安徽理工大學

《数据结构》课程设计说明书

题目: 城市通信网络的线路设计

学　　院　：　计算机科学与工程学院

专 业 班 级　：　计算机 22-4 班

学　　号　：　2022304335

学 生 姓 名　：　费晨馨

指 导 老 师　：　汤文兵

2023 年 11 月 30 日

数据结构课程设计执行计划书

课程设计名称	《数据结构》课程设计		
课程设计题目	城市通信网络的线路设计		
学期	2023-2024（1）	专业班级	计算机 2022-4 班
课程设计目的和要求	数据结构是为了将实际问题中所涉及的对象在计算机中表示出来并对它们进行处理。通过课程设计可以提高学生的思维能力，促进学生的综合应用能力和专业素质的提高。通过此次课程设计主要达到以下目的： ● 了解并掌握数据结构与算法的设计方法，具备初步的独立分析和设计能力； ● 初步掌握软件开发过程的问题分析、系统设计、程序编码、测试等基本方法和技能； ● 提高综合运用所学的理论知识和方法独立分析和解决问题的能力； 训练用系统的观点和软件开发一般规范进行软件开发，培养软件工作者所应具备的科学的工作方法和作风。		
课程设计内容	在选定一个数据结构课程设计的题目后，依据需求进行数据结构设计、数据操作设计、实现、其他功能模块实现，具体包括： 1、数据结构的逻辑分析和物理设计、实现 2、基于此物理数据结构的实现，设计实现相关操作（例如，查找、插入、删除、排序、统计分析等等） 3、按照课程设计题目具体要求，完成相关的逻辑功能和业务逻辑操作的实现 4、调试程序 5、撰写课程设计报告，包括分析、设计、实现和总结等部分 6、可视化程序实现（可选）		

<table>
<tr><td>课程设计的组织</td><td>（包括分组、负责教师名称、专业班级、学生人数、方式、起止时间）：
分组情况：费晨馨、于丁宇、鲍家伟、张盛泽、李宏金阳；
指导教师：汤文兵
专业班级：计算机 2022-4 班；
学生人数：5-6 人
方　　式：实践；
起止时间：14-18 周</td></tr>
<tr><td>课程设计进度安排</td><td><table><tr><td>名称</td><td>学时</td></tr><tr><td>课程设计选题</td><td>1</td></tr><tr><td>课程设计需求分析</td><td>2</td></tr><tr><td>课题总体设计</td><td>3</td></tr><tr><td>系统详细设计</td><td>5</td></tr><tr><td>系统模块设计、编码与调试</td><td>3</td></tr><tr><td>系统整合、运行与测试</td><td>1</td></tr><tr><td>编写课程设计说明书</td><td>4</td></tr><tr><td>课程设计效果演示与验收</td><td>1</td></tr></table></td></tr>
<tr><td>系主任审核意见</td><td>同意执行计划！
系主任（签章）：蒋社想</td></tr>
<tr><td>学院审核意见</td><td>同意！
教学院长（签章）：徐辉</td></tr>
</table>

安徽理工大学课程设计成绩评定表

学生姓名	费晨馨	**学号**	2022304335	**专业班级**	计算机 2022-4 班
课程设计题目	城市通信网络的线路设计				

指导老师评语：

成绩 _____

指导老师：

年　月　日

目　　录

1 问题分析及任务描述

1.1 问题分析

城市通信网络的线路设计：

（1）在 n 个城市之间建设通信网络，建立最小生成树；

（2）城市间的距离网采用邻接矩阵表示，也可用邻接表表示；

（3）分别用普里姆算法和克鲁斯卡尔算法求最小生成树；

（4）最小生成树中包括边及其权值，并显示得到的最小生成树的权值。

欲在 n 个城市间建立通信网络，则 n 个城市应铺 n-1 条线路；每条线路都会有对应的距离，n 个城市可能有 n(n-1)/2 条线路，要选择 n-1 条线路使通信网络的距离最短，即要求城市图的最小生成树，计算出其权值。

1.2 任务描述

在 n 个城市之间建设通讯网络，要求建立最小生成树。通讯网络的线路设计需要考虑城市间的距离，这可以通过邻接矩阵或邻接表进行表示。为了找到最优解，要求使用普里姆算法和克鲁斯卡尔算法分别求解最小生成树，并在结果中包括边及其权值。最终，设计需要显示得到的最小生成树的代价。任务分工及目标如下：

分工：

城市通讯网络设计： 建立通讯网络，确保所有城市之间都有通信线路，并采用最小生成树的方法实现。

距离表示： 考虑城市间的距离，可以选择邻接矩阵或邻接表的方式来表示城市之间的连接关系。

算法选择： 使用普里姆算法和克鲁斯卡尔算法两种不同的最小生成树算法来解决问题。

结果展示： 最终结果要包括最小生成树中的边及其权值，以及整个最小生成树的代价。

目标：

1. 确定城市通讯网络的整体结构，选择邻接矩阵或邻接表的表示方式。
2. 实现普里姆算法，确保其在城市通讯网络设计中的有效应用。
3. 实现克鲁斯卡尔算法，确保其在城市通讯网络设计中的有效应用。

