232 畅通工程](http://acm.hdu.edu.cn/showproblem.php?pid=1232)

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include<stdio.h>  
int a[1002];  
int find(int x)  
{  
 int r=x;  
 while(a[r]!=r)  
 r=a[r];  
 return r;  
}  
void merge(int x,int y)  
{  
 int fx=find(x);  
 int fy=find(y);  
 if(fx!=fy)  
 a[fx]=fy;  
}  
int main()  
{  
 int n,m,i,x,y,ans;  
 while(scanf("%d",&n),n)  
 {  
 for(i=1;i<=n;i++)  
 a[i]=i;  
 scanf("%d",&m);  
 while(m--)  
 {  
 scanf("%d%d",&x,&y);  
 merge(x,y);  
 }  
 for(ans=-1,i=1;i<=n;i++)  
 {  
 if(a[i]==i)  
 ans++;  
 }  
 printf("%d\n",ans);  
 }  
 return 0;  
}

[HDU1198 Farm Irrigation](http://acm.hdu.edu.cn/showproblem.php?pid=1198)

广搜的方法  15MS

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include<queue>  
#include<iostream>  
using namespace std;  
char map[200][200];  
char str[52][52];  
typedef struct{  
 int x, y;  
}POINT;  
POINT p, next;  
int n,m,dir[8][2] = {{-1, -1}, {-1, 0}, {-1, 1}, {0, -1}, {0, 1}, {1, -1}, {1, 0}, {1, 1}};  
queue <POINT> q;  
void bfs(int i, int j)  
{   
 int k;  
 p.x = i; p.y = j;  
 q.push(p);  
 while (!q.empty())  
 {  
 p = q.front();  
 q.pop();  
 for (k = 0; k < 8; k++) {  
 next.x = p.x + dir[k][0];  
 next.y = p.y + dir[k][1];  
 if (next.x >= 0 && next.x <= 3\*m && next.y >= 0 && next.y <= 3\*n)  
 if (map[next.x][next.y] == 'W') {  
 map[next.x][next.y] = '.';  
 q.push(next);  
 }  
 }  
 }  
}  
  
int main()  
{  
 int i,j,con;  
 while(scanf("%d%d",&m,&n),m>=0&&n>=0)  
 {  
 getchar();  
 for(i=0;i<=3\*m;i++)  
 {  
 for(j=0;j<=3\*n;j++)  
 map[i][j]='.';  
 }  
 for(i=0;i<m;i++)  
 {  
 gets(str[i]);  
 for(j=0;j<n;j++)  
 {  
 if(str[i][j]=='A'){map[3\*i][3\*j+1]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+1][3\*j]='W';}  
 else if(str[i][j]=='B'){map[3\*i][3\*j+1]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+1][3\*j+2]='W';}  
 else if(str[i][j]=='C'){map[3\*i+1][3\*j]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+2][3\*j+1]='W';}  
 else if(str[i][j]=='D'){map[3\*i+1][3\*j+1]='W';map[3\*i+1][3\*j+2]='W';map[3\*i+2][3\*j+1]='W';}  
 else if(str[i][j]=='E'){map[3\*i][3\*j+1]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+2][3\*j+1]='W';}  
 else if(str[i][j]=='F'){map[3\*i+1][3\*j]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+1][3\*j+2]='W';}  
 else if(str[i][j]=='G'){map[3\*i][3\*j+1]='W';map[3\*i+1][3\*j]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+1][3\*j+2]='W';}  
 else if(str[i][j]=='H'){map[3\*i][3\*j+1]='W';map[3\*i+1][3\*j]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+2][3\*j+1]='W';}  
 else if(str[i][j]=='I'){map[3\*i+1][3\*j+2]='W';map[3\*i+1][3\*j]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+2][3\*j+1]='W';}  
 else if(str[i][j]=='J'){map[3\*i+1][3\*j+2]='W';map[3\*i][3\*j+1]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+2][3\*j+1]='W';}  
 else if(str[i][j]=='K'){map[3\*i+1][3\*j+2]='W';map[3\*i][3\*j+1]='W';map[3\*i+1][3\*j+1]='W';map[3\*i+2][3\*j+1]='W';map[3\*i+1][3\*j]='W';}  
 }  
 }  
 con = 0;  
 for(i=0;i<=3\*m;i++)  
 {  
 for(j=0;j<=3\*n;j++)  
 if (map[i][j] == 'W')  
 {  
 bfs(i, j);  
 con++;  
 }  
 }  
 printf("%d\n", con);  
 }  
 return 0;  
}

