**江苏大学京江学院**

**课程设计报告**

**设计题目: 城市交通调度系统**

**院 系: 电气信息工程学院**

**专业班级:**

**学 号:**

**学生姓名:**

**指导教师: 段学庆**

**2024年 1 月 9 日**

[**课程设计报告** 1](#_Toc121393208)

[一、问题描述 3](#_Toc121393209)

[二、需求分析 3](#_Toc121393210)

[三、总体设计 3](#_Toc121393211)

[四、详细设计 4](#_Toc121393212)

[五、上机操作 9](#_Toc121393213)

[六、总结 12](#_Toc121393214)

一、问题描述

定义城市和道路的数据结构，可以使用图的表示方法（邻接矩阵或邻接表）。

每个城市可以包括城市名称、坐标等信息，每条道路可以包括道路编号、起点城市、终点城市、距离等信息。

二、需求分析

**1.城市创建和连接**

* 允许用户创建新城市并连接城市，形成城市之间的道路网络。
* 用户可以指定城市之间的距离，表示两个城市之间的道路。

**2.最短路径查询**

* 实现最短路径算法，允许用户查询两个城市之间的最短路径和距离。
* 展示算法执行过程中节点的状态变化和最终的最短路径。

**3.城市扩展**

* 允许用户在已有城市之间添加新的道路，以扩展道路网络。
* 用户可以选择添加双向道路或单向道路。

**4.道路权重修改**

* 允许用户修改已有道路的权重，即距离。
* 系统应能够重新计算最短路径，反映道路权重的变化。

**5.城市删除**

允许用户删除城市及其相关的道路，以模拟城市的变化或关闭。

**6.数据统计**

实现功能，能够统计每个城市的连接数量、总距离等信息。

**7.数据持久化**

使用文件来保存城市和道路的结构，以便下次程序运行时可以重新加载数据。

**8.用户友好界面**

通过命令行交互，使用户能够方便地使用系统的各项功能。

**9.错误处理**

考虑用户输入错误、文件读写错误等情况，给予相应的提示和处理。

**10.注释与文档**

* 在代码中添加适当的注释，说明关键步骤和算法。
* 提供文档，解释数据结构的选择和系统的设计原理。

三、总体设计

采用一个string数组存储站点名称，然后下标是该站点对应的编号。采用邻接表来存储站点之间的花费，采用邻接矩阵来存储站点之间的距离。然后可以采用弗洛伊德算法来实现最优决策的查找，以及采用宽度优先搜索来实现中转次数最少的路线。

弗洛伊德算法思路:

**初始化**：定义初始距离矩阵D\_0和节点序列矩阵S\_0。对角线上的元素表示不需要从自身到自身的距离，即为0。这里的节点序列矩阵相当于路线表，S\_{ij}=j表示，从节点i到节点j只需经过节点j即可。

**迭代**：对于矩阵D\_{k-1}中的每一个元素做三重操作。如果满足条件：d\_{ik}+d\_{kj}<d\_{ij}，其中(i≠k,j≠k,i≠j)，则进行下面的操作：

用d\_{ik}+d\_{kj}代替矩阵D\_{k-1}中的元素d\_{ij}，从而得到矩阵D\_k。

用k代替矩阵S\_{k-1}中的元素s\_{ij}，从而得到矩阵S\_k。

令k=k+1，如果k=n+1，停止，否则重复此步骤。

四、详细设计

定义一个结构体来表示城市和道路的信息。结构体中包含城市名称、坐标、道路编号、起点和终点信息等字段。此外，还需要定义一个邻接表来表示道路的连接关系，每个节点包含一个指向下一个节点的指针和一个指向道路信息的指针。提供函数来创建新城市和连接城市。函数需要接受城市名称、坐标等信息作为参数，并在邻接表中添加相应的节点和边。同时，还需要计算两个城市之间的距离，并将其存储在道路信息中。实现最短路径算法来查询两个城市之间的最短路径和距离。在算法执行过程中，需要动态更新节点的状态并记录最短路径。可以使用优先队列来选择下一个要遍历的节点。提供函数来添加新的道路和修改已有道路的权重（距离）。函数需要接受起点、终点、道路权重等信息作为参数，并在邻接表中添加相应的节点和边。同时，需要更新最短路径算法的相关数据结构以确保正确性。提供函数来删除城市及其相关的道路。函数需要接受城市名称作为参数，并在邻接表中删除相应的节点和边。同时，需要更新最短路径算法的相关数据结构以确保正确性。实现功能来统计每个城市的连接数量、总距离等信息。可以通过遍历邻接表来实现统计功能。使用文件来保存城市和道路的结构以及最短路径算法的相关数据结构。在程序启动时读取文件数据并重新加载数据结构，在程序退出时将数据写入文件以实现持久化存储。

