****

**实验报告：电网建设造价模拟系统**

2351041

刘浩田

完成日期:2024.10.6

# 索引

### 概述

### 项目目的

### 项目环境

### 项目背景

### 项目要求

### 项目实现

### 项目总结

# 概述

在现代社会中，电力作为基础设施的重要组成部分，其建设和管理的效率直接关系到城市的运行成本和居民的生活质量。随着城市化进程的加快，城市电网的规划和建设面临着新的挑战。如何在保证电力供应可靠性的同时，降低电网建设的总成本，成为了城市规划者和电力工程师需要解决的问题。本项目旨在设计一个算法，通过模拟电网建设过程，实现城市中n个小区之间的电网相互接通，同时使总工程造价最低，即寻求一个最小生成树（Minimum Spanning Tree, MST）的解决方案。

项目的核心在于构建一个有效的算法模型，该模型能够模拟电网的建设过程，并在保证每个小区都能接入电网的前提下，选择成本最低的电网连接方案。具体而言，项目需要处理的关键问题包括：如何在多个小区之间选择电网线路，以确保所有小区都能相互接通；如何在众多可能的连接方案中，找到总成本最低的方案；以及如何确保所选方案的可行性和经济性。

为了实现这一目标，项目采用了图论中的最小生成树算法，特别是Kruskal算法和Prim算法。这两种算法都是求解最小生成树的经典方法，能够有效地在所有可能的连接方案中找到成本最低的方案。Kruskal算法通过按权重（即电网线路的经济代价）排序所有边，然后逐个选取边加入到生成树中，直到所有节点都被连接。Prim算法则是从一个节点开始，逐步扩展生成树，每次添加连接生成树和非生成树节点之间权重最小的边。

在实现算法的过程中，项目采用了C++语言进行编程，利用了自定义的数据结构和算法库。具体来说，项目中定义了MyString、MyVector和MyPriorityQueue等自定义数据结构，分别用于处理字符串操作、动态数组和优先队列。这些自定义的数据结构不仅提高了代码的可读性和可维护性，还增强了算法的执行效率。

项目的关键技术点在于如何高效地处理大规模数据和复杂的图结构。为了实现这一目标，项目采用了邻接表的方式来表示图，这种表示方法能够灵活地处理图中的节点和边，特别适合于稀疏图的情况。同时，项目还实现了一个邻接表的类ALGraph，该类封装了图的基本操作，如节点的搜索、边的添加等，为最小生成树算法的实现提供了基础。

在项目实现过程中，项目还考虑了用户交互的便捷性。通过控制台输入输出，用户可以方便地与系统进行交互，输入小区数量、节点名称以及电网线路的经济代价等信息。系统会根据用户的输入，动态地构建电网模型，并计算出最低成本的电网建设方案。

总之，本项目通过模拟电网建设过程，不仅提供了一个解决城市电网建设成本问题的实用工具，还展示了图论算法在实际问题中的应用。通过本项目的实现，可以有效地降低电网建设的总成本，提高电力供应的可靠性，对于城市规划和电力管理具有重要的实际意义。

# 项目目的

本项目的核心目的在于设计并实现一个电网造价模拟系统，该系统能够针对一个城市中n个小区之间的电网连接问题，提供一个总工程造价最低的解决方案。随着城市化进程的不断推进，城市电网的规划和建设变得日益复杂，如何在保证电力供应的可靠性和稳定性的同时，降低电网建设的总成本，是城市规划者和电力工程师面临的重大挑战。本项目旨在通过计算机模拟和算法优化，探索电网建设的最优路径，实现成本效益最大化。

首先，项目旨在加强学生对图论算法的理解和应用。在计算机科学中，图论是一种研究图的结构以及图上进行的信息加工过程的数学理论。电网建设问题可以抽象为图论中的最小生成树问题，即在一个加权连通图中找到一棵权重最小的生成树。通过本项目，学生可以深入理解最小生成树算法的原理和实现，掌握它们在实际问题中的应用方法。特别是在处理大规模数据和复杂图结构时，这些算法的灵活性和高效性显得尤为重要。

其次，实验强调了算法优化的重要性。一个优秀的算法能够显著提高程序的执行效率，对于电网造价模拟系统而言，这直接关系到模拟结果的准确性和实用性。例如，在实现电网线路选择功能时，选择合适的搜索算法可以加快路径检索速度；在进行成本计算时，合理的算法可以减少计算的复杂度。通过本项目，学生可以练习设计和优化算法，提高解决实际问题的效率。

