These days, you can do all sorts of things online. For example, you can use various websites to make virtual friends. For some people, growing their social network (their friends, their friends’ friends, their friends’ friends’ friends, and so on), has become an addictive hobby. Just as some people collect stamps, other people collect virtual friends.

Your task is to observe the interactions on such a website and keep track of the size of each person’s network.

Assume that every friendship is mutual. If Fred is Barney’s friend, then Barney is also Fred’s friend.

**Input**

The first line of input contains one integer specifying the number of test cases to follow. Each test case begins with a line containing an integer FF, the number of friendships formed, which is no more than 100000100000. Each of the following FF lines contains the names of two people who have just become friends, separated by a space. A name is a string of 1 to 20 letters (uppercase or lowercase).

**Output**

Whenever a friendship is formed, print a line containing one integer, the number of people in the social network of the two people who have just become friends.

|  |  |
| --- | --- |
| **Sample Input 1** | **Sample Output 1** |
| 1  3  Fred Barney  Barney Betty  Betty Wilma | 2  3  4 |

这道题考察的是新知识……做不出来也在情理之中，考察并查集，并查集的资料见附录，如果知道的话比赛的时候做出来是很快的……

#include <iostream>

#include<sstream>

#include<iomanip>

#include<string>

#include<vector>

#include<stack>

#include<queue>

#include<algorithm>

#include<map>

#include<cmath>

#include<climits>

#define hash 997

#define MAX 100005

#define ll long long

using namespace std;

map<string, int>M;

vector<int>pre;

vector<int>pre\_num;

int now = 1;

int find(int num)

{

int copy = num;

while (pre[copy] != copy)

{

copy = pre[copy];

}

int temp = num;

int temp2;

while (pre[temp] != temp)

{

temp2 = pre[temp];

pre[temp] = copy;

temp = temp2;

}

return copy;

}

void join(int num1, int num2)

{

int Anum1 = find(num1), Anum2 = find(num2);

if (Anum1 != Anum2)

{

pre[Anum2] = Anum1;

pre\_num[Anum1] += pre\_num[Anum2];

pre\_num[Anum2] = 0;

}

}

void reset()

{

M.clear();

pre.resize(MAX);

pre\_num.resize(MAX);

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

{

pre[i] = i;

pre\_num[i] = 1;

}

now = 1;

}

int main()

{

int T;

cin >> T;

while (T--)

{

reset();

int num;

cin >> num;

while (num--)

{

string a, b;

int x, y;

cin >> a >> b;

if (!M[a])

{

x = now;

M[a] = now++;

}

else

x = M[a];

if (!M[b])

{

y = now;

M[b] = now++;

}

else

y = M[b];

join(x, y);

int target = find(x);

cout << pre\_num[target] << endl;

}

}

}