

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:

```
1. vector<int> add; //存放当前所有可达点
2. vector<int> p[n]; //存放可达关系
3. int m;
4. cin>>m;
5. for(int i = 0; i < m ; ++i){ //将可达关系读入 p 中
6.     int x,y;
7.     cin>>x>>y;
8.     p[x].push_back(y);
9. }
10. 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
11. vis[0]=1; //起点为 0,已经可达
12. add.push_back(0); //将起点设置为新增可达点
```

搜索函数设计:

```
1. void visit(vector<int> p[],vector<int>&add,int vis[],int start){
2.     if(start==add.size()) return; //无新增可达点时,跳出递归
3.     int start2=add.size(); //记录下下的新增可达点的起点
4.     for(int i = start ; i < add.size(); ++i){ //start~add.size()-1 是上次的新增可达点
5.         int t=add[i];
6.         for(int j=0;j<p[t].size();++j){
7.             int k=p[t][j];
8.             if(vis[k]==0){
9.                 vis[k]=1; //新增可达点在 vis[]里记录为 1 表示该点已经可达
10.                add.push_back(k); //并将其保存到 add 里
11.                //visit(p,add,vis,start2); //深度优先
12.            }
13.        }
14.        //visit(p,add,vis,start2); //用该点的全部儿子作为整体的深度优先
15.    }
16.    visit(p,add,vis,start2); //广度优先
17. }
```

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

```
1. #include<vector>
2. #include<iostream>
3. using namespace std;
4. void visit(vector<int> p[],vector<int>&add,int vis[],int start){
5.     if(start==add.size()) return;//无新增可达点时，跳出递归
6.     int start2=add.size();//记录下下次的新增可达点的起点
7.     for(int i = start ; i <add.size();++i){
8.         //start~add.size()-1 是上次的新增可达点
9.         int t=add[i];
10.        for(int j=0;j<p[t].size();++j){
11.            int k=p[t][j];
12.            if(vis[k]==0){
13.                vis[k]=1; //新增可达点在 vis[]里记录为 1 表示该点已经可达
14.                add.push_back(k); //并将其保存到 add 里
15.                //visit(p,add,vis,start2);//深度优先
16.            }
17.        }
18.        //visit(p,add,vis,start2);//用该点的全部儿子作为整体的深度优先
19.    }
20.    visit(p,add,vis,start2);//广度优先
21. }
22. int main(){
23.     int n;
24.     while(cin>>n&&n!=0){
25.         vector<int> add;//存放当前所有可达点
26.         vector<int> p[n];//存放可达关系
27.         int m;
28.         cin>>m;
29.         for(int i = 0; i <m ;++i){//将可达关系读入 p 中
30.             int x,y;
31.             cin>>x>>y;
32.             p[x].push_back(y);
33.         }
34.         int vis[n]={0};
35.         //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
36.         vis[0]=1;//起点为 0
37.         add.push_back(0);//将起点设置为新增可达点
38.         visit(p,add,vis,0);
39.         if(vis[n-1]==1) cout<<"I can post the letter"<<endl;
40.         else cout<<"I can't post the letter"<<endl;
41.     }
42. }
```

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

```
1. #include<iostream>
2. using namespace std;
3.
4. class Dset{//并查集类
5. private:
6.     int *p;
7.     int sum; //代表元总数
8.     int N;  //点的总数
9. public:
10.     Dset(int n){
11.         p = new int[n+1];
12.         for(int i =0 ;i<n+1 ;++i) p[i]=-1;
13.         N=n;
14.         sum = 0;
15.     }
16.     ~Dset(){
17.         delete []this->p ;
18.     }
19.     int find(int x) {
20.         return (p[x] == -1) ? x : p[x] = find(p[x]); //实现了路径压缩
21.     }
22.     void union_(int x, int y) {
23.         if(find(y) == find(x)) return;
24.         p[find(y)] = find(x);
```

```

25.     }
26.     int sum_of_father(){ //求代表元总数 sum 并返回 sum
27.         for(int i = 1; i<=N;++i){
28.             if(p[i]==-1) ++sum;
29.         }
30.         return sum;
31.     }
32. };
33. int main(){
34.     int n,m;
35.     cin>>n>>m;
36.     Dset f(n); //定义一个并查集类型变量 f
37.     for(int i = 0; i<=m;++i){
38.         int x,y;
39.         cin>>x>>y;
40.         f.union_(x,y);
41.     }
42.     cout<<f.sum_of_father();//输出代表元总数即连通块数目
43.     return 0;
44. }

```

4.程序运行与测试

1) Can I Post the letter

测试输入

```

1. 7
2. 1
3. 1 6
4.
5. 9
6. 4
7. 0 3
8. 3 2
9. 2 4
10. 4 8
11.
12. 9
13. 4
14. 0 3
15. 3 2

```

```
16. 4 2
17. 4 8
18.
19. 8
20. 3
21. 0 4
22. 2 7
23. 4 2
24.
25. 7
26. 5
27. 0 4
28. 2 5
29. 3 5
30. 5 6
31. 4 2
```

输出

```
1. I can't post the letter
2. I can post the letter
3. I can't post the letter
4. I can post the letter
5. I can post the letter
```

2) Connect components in undirected graph

测试输入 1

```
1. 7
2. 1
3. 1 6
```

输出 1: 6

测试输入 2

```
1. 9
2. 4
3. 1 3
4. 3 5
5. 2 4
6. 4 8
```

输出 2: 5

测试输入 3

```
1. 23
2. 12
3. 1 13
4. 23 5
5. 12 4
6. 4 8
7. 8 6
8. 9 3
9. 2 17
10. 21 3
11. 22 15
12. 16 17
13. 1 18
14. 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