《数据结构与算法》实验报告

年级、专业、班级	21 计卓 2 班	姓名	文红兵
实验题目	图算法实践		
实验时间	2022.12.8	实验地点	竹园四栋
实验成绩		实验性质	☐验证性 √设计性 ☐综合性

教师评价：

☐算法/实验过程正确； ☐源程序/实验内容提交 ☐程序结构/实验步骤合理；

☐实验结果正确； ☐语法、语义正确； ☐报告规范；

其他：

评价教师签名：

实验目的

1. 掌握图的存储结构与基本操作
2. 训练使用经典的图算法，通过编程解决不同难度问题的实践能力

二、实验项目内容

实验课题 1：

题目内容：

7-1 生化危机

解题思路：

1、通过邻接表储存图

2、通过深度搜索 DFS 寻找出道路

3、输出答案

源代码：

报告创建时间：

```cpp
#include <bits/stdc++.h>
using namespace std;
#define maxM 101
int m,n,k;
int st,ta;

// 图的存储，利用c++的动态数组vector来构建特殊的邻接表
// vector[u][1...end] 表示 u->vector[u][i] 的边
vector<vector<int>> gr(maxM,vector<int>(1));

void read(){  // 读取数据，存储进入图中
    scanf("%d%d%d",&m,&n,&k);
    for(int i=1;i<=n;++i){
        int temp;scanf("%d",&temp);
        gr[temp][0] = 1;
    }
    for(int i=1;i<=k;++i){
        int s,e;scanf("%d%d",&s,&e);
        gr[s].push_back(e);
        gr[e].push_back(s);
    }
    scanf("%d%d",&st,&ta);
}
bool tag = false;
bool visited[maxM];
void DFS(int u){  // 深度搜索寻找出正确的道路
    if(tag==true) return;
    if(visited[u]==true || gr[u][0]==0) return;
    visited[u] = true;
    if(u==ta) {
        tag=true;return;
    }
    for(int i=1;i<gr[u].size();++i){
        DFS(gr[u][i]);
    }
}
void ans(){  // 调用深度搜索，输出答案
    if(gr[ta][0]==0){
        printf("The city %d is not safe!\n",ta);
    }else {
        for(int i=1;i<gr[st].size();++i){
            DFS(gr[st][i]);
        }
        if(tag==true){
            printf("The city %d can arrive safely!\n",ta);
        }else{
            printf("The city %d can not arrive safely!\n",ta);
        }
    }
}
int main(){
    read();
    ans();
    system("pause");
    return 0;
}
```

时间与空间复杂度分析：

时间复杂度是 O(n^2)

空间复杂度是 O(n^2)