4. 设计和实现一个能够显示最小生成树及其代价的模块。

5. 成功建立城市通讯网络：确保通过最小生成树算法建立了一个完整的城市通讯网络。

6. 正确实现两种算法： 确保普里姆算法和克鲁斯卡尔算法都正确实现，并能够在设计中正确应用。

7. 结果可视化： 设计一个能够清晰展示最小生成树的边、权值和总代价的结果显示模块。

8. 算法效率： 考虑算法的时间和空间复杂度，确保设计的算法在实际使用中具有合理的效率。

通过以上分工和目标，这个设计旨在解决城市通讯网络线路设计问题，要求采用两种不同的最小生成树算法，并提供清晰的结果展示。

2 概要设计

2.1 基本思路

在数据结构课上，我学习了图的邻接矩阵表示法以及最小生成树的普里姆算法和克鲁斯卡尔算法等。在实验课上，我进行了相关的练习，加深了对这些概念和算法的理解。

我将在 VSCode 实验环境下，将一个 C++源文件改为进行 C++多文件编译、运行、测试，来完成我的课设。在同一项目文件夹中创建多个文件，main.cpp 主函数，AMGraph.h 存放 AMGraph 的类及其他数据结构，graph.cpp 存放 AMGraph 的成员函数等每个源文件，整个程序代码将实现图的相关操作和算法。

在 VSCode 中，我将在项目文件夹中打开一个终端窗口，并使用以下命令来编译、链接、运行我的项目。在编译和运行过程中，我将仔细检查每个函数的输出结果，以确保它们与预期结果一致。同时，我也将使用调试器来检查代码中的错误和异常情况。

我还学会利用网络进行对最小生成树的进一步深层次学习，尝试优化，利用贪心算法、dijkstra 算法、并查集、vector 容器等来实现对源代码的优化。

2.2 数据类型设计

邻接矩阵

1. 邻接矩阵表示法

在邻接矩阵中，图的连接关系用矩阵表示。如果图有 n 个顶点，那么矩阵 matrix 的大小就是 n×n。对于无向图，矩阵是对称的；对于有向图，矩阵可以是非对称的。

```
const int MAX_VERTICES = 100;
typedef struct {
    int vertices[MAX_VERTICES];
    int matrix[MAX_VERTICES][MAX_VERTICES];
    int numVertices;
    int numEdges;
} AMGraph;
```

2. 邻接表表示法

在邻接表中，每个顶点都有一个相邻顶点的链表。这种表示法更加节省空间，特别是对于稀疏图。

```
const int MAX_VERTICES = 100;
typedef struct Node {
    int vertex;
    int weight;
    struct Node* next;
} Node;
typedef struct {
    int numVertices;
    Node* adjLists[MAX_VERTICES];
} ALGraph;
```

在最小生成树算法中，还需要额外的数据结构来存储边的信息。一个常用的方式是使用结构体来表示边：

```
typedef struct {
    int startVertex;
    int endVertex;
    int weight;
} Edge;
```

本实验采取邻接矩阵法进行对最小生成树的求解。

```
//节点结构体
struct Vex {
    string code;
```

```
    string Name;
};
//边结构体
struct Edge {
    Vex vex1;
    Vex vex2;
    int weight;
};
//图 类
class Graph {
    private:
    int AdjMatrix[numMAX][numMAX];
    SqList<Vex>Vexs;
    SqList<Edge>Edges;
    int VexNum;
}
```

2.3 功能模块设计

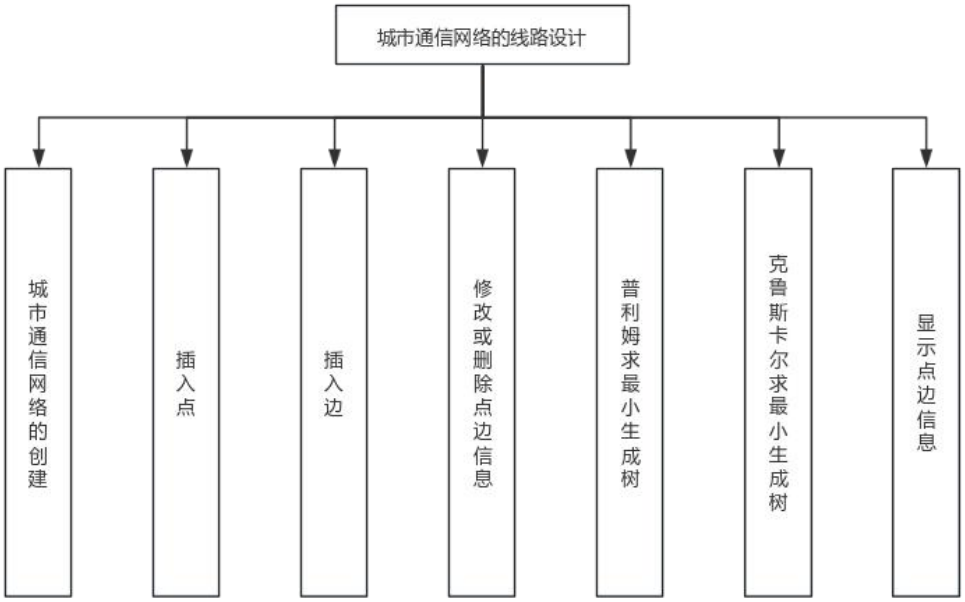

图 2-1 功能模块设计图

功能模块设计图如图 2-1 所示，模块划分为城市通信网络的创建、插入点和插入边、修改或删除点边信息、普利姆求最小生成树、克鲁斯卡尔求最小生成树和显示点边信息。

3 详细设计

3.1 函数调用关系

函数调用关系如图 3-1 所示，主函数通过菜单选项读取用户指令方式调用各子函数。

函数之间的调用关系主要分析图类（Graph）的成员函数之间的相互调用关系。以下是对函数调用关系的分析：

1.构造函数 Graph::Graph 调用了 SqList 的构造函数。

2. 插入顶点函数 Graph::InsertVex 调用了 SqList::GetLength、SqList::GetElem、SqList::InsertElem。

