|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据结构与算法》实验报告** |
|  |
| 实验四  图型结构及其应用  学 院: 机电工程与自动化   |  |  | | --- | --- | | 姓 名: | 宣元 | | 学 号: | 200320505 | | 专 业: | 自动化 | | 日 期: | 2022-05-19 | |

# 一、问题分析

任务一要求根据输入判断创建的图是否为连通图。任务二要求计算图中每个顶点的度，即计算与每个顶点相关联的边的条数。任务三要求计算图的聚类系数，即计算avg(所有与一个顶点相连的边的数量/这些顶点之间可以连出的最大边数)。任务四要求计算连通图的单源最短路径，且要求使用Dijkstra算法。任务五要求计算连通图的直径和半径，即最大的eccentricity和最小的eccentricity

# 、详细设计

## 2.1 设计思想

任务一要求判断图是否连通，可以从图的任意顶点出发进行深度优先或广度优先遍历，如果能遍历到所有结点，则说明图是连通图，否则图不连通。

任务二要求计算每个顶点的度，题中图的存储形式为邻接矩阵，计算每个顶点的度时只需要遍历该顶点对应的行（列），统计权重值小于MAX\_DIS（规定的最大距离，相当于无穷大）的顶点个数N，N-1即为该顶点的度（减去自己）。

任务三要求计算图的聚类系数，问题可以分解为计算每个顶点的聚类系数。首先遍历邻接矩阵的一行找到顶点的所有邻接点，然后枚举每一对邻接点（因为是无向图，先后顺序不同视为同一条边），就可以得到所有邻接点之间的边数，而n个邻接点最多能形成n(n-1)/2条边，由此可以计算出每个点的聚类系数。

任务四要求运用Dijkstra计算单源最短路径，运用传统的Dijkstra算法就可以解决问题。

任务五要求计算图的直径和半径。首先用Floyd算法求出每一对顶点之间的最短距离，然后对于某一个顶点，遍历它与所有其他顶点的最短距离，这个最短距离的最大值就是该顶点的eccentricity。图中最大和最小的eccentricity值即为图的直径和半径值。

## 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。）

本实验中采用的存储结构为图(Graph)和队列(STL)。

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。）

队列：

Queue.pop() 队头出队

Queue.push() 新成员入队

图：

int firstNeighbour(int x); 找到指定顶点的第一个邻接点

int nextNeighbour(int x, int y); 找到指定顶点除了y外的第一个邻接点

int isConnected(); 判断图是否连通，是返回1，否则返回0

void nodeDegree(vector<int> &node\_degree); 把图顶点的度信息写入node\_degree数组

double clusteringCoefficient(); 计算图的聚类系数

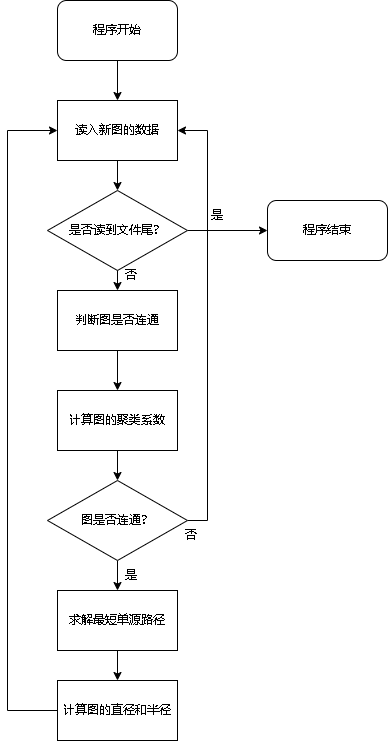
int dijkstra(int start, int end, vector<int> &path); 用Dijkstra算法计算单源最短路径

