**数据结构课程设计《电网建设造价模拟系统》项目说明文档**

2253893 苗君文 软件工程

**1. 项目简介**

**1.1. 项目背景**

图是由节点（或顶点）和边组成的数据结构，节点表示图中的对象，边表示对象之间的关系。最小生成树是一个连通图的生成树（Spanning Tree）中，边的权重之和最小的生成树。生成树是一个树，它包含图中的所有顶点，并且通过边将这些顶点相连而形成，同时不形成回路。这个概念在许多应用中都有实际意义，比如在通信网络、电路设计、城市规划等领域。找到一个图的最小生成树有助于优化资源的利用，减小总体成本。

常见的最小生成树算法包括Prim算法和Kruskal算法。这两种算法都属于贪心算法的范畴，通过每一步选择局部最优解来达到全局最优解。它们的目标都是找到一个包含所有顶点的树，并且边的权重之和最小。

**1.2. 项目要求**

假设一个城市有n个小区，要实现n个小区之间的电网都能够相互接通，构造这个城市n个小区之间的电网，使总工程造价最低。请设计一个能够满足要求的造价方案。

在每个小区之间都可以设置一条电网线路，都要付出相应的经济代价。n个小区之间最多可以有n（n-1）/2条线路，选择其中的n-1条使总的耗费最少。

**1.3. 输入格式**

本项目要求用户不断输入要执行的操作，若要创建顶点，则需要输入顶点个数及各个顶点的名称；若要添加边，则需要输入边的数量及各边的两个顶点及权值，构造或显示最小生成树或退出程序时不需要输入信息。

**1.4. 输出格式**

本项目针对输入有详细的输入错误处理，若用户输入错误，则程序会输出相关提示信息，并要求重新输入。而针对用户选择的操作，可以输出相应的操作结果，并最后输出最小生成树的顶点及边。

**2. 设计思路**

**2.1. 数据结构设计**

Prim算法通过贪心策略逐步构建最小生成树，在电网造价模拟系统中采用邻接链表、最小堆、最小生成树数据结构，使得程序能够满足应有的功能，这些数据结构的选择是出于算法特性和操作的高效性考虑。

首先，对于图的表示，采用邻接表的方式能够更灵活地处理不同的情况，尤其是在电网拓扑结构动态变化的场景下，链表结构的插入和删除操作更为便捷。在空间复杂度上，这种表示方法对于稀疏图比邻接矩阵有明显的空间优势，在电网建设中，通常节点之间的连接较为稀疏，并避免了存储大量不存在边的冗余信息。此外，对于每个顶点，邻接表直接存储了与之相邻的顶点信息，使得查找邻接顶点的操作更为高效。

其次，Prim算法需要快速找到权值最小的边，因此选择了最小堆作为辅助数据结构。使用最小堆的原因在于插入操作的时间复杂度为 O(log n)，在每次加入新的顶点时，需要将新的边插入最小堆中，而最小堆的插入操作的时间复杂度为对数级别，保证了边的插入效率。并且，删除操作的时间复杂度也为 O(log n)，每次选择最小权值的边，从最小堆中删除该边，同样具有对数级别的时间复杂度。此外，最小堆的性质保证了堆顶元素是当前最小的元素，因此能够在每次选择最小权值边时，快速取出根部元素。

最小生成树的节点和类的设计同样符合清晰的逻辑结构。MinSpanTree类中使用了一个数组edgevalue来存储最小生成树的边，这种设计简单而有效。在minHeap类中，使用Element结构表示堆中的元素，其中包括key和other两个成员变量。这种设计是为了与MSTEdgeNode的结构相匹配，方便堆中元素与最小生成树边的转换。

采用最小堆的优势在于能够高效地获取当前图中的最小权值边，而邻接表则为高效地表示稀疏图提供了便利。MinSpanTree类和Element结构的设计则是为了更好地配合实现Prim算法的逻辑。这样的设计使得算法实现更为清晰、高效。

**2.2. 类设计**

**2.2.1. MSTEdgeNode类（最小生成树结点类）**

成员变量：

* int tail, head：表示边的两个顶点位置。
* E key：表示边的权值。

成员函数：

* 构造函数：用于初始化边结点。
* bool operator<=和bool operator>：用于比较两个边结点的权值大小。

**2.2.2. MinSpanTree（最小生成树类）**

成员变量：

* MSTEdgeNode<T, E>edgevalue[maxEdges]：用边值数组表示最小生成树。
* int maxSize：数组的最大元素个数。
* int n：当前个数。

成员函数：

* 构造函数：初始化 maxSize 和 n。
* int Insert(MSTEdgeNode<T, E> item)：插入一条边到最小生成树中。
* void getEdgevalue(int i, int &tail, int &head, E &key)：获取第 i 条边的信息。
* int getEdgeNum()：获取当前最小生成树的边数。