3. 删除顶点函数 Graph::DeleteVex 调用了 SqList::GetLength、SqList::GetElem、SqList::DeleteElem、Graph::isConnect。

4. 更新顶点函数 Graph::UpdateVex 调用了 SqList::GetLength、SqList::GetElem、SqList::SetElem。

5. 插入边函数 Graph::InsertEdges 调用了 qList::GetLength、SqList::GetElem、SqList::InsertElem

6. 删除边函数 Graph::DeleteEdges 调用了 SqList::GetLength、SqList::GetElem、SqList::DeleteElem、Graph::isConnect。

7. 更新边函数 Graph::UpdateEdges 调用了 SqList::GetLength、SqList::GetElem、SqList::SetElem。

9.Kruskal 最小生成树函数 Graph::KruskalMinTree 调用了 Graph::isConnect。

10. 判断两个顶点是否连通函数 Graph::isConnect 调用了 SqList::GetLength、SqList::GetElem。

11. 显示图信息函数 Graph::showMassage 调用 SqList::GetLength、SqList::GetElem。

这些函数之间的调用关系主要涉及对顶点和边的操作，其中 SqList 类用于存储顶点和边的信息。在一些操作中，需要判断两个顶点是否连通，这种情况下会调用 Graph::isConnect 函数。函数调用关系相对清晰，主要集中在对顶点和边的

