并查集写法

findFather函数

```
int findFather(int x)
{
    int a = x;
    while (x != father[x])
        x = father[x];
    while(a!=father[a])
    {
        int temp = a;
        a = father[a];
        father[temp] = x;
    }
    return x;
}
```

Union函数

```
void Union(int a,int b)
{
    int faA = findFather(a);
    int faB = findFather(b);
    father[faA] = faB;
}
```

以上就是并查集最主要的两个函数。

findFather函数的第二个while是压缩路径，使得集合都指向root点，使查找复杂度降为$O(1)$

例题　1107 Social Clusters (30分)

When register on a social network, you are always asked to specify your hobbies in order to find some potential friends with the same hobbies. A **social cluster** is a set of people who have some of their hobbies in common. You are supposed to find all the clusters.

Input Specification:

Each input file contains one test case. For each test case, the first line contains a positive integer *N* (≤1000), the total number of people in a social network. Hence the people are numbered from 1 to *N*. Then *N* lines follow, each gives the hobby list of a person in the format:

*K****i*: *h****i*[1] *h****i*[2] ... *h****i*[*K****i*]

where *K****i* (>0) is the number of hobbies, and *h****i*[*j*] is the index of the *j*-th hobby, which is an integer in [1, 1000].

Output Specification:

For each case, print in one line the total number of clusters in the network. Then in the second line, print the numbers of people in the clusters in non-increasing order. The numbers must be separated by exactly one space, and there must be no extra space at the end of the line.

Sample Input:

```
8
3: 2 7 10
1: 4
2: 5 3
1: 4
1: 3
1: 4
4: 6 8 1 5
1: 4
```

Sample Output:

```
3
4 3 1
```

思路

是一道很典型的并查集题目。不过是我不太熟悉的类型。记得多复习。

参考代码

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
vector<int> father, isRoot, course;
const int inf = 1010;
int findFather(int x)
{
    int a = x;
    while (x != father[x])
        x = father[x];
    while(a!=father[a])
    {
        int temp = a;
        a = father[a];
        father[temp] = x;
    }
    return x;
}
void Union(int a,int b)
{
    int faA = findFather(a);
    int faB = findFather(b);
    father[faA] = faB;
}
bool cmp(int a,int b)
{
    return a > b;
}
int main()
```

```
{
    int number = 0, t = 0, temp = 0;
    cin >> number;
    course.resize(inf);
    isRoot.resize(number + 1);
    father.resize(number + 1);
    for (int i = 0; i < number + 1; i++)
        father[i] = i;
    for(int i=1;i<number+1;i++)
    {
        scanf("%d: ", &t);
        for(int j=0;j<t;j++)
        {
            cin >> temp;
            if (course[temp] == 0)
                course[temp] = i;
            Union(i, findFather(course[temp]));
        }
    }
    for (int i = 1; i < number + 1; i++)
        isRoot[findFather(i)]++;
    sort(isRoot.begin(), isRoot.end(), cmp);
    temp = 0;
    for (int i = 0; i < number + 1; i++)
        if (isRoot[i] != 0)
            temp++;
    cout << temp << endl << isRoot[0];
    for (int i = 1; i < temp; i++)
        cout << " " << isRoot[i];
    return 0;
}
```