|  |
| --- |
| 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: | 宗晴 | | 学 号: | 200110513 | | 专 业: | 计算机科学与技术 | | 日 期: | 2021-05-01 | |

# 一、问题分析

题目1/（1）要解决的问题是构建无向图，判断该图是否连通，并计算每个结点的度。可以用邻接矩阵或者邻接表来构建无向图，此处使用了邻接矩阵。可以借助深度优先遍历来判断图是否连通，遍历整个邻接矩阵来计算每个结点的度。

题目1/（2）要解决的问题是计算该无向图的聚类系数。可以先求出单个结点的聚类系数，即所有与它相连的结点之间所连的边数（先用一个数组存储与她相连的结点，再遍历该数组，求出这些结点之间所连的边数），除以这些结点之间可以连出的最大边数。然后求出所有结点的聚类系数的均值，即整张图的聚类系数。

题目1/（3）要解决的问题是计算结点1到结点3 的单源最短路径。可以利用Dijkstra算法求解。用一个parent数组记录从结点1到每个结点的最短路径上，该结点的前驱结点。当找到到结点3的最短路径时，即可停止搜索，利用parent数组回溯寻找从结点1到结点3 的最短路径（反向）并逆序输出。

题目1/（4）要解决的问题是计算该图的直径和半径。可以利用Floyd算法求解。先利用Floyd算法求出每两个结点之间的最短路径长度，存储在一个二维数组中。接着遍历该数组，求出每个节点的离心率，即它到其他节点的节点距离的最大值，存储在一个一维数组中。最后求图直径，即图中所有节点的最大离心率；求图半径，即图中所有节点的最小离心率。

题目2要解决的问题是根据题目所提供的深圳地铁线路图自行建图，并判断该线路图是否连通；求出线路图中换乘线路最多的站点，并求出其换乘线路数；求出从大学城站到机场站所需的最少时间，并打印推荐路径上的站点名称；求出该线路图的直径和半径。这些问题均可利用问题1中的函数求解。用邻接矩阵构建该图，注意在建图时要设置一个标记数组记录每个站点作为端点站点的次数，因为求某个站点的换乘线路时，该站点可以作为端点站，需要加以区分。

# 二、详细设计

## 2.1 设计思想

使用邻接矩阵存储无向图。

1\_1、判断该图是否连通，并计算每个结点的度。

首先借助深度优先遍历来判断图是否连通。在函数中申请visit数组用来标记顶点是否已经被访问过，将visit数组中的值均置为0。从第0个顶点开始深度优先遍历该图。将该图、visit数组和当前顶点编号传入DFS函数。在DFS函数 中，将当前顶点标记为已访问，继续调用该DFS函数深度优先遍历当前顶点的相邻顶点中未被访问过的顶点。遍历结束后，检查visit数组中是否有顶点未被访问过，若有，则说明该图不连通，释放visit数组后返回0，否则说明连通，释放visit数组后返回1。

接着计算每个节点的度。将每个顶点的度初始化为0。遍历所有顶点，对于每个顶点，遍历一遍所有顶点，若顶点与当前顶点的距离不是无穷大（即两顶点相邻），并且它们是不同的顶点，那么就将该顶点的度加一。遍历完后即求得每个顶点的度。

1\_2、计算该无向图的聚类系数

首先，申请一个LCC数组用来存储每个顶点的聚类系数，一个neighbour数组用来存储与当前顶点相邻的顶点编号。定义变量counter用来记录与当前顶点相邻的顶点个数，pairs用来记录与当前顶点相邻的顶点之间所连的边数。

然后遍历所有顶点，对于每个顶点再遍历一遍所有顶点，求出与当前顶点相邻的顶点个数，并将相应的顶点编号存储在neighbour数组中。在该循环中，遍历neighbour中的顶点，对于每个顶点再遍历一遍neighbour数组，求与当前顶点相邻的顶点之间所连的边数，存入pairs中。考虑特殊情况，如果当前顶点的度为0或1时，聚类系数均为0，否则用公式求出该顶点的聚类系数（需要强制转换）。