并查集的方法  0MS

这里用的递归法压缩路径，感觉也挺好的，代码很短，以前都是用非递归形式。在合并的时候，按秩合并很重要，每次合并，总把深度小的树合并到深度大的树里面，以防止树退化成链。这样可以减少树的深度，使得查找时间很均匀，更高效的查找。

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include<stdio.h>  
#include<string.h>  
int bin[2502],rank[2502];  
char map[55][55];  
char mat1[100][100];//上下连通  
char mat2[100][100];//左右连通  
int up[8],down[8],left[8],right[8];  
void prepare()  
{  
 int i,j;  
 up[0]='A'; up[1]='B'; up[2]='E'; up[3]='G'; up[4]='H'; up[5]='J'; up[6]='K';  
 down[0]='C'; down[1]='D'; down[2]='E'; down[3]='H'; down[4]='I'; down[5]='J'; down[6]='K';  
 left[0]='A'; left[1]='C'; left[2]='F'; left[3]='G'; left[4]='H'; left[5]='I'; left[6]='K';  
 right[0]='B'; right[1]='D'; right[2]='F'; right[3]='G'; right[4]='I'; right[5]='J'; right[6]='K';  
 memset(mat1,0,sizeof(mat1));  
 memset(mat2,0,sizeof(mat2));  
 for(i=0;i<7;i++)  
 {  
 for(j=0;j<7;j++)  
 {  
 mat1[down[i]][up[j]]=1;  
 mat2[right[i]][left[j]]=1;  
 }  
 }  
}  
  
int find(int x)  
{  
 if(x!=bin[x])  
 return bin[x]=find(bin[x]);  
 return x;  
}  
  
void merge(int x,int y)  
{  
 int fx=find(x);  
 int fy=find(y);  
 if(fx!=fy)  
 {  
 if(rank[fx]>rank[fy])  
 bin[fy]=fx;  
 else if(rank[fx]<rank[fy])  
 bin[fx]=fy;  
 else  
 {  
 bin[fx]=fy;  
 rank[fy]++;  
 }  
 }  
}  
  
int main()  
{  
 int n,m,i,j;  
 prepare();  
 while(scanf("%d%d",&n,&m),n+m!=-2)  
 {  
 getchar();  
 for(i=0;i<n;i++)  
 gets(map[i]);  
 for(i=0;i<m\*n;i++)  
 bin[i]=i, rank[i]=1;  
 for(i=0;i<n;i++)  
 {  
 for(j=0;j<m;j++)  
 {  
 if(j<m-1 && mat2[map[i][j]][map[i][j+1]]==1)  
 merge(i\*m+j,i\*m+j+1);  
 if(i<n-1 && mat1[map[i][j]][map[i+1][j]]==1)  
 merge(i\*m+j,i\*m+j+m);  
 }  
 }  
 int con=0;  
 for(i=0;i<m\*n;i++)  
 if(bin[i]==i)  
 con++;  
 printf("%d\n",con);  
 }  
 return 0;  
}

[HDU1558 Segment set](http://acm.hdu.edu.cn/showproblem.php?pid=1558)

第一点：如何判断两条线段是否有交点。

第二点：快速查找某条线段所在集合里的线段条数，用并查集实现。

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include<stdio.h>  
#define N 1005  
typedef struct node{  
 double x,y;  
}NODE;  
NODE stap[N],endp[N];  
int cnt[N],rank[N],bin[N];  
int index,n;  
void init()  
{  
 index=1;  
 for(int i=1;i<=n;i++)  
 {  
 bin[i]=i;  
 rank[i]=1;  
 cnt[i]=1;  
 }  
}  
  