**2.2.3. Element类**

成员变量：

* T key：表示堆中元素的键值。
* E other：表示堆中元素的其他信息。

成员函数：

* Element<T, E>& operator=：赋值运算符，用于复制元素。
* bool operator<=、bool operator<、bool operator>=、bool operator>：用于比较两个元素的键值大小。

**2.2.4. minHeap类**

成员变量：

* MSTEdgeNode<T, E>\* heap：堆的数组表示。
* int currentSize：当前堆中元素个数。
* int maxHeapSize：最大堆容量。

成员函数：

* 构造函数：初始化堆的大小。
* 析构函数：释放堆的内存。
* void siftDown(int start, int m)：最小堆的下滑调整。
* void siftUp(int start)：最小堆的上滑调整。
* bool Insert(const MSTEdgeNode<T, E>& x)：插入一个元素到堆中。
* bool Remove(MSTEdgeNode<T, E>& x)：删除堆顶元素。

**2.2.5. Edge类**

成员变量：

* int dest：边的目标顶点位置。
* E cost：边的权值。
* Edge<T, E>\* link：指向下一条边的指针。

**2.2.6. Vertex类**

成员变量：

* T data：顶点的名称。
* Edge<T, E>\* adj：边链表的头指针。

**2.2.7. Graphlnk类**

成员变量：

* Vertex<T, E> NodeTable[maxVertices]：顶点表（各边链表的头结点）。
* int numVertices：当前图的顶点个数。
* int numEdges：当前图的边数。

成员函数：

* 构造函数：初始化顶点表。
* 析构函数：释放图的内存。
* int getVertexPos(T vertex)：取顶点 vertex 在结点表中的序号。
* T getValue(int i)：获取顶点 i 的值。
* E getWeight(int v1, int v2)：获取边(v1, v2)的权值。
* int insertVertex(T vertex)：在图中插入一个顶点。
* bool insertEdge(int v1, int v2, E cost)：在图中插入一条边。
* int getFirstNeighbor(int v)：取顶点 v 的第一个邻接顶点。
* int getNextNeighbor(int v, int w)：取 v 的邻接顶点 w 的下一个邻接顶点。
* int NumOfVertices()：获取顶点个数。
* int NumOfEdges()：获取边数。
* bool searchVertex(T vertex)：查找顶点是否存在。
* void clear()：清除图中边和点。

**2.3. 主程序设计**

首先，程序通过用户输入提供了一个菜单界面，其中包括创建电网顶点、添加电网边、构造最小生成树、显示最小生成树和退出程序等操作选项。通过输入选择相应的操作，用户可以依次完成创建电网顶点、添加电网边、以及利用Prim算法构造最小生成树。在这个过程中，使用了图的邻接表表示，通过Graphlnk类实现图的创建和边的插入。同时，程序使用了最小堆数据结构，通过minHeap类实现Prim算法中的堆操作，用于高效地选择最小权值的边。

程序中使用了MinSpanTree类来表示最小生成树，该类通过MSTEdgeNode结构体来存储边的信息，并提供了插入边、获取边信息等操作。在主程序中，通过调用Prim函数，利用Prim算法构造最小生成树，并将结果存储在MinSpanTree对象中。最后，用户可以选择显示最小生成树的顶点和边，提供了清晰的输出以展示构建的电网最小生成树。

整体设计思路强调了面向对象的原则，将功能划分为独立的类，提高了程序的可读性和可维护性。通过用户交互、图的表示和最小生成树算法，主程序实现了一个功能完备的电网建设造价模拟系统。

**2.4. 输入错误处理设计**

**2.4.1. 单个参数的输入错误处理**

在输入操作选项时,使用 while 循环不断尝试获取用户输入，直到输入满足要求。使用cin.fail()及对应的大小写字母来判断输入是否出错，检查输入是否在有效范围内。如果输入无效，则输出错误信息，清除输入缓冲区并忽略之后的字符，并重新输出输入的提示信息。通过一个字符来获取输入一个数之后的字符以检查输入字符的个数，若不正确，也输出错误信息，并重新输出输入的提示信息，执行相应清除操作。如果输入有效，则跳出循环。

类似的操作结构可以用在输入顶点个数,输入各顶点名称,输入边的个数,输入边的两个顶点及其权值,输入生成最小生成树的起始结点等处。

**2.4.2. 电网系统相关的输入错误处理**

在各个操作选项中，都有对电网操作的相关输入错误处理。在选项B添加电网操作时，如果还未创建顶点就创建边，就会输出提示信息，让用户先创建电网顶点。在输入边的时候，如果插入的边已经存在于图中，则会提示无需插入。在选项C构造最小生成树时，如果电网中没有顶点或没有边，则会提示用户先创建电网。输入起始顶点时，如果在图中找不到这个顶点，则需要重新输入。在选项D显示最小生成树中，如果电网无内容，则会提示用户先创建电网内容；如果还未经过选项C生成最小生成树，则会提示用户先生成最小生成树。