再次，实验着重于提升学生的面向对象编程能力。面向对象编程是一种重要的编程范式，它通过将数据和操作封装成对象，提高了代码的可重用性和可维护性。在本项目中，学生需要构建多个类来表示不同的概念，如小区、电网线路等，并设计类的属性和方法来实现系统的功能。这要求学生不仅要掌握面向对象编程的语法，还要学会如何运用面向对象的思想进行系统设计。

此外，实验还旨在培养学生的软件工程意识。在实际的软件开发过程中，除了编写代码，还需要进行需求分析、系统设计、测试和维护等工作。通过本项目，学生可以体验软件开发的全过程，了解各个阶段的任务和要求。特别是在系统设计阶段，学生需要考虑系统的结构、模块划分、接口设计等问题，这对于提高学生的系统分析和设计能力具有重要意义。

最后，实验还关注于提升学生的用户体验意识。一个好的软件不仅要功能强大，还要易于使用。在本项目中，学生需要设计用户友好的交互界面，确保用户能够方便地进行操作。这要求学生不仅要关注程序的内部逻辑，还要考虑用户的需求和习惯，培养从用户的角度思考问题的能力。

综上所述，本项目目的在于通过实际操作，使学生在多方面得到锻炼和提升，为将来的学习和工作打下坚实的基础。通过本项目，学生不仅能够提高自己的编程能力和系统设计能力，还能够培养良好的软件开发习惯和用户意识，为成为一名优秀的软件工程师奠定基础。同时，项目的实际应用价值在于为城市电网建设提供一个成本优化的解决方案，有助于提高电力资源的利用效率和城市的可持续发展能力。

# 项目环境

## 系统：Windows11

## 编译器：VisualStudio2022x86

## 语言：C++

# 项目背景

在21世纪的城市发展中，电力供应的稳定性和可靠性对于城市的运行至关重要。随着城市规模的扩大和人口的增长，城市电网的建设和优化成为了一个复杂而迫切的问题。城市电网不仅需要保证电力的稳定供应，还要考虑到建设成本和后期维护的经济性。在这样的背景下，如何以最低的成本实现城市中各个小区之间的电网相互接通，成为了电力工程和城市规划领域的一个重要课题。

传统的电网规划方法往往依赖于专家经验和手工计算，这种方法不仅效率低下，而且难以适应大规模和复杂电网的规划需求。随着信息技术的发展，特别是图论算法在计算机科学中的应用，为电网规划提供了新的思路和工具。通过构建数学模型，利用算法自动寻找最优解，可以大大提高电网规划的效率和准确性。

本项目正是在这样的需求驱动下进行的设计和实现。项目旨在通过模拟电网建设过程，采用图论中的最小生成树算法，为城市电网提供一个电子化的解决方案，以应对日益增长的电网建设需求，并解决传统规划方式中存在的问题。通过本项目的实施，可以有效地降低电网建设的总成本，提高电力资源的利用效率，对于城市的可持续发展具有重要意义。

# 项目要求

本项目旨在开发一个电网造价模拟系统，该系统能够满足城市电网规划的基本需求，提供高效的信息管理功能，并确保电网建设的总成本最低。以下是对项目的详细要求：

## 6.1 功能要求

系统需要实现以下核心功能：

顶点创建：能够让用户输入城市中小区的数量，并为每个小区创建一个顶点。

边的添加：允许用户输入每对小区之间的电网线路的经济代价，并添加到图中。

最小生成树构建：系统应能够根据用户输入的数据，使用最小生成树算法计算出连接所有小区的最低成本方案。

结果展示：系统应能够展示最小生成树的详细结果，包括每条线路连接的小区和对应的成本。

## 6.2 数据结构要求

数据组织：建议使用邻接表来存储图数据，以支持高效的数据插入和删除操作。

数据完整性：确保所有输入数据的完整性和准确性，如检查必填项、格式等。

## 6.3 用户界面要求

操作简便：用户界面应直观易用，操作流程简单明了。

输入验证：系统应提供输入验证，确保用户输入的数据符合要求。

错误处理：对于用户的错误操作，系统应提供清晰的错误提示。

## 6.4 可复用性和可维护性

模块化设计：系统应采用模块化设计，便于后期维护和升级。

代码可读性：代码应具有良好的可读性，注释清晰，便于他人理解和修改。

## 6.5 测试要求

测试用例：为每个功能编写详细的测试用例，确保功能的正确性和稳定性。

边界测试：进行边界条件测试，确保系统在极端情况下也能正常工作。

## 6.6 性能要求

响应时间：系统应能够快速响应用户的操作请求，特别是在进行最小生成树计算时，应保证算法的执行效率。

数据处理能力：系统应能够处理大规模数据，即使在小区数量达到最大限制时也能保持稳定运行。

通过满足以上要求，电网造价模拟系统将能够为用户提供一个功能全面、操作简便、安全可靠的电网规划管理工具，有效提升电网规划工作的效率和质量。系统不仅能够帮助城市规划者和电力工程师优化电网建设方案，还能够为城市电网的可持续发展提供技术支持。