以下是详细代码：

int g[N][N];

用二维数组g来表示邻接图。

typedef struct node {

int name;

int val;

node\* next;

};

自定义一个结构体node来表示边表，其中name指的是站点的id，val指的是到站点的开销。

node \*list[N];

list是顶点表。

string zhandian[N]={ "a","b","c","d","e","f","f"};

默认有七个站点，可以根据自己的需要进行相对应的增加和减少。

int find(string str) {

for (int i = 0; i < cnt; i++) {

if (zhandian[i] == str) {

return i;

}

}

return -1;

}

Find函数实现了站点名字到站点编号的转换。

增加站点伪代码：

先输入站点名称

再输入所要增加边数

然后使用一个循环读取新增信息添加到图里面。其中邻接矩阵的数据增加采用数组赋值即可。而邻接表的增加需要新建结点并更新顶点表的第一个边表，这样可以节省增加的时间。

删除站点伪代码分析：

首先先读取删除的站点的名称

然后使用find函数转换成对应的编号。

然后遍历邻接矩阵的每一个点，只要跟该编号相关的点的信息都置为0.然后遍历每一个在邻接矩阵的结点，如果是name为该编号的结点全部释放。最后将该编号的顶点表里面的所有结点清空。

弗洛伊德算法伪代码：

距离数组dis赋值初始化，后继结点path也复制初始化。

枚举顶点k ∈ [1,n]

以顶点k为中间点，枚举所有顶点对i和j（i ∈ [1,n],j ∈1[1,n]）

如果dis[i][k] + dis[k][j] <dis[i][j]成立

赋值dis[i][j] = dis[i][k] + dis[k][j] 并且更新路径后继节点path[i][j]=k

宽度优先算法伪代码：

前驱结点数组初始化。

新建一个空队列，把起始点送进队列，前驱节点赋值。

While 队列不为空

取出队列的第一个元素cur，并将其出队。

一次遍历跟cur邻接的所有顶点，如果有没被访问过的顶点则将该顶点的前驱置为cur，且将该顶点进队。

继续while循环体。

弗洛伊德算法代码:

void foly1() {

    string z1, z2;

    cout << "请输入起始站跟终点站" << endl;

    cin >> z1 >> z2;

    int sta = find(z1);

    int des = find(z2);

    if (sta == -1) cout << "没有该起始站点" << endl;

    if (des == -1) cout << "没有该终点站" << endl;

    if (sta != -1 && des != -1) {

        int dis[N][N];

        int path[N][N];

        //初始化

        for (int i = 0; i < cnt; i++) {

            for (int j = 0; j < cnt; j++) {

                if (g[i][j] == 0)dis[i][j] = 999999;//设置一个较大的值

                else dis[i][j] = g[i][j];

                path[i][j] = j;

            }

        }

        for (int k = 0; k < cnt; k++) {

            for (int i = 0; i < cnt; i++) {

                for (int j = 0; j < cnt; j++) {

                    if (dis[i][k] + dis[k][j] < dis[i][j]) {

                        dis[i][j] = dis[i][k] + dis[k][j];

                        path[i][j] = path[i][k];

                    }

                }

            }

        }

        cout << zhandian[sta] << "->" << zhandian[des] << "最短距离为" << dis[sta][des] << endl;

        cout << "path: ";

        while (sta != des) {

            cout << zhandian[sta] << "->";

            sta = path[sta][des];

        }

        cout << zhandian[sta] << endl;

    }

}

void foly2() {

    string z1, z2;

    cout << "请输入起始站跟终点站" << endl;

    cin >> z1 >> z2;

    int sta = find(z1);

    int des = find(z2);

    if (sta == -1) cout << "没有该起始站点" << endl;

    if (des == -1) cout << "没有该终点站" << endl;

    if (sta != -1 && des != -1) {

        int dis[N][N];

        int path[N][N];

        for (int i = 0; i < cnt; i++) {

            for (int j = 0; j < cnt; j++) {

                dis[i][j] = 999999;//设置一个较大的值

                path[i][j] = j;

            }

        }

        node\* cur;

        for (int i = 0; i < cnt; i++) {

            cur = list[i];

            while (cur != NULL) {

                dis[i][cur->name] = cur->val;

                cur = cur->next;

            }

        }

        for (int k = 0; k < cnt; k++) {

            for (int i = 0; i < cnt; i++) {

                for (int j = 0; j < cnt; j++) {

                    if (dis[i][k] + dis[k][j] < dis[i][j]) {

                        dis[i][j] = dis[i][k] + dis[k][j];

                        path[i][j] = path[i][k];;

                    }

                }

            }

        }

        cout << zhandian[sta] << "->" << zhandian[des] << "最短花费为" << dis[sta][des] << endl;

        cout << "path: ";

        while (sta != des) {

            cout << zhandian[sta] << "->";

            sta = path[sta][des];

        }

        cout << zhandian[sta] << endl;

    }

}

通过佛洛依德算法可以计算出最短的路径.

