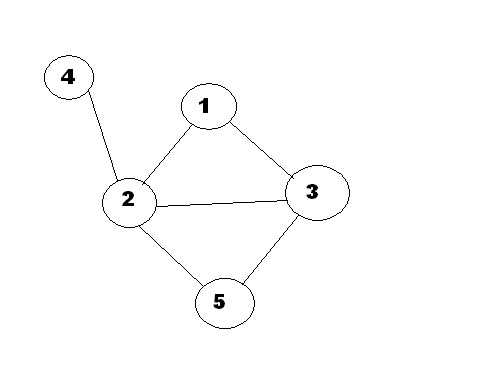
**网络寻路**

问题描述

X 国的一个网络使用若干条线路连接若干个节点。节点间的通信是双向的。某重要数据包，为了安全起见，必须恰好被转发两次到达目的地。该包可能在任意一个节点产生，我们需要知道该网络中一共有多少种不同的转发路径。

源地址和目标地址可以相同，但中间节点必须不同。

如下图所示的网络。



1 -> 2 -> 3 -> 1 是允许的

1 -> 2 -> 1 -> 2 或者 1 -> 2 -> 3 -> 2 都是非法的。

输入格式

输入数据的第一行为两个整数N M，分别表示节点个数和连接线路的条数(1<=N<=10000; 0<=M<=100000)。

接下去有M行，每行为两个整数 u 和 v，表示节点u 和 v 联通(1<=u,v<=N , u!=v)。

输入数据保证任意两点最多只有一条边连接，并且没有自己连自己的边，即不存在重边和自环。

输出格式

输出一个整数，表示满足要求的路径条数。

样例输入1

3 3  
1 2  
2 3  
1 3

样例输出1

6

样例输入2

4 4  
1 2  
2 3  
3 1  
1 4

样例输出2

10

锦囊1

图论构图，动态规划。

本题的C++参考代码如下：

#include<cstdio>

#include<cstring>

#define MAXN 10010

#define MAXM 100010

int Du[MAXN],U[MAXM],V[MAXM];

int main()

{

int n,m;

long long ans=0;

scanf("%d%d",&n,&m);

memset(Du,0,sizeof(Du));

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

scanf("%d%d",&U[i],&V[i]);

Du[U[i]]++;

Du[V[i]]++;

}

for(int i=0;i<m;i++)if(Du[U[i]]>1&&Du[V[i]]>1)ans+=(Du[U[i]]-1)\*(Du[V[i]]-1)\*2;

printf("%I64d\n",ans);

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#define MAXN 10001

#define MAXM 100001

int main()

{

int i, j;

int n, m;

int edge[MAXN]={0}, u[MAXM]={0}, v[MAXM]={0};

int ans = 0;

scanf("%d%d", &n, &m);

for(i=1; i<=m; i++){

scanf("%d%d", &u[i], &v[i]);

edge[u[i]] ++;

edge[v[i]] ++;

}

for(i=1; i<=m; i++){

if(edge[u[i]]>1 && edge[v[i]]>1){

ans += (edge[u[i]]-1) \* (edge[v[i]]-1) \* 2;

}

}

printf("%d", ans);

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader bfr = new BufferedReader(

new InputStreamReader(System.in));

StringTokenizer tok = new StringTokenizer(bfr.readLine());

int n = Integer.parseInt(tok.nextToken());

int m = Integer.parseInt(tok.nextToken());

int i, count = 0;

ArrayList<Integer>[] way = new ArrayList[n];

for (i = 0; i < n; i++)

way[i] = new ArrayList<Integer>();

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

tok = new StringTokenizer(bfr.readLine());

int x = Integer.parseInt(tok.nextToken()) - 1;

int y = Integer.parseInt(tok.nextToken()) - 1;

way[x].add(y);

way[y].add(x);

}

for (i = 0; i < n; i++) {

int a = i, ll = way[i].size();

for (int ii = 0; ii < ll; ii++) {

int aa = way[a].get(ii);

int lll = way[aa].size();

for (int iii = 0; iii < lll; iii++) {

int aaa = way[aa].get(iii);

if (aaa == a)

continue;

count += way[aaa].size();

count--;

}

}

}

System.out.println(count);

}

}