中山大学计算机学院本科生实验报告

（2021 学年第 1 学期）

课程名称：Data structures and algorithms　　任课教师：张子臻

年级	20 级	专业（方向）	软件工程
学号	20337270	姓名	钟海财
电话	13397996670	Email	2940599563@qq.com
开始日期	2021/12/2	完成日期	2021/12/8

1. 实验题目

1) 连通性问题

2) 家族查询

2.实验目的

1）关系 R 具有对称性和传递性。数对(p,q)表示 pRq,p 和 q 是 0 或自然数,p 不等于 q。要求写一个程序将数对序列进行过滤：如果一个数对可以通过前面数对的传递性得到，则将其滤去。

2）实现家族查询，判断输入的 x，y 是否为亲戚：

第一行为数据组数 T(T <= 20)

对于每组数据，第一行有两个整数 n、m(1 <= n, m <= 5000)，表示有 n 个人，编号 1~n，其中存在 m 个亲戚关系。

接下来 m 行，每行有两个整数 u、v(u != v, 1 <= u, v <= n)，表示 u 和 v 之间有亲戚关系。

然后是询问组数 q(1 <= q <= 5000)

接下来 q 行，每行有两个整数 u、v(u != v, 1 <= u, v <= n)，询问 u 和 v 之间是否有亲戚关系。

3.程序设计

1) 连通性问题

设计思路

由于关系 R 具有自反性和传递性，对于任意已经连通的数字集合，如：a1-a2-a3-…-an 构成并查集{a1,a2,…,an}：

1）对于集合中的任意两个数字，它们组成的数对都是已经连通的，再出现时需要滤去；

2）且对于集合中的任何数字，它们都有相同的代表元。

则我们可用并查集记录连通性，对于读入的数对(x,y)：当 x 的代表元!=y 的代表元时，说明在已有记录的连通关系中这两个数未连通，这时可将这两个数字分别记录到两个 vector 类型的变量中；当 x 的代表元==y 的代表元时，则说明已连通，需要滤去，则不记录到上述两个 vector 类型的变量中。

并将实现的函数 find，union_ 封装到一个类里，方便调用。

过滤完所有数据后，将两个 vector 类型的变量里存储的数据输出。

代码

```
#include<iostream>
using namespace std;
#include<vector>
class Dset{  //并查集的类
private:
    int *p;
public:
```

```
    Dset(int n){
        p = new int[n+1];
        for(int i =0 ;i<n+1 ;++i) p[i]=-1;
    }
    ~Dset(){
        delete []this->p ;
    }
    int find(int x) {  //返回代表元
        return (p[x] == -1) ? x : p[x] = find(p[x]);//实现了路径压缩
    }
    void union_(int x, int y) {
        if(find(y) == find(x)) return; //x，y 代表元相同则返回
        p[find(y)] = find(x);
    }
};
int main(){
    int n = 100000;
    Dset f(n);
    vector<int> a,b;  //用于存放过滤后的数据
    int x,y;
    while(cin>>x){
        cin>>y;
        if(f.find(x)!=f.find(y)){ //当两数的代表元不同时将两数及其关系记录
           f.union_(x,y);
           a.push_back(x);
           b.push_back(y);
        }
    }
    for(int i = 0 ; i <a.size();++i){  //输出过滤后的数据
        cout<<a[i]<<" "<<b[i]<<endl;
    }
    return 0;
}
```