操作上。

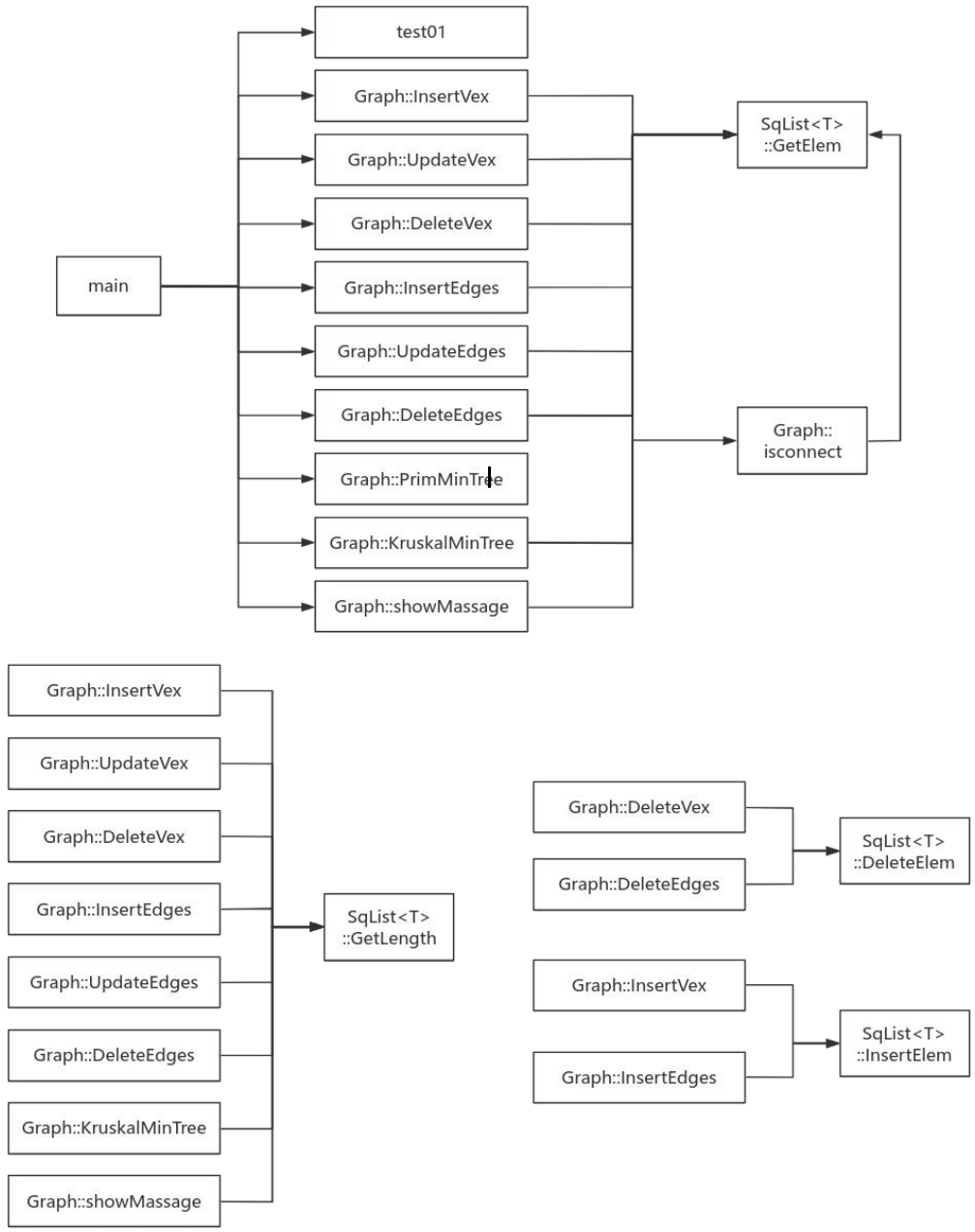

图 3-1 函数调用关系图

3.1 PrimMinTree 函数

函数功能：PrimMinTree 函数是用 Prim 算法实现图的最小生成树。该算法从一个起始节点出发，逐步选择与当前生成树相连的最小权重边连接的节点，直到所有节点都包含在生成树中。

返回值：函数返回一个整数，通常用于表示函数执行成功与否，这里返回值为 0。

入口参数：无，但函数使用了 Graph 类中的成员变量和邻接矩阵信息。

出口参数：通过参数 Edge aPath[] 返回最小生成树的边信息。

算法思想：

从第一个点出发，初始化最短边数组 lowcost[] 和链接顶点数组 closest[]。

通过循环，逐步选择与当前生成树相连的最小权重边所连接的节点，并更新 lowcost[] 和 closest[]。

循环结束后，输出生成树的边信息，即节点之间的连接关系。

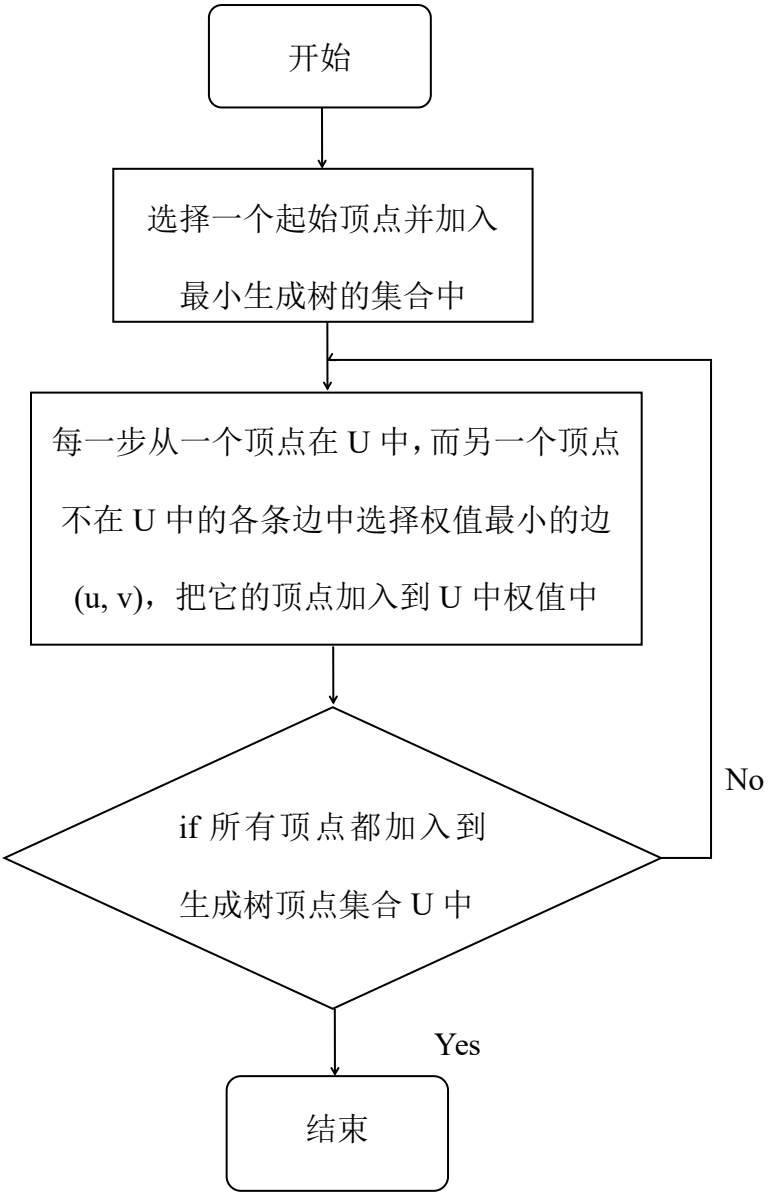

图 3-2 PrimMinTree 函数流程图

3.2 KruskalMinTree 函数

函数功能：KruskalMinTree 函数是用 Kruskal 算法实现图的最小生成树。该算法通过对所有边按权值排序，然后逐个选择最小权值的边，确保该边不与已选择的边构成环，直到选择了足够数量的边构成最小生成树。

返回值：无。入口参数：无，但函数使用了 Graph 类中的成员变量和邻接矩阵信息。出口参数：通过参数 Edge aPath[] 返回最小生成树的边信息。

算法思想：从图中取出所有边，按照权值从小到大进行排序。初始化一个边数组 outEdges 和一个变量 outNumber 用于记录最终的生成树边和数量。遍历排序后的边，依次选择最小权值的边，检查是否与已选的边构成环路。如果不构成环路，则将该边加入 outEdges，并更新相关信息。遍历完成后，输出最小生成树的边信息。sort 函数对 Kruskal 算法中边数组进行排序的实现方式有三种：

```
const int MAXM = 1e5 + 7; // 图中边的最大数量
struct Edge
{
    int u, v, w; // 边的起点、终点、权值
} edges[MAXM]; // 存储图中所有边
```