int Diameter(); 计算图的直径

int Radius(); 计算图的半径

void printPath(vector<int> diameter\_path); 根据path数组打印路径

## 2.3 程序整体流程

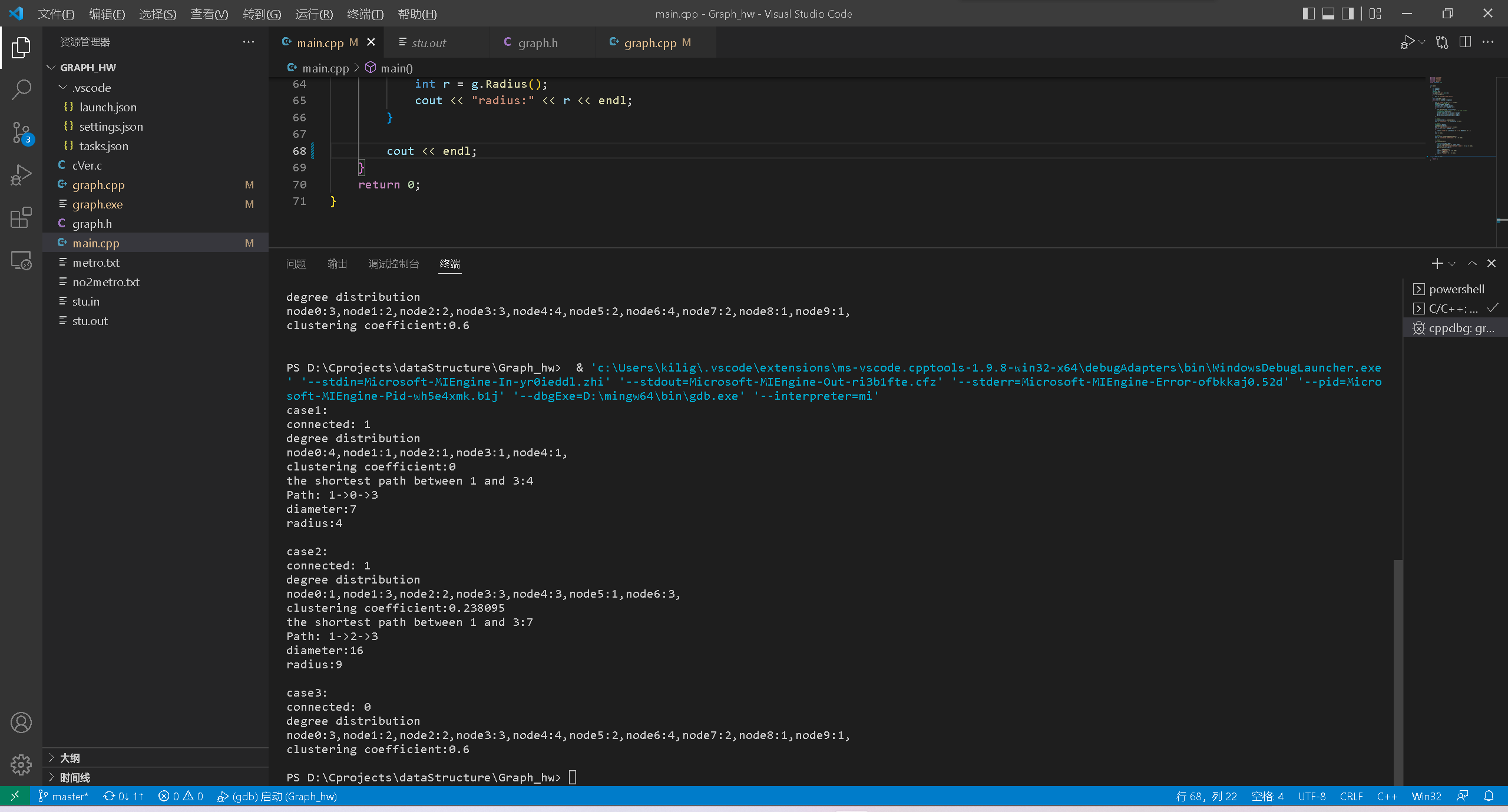


# 三、用户手册

1. 输入数据的方式：程序从文本文件中读取数据。数据格式如下：对于每张图，先在前两行分别输入图的顶点数和边数，之后按照（起点 终点 权重）的方式添加边，一行只输入一条边的数据。
2. 操作方式：创建图，调用其不同的成员函数即可以实现不同的功能。

# 四、结果

程序正确运行的结果截图。



# 五、总结

该实验涉及到了图形结构的基本操作，总体来说贴近课本的教学内容，实现功能较为简单，但用C++从头写代码还是有些繁琐，同时也让我学习了C++文件流读取文件的方法。