2) 家族查询

设计思路

由于亲戚的亲戚仍是亲戚，要判断 x，y 是否具有亲戚关系，把亲戚关系换成上题的连通性，发现本质相同，所以我们同样可使用并查集解决该问题。

首先将所有亲戚记录进并查集里，再根据 x，y 是否有相同代表元（亲戚）

判断 x，y 是否为亲戚。

并查集函数设计和上题相同，类里增加一个函数 is_common，判断 x，y 是否具有相同代表元（亲戚）并输出结果。

代码

```
#include<iostream>
using namespace std;
class Dset{
private:
    int *p;
public:
    Dset(int n){
        p = new int[n+1];
        for(int i =0 ;i<n+1 ;++i) p[i]=-1;
    }
    ~Dset(){
        delete []this->p ;
    }
    int find(int x) {  //寻找代表元
        return (p[x] == -1) ? x : p[x] = find(p[x]);//实现了路径压缩
    }
    void union_(int x, int y) {  //记入并查集
        if(find(y) == find(x)) return;
        p[find(y)] = find(x);
    }
    void is_common( int x,int y){  //判断 x，y 是否具有相同代表元（亲戚）并输出
        if(find(x)==find(y)) cout<<"Yes"<<endl;
        else cout<<"No"<<endl;
    }
};
int main(){
    int T;
    cin>>T;
    while(T>0){
        --T;
        int n,m;
        cin>>n>>m;
        Dset family(n);  //定义一个并查集
        int x,y;
        for(int i = 0 ; i <m;++i){
            cin>>x>>y;
            family.union_(x,y);  //将关系记入并查集
        }
```

```
        int q;
        cin>>q;
        for(int i = 0 ; i <q;++i){
            cin>>x>>y;
            family.is_common(x,y);  //输出判断结果
        }
        cout<<endl;
    }
    return 0;
}
```

4.程序运行与测试

1) 连通性问题

测试输入 1

```
3 4
4 9
8 0
2 3
5 6
2 9
5 9
7 3
4 8
5 6
0 2
6 1
```

输出 1

```
3 4
4 9
8 0
2 3
5 6
5 9
7 3
4 8
6 1
```

测试输入 2

```
1 563
571 808
585 479
350 895
822 746
174 858
301 571
303 14
91 364
147 165
988 445
119 4
8 377
531 571
571 607
607 301
450 352
57 607
783 808
783 571
```

输出 2

```
1 563
571 808
585 479
350 895
822 746
174 858
301 571
303 14
91 364
147 165
988 445
119 4
8 377
531 571
571 607
450 352
57 607
783 808
```

测试输入 3

```
1 3
3 9
9 6
9 5
5 3
8 3
2 8
19 2
21 3
21 7
22 21
22 6
```

输出 3

```
1 3
3 9
9 6
9 5
8 3
2 8
19 2
21 3
21 7
22 21
```

2) 家族查询

测试输入 1

```
2
3 1
2 3
2
1 2
2 3
4 2
1 2
1 4
3
1 2
1 3
2 4
```

输出 1

```
No
Yes

Yes
No
Yes
```

测试输入 2

```
2
4 2
2 3
4 3
3
1 2
2 3
4 3

5 3
1 2
1 4
5 2
4
1 2
1 3
2 4
1 5
```

输出 2

```
No
Yes
Yes

Yes
No
Yes
Yes
```

测试输入 3

```
2
4 3
2 3
4 3
1 2
3
2 4
1 3
1 4

6 3
1 2
1 6
5 2
4
6 2
5 3
2 4
1 5
```

输出 3

```
Yes
Yes
Yes

Yes
No
No
Yes
```

5.实验总结与心得

通过本次实验，我发现如果处理的关系 R 具有自反性和传递性（如家族查询中的亲戚关系），我们可以使用并查集来处理，只需判断两个元素是否具有相同的代表元，则可以判断这两个元素是否具有关系 R。

此外，我还遇到一个问题，uion_函数缺少“if(find(y) == find(x)) return;”这句时错误！

```
void union_(int x, int y) {
    if(find(y) == find(x)) return;
    p[find(y)] = find(x);
}
```

因为如果 x，y 的祖先相同时，会让祖先 find(x)的祖先变为祖先自身 find(x), 即 p[find(x)]=find(x)，这样就会形成一个环，使 find 函数陷入死循环。

附录、提交文件清单