并查集，不相交集数据结构，判断两个事物是否为同一类或有同一个祖先。

算法思想是，先用一维数组存储数据，每个下标都存储下标本身。比如说f[1] = 1,f[2] = 2,f[3] = 3.. 就是说1的祖先是1，2的祖先是2，3的祖先3。改变f数组的值后，f[1] = 1,f[2] = 1,f[3] = 1; 就是说1的祖先是1，2的祖先是1，3的祖先1。

存储完成后，开始合并。这里用一个递归去找爹，不断地找，直到f[x] = x;

Function zhaoDie(v)

{

if(f[v] == v)

{

return v;

}

else

{

f[v] = zhaoDie(f[v]);

//这里f[v]赋值，是为了把递归的路上把不相等f[]改成祖先的值。

return f[v];

}

}

for(i = 1;i <= (找祖先的条件个数m);i++)

{

heBing(v, u);

}

Function heBing(v, u)

{

if(zhaoDie(v) != zhaoDie(u))

{

f[zhaoDie(v) ] = zhaoDie(u) ;

}

}

最后扫描的时候判断f数组存的值跟下标是否相等。

最后输出有多少个独立的

注：m是m个找祖先线索，n是个数

输入格式：

10 9 n m

1 2

3 4

5 2

4 6

2 6

8 7

9 7

1 6

2 4