广度优先算法

void bfs() {

    string z1, z2;

    cout << "请输入起始站跟终点站" << endl;

    cin >> z1 >> z2;

    int sta = find(z1);

    int des = find(z2);

    if (sta == -1) cout << "没有该起始站点" << endl;

    if (des == -1) cout << "没有该终点站" << endl;

    if (sta != -1 && des != -1) {

        queue<int> q;

        q.push(sta);

        int cur = 0;

        int pre[N];

        memset(pre, -1, sizeof(pre));

        pre[sta] = sta;

        while (!q.empty()) {

            cur = q.front();

            q.pop();

            for (int i = 0; i < N; i++) {

                if (g[cur][i] != 0 && pre[i] == -1) {

                    pre[i] = cur;

                    q.push(i);

                }

            }

        }

        int count = 0;

        int ans[N];

        //ans[count++] = sta;

        cur = des;

        while (cur != sta) {

            ans[count++] = cur;

            cur = pre[cur];

        }

        cout << zhandian[sta] << "->" << zhandian[des] << "最少中转次数为" << count - 1 << endl;

        cout << "path:" << zhandian[sta];

        for (int i = count - 1; i >= 0; i--) {

            cout << "->" << zhandian[ans[i]];

        }

        cout << endl;

    }

}

广度优先搜索是一种迭代的搜索算法，其主要思想是从起始节点开始，逐层遍历相邻节点，直到找到目标节点或遍历完整个图。这种搜索方式常用于以下情况：

最短路径问题：广度优先搜索可以用于寻找无权图中两个节点之间的最短路径。例如，在迷宫游戏中，我们可以使用广度优先搜索来找到从起点到终点的最短路径。网络分析：广度优先搜索可以用于分析社交网络或互联网中的关系。例如，寻找两个人之间的最短社交路径或确定网页之间的相关性。

生成树和图的连通性：广度优先搜索可以用于生成树的构建和判断图的连通性。例如，在无向图中，我们可以使用广度优先搜索来构建最小生成树或判断两个节点是否连通。

五、上机操作



图5-1 初始化站点

一开始先默认有七个站点，如上所示。Zhandian数组存储名称，cnt来存储站点个数。

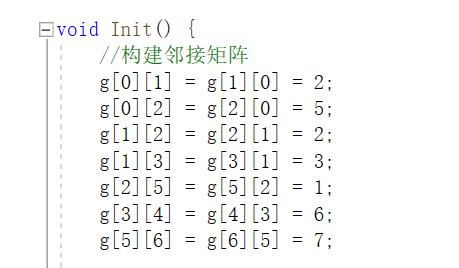


图5-2 构建邻接矩阵

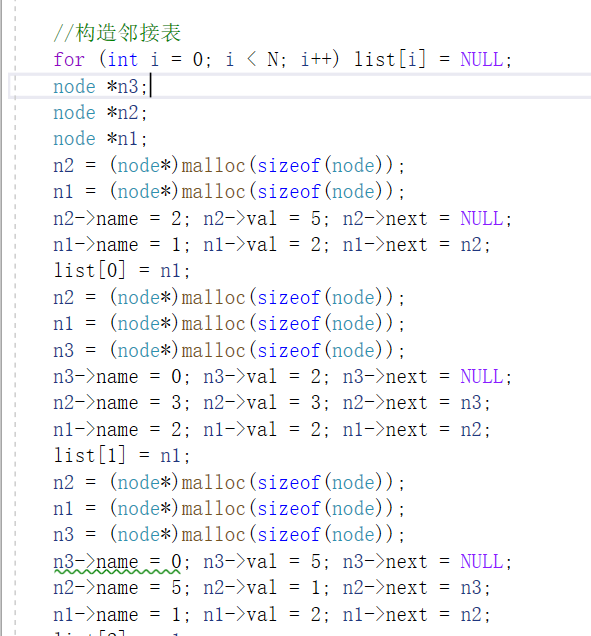


图5-3 构建邻接矩阵

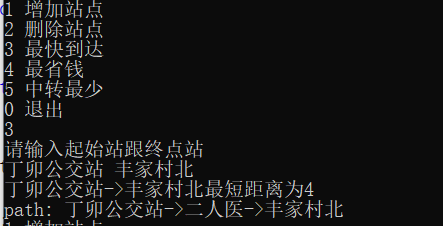


图5-4 最快到达

可以看到丁卯公交站是编号为0的点，丰家村北是编号为2的点。0->2的直接距离为5，可有邻接矩阵读取，0->1的距离为2，1->2的距离也为2,所以丁卯公交站经过二人医站到达丰家村北为距离最短。测试结果跟分析结果一致。