int Find(int x)  
{  
 if(x!=bin[x])  
 return bin[x]=Find(bin[x]);  
 return x;  
}  
void Merge(int x,int y)  
{  
 int fx=Find(x);  
 int fy=Find(y);  
 if(fx!=fy)  
 {  
 if(rank[fx]>rank[fy])  
 {  
 bin[fy]=fx;  
 cnt[fx]+=cnt[fy];  
 }  
 else  
 {  
 if(rank[fx]==rank[fy])  
 rank[fy]++;  
 bin[fx]=fy;  
 cnt[fy]+=cnt[fx];  
 }  
 }  
}  
  
int Value(NODE s,NODE e,NODE t)  
{  
 double ans=(e.x-s.x)\*(t.y-s.y)-(t.x-s.x)\*(e.y-s.y);  
 if(ans<0)  
 return -1;  
 else if(ans>0)  
 return 1;  
 else  
 return 0;  
}  
int Cross(NODE s1,NODE e1,NODE s2,NODE e2)  
{  
 //if(!(Value(s1,e1,s2)\*Value(s1,e1,e2)>0) && !(Value(s2,e2,s1)\*Value(s2,e2,e1)>0))  
 if((Value(s1,e1,s2)\*Value(s1,e1,e2)<=0) && (Value(s2,e2,s1)\*Value(s2,e2,e1)<=0))  
 return 1;  
 return 0;  
}  
  
void Add(NODE s,NODE e)  
{  
 for(int i=1;i<index;i++)  
 {  
 if(Cross(s,e,stap[i],endp[i]))  
 Merge(i,index);  
 }  
}  
  
int main()  
{  
 int cas,x,fx;  
 char ch[4];  
 scanf("%d",&cas);  
 while(cas--)  
 {  
 scanf("%d",&n);  
 init();  
 while(n--)  
 {  
 scanf("%s",ch);  
 if(ch[0]=='P')  
 {  
 scanf("%lf%lf%lf%lf",&stap[index].x,&stap[index].y,&endp[index].x,&endp[index].y);  
 Add(stap[index],endp[index]);  
 index++;  
 }  
 else{  
 scanf("%d",&x);  
 fx=Find(x);  
 printf("%d\n",cnt[fx]);  
 }  
 }  
 if(cas)  
 puts("");  
 }  
 return 0;  
}

[PKU1182 食物链](http://124.205.79.250/JudgeOnline/problem?id=1182)

一组测试实例，写成多组的话会WA，道理同PKU2492一样，用到偏移量

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include<stdio.h>  
#define N 50005  
int bin[N],offset[N];  
int ans;  
int find(int x)  
{  
 if(x!=bin[x])  
 {  
 int t=find(bin[x]);  
 offset[x]=(offset[x]+offset[bin[x]])%3;  
 return bin[x]=t;  
 }  
 return x;  
}  
void merge(int d,int x,int y)  
{  
 int fx=find(x);  
 int fy=find(y);  
 if(fx==fy)  
 {  
 if(d==1)  
 {  
 if(offset[x]!=offset[y])  
 ans++;  
 }  
 else  
 {  
 if((offset[x]-offset[y]+3)%3!=1)  
 ans++;  
 }  
 }  
 else  
 {  
 if(d==1){  
 bin[fx]=fy;  
 offset[fx]=(offset[y]-offset[x]+3)%3;  
 }  
 else{  
 bin[fx]=fy;  
 offset[fx]=(offset[y]-offset[x]+4)%3;  
 }  
 }  
}  
int main()  
{  
 int n,k,i,d,x,y;  
 scanf("%d%d",&n,&k);  
 ans=0;  
 for(i=1;i<=n;i++)  
 {  
 bin[i]=i;  
 offset[i]=0;  
 }  
 while(k--)  
 {  
 scanf("%d%d%d",&d,&x,&y);  
 if(x>n || y>n || (d==2&&x==y)){  
 ans++;  
 continue;  
 }  
 merge(d,x,y);  
 }  
 printf("%d\n",ans);  
 return 0;  
}

[HDU1811 Rank of Tetris](http://acm.hdu.edu.cn/showproblem.php?pid=1811)

拓扑排序+并查集

[如何去理解 拓扑排序算法](http://www.cnblogs.com/shanyou/archive/2006/11/16/562861.html)