第一种，比较函数，按边权值从小到大排序：

```
bool cmp(const Edge &a, const Edge &b)
{
    return a.w < b.w;
}
```

第二种，操作符重载：

```
struct Edge
{
    int u, v, w; // 边的起点、终点、权值
    bool operator<(const Edge& other) const { return w < other.w; }
} edges[MAXM]; // 存储图中所有边
sort(edges, edges + m);
```

第三种，使用匿名 lambda 函数如下：

```
sort(edges, edges + m, [](const Edge &a, const Edge &b) {
    return a.w < b.w;
});
```

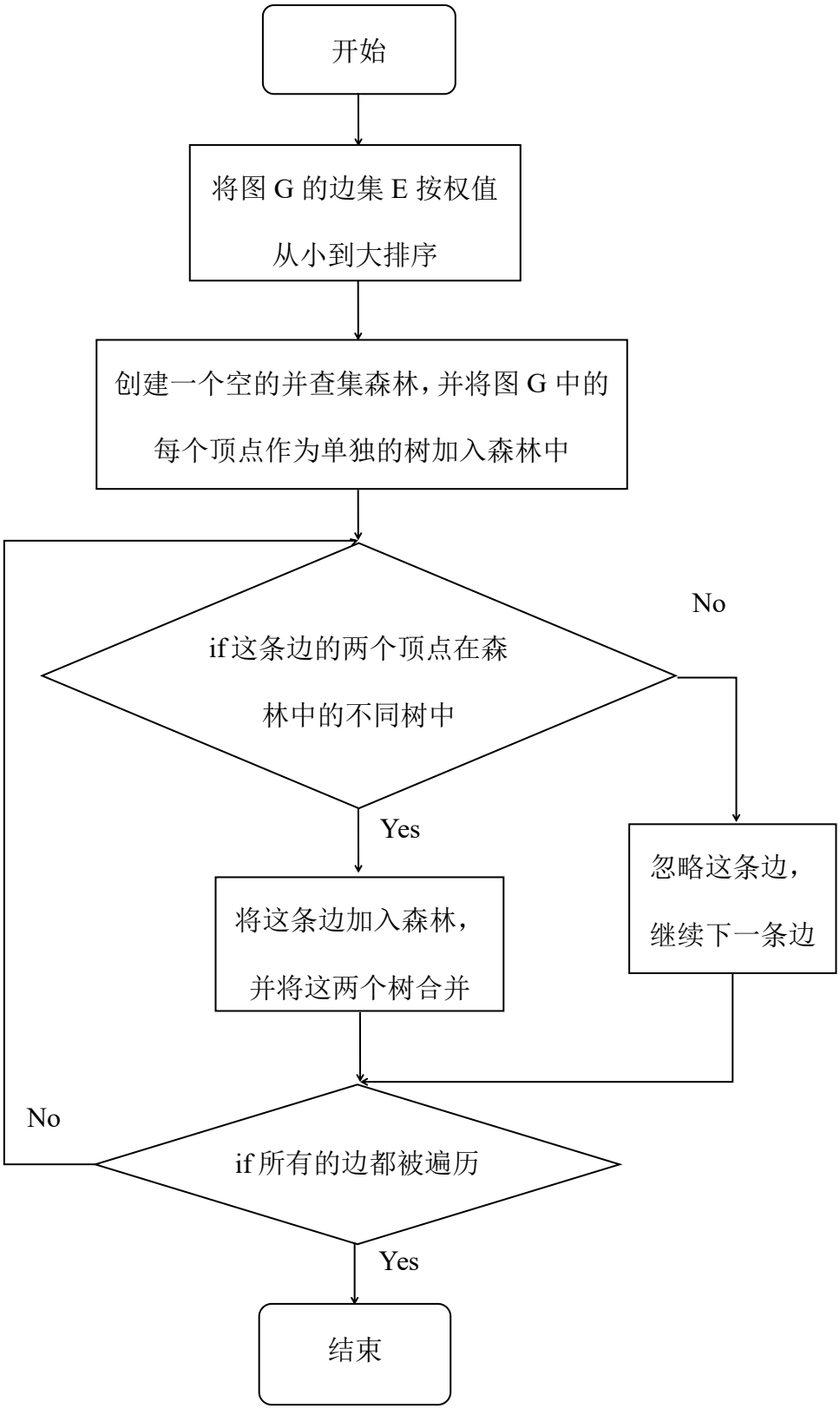

图 3-3 KruskalMinTree 函数流程图

3.3 Graph 类成员函数

以下是对 Graph 类每个函数的功能、返回值、入口参数、出口参数和算法思想的简要描述：

Graph::Graph()：功能：构造函数，初始化图的邻接矩阵。返回值：无。入口参数：无。出口参数：无。算法思想：使用双重循环初始化邻接矩阵，对角线元素为 0，其余为一个较大的值（代表无穷大）。

