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	2021/12/9	完成日期	2021/12/16

1. 实验题目

1）Can I Post the letter

2）Connect components in undirected graph

2.实验目的

1）给出有向图中 n 个点间的 m 个可达关系，判断起点 0 是否存在到终点 n-1 的有向路径

2）给出无向图中 n 个点间的 m 个连通关系，求出连通块的数量

3.程序设计

1）Can I Post the letter

设计思路

由于是有向图，可达关系不具有对称性，不能使用并查集（开始我用并查集但能过评测，应该是测试样例不充分的问题），所以我使用广度/深度优先搜索：

初始准备：

```
vector<int> add;//存放当前所有可达点
vector<int>p[n];//存放可达关系
int m;
cin>>m;
for(int i = 0; i <m ;++i){//将可达关系读入 p 中
    int x,y;
    cin>>x>>y;
    p[x].push_back(y);
}
int vis[n]={0}; //用于存放 0 的可达点:vis[k]=1 表示 0 可达 k,vis[k]=0 表示 0 不可达 k
vis[0]=1;//起点为 0,已经可达
add.push_back(0);//将起点设置为新增可达点
```

搜索函数设计：

```
void visit(vector<int> p[],vector<int>&add,int vis[],int start){
    if(start==add.size()) return;//无新增可达点时，跳出递归
    int start2=add.size();//记录下次的新增可达点的起点
    for(int i = start ; i <add.size();++i){//start~add.size()-1 是上次的新增可达点
        int t=add[i];
        for(int j=0;j<p[t].size();++j){
            int k=p[t][j];
            if(vis[k]==0){
                vis[k]=1; //新增可达点在 vis[]里记录为 1 表示该点已经可达
                add.push_back(k);  //并将其保存到 add 里
                //visit(p,add,vis,start2);//深度优先
             }
        }
        //visit(p,add,vis,start2);//用该点的全部儿子作为整体的深度优先
    }
    visit(p,add,vis,start2);//广度优先
}
```

这里将起点 0 的所有可达点存放到 vector<int>&add 里，利用 start==add.size() 来判断是否有新增可达点，以及新增可达点在 add 里的位置为 start~add.size()-1.

且在不同的位置遍历新增可达点，可形成深度优先和广度优先两种图的遍历方法。

代码

```
#include<vector>
#include<iostream>
using namespace std;
void visit(vector<int> p[],vector<int>&add,int vis[],int start){
    if(start==add.size()) return;//无新增可达点时，跳出递归
    int start2=add.size();//记录下次的新增可达点的起点
    for(int i = start ; i <add.size();++i){
//start~add.size()-1 是上次的新增可达点
        int t=add[i];
        for(int j=0;j<p[t].size();++j){
            int k=p[t][j];
            if(vis[k]==0){
                vis[k]=1; //新增可达点在 vis[]里记录为 1 表示该点已经可达
                add.push_back(k);  //并将其保存到 add 里
                //visit(p,add,vis,start2);//深度优先
            }
        }
        //visit(p,add,vis,start2);//用该点的全部儿子作为整体的深度优先
    }
    visit(p,add,vis,start2);//广度优先
}
int main(){
    int n;
    while(cin>>n&&n!=0){
        vector<int> add;//存放当前所有可达点
        vector<int>p[n];//存放可达关系
        int m;
        cin>>m;
        for(int i = 0; i <m ;++i){//将可达关系读入 p 中
            int x,y;
            cin>>x>>y;
            p[x].push_back(y);
        }
        int vis[n]={0};
//用于存放 0 的可达点:vis[k]=1 表示 0 可达 k,vis[k]=0 表示 0 不可达 k
        vis[0]=1;//起点为 0
        add.push_back(0);//将起点设置为新增可达点
        visit(p,add,vis,0);
        if(vis[n-1]==1) cout<<"I can post the letter"<<endl;
        else cout<<"I can't post the letter"<<endl;
    }
}
```

2）Connect components in undirected graph

设计思路

由于是连通无向图，连通性具有对称性和传递性，由上一次实验的经验可知，我们可使用“并查集”来处理，所有连通的点组成一个连通块，且其具有相同的代表元，同时在并查集中代表元对应的值为-1，所以我们只需：

先将所有连通关系读入并查集中，再求出并查集中所有值为-1 的点的数目 sum（代表元的数目），则 sum 为所求的连通块的数目。

并查集类和上次基本相同：在类里新增两个变量：N 记录点的总数，sum 记录代表元的总数；新增一个求代表元总数函数，返回 sum。

代码

```
#include<iostream>
using namespace std;

class Dset{//并查集类
private:
    int *p;
    int sum; //代表元总数
    int N;  //点的总数
public:
    Dset(int n){
        p = new int[n+1];
        for(int i =0 ;i<n+1 ;++i) p[i]=-1;
        N=n;
        sum = 0;
    }
    ~Dset(){
        delete []this->p ;
    }
    int find(int x) {
        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;
        p[find(y)] = find(x);
```

```
    }
    int sum_of_father(){ //求代表元总数 sum 并返回 sum
        for(int i = 1; i<=N;++i){
            if(p[i]==-1) ++sum;
        }
        return sum;
    }
};
int main(){
    int n,m;
    cin>>n>>m;
    Dset f(n);//定义一个并查集类型变量 f
    for(int i = 0; i<=m;++i){
        int x,y;
        cin>>x>>y;
        f.union_(x,y);
    }
    cout<<f.sum_of_father();//输出代表元总数即连通块数目
    return 0;
}
```

4.程序运行与测试

1）Can I Post the letter

测试输入

```
7
1
1 6

9
4
0 3
3 2
2 4
4 8

9
4
0 3
3 2
```

```
4 2
4 8

8
3
0 4
2 7
4 2

7
5
0 4
2 5
3 5
5 6
4 2
```

输出

```
I can't post the letter
I can post the letter
I can't post the letter
I can post the letter
I can post the letter
```

2）Connect components in undirected graph

测试输入 1

```
7
1
1 6
```

输出 1：　　6

测试输入 2

```
9
4
1 3
3 5
2 4
4 8
```

输出 2：　　5

测试输入 3

```
23
12
1 13
23 5
12 4
4 8
8 6
9 3
2 17
21 3
22 15
16 17
1 18
19 11
```

输出 3：　　11

5.实验总结与心得

由于有向图的可达性只具备传递性，不具备对称性，所以不能使用并查集解决该类问题。

而无向图的连通性具备对称性和传递性，由上次实验的经验可知，可以用并查集解决该类问题。

附录、提交文件清单