苏 州 大 学 实 验 报 告

院、系	计算机学院	年级专业	19 计算机类	姓名	张昊	学号	1927405160
课程名称	数据结构课程实践					成绩	
指导教师	孔芳	同组实验者	无		实验日期	2021 年 1 月 4 日	

实 验 名 称 社交网络图

一、问题描述及要求

图-社交网络模型

设计并实现一个社交网络模型图。

要求：

（1）每个人的信息是一个顶点，两个人相互认识则构成边。

（2）根据输入的任意两个人的信息，给出他们之间的联系路径，最少经过多少人构成联系。

（3）可根据自己的创意添加更多的功能。

二、概要设计

1. 问题简析

本次实验内容是设计并实现一个社交网络模型图。实验需要在实现了图在计算机中的表示后，人抽象为顶点，两个人互相认识的关系抽象为边，并设计找到两个人互相联系的路径的算法。

2. 程序界面设计

程序启动，首先引导用户创建社交网络图，如下所示。

```
The number of people:5
The name of people:A B C D E
People's id is numbered from 0 to 4.
The number of A's friends:3
A's friends are (use id):1 2 3
The number of B's friends:2
B's friends are (use id):0 2
The number of C's friends:3
C's friends are (use id):0 1 4
The number of D's friends:1
D's friends are (use id):0
The number of E's friends:1
E's friends are (use id):2
```

之后会依次测试各功能，由用户输入测试组数。详见第 4 页运行结果。

3. 程序结构设计思路

程序将人抽象为图的顶点，两个人互相认识的关系抽象为图的边，依次构建社交网络图，并采用邻接矩阵的方式来存储。根据题目要求，并丰富社交网络图能执行的

工作，为社交网络图添加两个实用方法：找出给定两个人（用编号表示）之间的最短联系路径，其中包含了最少经过多少人构成联系的信息；找出给定的人可能认识的人。

将以上属性和方法包装为社交网络图类，在主函数中提供测试这个类的接口，用用户测试和使用。

三、详细设计

1 寻找最短联系路径

寻找最短联系路径的算法是基于 Dijkstra 算法来实现的。Dijkstra 算法是求给定源点到其余各点的最短路径，本算法中只需按照终点取其中一个路径即可。

Dijkstra 算法的基本思想是将顶点集合 V 分成两个集合，一类是生长点的集合 S，包括源点和已经确定最短路径的顶点；另一类是非生长点的集合 V-S，包括所有尚未确定最短路径的顶点，并使用一个待定路径表来存储当前从源点到每个非生长点的最短路径，按照路径长度的递增次序来产生最短路径。

在实现中，对邻接矩阵做了微调，用 inf 来表示两人不认识，即不存在边，用 1 比表示两人认识，即存在边。这样做的好处是后续稍作修改就可以表示两人之间的陌生程度，即两人之间需要互相直接联系的代价是多少。使用了辅助数组 dist[n]：元素 dist[i]表示当前所找到的从源点 v 到终点 vi 的最短路径长度。初态为:若从 v 到 vi 有边，则 dist[i]为边上的权值（这里都是 1）；否则置 dist[i]为 inf（定义为 0x3f3f3f）。同时也使用了辅助数组 path[n]：元素 path[i]是一个字符串，表示当前从源点 v 到终点 vi 的最短路径。

具体实现可见如下代码：

```
1.  std::string SocialGraph::short_path(int source, int destination) {
2.    if ((source >= vertex_num_ || source < 0)
3.        || (destination >= vertex_num_ || destination < 0)) {
4.      std::cerr << "out of range" << std::endl;
5.      exit(3);
6.    }
7.    int i, k, num, dist[MaxSize];
8.    std::string path[MaxSize];
9.    for (i = 0; i < vertex_num_; i++) {
10.     dist[i] = edge_[source][i];
11.     if (dist[i] == 1) {
12.       path[i] = vertex_[source] + " > " + vertex_[i];
13.     } else { path[i] = ""; }
14.   }
15.   for (num = 1; num < vertex_num_; num++) {
16.     k = min(dist, vertex_num_);
17.     for (i = 0; i < vertex_num_; i++)
18.       if (dist[i] > dist[k] + edge_[k][i]) {
19.         dist[i] = dist[k] + edge_[k][i];
20.         path[i] = path[k] + " > " + vertex_[i];
```

```
      }
    dist[k] = 0;
  }
  return path[destination];
}
```