最后将所有顶点的聚类系数求平均值，即得整张图的聚类系数。释放申请的内存后返回聚类系数的值。

1\_3、若图连通，计算结点1到结点3 的单源最短路径。

利用dijkstra算法实现。（增强健壮性）首先判断传入的始末顶点是否存在，若存在则继续，否则输出提示信息后直接退出函数。

定义变量fixed用来固定每轮循环确定的最小路径的顶点,fixed\_p存储上一轮循环固定的顶点,min\_len存储每轮循环求得的最短路径,num存储所求路径上的顶点个数。申请S数组作为源点集，值为1表示在源点集中，为0表示不在。申请len数组存储源点到各点的最短路径长度。申请parent数组存储最短路径中该顶点的前一个顶点，用来回溯以便找到路径。申请temp数组将路径上的顶点转移到path数组中（因为初始为逆序）。初始化数组及变量，将源点集中的值均置为0，len数组中的值均置为无穷大。将初始顶点放入源点集，并将它在len数组中对应的值置为0。

若已确定从始点到终点的路径，则循环结束，否则继续循环。遍历所有不在源点集中的顶点，若它与源点集中的顶点直接相连且存在更短的路径，则更新它的最短路径长度，由于此时产生变化的只可能是与上一轮固定的顶点相邻的顶点，因此将这一轮产生变化的顶点的前驱顶点置为上一轮确定的顶点。然后找出不在源点集中的顶点中，与始点的距最短的顶点，将该点固定。

通过parent数组回溯找到从始点到终点的最短路径，存入temp数组中（为逆序），并求出该路径上的顶点个数。将逆序的路径正向存储到path数组中。

释放申请的内存空间。

返回最短路径长度。

1\_4、若图连通，计算该图的直径和半径。

利用Floyd算法实现。申请eccentricity数组存储每个顶点的离心率，申请distance数组存储两个顶点之间的最短路径长度。

首先将原矩阵复制到最短路径长度矩阵中，然后依次添加可允许经过的顶点，更新最短路径长度。遍历完所有顶点后即可求得每两个顶点之间的最短路径长度。

对于每个顶点，遍历它所在行的distance数组，求出求每个顶点的离心率，即该节点到其他节点的节点距离的最大值，存入eccentricity数组中。遍历该数组，求得该图的直径（图中所有节点的最大离心率）和半径（图中所有节点的最小离心率）。