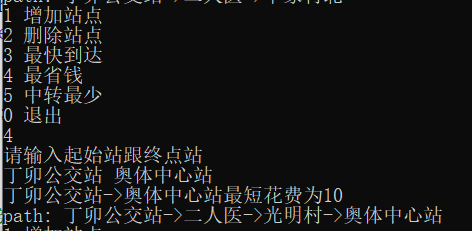


图5-5 最省钱

分析如上所示，数据可由邻接表读取。

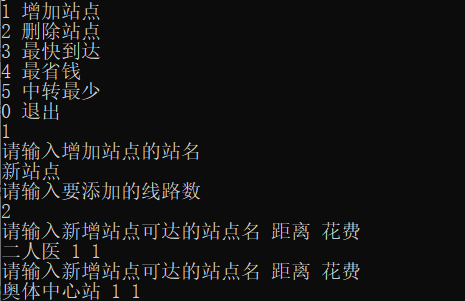


图5-6 增加站点

现在在二人医站跟奥体中心站直接增加一个新站点。

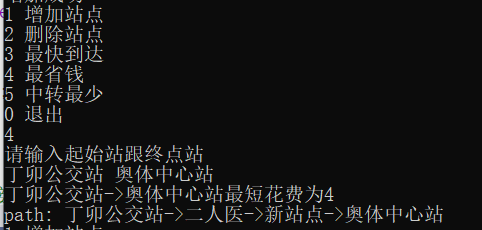


图5-7 最省钱

可以看到增加新站点之后的最省钱方案发生变化，跟预期一样。

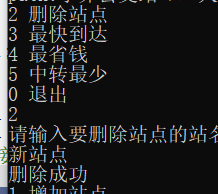


图5-8删除站点

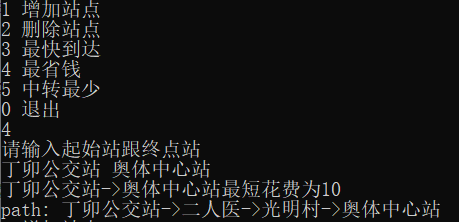


图5-9 最省钱

可以看到删除站点之后最省钱的方案变回一开始的方案，说明删除站点成功。

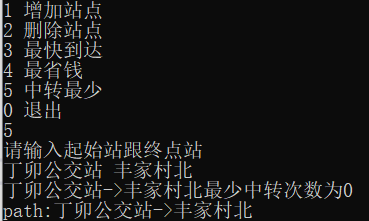


图5-10 中转最少

可以看到不需要中转即可到达，跟预期一致

六、总结及参考文献

本次课程设计主要围绕城市交通调度系统展开，涉及城市和道路的数据结构定义、最短路径算法实现、城市扩展、道路权重修改、城市删除等功能。采用邻接矩阵或邻接表表示城市和道路信息，使用弗洛伊德算法查找最优决策，宽度优先搜索实现中转次数最少的路线。同时，系统需具备数据持久化、用户友好界面、错误处理等功能。总体设计上，采用邻接表存储站点之间的花费，邻接矩阵存储站点之间的距离，并通过适当的注释和文档解释关键步骤和算法。用户可以创建新城市并连接城市，查询最短路径，扩展道路网络，修改道路权重，删除城市等。系统也具备数据统计和数据持久化功能，方便用户使用和保存数据。

由于上学期的C语言跟这学期的数据结构都算不上真正的懂，对于书上的稍微难点的知识就是是而非的，所以我只是对老师的程序理解，我也试着去改变了一些变量，自己也尽量多的去理解老师做程序的思路。当我第一天坐在那里的时候，我就不知道该做些什么，后来我只有下来自己看了一遍书来熟悉下以前学过的知识。

简而言之，这次的作业着实让我受益匪浅，不光光是在计算机应用能力上，更让我提高了解决问题和获取知识的能力。一事通则百事明，这次试验让我学到了知识，开阔了眼界，我会把这次试验的经历当做是我的大学生涯中重要的一件事，并以此为提醒，在以后的学习生活中，以实现自己的目标理想为目的，不断努力，让自己的父母更是让自己骄傲。

参考文献：《数据结构教程》清华大学出版社，CSDN，谭浩强《C语言程序设计》