1076. Forwards on Weibo (30)

时间限制

3000 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

Weibo is known as the Chinese version of Twitter. One user on Weibo may have many followers, and may follow many other users as well. Hence a social network is formed with followers relations. When a user makes a post on Weibo, all his/her followers can view and forward his/her post, which can then be forwarded again by their followers. Now given a social network, you are supposed to calculate the maximum potential amount of forwards for any specific user, assuming that only L levels of indirect followers are counted.

**Input Specification:**

Each input file contains one test case. For each case, the first line contains 2 positive integers: N (<=1000), the number of users; and L (<=6), the number of levels of indirect followers that are counted. Hence it is assumed that all the users are numbered from 1 to N. Then N lines follow, each in the format:

M[i] user\_list[i]

where **M[i]** (<=100) is the total number of people that user[i] follows; and **user\_list[i]** is a list of the M[i] users that are followed by user[i]. It is guaranteed that no one can follow oneself. All the numbers are separated by a space.

Then finally a positive K is given, followed by K **UserID**'s for query.

**Output Specification:**

For each **UserID**, you are supposed to print in one line the maximum potential amount of forwards this user can triger, assuming that everyone who can view the initial post will forward it once, and that only L levels of indirect followers are counted.

**Sample Input:**

7 3

3 2 3 4

0

2 5 6

2 3 1

2 3 4

1 4

1 5

2 2 6

**Sample Output:**

4

5

[提交代码](https://www.patest.cn/contests/pat-a-practise/1076)

对于每个user，在其关注人的粉丝列表里加入其地址；

如此即可建立图

对于图中目标点的每个粉丝进行搜索，规则为：

1. 初始层数为0，每进行一次搜索，其层数增加一层，到达目标层数，即统计后返回
2. 每当一个顶点被访问到，在vst中留下当时的层数，可以被再次访问，若再次访问层数大于或等于前面的函数，则剪枝，不然则更新状态，继续进行下一层的访问。
3. 统计vst中不为零的值，注意此处起始点可能会被访问到，需要在函数里进行特判。

#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

#include<sstream>

#include<stack>

#include<map>

#include<cstring>

#include<climits>

#define MAX 1001

using namespace std;

struct user

{

int id;

vector<user\*>follower;

};

int vst[MAX] = {0},inti;

int flag = 0;

vector<user\*>all\_user\_list;

int all\_user\_num, indirect\_limit;

void dfs(int current, int deepth)

{

if (current == inti&&flag)

return;

if (deepth>indirect\_limit)

return;

if (vst[current] == 0)

vst[current] = deepth;

else if (vst[current] > deepth)

vst[current] = deepth;

else

{

return;

}

flag = 1;

for (int i = 0; i < all\_user\_list[current]->follower.size(); i++)

{

dfs(all\_user\_list[current]->follower[i]->id-1, deepth + 1);

}

}

void output()

{

int total = 0;

for (int i = 0; i < all\_user\_num; i++)

{

if (vst[i]<=indirect\_limit&&vst[i]!=0)

{

total++;

vst[i] = 0;

}

}

cout << total << endl;

}

int main()

{

cin >> all\_user\_num >> indirect\_limit;

for (int i = 0; i < all\_user\_num; i++)

{

user\*temp;

temp = new user;

temp->id = i + 1;

all\_user\_list.push\_back(temp);

}

for (int i = 0; i < all\_user\_num; i++)

{

int num;

scanf("%d",& num);

for (int j = 0; j < num; j++)

{

int now;

scanf("%d",& now);

all\_user\_list[now-1]->follower.push\_back(all\_user\_list[i]);

}

}

int test\_num;

cin >> test\_num;

for (int i = 0; i < test\_num; i++)

{

int now;

scanf("%d",&now);

inti=now-1;

dfs(now - 1, 0);

output();

flag = 0;

}

}