释放申请的内存，返回。

1. 根据题目所提供的深圳地铁线路图自行建图。可以用邻接矩阵构建该图，注意在建图时要设置一个标记数组记录每个站点作为端点站点的次数，用以后续区分。

2\_1、判断该线路图是否连通。利用1\_1题中的函数判断该线路图是否连通。

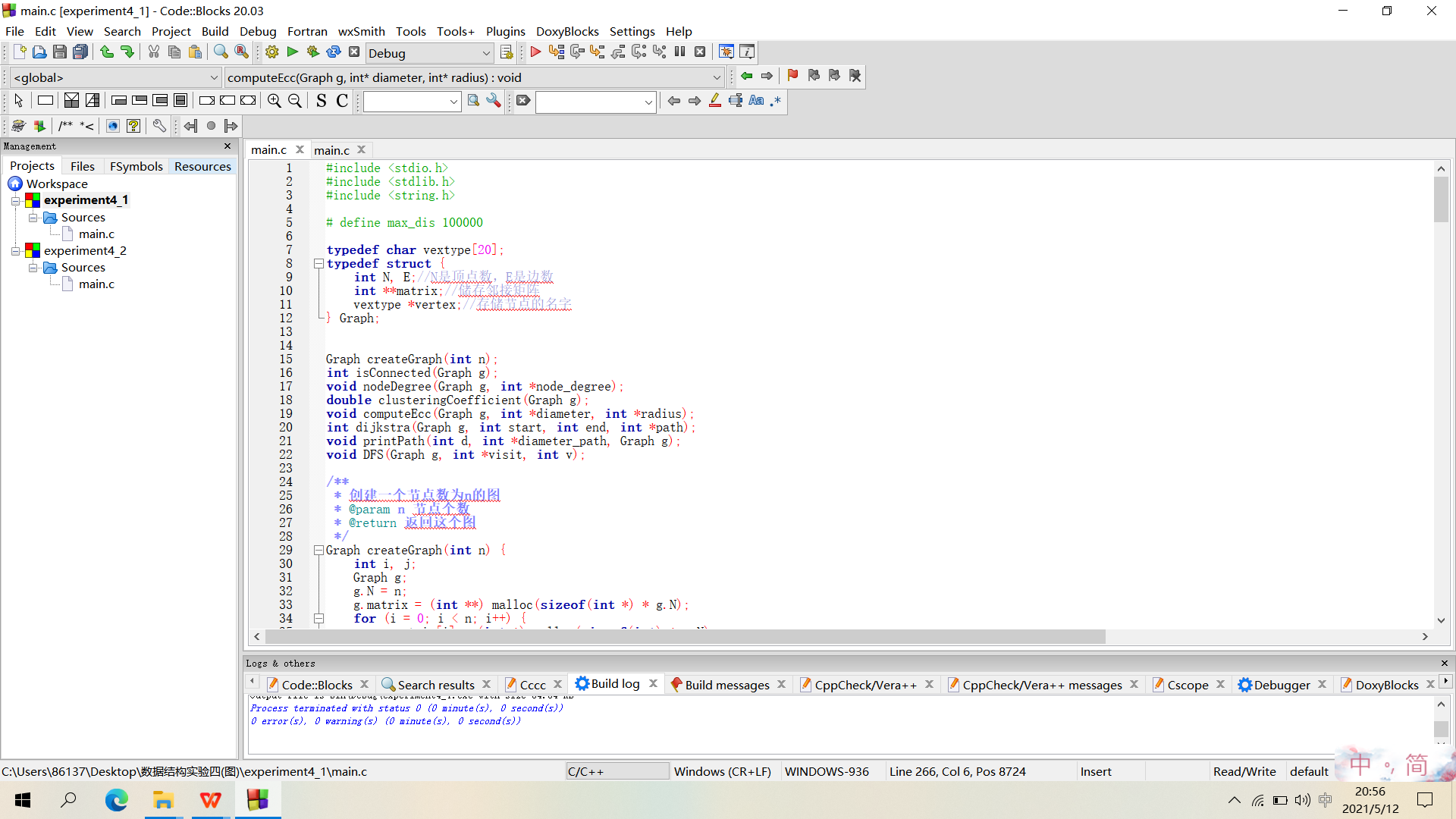
2\_2、求出线路图中换乘线路最多的站点，并求出其换乘线路数。利用1\_1题中的函数求出每个顶点的度，需要考虑站点为端点站点的情况，即加上每个站点作为端点站点的次数（在建图时已经记录），除以2就得到每个站点经过的线路数。寻找换乘线路数最大的顶点，并将该顶点返回。

2\_3、求出从大学城站到机场站所需的最少时间，并打印推荐路径上的站点名称。用图中的路径长度表示两站点之间的时间。首先找到分别表示大学城站和机场站的顶点编号，利用1\_3题中的函数求出这两个顶点之间的最短路径，即所需的最少时间，并打印出该路径上的顶点编号所对应的站点名称。

2\_4、求出该线路图的直径和半径。利用1\_4题中的函数求出该线路图的直径和半径。

## 2.2 存储结构及操作

(1) 存储结构（一般为自定义的数据类型，比如单链表，栈等。）



使用邻接矩阵存储图，其中无穷大定义为100000。

1. 涉及的操作（一般为自定义函数，可不写过程，但要注明该函数的含义。）

* Graph createGraph(int n);

参数：节点个数

功能：创建一个节点数为n的图

返回值：返回这个图

* int isConnected(Graph g);

参数：图

功能：判断图是否连通

返回值：连通返回1，否则返回0

* void nodeDegree(Graph g, int \*node\_degree);

参数：图、记录每个点的度的数组

功能：计算每个点的度

返回值：无返回值

* double clusteringCoefficient(Graph g);

参数：图

功能：计算图的聚类系数

返回值：返回聚类系数

* void computeEcc(Graph g, int \*diameter, int \*radius);

参数：图、指向直径变量的指针、指向半径变量的指针

功能：利用Floyd算法计算图的直径和半径

返回值：无返回值

* int dijkstra(Graph g, int start, int end, int \*path);