Graph::~Graph()：功能：析构函数，释放资源。返回值：无。入口参数：无。出口参数：无。算法思想：释放动态分配的资源。

Graph::InsertVex(Vex svex)：功能：插入顶点。返回值：成功插入返回 1，否则返回 0。入口参数：顶点对象 svex。出口参数：无。 算法思想：通过遍历顶点列表，判断是否存在相同顶点，如果不存在则插入。

Graph::DeleteVex(Vex svex)：功能：删除顶点。返回值：成功删除返回 1，否则返回 0。入口参数：顶点对象 svex。出口参数：无。 算法思想：遍历顶点列表，找到对应顶点并删除，同时删除相关的边。

Graph::UpdateVex(Vex svex)：功能：更新顶点。返回值：成功更新返回 1，否则返回 0。入口参数：顶点对象 svex。出口参数：无。 算法思想：遍历顶点列表，找到对应顶点并更新。

Graph::InsertEdges(Edge sedge)：功能：插入边。返回值：成功插入返回 1，否则返回 0。入口参数：边对象 sedge。出口参数：无。算法思想：遍历边列表，判断是否存在相同边，如果不存在则插入。

Graph::DeleteEdges(Edge sedge)：功能：删除边。返回值：成功删除返回 1，否则返回 0。入口参数：边对象 sedge。出口参数：无。 算法思想：遍历边列表，找到对应边并删除，同时更新邻接矩阵。

Graph::UpdateEdges(Edge sedge)：功能：更新边。返回值：成功更新返回 1，否则返回 0。入口参数：边对象 sedge。出口参数：无。算法思想：遍历边列表，找到对应边并更新。

Graph::isConnect(Edge b, Edge outEdges[], int n)：功能：判断两个顶点是否连通。返回值：顶点是否连通，1 表示是，0 表示否。入口参数：边对象 b，边数组 outEdges[]，数组长度 n。出口参数：无。算法思想：利用拓扑排序判断是否有环路，判断两个顶点是否连通。

Graph::showMassage()：功能：显示图信息。返回值：无。入口参数：无。出口参数：无。算法思想：输出图的顶点和边信息。

3.3 SqList 模板类成员函数

以下是对顺序表模板类 SqList 的各个函数的功能、返回值、入口参数、出口参数和算法思想的简要描述：

SqList::SqList(int size = DEFAULT_SIZE)：功能：构造函数，创建一个空表。返回值：无。入口参数：可选参数 size，表示顺序表的最大容量，默认为 DEFAULT_SIZE。出口参数：无。算法思想：分配内存空间，初始化顺序表的长度和最大容量。

SqList::~SqList()：功能：析构函数，释放动态分配的内存。返回值：无。入口参数：无。出口参数：无。算法思想：释放动态分配的内存空间。

SqList::GetLength() const：功能：获取顺序表的长度。返回值：当前顺序表的长度。入口参数：无。出口参数：无。算法思想：返回顺序表的长度。

SqList::GetElem(int i, T& e) const：功能：获取顺序表中第 i 个元素的值。返回值：成功获取返回 0，未找到返回 1。入口参数：整数 i，表示要获取的元素的位置；引用参数 e，用于存储获取到的元素的值。出口参数：引用参数 e 存储获取到的元素的值。

算法思想：判断位置 i 是否合法，如果合法则将相应位置的元素值存储到参数 e 中。

SqList::SetElem(int i, const T& e)：功能：修改顺序表中第 i 个元素的值。返回值：成功修改返回 0，未找到返回 1。入口参数：整数 i，表示要修改的元素的位置；常量引用参数 e，表示新的元素值。出口参数：无。

算法思想：判断位置 i 是否合法，如果合法则将相应位置的元素值修改为参数 e。

SqList::DeleteElem(int i, T& e)：功能：删除顺序表中第 i 个元素。返回值：成功删除返回 0，未找到返回 1。入口参数：整数 i，表示要删除的元素的位置；引用参数 e，用于存储删除的元素的值。出口参数：引用参数 e 存储删除的元素的值，顺序表的长度减一。算法思想：判断位置 i 是否合法，如果合法则将相应位置的元素值存储到参数 e 中，并将该位置后面的元素向前移动一个位置。

SqList::InsertElem(const T& e)：功能：在顺序表的表尾插入元素。返回值：成功插入返回 0，插入失败返回 1。入口参数：常量引用参数 e，表示要插入的元素的值。出口参数：无。算法思想：将元素插入到顺序表的最后，并将顺序表的长度加一。

4 运行与测试

4.1 编译运行

vs code 环境下多文件编译运行编译运行。多文件如下：

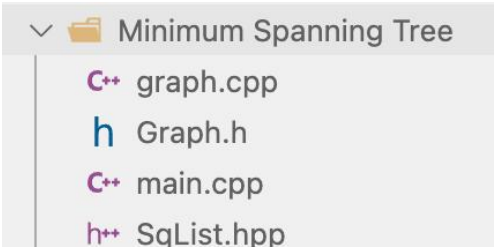

图 4-1 多文件

1.终端输入命令行，分别编译 main.cpp 和 graph.cpp 为二进制文件 main.o graph.o ：

```
g++ -c main.cpp -o main.o
g++ -c graph.cpp -o graph.o
```

2.将二进制文件 main.o 和 graph.o 链接，并生成可执行文件 a.out ：

```
g++ main.o graph.o -o a.out
```

