C – POJ 2942 点双连通分量

1.题目大意

从 N 个骑士中找出奇数个坐在圆桌边开会，使相邻的骑士不互相仇恨。

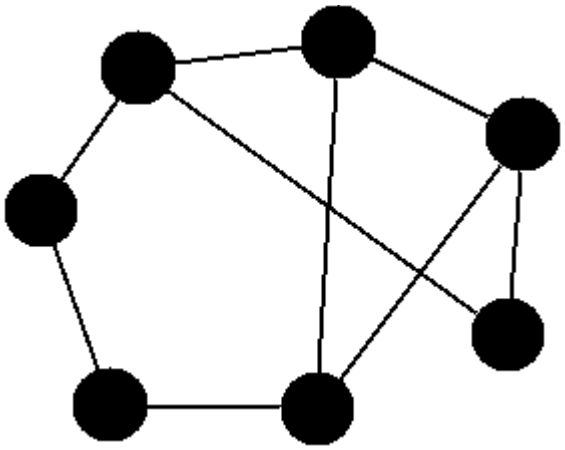
2.解题思路

a.建立补图

题目中给出的是“相互仇恨”的列表。我们要建立一个补图，使友好的骑士之间连边。

b.点双连通分量

概念：点双连通分量是指在这个连通分量中，每两个点都有两条完全不同的路径可到达。也就是去掉这个图的任意一个点。连通分量中的点两两之间依然可达。如图所示。



题目要求只要相邻的骑士不为互相仇恨，所以对于每一个双连通分量来说，都是有可能满足条件的。而对于图中的环来说，就无法满足”最大“这个条件。

c.奇环

如上图的点双连通分量，是不满足题目条件的，因为对于整个分量来说，是不成环的。所以又引出一个问题，如何判断奇环。

统计数目是不可行的，上图就是反例。所以我们使用 DFS + 染色的方法来进行判断。条件如下：

性质：如果此连通分量中存在奇环，那么此连通分量中的所有点都可以在一个奇环内

设连通图简化为(a)=(b)，(a)为奇环。若(b)为奇环，明显符合条件。如(b)为偶环，则(ab)也必是奇环。

3.解题代码

Problem: 2942 **User:** wizmann
Memory: 9748K **Time:** 1297MS

Language: G++ **Result:** Accepted

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <stack>
#include <algorithm>

using namespace std;

#define print(x) cout<<x<<endl
#define input(x) cin>>x
#define SIZE 1010

typedef long long llint;

typedef struct edge
{
        int st,end;
        edge(int a,int b){st=a;end=b;}
        inline void setedge(int a,int b){st=a;end=b;}
}edge;

int cnc[SIZE][SIZE];//连通临接表
char visit[SIZE][SIZE];//是否访问过
int dfn[SIZE],low[SIZE];//时间戳
int lv;//DFS层数
int g[SIZE][SIZE];//原始图
int in[SIZE];
stack<edge> st;
int color[SIZE];
int oddcir[SIZE];

int n,m;

bool bio(int pos,int c)
{
        color[pos]=c;
        for(int i=1;i<=cnc[pos][0];i++)
        {
                int now=cnc[pos][i];
                if(in[now])
                {
                        if(color[pos]==color[now]) return false;
                        if(color[now]==-1&&!bio(now,1-c)) return false;
                }
        }
        return true;
}

void slove(int pos)
{
        memset(in,0,sizeof(in));
        while(1)
```

```
        {
                edge now=st.top();
                st.pop();
                in[now.st]=in[now.end]=1;//该点在此点双连通分量中
                if(now.st==pos) break;
        }
        memset(color,-1,sizeof(color));
        if(!bio(pos,0))
        {
                for(int i=1;i<=n;i++) oddcir[i]|=in[i];
        }
}

void tarjan(int pos)
{
        dfn[pos]=low[pos]=++lv;
        for(int i=1;i<=cnc[pos][0];i++)
        {
                int u=cnc[pos][i];
                if(visit[pos][u]) continue;//判重边 pos->u
                visit[pos][u]=visit[u][pos]=1;//无向图

                if(!dfn[u])//如果 u 点没有被访问过
                {
                        st.push(edge(pos,u));//压栈
                        tarjan(u);//向下搜索
                        low[pos]=min(low[pos],low[u]);//low[x]表示 x 所能到达的
最小深度的顶点
                        if(dfn[pos]<=low[u]) slove(pos);//如果不能有比 pos 更小
的深度

//找出找出双连通分量，进行处理
                }
                else low[pos]=min(low[pos],dfn[u]);
        }
}

void pb(int x,int y)//建立伪链表
{
        cnc[x][0]++;
        cnc[x][cnc[x][0]]=y;
}

int main()
{
        int a,b;
        while(input(n>>m) && n+m)
        {
                memset(cnc,0,sizeof(cnc));
                memset(visit,0,sizeof(visit));
                memset(dfn,0,sizeof(dfn));
                memset(low,0,sizeof(low));
                memset(g,-1,sizeof(g));
                memset(oddcir,0,sizeof(oddcir));
                while(!st.empty()) st.pop();
                lv=0;
                //I hate this... = =...

                for(int i=0;i<m;i++)
```

```
                {
                        scanf("%d%d",&a,&b);
                        g[a][b]=g[b][a]=0;
                }
                for(int i=1;i<=n;i++)
                {
                        for(int j=i+1;j<=n;j++)
                        {
                                if(g[i][j])
                                {
                                        pb(i,j);
                                        pb(j,i);
                                }
                        }
                }//建立补图

                for(int i=1;i<=n;i++)
                {
                        if(!dfn[i]) tarjan(i);
                }

                llint ans=0;
                for(int i=1;i<=n;i++)
                {
                        if(oddcir[i]) ans++;
                }
                print(n-ans);
        }
        return 0;
}
```