**3. 功能实现**

本项目的关键就在于Prim算法生成最小生成树，因此将仅介绍Prim函数的具体功能实现。

（1）初始化操作：首先，创建一个MSTEdgeNode对象ed用于存储最小生成树的边信息。接着，获取图G的顶点总数n和边总数m。根据给定的起始顶点u0在图G中找到对应的顶点位置，标记为u。然后，创建一个最小堆H用于存储候选边，最大容量为图的边数m。并且创建一个布尔数组Vmst，表示顶点是否已经加入最小生成树。初始时，将起始顶点u标记为已加入生成树。

（2）迭代构建最小生成树：使用一个循环，直到生成树包含所有顶点。在循环内，获取顶点u的第一个邻接顶点v。遍历顶点u的所有邻接顶点，如果顶点v尚未加入最小生成树，则将对应的边信息（ed）加入最小堆H中，其中ed.head为邻接顶点v，ed.tail为顶点u，ed.key为边的权值。从最小堆H中取出具有最小权值的边ed，若边的头部ed.head尚未加入最小生成树，则将该边加入最小生成树，并更新当前顶点u为边的头部ed.head，标记该顶点u已加入生成树。

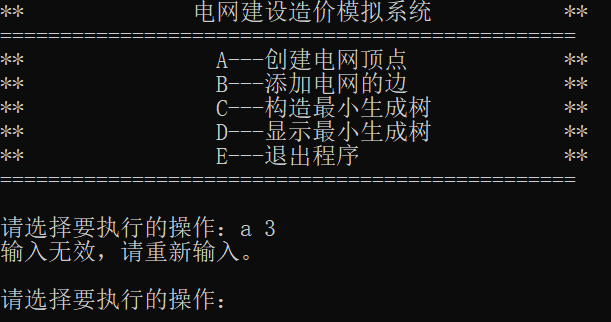
重复上述步骤直至最小生成树包含所有顶点。

（3）结果输出：函数执行完毕后，最小生成树的边信息存储在MST对象中，用户可以通过调用MST.Insert来获取最小生成树的边。每次从最小堆中取出的边都是当前生成树与未加入生成树的顶点之间的最小权值边，确保了构建最小生成树的正确性。