3.执行 a.out ：

```
./a.out
```

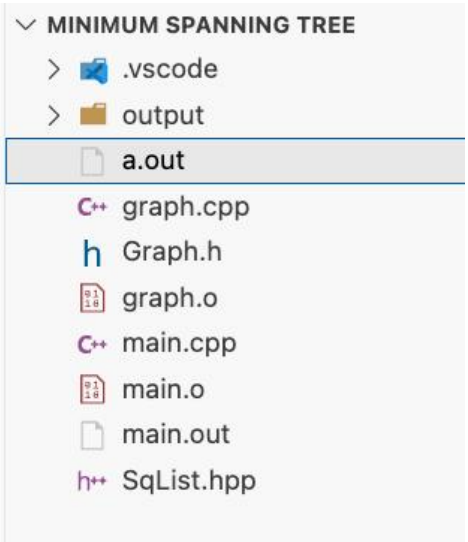

图 4-2 编译运行

4.2 测试

1. 添加城市案例并求最小生成树如下：

```
------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
1
Added successfully...

------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
7
Vertex information:
V1--北京
V2--成都
V3--武汉
V4--上海
V5--广州
V6--深圳
Edge information:
V1--V2 : 6
V1--V3 : 1
V1--V4 : 5
V2--V3 : 5
V2--V5 : 3
V3--V4 : 5
V3--V5 : 6
V3--V6 : 4
V4--V6 : 2
V5--V6 : 6
```

图 4-3 添加测试案例并展示

```
------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
4
The Prim algorithm implements the minimum spanning tree:
v2-v3 : 5
v3-v1 : 1
v4-v6 : 2
v5-v2 : 3
v6-v3 : 4
weight:15
------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
5
The Kruskal algorithm implements the minimum spanning tree:
v3-v1 : 1
v4-v6 : 2
v5-v2 : 3
v6-v3 : 4
v2-v3 : 5
weight:15
```

图 4-4 求测试案例的最小生成树及权值

2.添加点并展示如下：

```
------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
2
Enter the number of vertices to be added:
2
Enter the vertex :
v7 杭州
Enter the vertex :
v8 苏州
------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
7
Vertex information:
V1--北京
V2--成都
V3--武汉
V4--上海
V5--广州
V6--深圳
v7--杭州
v8--苏州
Edge information:
V1--V2 : 6
V1--V3 : 1
V1--V4 : 5
V2--V3 : 5
V2--V5 : 3
V3--V4 : 5
V3--V5 : 6
V3--V6 : 4
V4--V6 : 2
V5--V6 : 6
```

图 4-5 添加点并展示

3.添加边并展示如下：

```
------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
3
Enter the number of edges to be added:
2
Input(vex1 vex2 weight):
v7 v8 9
Input(vex1 vex2 weight):
v7 v1 5
------------------------------------------------
1. Test city map case
2. Add vertex
3. Add edge
4. Prim algorithm
5. Kruskal algorithm
6. Delete and modify information
7. Print vertex and edge information
8. Exit
------------------------------------------------
7
Vertex information:
V1--北京
V2--成都
V3--武汉
V4--上海
V5--广州
V6--深圳
v7--杭州
v8--苏州
Edge information:
V1--V2 : 6
V1--V3 : 1
V1--V4 : 5
V2--V3 : 5
V2--V5 : 3
V3--V4 : 5
V3--V5 : 6
V3--V6 : 4
V4--V6 : 2
V5--V6 : 6
v7--v8 : 9
v7--v1 : 5
```

图 4-6 添加边并展示求最小生成树

4.添加边后再求最小生成树如下：

```
4
The Prim algorithm implements the minimum spanning tree:
v2-v3 : 5
v3-v1 : 1
v4-v6 : 2
v5-v2 : 3
v6-v3 : 4
v7-v1 : 5
v8-v7 : 9
weight:29
```

图 4-7 添加边后展示求最小生成树

5.修改点边信息如下：

```
6
1. Modify vertex
2. Delete vertex
3. Modify edge
4. Delete edge
1
Enter the modified vertex number:
v7
hangzhou
```

```
6
1. Modify vertex
2. Delete vertex
3. Modify edge
4. Delete edge
3
Enter the modified edge information:
v7 v8 99
The update was successful...
```

```
7
Vertex information:
V1--北京
V2--成都
V3--武汉
V4--上海
V5--广州
V6--深圳
v7--hangzhou
v8--苏州
Edge information:
V1--V2 : 6
V1--V3 : 1
V1--V4 : 5
V2--V3 : 5
V2--V5 : 3
V3--V4 : 5
V3--V5 : 6
V3--V6 : 4
V4--V6 : 2
V5--V6 : 6
v7--v8 : 9
v7--v1 : 5
```

```
7
Vertex information:
V1--北京
V2--成都
V3--武汉
V4--上海
V5--广州
V6--深圳
v7--hangzhou
v8--苏州
Edge information:
V1--V2 : 6
V1--V3 : 1
V1--V4 : 5
V2--V3 : 5
V2--V5 : 3
V3--V4 : 5
V3--V5 : 6
V3--V6 : 4
V4--V6 : 2
V5--V6 : 6
v7--v8 : 99
v7--v1 : 5
```