参数：图、起点、终点、存储从start到end的最短路径的数组

功能：使用dijkstra算法计算单源最短路径

返回值：返回路径长度

* void printPath(int d, int \*diameter\_path, Graph g);

参数：路径长度、储存路径的数组、图

功能：根据距离d和路径数组path输出路径

返回值：无返回值

* void DFS(Graph g, int \*visit, int v);

参数：图、标记已访问顶点的数组、当前顶点

功能：深度优先遍历

返回值：无返回值

* int maxLineDegree(Graph g, int \*line\_degree, int \*flag);

参数：图、记录每个点的度的数组、存储站点为端点站点的次数的数组

功能：计算每个点的换乘线路数，并找到换乘线路数最大的顶点

返回值：返回换乘线路数最大的顶点编号

## 2.3 程序整体流程

第一部分：



第二部分：



核心算法流程：

* int isConnected(Graph g);



* void DFS(Graph g, int \*visit, int v);



* void nodeDegree(Graph g, int \*node\_degree);



* double clusteringCoefficient(Graph g);



* int dijkstra(Graph g, int start, int end, int \*path);



* void computeEcc(Graph g, int \*diameter, int \*radius);



# 三、用户手册

输入数据的方式以及实现各种功能的操作方式：

1. 第一部分：对于每一张图，用户应该输入如下数据：第一行为一个整数，代表结点数；第二行为一个整数，代表边数；后续每一行为三个整数，用空格隔开，前两个整数为顶点编号，这两个顶点之间有一条边，第三个整数代表这条边的权重，一共有边数行数据。若要继续存下一张图的数据，则从下一行开始，按照上述规则输入。

对于每一张图，程序会在每一行依次输出如下内容：1、输出“case x:”，x代表这是第几张图。2、输出“connected: ”以及0或1,0代表该图非连通，1代表该图连通。3、输出“degree distribution:”。4、对每个顶点会输出“nodei:j,”其中i为顶点名称，j为顶点的度。5、接着程序会输出“clustering coefficient:”以及该无向图的聚类系数。6、然后程序会输出“the shortest path between 1 and 3: ”以及顶点1和顶点3之间的最短路径长度，并且在下一行输出“Path: ”以及该最短路径上的全部顶点。7、输出“diameter:”以及图直径，下一行输出“radius:”以及图半径。（如果该图是非连通图，则程序不会输出第6点和第7点）如果有下一张图的话，程序会空开一行，继续按照上面的方式输出下一张图的信息。

1. 第二部分：首先，用户在第一行输入所有站点个数，然后在接下来的每一行中输入站点编号以及对应的站点名称，两个数据之间用空格隔开，行数等于站点个数。程序会将站点名称与编号对应存储。

然后，用户在接下来的部分中，需要再次输入所有线路上的站点总数，在下一行输入线路总数。换行开始输入每一条线路的信息。对于每一条线路，需要先输入该条线路上的站点总数，在接下来的每一行中输入两个数据，前者代表站点编号，后者代表从上一站到这一站的时间，两者用空格隔开，行数等于该条线路上的站点总数，其中第一行的第二个数据为0（从起始站到该站点所需时间为0）。换行继续输入下一条线路的信息，直至线路数等于上述所输的线路总数。程序会将每两个站点之间所需的时间存储为该两顶点之间边的权重。同时，程序会记录站点是端点站点的次数。

