**Travel**

**Time Limit: 10000/2000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 2580    Accepted Submission(s): 845**

Problem Description

      One day, Tom traveled to a country named BGM. BGM is a small country, but there are N (N <= 100) towns in it. Each town products one kind of food, the food will be transported to all the towns. In addition, the trucks will always take the shortest way. There are M (M <= 3000) two-way roads connecting the towns, and the length of the road is 1.  
      Let SUM be the total distance of the shortest paths between all pairs of the towns. Please write a program to calculate the new SUM after one of the M roads is destroyed.

Input

      The input contains several test cases.  
      The first line contains two positive integers N, M. The following M lines each contains two integers u, v, meaning there is a two-way road between town u and v. The roads are numbered from 1 to M according to the order of the input.  
      The input will be terminated by EOF.

Output

      Output M lines, the i-th line is the new SUM after the i-th road is destroyed. If the towns are not connected after the i-th road is destroyed, please output “INF” in the i-th line.

Sample Input

5 4

5 1

1 3

3 2

5 4

2 2

1 2

1 2

Sample Output

INF

INF

INF

INF

2

2

Source

[2008 Asia Chengdu Regional Contest Online](http://acm.hdu.edu.cn/search.php?field=problem&key=2008+Asia+Chengdu+Regional+Contest+Online&source=1&searchmode=source)

题目是要求每次删除地i条边后以1,2,3,...n为起始点到其他点的最短路和的总和

分析:从题意可以想到用spfa,毕竟spfa是求单源多点的最短路的好[算法](http://www.xuebuyuan.com/category/%E7%AE%97%E6%B3%95" \o "算法" \t "_blank), 所以可以想到每次删除第i条边时，我们只需要暂时把这条边长度赋值无穷大,然后对1,2,3,....n各点进行spfa然后求和,这是最初的思想,但是 看复杂度m\*n\*m=9\*10^8肯定会超时,第一个m代表删除m条边,第二个n表示每次删除一条边后就对1,2,3....n个点进行spfa,第三个 m表示spfa的复杂度(由于边长度为1所以每个点其实只要入队一次,所以复杂度是m),在m\*n\*m中我们能优化的只能是第一个m了,根据上面n表示的 意思我们知道每起始个点都要因为m条边而进行spfa  
m次,但是其实以起始点i开始的最短路边只有n-1,条,也就是说有m-n+1条边与以起始点i开始的最短路无关(即影响不到),那么如何去判断什么时候 进行spfa呢??在这里引进一个数组flag[i][u][v]表示以点i作为起始点的最短路是否含有边u-v,所以我们要预先进行spfa来标记 flag

这样就能根据flag[i][u[v]]来决定什么时候进行spfa,所以复杂度就变成了n\*n\*m,第一个n表示每个点最多要进行n-1(o(n))次

#include <iostream>

#include <cstdio>

#include <cstring>

#include <memory.h>

#include <queue>

#include <map>

#include <algorithm>

#include <cstdlib>

#include <string>

#include <iomanip>

#define INF 99999999

using namespace std;

const int MAX = 100+10;

int dist[MAX], Edgenum[MAX][MAX];

bool mark[MAX],flag[MAX][MAX][MAX];

int size,head[MAX],n,m,pos[3002],sum[MAX];

struct node

{

int v,w;

int next;

node(){}

node(int V,int W,int Next):v(V),w(W),next(Next){}

}edge[6500];

void Init(int num)

{

memset(head, -1, sizeof(int)\*(num + 2));

memset(Edgenum, 0, sizeof Edgenum);

memset(flag, false, sizeof(flag));

size = 0;

}

void put\_edge(int u,int v,int w)

{

edge[size] = node(v,w,head[u]);

head[u] = size++;

}

int spfa(int s,bool p)

{

int ans = 0;

queue<int>enqueue;

for(int i = 1; i <= n; ++i)

{

dist[i] = INF;

mark[i] = false;

}

dist[s] = 0;

mark[s] = true;

enqueue.push(s);

while(!enqueue.empty())

{

int u = enqueue.front();

enqueue.pop();

mark[u] = false;

for(int j = head[u]; j !=-1; j = edge[j].next)

{

int v = edge[j].v;

if(dist[v] > dist[u] + edge[j].w)

{

dist[v] = dist[u] + edge[j].w;

if(!mark[v])

{

if(p)

flag[s][u][v] = flag[s][v][u] = true;

enqueue.push(v);

mark[v] = true;

}

}

}

}

for(int i = 1; i <= n; ++i)

{

if(dist[i] == INF)

return INF;

else

ans+=dist[i];

}

return ans;

}

int main()

{

int ans,u,v,temp;

while(cin >> n >> m)

{

Init(n);

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

{

cin >> u >> v;

pos[i] = size;

put\_edge(u,v,1);

put\_edge(v,u,1);

++Edgenum[u][v];

++Edgenum[v][u];

}

ans = temp = 0;

for(int i = 1; i <= n; ++i)

{

sum[i] = spfa(i,1);

if(sum[i] == INF)

{

ans = INF;

break;

}

else

ans+=sum[i];

}

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

{

u = edge[pos[i]+1].v;

v = edge[pos[i]].v;

int s = ans;

if(ans == INF)

cout << "INF" << endl;

else if(Edgenum[u][v] - 1 > 0)

cout << ans << endl;

else

{

edge[pos[i]+1].w = INF;

edge[pos[i]].w= INF;

for(int j = 1; j <= n; ++j)

{

if(flag[j][u][v])

{

temp = spfa(j,0);

if(temp == INF)

{

cout << "INF" << endl;

break;

}

else

s+=temp-sum[j];

}

}

if(temp != INF)

cout << s << endl;

edge[pos[i]+1].w = 1;

edge[pos[i]].w = 1;

}

}

}

return 0;

}