图 4-8 修改点边信息并展示

5 总结

一、系统优缺点总结

在这次图的最小生成树的课设中，我实现了基于普里姆算法和克鲁斯卡尔算法的图的最小生成树生成。以下是本次课设的优缺点总结：

优点：

算法实现：成功实现了普里姆算法和克鲁斯卡尔算法，能够根据输入的图生成最小生成树；

代码结构清晰：代码结构清晰，易于理解和维护；

调试功能：系统提供了调试功能，方便发现和解决潜在的问题。

缺点：

输入验证：目前系统没有对输入进行严格的验证，可能导致一些无效或错误的输入；

性能优化：在处理大型图时，算法的运行时间可能较长，需要进一步优化；

图形界面：系统没有提供图形界面，只能通过命令行交互，繁琐不够直观。

二、实现了什么

建立图与读取图的邻接矩阵表示。

使用普里姆算法和克鲁斯卡尔算法生成最小生成树。

输出最小生成树的权重。

三、未实现或待完善的部分

输入验证：需要添加对输入的验证，确保输入的邻接矩阵有效。

性能优化：需要进一步优化算法，减少运行时间。

图形界面：可以考虑添加图形界面，使交互更加直观。

四、程序调试中发现的问题和解决办法

在程序调试过程中，我们发现了一些问题并找到了相应的解决办法：

输入格式错误：有时输入的邻接矩阵格式不正确，导致程序无法正确读取。解决办法是增加输入验证，确保输入格式正确。

内存溢出：在处理大型图时，可能会出现内存溢出的问题。解决办法是优化算法，减少内存使用。

调试输出问题：在调试过程中，有时输出结果不正确。解决办法是仔细检查

代码逻辑，确保算法实现正确。

编译器版本问题：有的编译器不支持 C++11 的某些语法。例如初始化列表语法，在 C++11 及更高版本中，可以使用大括号 {} 来初始化 vector。但在旧的编译器中就可能会导致错误。如果编译器版本较旧，只能使用传统的构造函数初始化 vector，调用 push_back 函数。

五、主要任务及学习到的东西

在这次程序设计训练中，我的主要任务是实现图的最小生成树算法和进行系统调试。通过这次训练，我学习到了以下内容：

1.图的最小生成树算法原理及其实现方法。例如用 C++ STL 中的 sort 函数对 Kruskal 算法中边数组进行排序的实现方式：

第一种实现方式是使用自定义比较函数。第二种实现方式是使用结构体的操作符重载。可以定义一个结构体，重载小于操作符，然后将结构体对象作为 sort 函数的第三个参数传入，实现对边按照权值从小到大进行排序。第三种实现方式是使用 lambda 函数。可以直接在 sort 函数的第三个参数位置定义一个匿名 lambda 函数，实现对边按照权值从小到大进行排序。使用 lambda 函数可以减少代码的编写量，同时使代码更加清晰、易读。

2.如何设计并实现一个完整的系统，包括输入输出、算法实现和调试等功能。

3.在遇到问题时如何进行调试和解决。

4.如何优化代码性能和提高代码质量。

六、可以加强的地方及今后的目标

在今后的学习和实践中，我还要加强以下方面：

1.提高算法性能：进一步优化图的最小生成树算法，提高处理大型图的能力。尝试图的其他表示方法进行图的基本操作。

2.学习图形界面开发：为了使系统更加直观和易用，以 Qt 为主，学习图形界面开发技术，实践学习这个跨平台的 c++库，开发 GUI 应用程序。

3.提高代码质量：通过不断学习和实践，提高代码质量，减少错误和漏洞。

4.坚持探索学习：计算机发展迅猛，更新速度快，坚持不断提出问题、解决问题，达到终身学习的境界。

5.我今后的目标是在学习和实践中不断提高自己的能力和水平，特别是学好 c++，阅读优秀的 C++、数据结构代码和书籍，了解最佳实践和设计模式。为未来的项目和挑战做好准备；学习 C++的面向对象编程特性，如类、对象、继承、多态等；学习 C++的标准库和 STL（Standard Template Library），这些库提供了许多有用的容器、算法和函数；尝试做 c++和数据结构等项目，不断试错，通过实践来提高自己的编程技能。

参考文献

[1] Stephen Prata.C++ Primer Plus 第6版中文版[M]. 人民邮电出版社, 2020.

[2] 程杰.大话数据结构[M]. 清华大学出版社, 2020.

[3] Thomas H.Cormen / Charles E.Leiserson / Ronald L.Rivest / Clifford Stein. 算法导论[M]. 机械工业出版社. 2012

[4] Stanley B. Lippman/ Josée Lajoie/Barbara E. Moo. C++ Primer 中文版（第5版）. 电子工业出版社, 2013.

[5] Josuttis N M. The C++ standard library: a tutorial and reference[J]. 2012.

[6] Gottschling P. Discovering Modern C++[M]. Addison-Wesley Professional, 2021.

[7] Hubbard J R. Programming with C++[J]. 2021.

[8] Reddy M. API Design for C++[M]. Elsevier, 2011.

[9] Stroustrup B. A Tour of C++[M]. Addison-Wesley Professional, 2022.

[10] Lam M. Data structures and algorithms[J]. 2015.

[11] Legat B, Dowson O, Garcia J D, et al. MathOptInterface: a data structure for mathematical optimization problems[J]. INFORMS Journal on Computing, 2022, 34(2): 672-689.