2 寻找可能认识的人

找出给定的人可能认识的人的算法思路借鉴了 Floyd 算法以及离散数学中图论可达性矩阵的定义。邻接矩阵 A 的 n 次幂A^n表示了两结点之间通过 n 个结点可达的关系。这里定义为两个人如果彼此不认识，但是通过一个人就可以认识，即两人的最短联系路径长度为 2 时，认为两个人互为可能认识的人。由此，对邻接矩阵中的 inf 置为 0，并求其二次幂，找到结果中给定的人对应的一行中不为 0 的所有顶点对应的人，即为所求。

要注意的一点是，由于人与人之间的认识关系可能存在回路，需要在之前求出的结果中除去已经认识的人，即邻接点即可。

具体实现可以参考如下代码：

```
std::string SocialGraph::people_may_know(int people) {
  if (people >= vertex_num_ || people < 0) {
    std::cerr << "out of range" << std::endl;
    exit(3);
  }
  int edge_2[MaxSize][MaxSize]{};
  for (int i = 0; i < vertex_num_; i++) {
    for (int j = 0; j < vertex_num_; j++) {
      for (int k = 0; k <= vertex_num_; k++) {
        int x = 0, y = 0;
        if (edge_[i][k] == 1) { x = 1; }
        if (edge_[k][j] == 1) { y = 1; }
        edge_2[i][j] += x * y;
      }
    }
  }
  std::ostringstream out;
  for (int i = 0; i < vertex_num_; ++i) {
    if (people != i && edge_2[people][i] >= 1 && edge_[people][i] != 1) {
      out << vertex_[i] << ' ';
    }
  }
  return out.str();
}
```

四、实验结果测试与分析

使用一组数据，以手动输入的方式来测试算法的正确性。

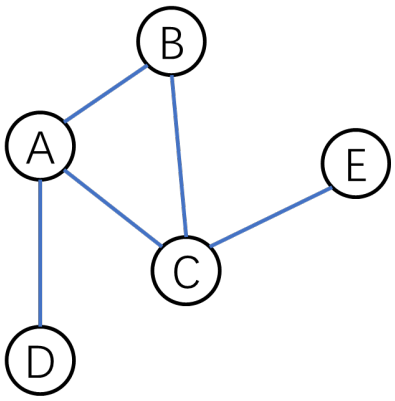
```
The number of people:5
The name of people:A B C D E
People's id is numbered from 0 to 4.
The number of A's friends:3
A's friends are (use id):1 2 3
The number of B's friends:2
B's friends are (use id):0 2
The number of C's
friends:3
C's friends are (use id):0 1 4
The number of D's friends:1
D's friends are (use id):0
The number of E's friends:1
E's friend
s are (use id):2

Shortest path Test
Number of tests:5
The id of person 1:0
The id of person 2:2
The shortest path between A and C is:A > C
The id of person 1:1
The id of person 2:4
The shortest path between B and E is:B > C > E
The id of person 1:3
The id of person 2:4
The shortest path between D and E is:D > A > C > E
The id of person 1:0
The id of person 2:4
The shortest path between A and E is:A > C > E
The id of person 1:1
The id of person 2:3
The shortest path between B and D is:B > A > D

People who may know Test
Number of tests:5
The id of person:0
Person A may know: E
The id of person:1
Person B may know: D E
The id of person:2
Person C may know: D
The id of person:3
Person D may know: B C
```

```
The id of person:4
Person E may know: A B
```

这里给出的社交网络图可以用下图表示，从图中可以看出，程序运行结果符合预期。



五、小结

通过这次实验，我复习了图的邻接矩阵存储方式，对求最短路径的算 Floyd 和 Dijkstra 法有了进一步的认识。

六、附录

1. **源代码路径**：C++源代码位于附件中 src 目录下。
2. **文件编码**：UTF-8；**行分隔符**：CRLF。
3. **实验环境**：Windows 操作系统，MSVC 编译器，基于 cmake 构建；在 CLion 集成开发环境中调试运行通过。手动编译运行方法为（在 Linux/Unix shell 中）：

```
$ mkdir build  # pwd: src/
$ cd build
$ cmake ..
$ make
$ ./social
```