# 项目实现

为了实现一个电网造价模拟系统，该系统能够确保城市中n个小区之间的电网相互接通，并且总工程造价最低，设计并实现了一个基于图论的最小生成树算法——克鲁斯卡尔算法（Kruskal's algorithm）。

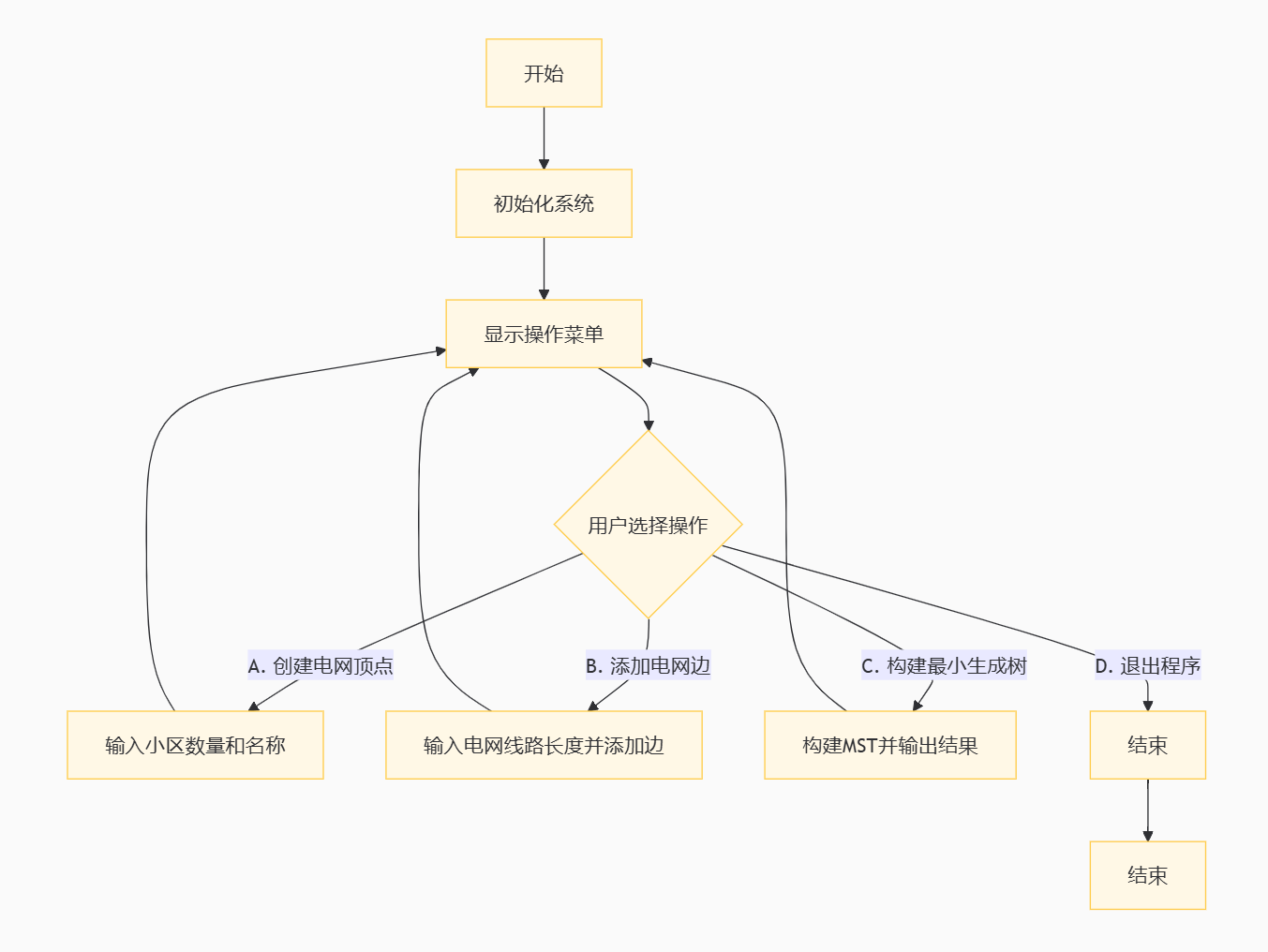


图 项目流程图

## 7.1 类设计和实现

### ALGraph 类

ALGraph 类是整个系统的核心，负责表示和管理图的顶点和边，以及构建最小生成树。

顶点和边的定义：

VNode 结构体表示图中的顶点，包含小区的名称和指向第一条边的指针。

ArcNode 结构体表示图中的边，包含邻接顶点的索引、指向下一条边的指针和邻接顶点的名称。

Road 结构体表示两个小区之间的电网线路，包含起始小区名称、结束小区名称和线路长度。

图的创建：

CreatGraph 方法用于输入小区数量和名称，初始化图的顶点。该方法通过循环接收用户输入的小区名称，并使用MyString类来存储这些名称，确保字符串操作的安全性和便捷性。

AddEdges 方法用于输入每对小区之间的电网线路长度，并添加到图中。该方法通过双层循环获取每对小区之间的线路长度，并创建ArcNode对象来表示这些边，存储在对应的顶点中。

最小生成树的构建：

CreatMST 方法用于构建最小生成树，并输出结果。该方法首先通过用户输入获取起始节点名称，然后初始化一个MyVector<bool>类型的visited数组来标记已访问的顶点，以及一个MyVector<Road>类型的mst数组来存储最小生成树中的边。

addEdgesToPQ 方法用于将图中的边添加到优先队列中，以便选择最小代价的边。该方法遍历每个顶点的所有邻接边，并将它们添加到优先队列中。

|  |
| --- |
| 附件、ALGraph 类部分代码与附属结构体 |
| struct Road {  MyString first;  MyString second;  int length;  Road()  : first("")  , second("")  , length(0)  {  }  Road(MyString f, MyString s, int l)  : first(f)  , second(s)  , length(l)  {  }  };  typedef struct ArcNode {  int adjvex;  struct ArcNode\* nextarc;  MyString sec\_name;  ArcNode()  : adjvex(0)  , nextarc(nullptr)  , sec\_name("")  {  }  } ArcNode;  typedef struct VNode {  MyString Name;  ArcNode\* firstarc;  VNode()  : Name("")  , firstarc(nullptr)  {  }  } VNode;  class ALGraph {  private:  