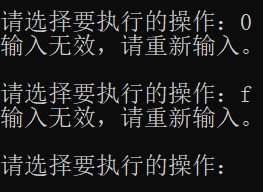
**4. 测试结果（包含边界测试）**

**4.1. 选择要执行的操作**

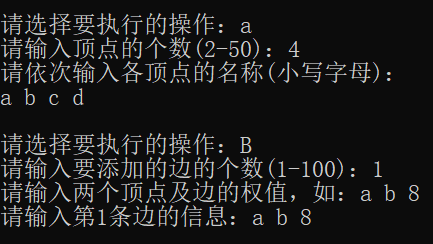
（1）输入的个数超过一个



（2）输入的内容非法



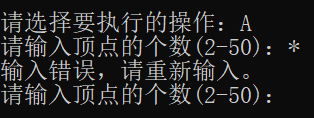
（3）输入正确（大小写均可）即可进入相应的操作



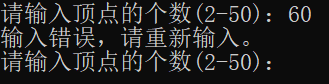
**4.2. 创建电网顶点（选项A）**

**4.2.1. 输入顶点的个数**

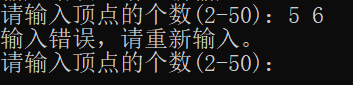
（1）输入的类型错误



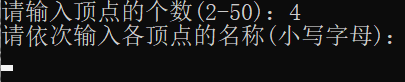
（2）输入的范围错误



（3）输入的个数超过一个

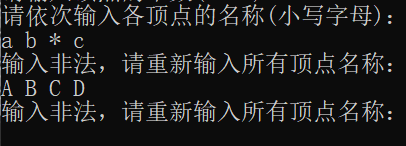


（4）输入正确即可进入下一环节

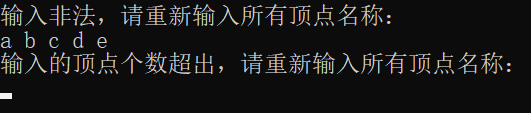


**4.2.2. 输入各顶点的名称（小写字母）**

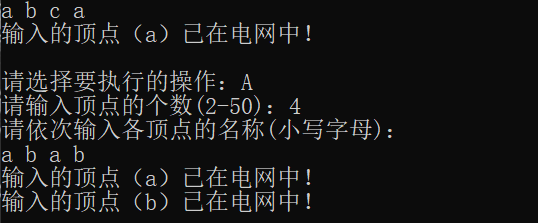
（1）输入内容中有非法字符



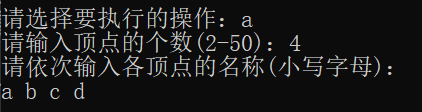
（2）输入的个数超过用户规定的顶点个数



（3）输入的顶点已存在于电网中，则提示相关信息

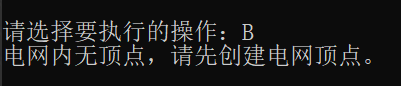


（4）输入的顶点均为新顶点



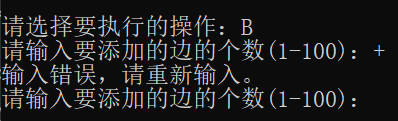
**4.3. 添加电网的边（选项B）**

**4.3.1. 电网内无顶点时，提示相关信息。**

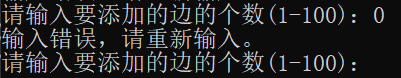


**4.3.2. 输入要添加的边的个数**

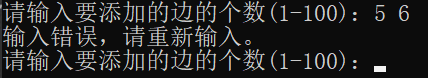
（1）输入的类型错误



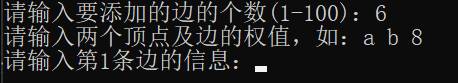
（2）输入的范围错误



（3）输入的个数超过一个

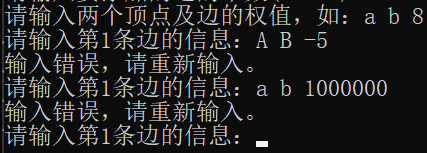


（4）输入正确即可进入下一环节

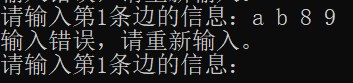


**4.3.3. 输入边的信息**

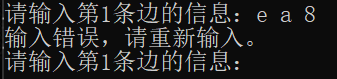
（1）输入字符非法或权值非法



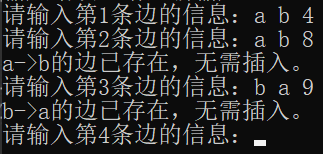
（2）输入的个数超过3个



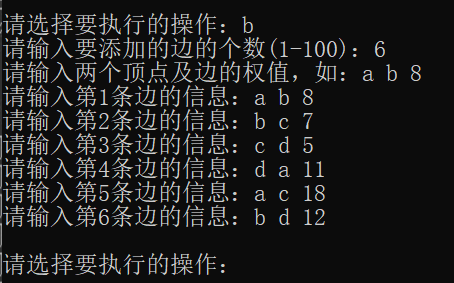
（3）输入的顶点不存在于电网中



（4）输入的边已存在于电网中（双向可行），则提示相关信息（无需插入）

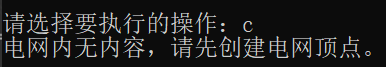


（5）输入的边都为新边，即输入正确



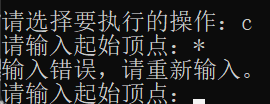
**4.4. 构造最小生成树（选项C）**

**4.4.1. 电网内无内容（无顶点或无边）时，提示相关信息。**

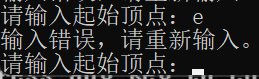


**4.4.2. 输入起始顶点**

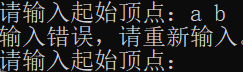
（1）输入的起始顶点为非法字符



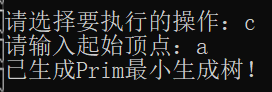
（2）输入的起始顶点不存在于电网中



（3）输入的个数超过一个

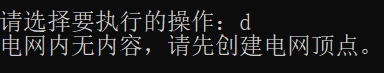


（4）输入正确，即输入的顶点存在于电网中

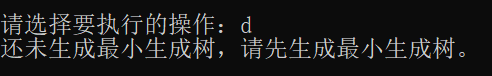


**4.5. 显示最小生成树（选项D）**

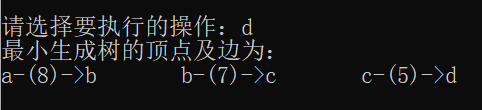
**4.5.1. 电网内无内容（无顶点或无边）时，提示相关信息。**



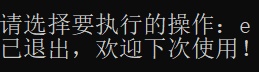
**4.5.2. 未生成最小生成树，提示相关信息。**



**4.5.3. 有最小生成树的情况，可以输出最小生成树的顶点及边**



**4.6. 退出程序，输出提示信息**



**4.7. 程序的测试结果总览（此处仅展示输入合法的情况）**

