|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据结构》实验报告** |
|  |
| 实验四  图型结构及其应用  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: | 苏亦凡 | | 学 号: | 200111229 | | 专 业: | 计算机科学与技术 | | 日 期: | 2021-04-27 | |

# 一、问题分析

第一部分：

1. 建图：需要设计图的储存结构，并考虑如何读取用户输入并按照输入构建图。
2. 判断图是否连通：可以转换为一个遍历问题：按照邻接关系遍历顶点能否一次遍历整个图的所有顶点。涉及到遍历的实现。
3. 计算图中每个顶点的度：即求取图中每个顶点相关联的边的条数。而由于此题限制两个顶点间最多只有一条边相连，且是无环图，所以可通过计算每个顶点相邻接的顶点数来计算度数。
4. 计算图的聚类系数：即为所有点聚类系数的平均值。求取顶点的聚类系数，需要了解相邻顶点与他们之间的连通程度。
5. 若图连通，使用Dijkstra算法计算单源最短路径。重点在于根据设计的图的存储结构实现算法。
6. 若图连通，计算图的直径（diameter），半径（radius）。重点在于求取任意两节点间最短路径，可采用Prim算法。

第二部分：

该部分与第一部分的重复率很高，只需要根据地铁的特点重新设计图的存储结构，根据输入文件设计文件读取的操作。其他函数基本可以使用第一部分中的现有的代码。

# 二、详细设计

## 2.1 设计思想

第一部分：

1. 建图：数据的输入形式为边的起点、终点与权重。由于是无向图，故起点与终点可互换，所以需要在邻接矩阵中赋值两次。
2. 判断图是否连通：为方便起见，对下标为0的顶点做BFS，同时用一visited数组记录各顶点的访问情况。若一次BFS后仍有节点未被访问，则图不连通，反之则连通。
3. 计算图中每个顶点的度。在无向简单图中，顶点的度等于与顶点邻接的顶点个数。扫描邻接矩阵，当不为对角线元素且值不为无限大时计数。最终求得的就是顶点的度。
4. 计算图的聚类系数。问题可拆开成两个子问题。
   1. 求一个顶点的聚类系数。创建一个线性存储结构储存所有与该顶点邻接的顶点并记录顶点数n。首先处理n=0的特殊情况，直接输出0。n0时，用双重循环遍历储存的顶点中所有两两组合的情况，当两个顶点邻接时计数，循环结束时计数为m。顶点的聚类系数为：，注意将分子或分母转换为double后进行运算。
   2. 求所有顶点聚类系数的平均值。
5. 若图连通，使用Dijkstra算法计算单源最短路径：算法的实现需要使用length与flag两个辅助数组，由于需要求取路径，还需要使用pre数组记录前驱元。算法的实现中最重要的是如何对length中的数组元素进行调整，使之成为到起点的最短路径和。最后输出路径时，若采取常规的方法需要对数组做一次逆序，若我们转换思路，利用无向图的特性，以end为起点，则得到的路径为正序，无需进行逆序。
6. 若图连通，计算图的直径（diameter），半径（radius）：采取Prim计算各对顶点间的最短路径长，再计算每个顶点到其他顶点的最大距离，从中选出最大的值，记为直径；选出最小的值，记为半径。

第二部分：

该部分与第一部分算法几乎一致。 对于第二个问题，考虑在图的数据结构中定义一个数组储存经过每一个顶点的线路数。

## 2.2 存储结构及操作

(1) 存储结构

第一部分：

typedef char vextype[20];

typedef struct

{

    int N, E;        //N是顶点数，E是边数

    int \*\*matrix;    //储存邻接矩阵

    vextype \*vertex; //存储节点的名字

} Graph;

第二部分：

typedef char vextype[20];

typedef struct

{

    int N, E, L;     //顶点数，边数，路线数

    int \*\*matrix;    //邻接矩阵

    vextype \*vertex; //节点名字

    int \*ID;         //节点ID(为了避免节点ID与节点下标不匹配)

    int \*num;        //记录有几条线路经过节点

} Graph;