MyVector<VNode> vertices;  int vexnum, arcnum;  public:  ALGraph()  : vexnum(0)  , arcnum(0)  {  }  int SearchMost(ArcNode\* firstarc, MyString& second)  {    }  unsigned int SearchVex(const MyString Name)  {    }  void CreatGraph()  {    }  void AddEdges()  {    }  void CreatMST()  {    }  struct CompareRoad {  bool operator()(const Road& r1, const Road& r2)  {  return r1.length > r2.length;  }  };  void addEdgesToPQ(int nodeIndex, MyPriorityQueue<Road, MyVector<Road>, CompareRoad>& pq, const MyVector<bool>& visited)  {  }; |

### MyPriorityQueue 类

MyPriorityQueue 类是一个模板类，用于实现优先队列，存储边并按照长度从小到大的顺序排列。

主要方法：

push 方法用于向优先队列中添加一条边。该方法将新边添加到容器的末尾，并调用heapify\_up方法来维护堆的性质。

pop 方法用于从优先队列中移除并返回代价最小的边。该方法将容器的第一个元素（即最小边）与最后一个元素交换，然后移除最后一个元素，并调用heapify\_down方法来维护堆的性质。

top 方法用于获取代价最小的边。该方法返回容器的第一个元素，即最小边。

|  |
| --- |
| 附件、MyPriorityQueue 类部分代码 |
| template <typename T, typename Container = MyVector<T>, typename Compare = std::less<typename Container::value\_type>>  class MyPriorityQueue {  private:  Container c;  Compare comp;  void heapify\_up(size\_t index)  {    }  void heapify\_down(size\_t index)  {    }  public:  MyPriorityQueue() = default;  void push(const T& value)  {    }  void pop()  {    }  const T& top() const { return c[0]; }  bool empty() const { return c.empty(); }  size\_t size() const { return c.size(); }  }; |