实验课题 2：

题目内容：

7-2 大众情人

解题思路：

1、用邻接表储存图

2、用 floyd 算法求取全源最短路径，经过测试 Dijsktra 算法会超时，因此不能使用

3、通过求取的距离感求取异性缘

4、最后通过求取的异性缘找出大众情人

源代码：

```
#include <bits/stdc++.h>
using namespace std;
#define maxN 501
#define inf 0x3f3f3f3f
int n,dis[maxN][maxN];
double opSex[maxN];

struct edge{  // 图的存储，邻接表
    int end;
    int weight;
    edge* next;
    edge(int _end,int _weight):end(_end),weight(_weight),next(NULL){}
};
struct vex{
    char gender;
    edge* first;
    vex():first(NULL){};
} vexs[maxN];
void addedge(int s,int e,int w){
    edge* temp = vexs[s].first;
    vexs[s].first = new edge(e,w);
    vexs[s].first->next = temp;
}
```

```
25  void read(){  // 读取数据，储存进图
26      scanf("%d",&n);
27      for(int i=1;i<=n;++i){
28          getchar();  // 残留一个换行符，提取掉
29          char g;
30          int k;
31          scanf("%c %d",&g,&k);
32          vexs[i].gender = g;
33          for(int j=0;j<k;++j){
34              int e,w;
35              scanf("%d:%d",&e,&w);
36              addedge(i,e,w);
37          }
38      }
39  }
```

```
40  // 通过floyd算法求取距离感
41  // 至于选取floyd的原因，经过测试，n次调用Dijsktra最后一个测试点会超时
42  void floyd(){
43      for(int i=1;i<=n;++i){
44          for(int j=1;j<=n;++j){
45              dis[i][j] = INT_MAX;
46          }
47      }
48      for(int i=1;i<=n;++i){
49          for(edge* it = vexs[i].first;it!=NULL;it=it->next){
50              dis[i][it->end] = it->weight;
51          }
52      }
53      for(int k=1;k<=n;++k){
54          for(int i=1;i<=n;++i){
55              for(int j=1;j<=n;++j){
56                  if(dis[i][k]!=INT_MAX && dis[k][j]!=INT_MAX){
57                      dis[i][j] = min(dis[i][j],dis[i][k]+dis[k][j]);
58                  }
59              }
60          }
61      }
62  }
63
```

```
64  void OpSex(){  // 通过距离感求取异性缘
65      for(int i=1;i<=n;++i){
66          opSex[i] = inf;
67          for(int j=1;j<=n;++j){
68              if(vexs[j].gender == 'M' && vexs[i].gender == 'F')
69                  opSex[i] = min(opSex[i],(double)1/(double)dis[j][i]);
70              else if(vexs[j].gender == 'F' && vexs[i].gender == 'M')
71                  opSex[i] = min(opSex[i],(double)1/(double)dis[j][i]);
72          }
73      }
74  }
```

```cpp
void ans(){  // 通过异性缘寻找出大众情人，并且输出答案
    double maleMax=-inf,femaleMax=-inf;
    for(int i=1;i<=n;++i){
        if(vexs[i].gender=='F' && femaleMax < opSex[i]){
            femaleMax = opSex[i];
        }else if(vexs[i].gender=='M' && maleMax < opSex[i]){
            maleMax = opSex[i];
        }
    }
    vector<int> v;
    for(int i=1;i<=n;++i){
        if(vexs[i].gender=='F' && femaleMax == opSex[i]){
            v.push_back(i);
        }
    }
    for(int i=0;i<v.size();++i){
        if(i!=v.size()-1) cout<<v[i]<<" ";
        else cout<<v[i];
    }
    cout<<endl;
    vector<int> v1;
    for(int i=1;i<=n;++i){
        if(vexs[i].gender=='M' && maleMax == opSex[i]){
            v1.push_back(i);
        }
    }
    for(int i=0;i<v1.size();++i){
        if(i!=v1.size()-1) cout<<v1[i]<<" ";
        else cout<<v1[i];
    }
}
```

```cpp
int main() {
    read();
    floyd();
    OpSex();
    ans();
    system("pause");
    return 0;
}
```

时间与空间复杂度分析：

时间复杂度为 O(n^3)

空间复杂度为 O(n^2)

（3）**思考题**

对于无边权重的无向图，可以用广度优先遍历(BFS)算法求从起点到其它所有结点的最短路径，并且使用的 FIFO 队列的长度为 O(|V|),即结点的数量。而有权重的图，使用 Dijsktra 算法求最短路径，通常需要使用长度为 O(|E|)的优先队列。造成这一区别的主要原因是什么？如果要让 Dijsktra 算法也只使用 O(|V|)长度的优先队列，该如何处理？

答：

区别的原因：BFS 算法主要针对节点进行遍历，只有无权重才可以，而 Dijsktra 算法虽然也是针对节点，但是也因为需要储存边来找出最小值的原因，因此需要 O(|E|)的优先队列。

处理方法：如果只使用 O(|V|)长度的优先队列，可以先对**邻接表**的**每个顶点的相邻边**根据权重进行排序。每次入队时，一个顶点只进入未被使用且有效的权重最小的边，这样就可以实现 O(|V|)长度的优先队列。为了方便排序和遍历，邻接表可以用 c++ STL 库的 vector 实现。