(2) 涉及的操作

第一部分与第二部分共用

Graph createGraph(int n); //创建图

int isConnected(Graph g); //判断图是否连通

void nodeDegree(Graph g, int \*node\_degree); //求取顶点的度数

double clusteringCoefficient(Graph g); //求取图的聚类系数

void computeEcc(Graph g, int \*diameter, int \*radius); //求取图的半径与直径

int dijkstra(Graph g, int start, int end, int \*path); //求单源最短路径

void printPath(int d, int \*diameter\_path, Graph g); //打印路径

void DFS(Graph g, int v, int visited[]); //DFS

double CSofOneNode(Graph g, int v);      //单个顶点的聚类系数

void DestoryTree(Graph g); //清除树

## 2.3 程序整体流程（略）

# 三、用户手册

如：(1)输入数据的方式；(2)实现各种功能的操作方式等。

（1）输入数据的方式：

第一部分通过同级目录下的stu.in输入数据，输入格式为：

第一行为节点数，第二行为边数，接下去每行为起点下标、终点下标、权，用逗号隔开。

第二部分通过同级目录下的metro.txt与no2metro.txt输入数据，输入格式为：

no2metro.txt：（编码格式为GBK）

第一行为站的总数，接下去每行分别为站点ID与站名，用一个Tab隔开。

metro.txt：

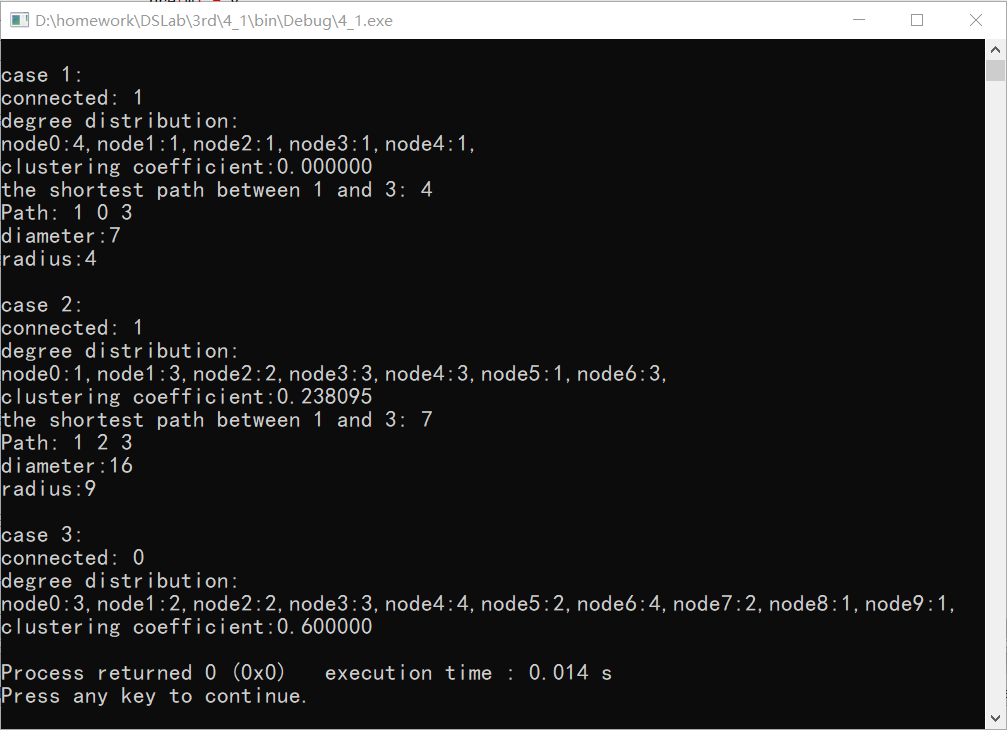
第一行为站的总数，第二行为线路的总数，第三行开始根据线路分组，每组第一行为当前线路所经过站的总数，接下去每行都是一个二元组，分别表示此站，上一站到此站的时间（规定起点站这个值为0），用一个Tab隔开。

实现各功能的操作方式：

下面描述实现各功能的方法：

1. 判断图是否连通。调用isConnected函数，传入参数为图，输出为int类型，若连通则输入1，若不连通则输出0.
2. 计算图每个顶点的度。调用nodeDegree函数，传入参数为图，以及用于接收每个顶点度的数组头指针。
3. 计算图的聚类系数。调用clusteringCoefficient函数，传入参数为图，返回值为double类型图的聚类系数。
4. 计算单源最短路径。调用dijkstra函数，传入参数为图，起点下标、终点下标以及用于接收路径的数组头指针，返回值为最短路径长。
5. 计算图的直径与半径。调用computeEcc函数，传入参数为图，用于接收直径值的指针，用于接收半径值的指针。

# 四、结果





# 五、总结

该实验涉及到的数据结构和算法，以及遇到的问题和收获。

该实验主要涉及的数据结构为图，辅助的数据结构主要为线性表。主要涉及的算法有DFS、Dijkstra算法、Prim算法。整体实验流程思路清晰，几乎没有遇到难点。

第一部分由于文件读取由模板给出，且文件没有出现中文，因此在运行的时候没有出现问题。但在第二部分的运行中，在一开始我的控制台输出站名的时候出现了乱码，最后发现原因是我将文件的字符集从GBK转换成了UTF-8，而windows的控制台默认只能输出GBK的中文字符。

此次实验加深了我对图的数据结构以及相关算法的理解，总体来说收获颇丰。