### MyString 类

MyString 类是一个自定义字符串类，用于处理字符串的输入输出，增强了系统的健壮性。

构造函数和析构函数：

提供了默认构造函数、从C风格字符串构造的构造函数、拷贝构造函数和析构函数。这些方法确保了字符串的正确创建和销毁，避免了内存泄漏。

重载了赋值运算符和比较运算符，以及输入输出运算符。这些重载操作符使得MyString类能够方便地与标准输入输出流配合使用。

|  |
| --- |
| 附件、MyString 类部分代码 |
| class MyString {  private:  char\* data;  size\_t \_size;  public:  MyString()  : data(new char[1])  , \_size(0)  {  data[0] = '\0';  }  MyString(const char\* str)  : \_size(strlen(str))  {  data = new char[\_size + 1];  strcpy(data, str);  }  MyString(const MyString& other)  : \_size(other.\_size)  {  data = new char[\_size + 1];  strcpy(data, other.data);  }  ~MyString()  {  delete[] data;  }  MyString& operator=(const MyString& other)  {    }  bool operator==(const MyString& other) const  {  }  friend std::ostream& operator<<(std::ostream& os, const MyString& str)  {    }  friend std::istream& operator>>(std::istream& is, MyString& str)  {    }  const char\* c\_str() const { return data; }  size\_t size() const { return \_size; }  bool empty() const { return \_size == 0; }  }; |

### MyVector 类

MyVector 类是一个模板类，模拟标准库中的vector，用于动态数组的实现。

构造函数和析构函数：

提供了默认构造函数和析构函数，以及拷贝构造函数和赋值运算符。这些方法确保了动态数组的正确创建和销毁。

push\_back 方法用于在数组末尾添加元素。如果当前容量不足以容纳新元素，该方法会调用realloc方法来扩大容量。

pop\_back 方法用于移除数组末尾的元素。该方法减少了数组的\_size属性。

提供了索引操作符operator[]，以及size和empty方法。这些方法使得MyVector类能够方便地访问和操作动态数组中的元素。

|  |
| --- |
| 附件、MyVector类部分代码 |
| template <typename T>  class MyVector {  private:  T\* data;  size\_t \_size;  size\_t \_capacity;  void realloc(size\_t new\_capacity)  {    }  public:  MyVector()  : data(nullptr)  , \_size(0)  , \_capacity(0)  {  }  ~MyVector()  {  delete[] data;  }  MyVector(const MyVector& other)  {  }  MyVector& operator=(const MyVector& other)  {    }  void push\_back(const T& value)  {    }  void pop\_back()  {    }  T& operator[](size\_t index) { return data[index]; }  const T& operator[](size\_t index) const { return data[index]; }  T& back() { return data[\_size - 1]; }  const T& back() const { return data[\_size - 1]; }  size\_t size() const { return \_size; }  bool empty() const { return \_size == 0; }  T\* begin() { return data; }  const T\* begin() const { return data; }  T\* end() { return data + \_size; }  const T\* end() const { return data + \_size; }  }; |

## 7.2 主函数设计

主函数 main 控制程序的执行流程，处理用户输入和显示输出结果。

用户交互：

程序首先显示操作菜单，用户可以选择创建电网顶点、添加电网的边、构建生成方案或退出程序。该菜单通过循环显示，直到用户选择退出。

根据用户的选择，程序执行相应的操作。例如，当用户选择创建电网顶点时，程序调用ALGraph类的CreatGraph方法来创建图的顶点。

操作流程：

用户首先创建电网顶点，然后添加电网的边，最后构建生成方案。在每个步骤中，程序都会提供相应的提示信息，并处理用户输入的数据。

在构建生成方案时，程序会输出总工程造价最低的电网连接方案。该方案通过遍历mst数组并使用MyString类的输出运算符来显示每条边的信息。

|  |
| --- |
| 附件、main函数部分代码 |
| int main()  {  int generated = 0;  bool quit = false;  char op;  ALGraph grapher;  while (!quit) {  cout << "菜单" << endl;  cout << "\n请输入选项\n";  cin >> op;  switch (op) {  case 'A':  if (!generated) {  grapher.CreatGraph();  generated++;  } else {  cout << "电网已生成\n";  }  break;  case 'B':  if (generated == 1) {  grapher.AddEdges();  generated++;  } else {  cout << "电网未生成\n";  }  break;  case 'C':  if (generated == 2) {  grapher.CreatMST();  } else {  cout << "电网未生成\n";  }  break;  case 'D':  quit = true;  break;  default:  cout << "无此选项，请重输\n";  cin.clear();  cin.ignore(MAX\_SIZE, '\n');  break;  }  }  return 0;  } |

