网络攻击

一. 考察内容:

图论 图论计数

二. 题目分析:

[题目大意]

在一个无向图上的割掉两条边，使得图不连通，求出所有方案数。

[写题思路]

考虑对原图从一点开始进行深度优先遍历，求出dfs树，定义“横叉边”为一棵树上两个没有血缘关系的点之间连接的一条边，我们可以知道，无向图的dfs数上没有横叉边，所有的非树边都是返祖边，所以我们考虑3种情况，1.割掉两条树边，2.割掉一条树边和一条非树边，3.割掉两条非树边。

由树的定义可知，割掉两条非树边不能使图不连通，所以总方案数就是情况1和情况2的总和。

定义树边被非树边“覆盖”为非树边两点之间的树上通路经过的所有边，一条链每被覆盖一次，就相当于多了一条从链底部（靠近叶子的一端）到链顶部（另一端）多了一条通路，所以我们可以推出，当一条边没有被覆盖过时，割掉这条边和任意一个非树边即可使图不连通，当一条边被覆盖一次时，同时割掉该边和覆盖他的非树边可以使得整个图不连通，如果一条边被覆盖多次，则无论如何都不能通过情况2使图不连通。

考虑情况1，如果某条树边没有被覆盖过，那么他在原图中就是桥，割掉他和其他任意一条树边（非树边也可以，但是在上一段落中已经计算过了）即可使图不连通，当两条边满足他们的覆盖情况相同时，割掉这两条边也可以满足要求。

统计上述两种情况的方案数之和即可。

[实现]

其中实现难点在于如何统计每个树边被哪些非树边覆盖和覆盖次数，覆盖次数考虑在dfs树上用差分实现，差分的汇总可以在递归返回是求得。

那么如何求每个树边被哪些非树边覆盖呢?

我们考虑使用随机化算法，对于任意一个非树边，给他赋一个随机值，将树边的覆盖情况用这些覆盖他的非树边随机值的异或值表示，同样使用差分来统计即可。

三. 代码实现:

```
#define _CRT_SECURE_NO_DEPRECATE
/************************
*创建时间：2018 08 20
*文件类型：源代码文件
*题目来源：COGS
*当前状态：已通过
*备忘录：图论 dfs树
*作者：HtBest
************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
```

```cpp
#include <queue>
#include <time.h>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 2001
#define MAXM 100001
#define INF 1000000000
int n,m,head[MAXN],_edge,dfsx[MAXN],_dfsx,cover[MAXN],v[MAXN];
long long ans;
struct EDGE
{
    int a,b,next;
    EDGE(int a=0,int b=0,int next=0):a(a),b(b),next(next){}
}edge[2*MAXM];
/* Variable explain:
n:节点数
m:边数
head:每个点的第一条出边
edge:邻接表
_edge:邻接表标记
dfsx:dfs序
_dfsx:dfs序标记
cover:树边的覆盖次数
v:树边的异或值
ans:答案
*/
void adde(int a,int b)
{
    edge[++_edge]=EDGE(a,b,head[a]);
    head[a]=_edge;
}
void read()
{
    int ls1,ls2;
    // freopen("networkattack.in","r",stdin);
    // freopen("networkattack.out","w",stdout);
    scanf("%d%d",&n,&m);
    _edge=1;
    for(int i=1;i<=m;++i)scanf("%d%d",&ls1,&ls2),adde(ls1,ls2),adde(ls2,ls1);
    return;
}
void dfs(int a,int l)
{
    // printf("%d\n",a);
    dfsx[a]=++_dfsx;
    for(int i=head[a];i;i=edge[i].next)
    {
        int b=edge[i].b;
        if(i==(l^1)||dfsx[b]>dfsx[a])continue;
        if(!dfsx[b])
        {
            // printf("next:");//输出建树过程
            dfs(b,i);
            cover[a]+=cover[b];
            v[a]^=v[b];
        }
        else //非树边
        {
            int ls1=rand()%INF;
            --cover[b];++cover[a];
```

```
            v[b]^=ls1,v[a]^=ls1;
        }
    }
}
int main()
{
    srand(time(0)+clock());
    read();
    dfs(1,0);
    for(int i=2;i<=n;++i)
    {
        if(cover[i]==0)ans+=m-n+1;
        if(cover[i]==1)++ans;
        for(int j=i+1;j<=n;++j)
            if(!v[i]||!v[j]||v[i]==v[j])++ans;
    }
    printf("%lld\n",ans);
    return 0;
}
```

<题目跳转> <查看代码>