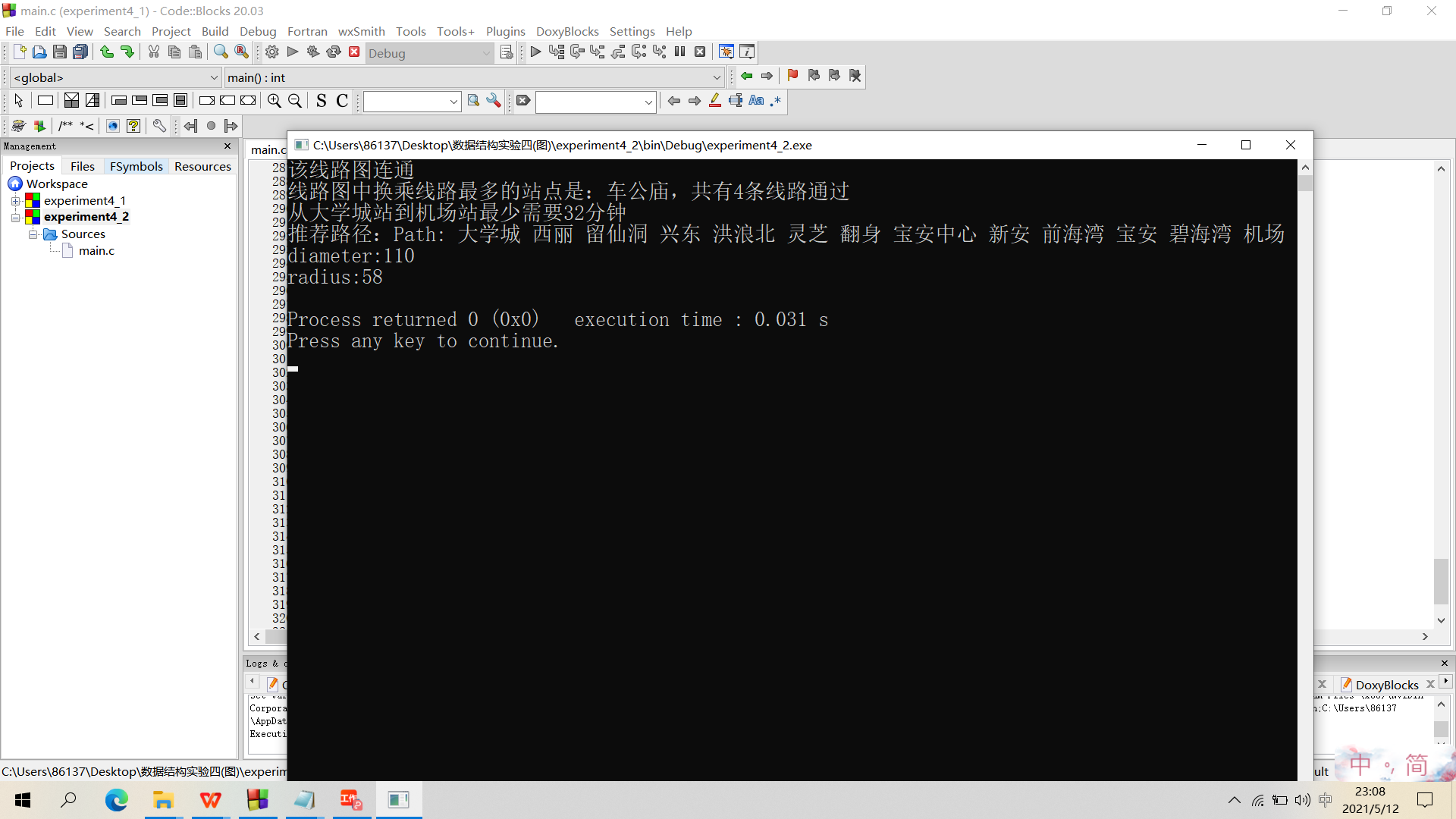
接着，程序会用第一部分中所写函数判断该图是否连通并输出相应提示信息。然后，程序会求出每个顶点的度，同时借助存储的端点站点次数信息求出每个站点经过的线路数，输出换乘线路数最大的顶点名称以及通过它的线路数。然后，程序会找到分别表示大学城站和机场站的顶点编号，利用上述函数求出这两个顶点之间的最短路径，即所需的最少时间，并打印出该路径上的顶点编号所对应的站点名称（即推荐路径）。最后，继续利用上述函数求出该图的直径和半径。

# 结果

第一部分：

# 

第二部分：



# 总结

该实验涉及到的数据结构是图。考察了图的存储、遍历、求顶点的度、利用Dijkstra算法求单源最短路径、利用Floyd算法计算图的直径和半径等算法。同时利用这些算法求解实际问题，即分析深圳地铁线路的实际情况。

在这次实验中，我加深了对图相关操作的理解与熟悉，代码能力有了很大的提升。同时，这是我第一次在实验课中接触到，将算法用于求解实际问题，这让我印象深刻，并使我深深感受到了利用计算机求解实际问题的便捷。这也使得我对后续的学习充满了兴趣。