## 7.3 构建最小生成树

构建最小生成树的过程如下：

初始化：

用户输入小区数量和名称，程序创建图的顶点。该步骤通过CreatGraph方法实现，用户输入的每个小区名称都被存储在MyString对象中，确保了字符串操作的安全性。

用户输入每对小区之间的电网线路长度，程序添加边到图中。该步骤通过AddEdges方法实现，用户输入的每条边的长度都被存储在ArcNode对象中，并添加到对应顶点的邻接表中。

添加边到优先队列：

程序遍历每条边，将它们添加到优先队列中。该步骤通过addEdgesToPQ方法实现，边按照长度从小到大的顺序被添加到MyPriorityQueue中。

选择最小代价边：

程序从优先队列中依次取出代价最小的边，确保添加的边不会形成环。该步骤通过循环实现，每次循环都从优先队列中取出最小边，并检查是否形成环。

重复此过程，直到选择了n-1条边。该步骤通过检查mst数组的大小来控制，当大小达到n-1时停止循环。

输出结果：

程序输出最终的电网连接方案，包括每条线路的起始小区、结束小区和长度。该步骤通过遍历mst数组并使用MyString类的输出运算符来实现，确保了输出的可读性。

通过以上实现，我成功构建了一个电网造价模拟系统，该系统能够确保城市中n个小区之间的电网相互接通，并且总工程造价最低。系统的实现不仅提高了电网建设的效率，还降低了成本，具有良好的实用价值。

|  |
| --- |
| 附件、CreatMST()函数 |
| void CreatMST()  {  MyString startName;  cout << "请输入开始建立电网的节点名称\n";  while (1) {  cin >> startName;  if (SearchVex(startName)) {  break;  } else {  cout << "无此节点，请重输\n";  }  }  MyVector<bool> visited;  for (int i = 0; i < vexnum; ++i) {  visited.push\_back(false);  }  MyVector<Road> mst;  MyPriorityQueue<Road, MyVector<Road>, CompareRoad> pq;  int startIndex = SearchVex(startName) - 1;  visited[startIndex] = true;  addEdgesToPQ(startIndex, pq, visited);  while (!pq.empty() && mst.size() < size\_t(vexnum - 1)) {  Road minRoad = pq.top();  pq.pop();  int secondIndex = SearchVex(minRoad.second) - 1;  if (!visited[secondIndex]) {  visited[secondIndex] = true;  mst.push\_back(minRoad);  addEdgesToPQ(secondIndex, pq, visited);  }  }  cout << "电网建立结果为：\n";  for (size\_t i = 0; i < mst.size(); ++i) {  const Road& road = mst[i];  cout << road.first << " - " << road.second << ": " << road.length << endl;  }  } |

# 项目总结

在本次课程设计中，我成功地设计并实现了一个电网造价模拟系统，旨在为城市中的n个小区构建一个总工程造价最低的电网连接方案。通过这个项目，我不仅巩固了对数据结构和算法的理解，还提升了面向对象编程和系统设计的能力。以下是对整个项目的总结：

## 8.1 学习体会

理论与实践相结合：通过将课堂上学到的数据结构和算法知识应用到实际项目中，我深刻体会到了理论知识在解决实际问题中的重要性。特别是在实现最小生成树算法时，对图论和排序算法的理解显得尤为重要。

面向对象编程：项目让我更加熟悉了面向对象的设计原则，如封装、继承和多态。通过定义清晰的类和对象，我提高了代码的可读性和可维护性。例如，ALGraph类封装了图的所有相关操作，MyString和MyVector类分别封装了字符串和动态数组的操作。

系统设计经验：在项目开发过程中，我学习了如何规划和设计一个完整的系统，包括需求分析、系统架构设计、模块划分等。特别是在系统设计阶段，我需要考虑系统的结构、模块划分、接口设计等问题，这对于提高我的系统分析和设计能力具有重要意义。

## 8.2 技术收获

C++编程技能：通过本项目，我提高了C++编程技能，特别是在STL的使用、异常处理和输入输出操作等方面。我学会了如何自定义模板类，如MyPriorityQueue和MyVector，来模拟标准库中的容器和算法。

数据结构应用：我深入理解了链表、优先队列等数据结构的特点和适用场景，掌握了它们在实际问题中的应用方法。特别是在处理动态数据时，优先队列结构的灵活性和高效性显得尤为重要。

算法实现：项目中实现了多种算法，如排序算法、查找算法等，提高了我的算法设计和优化能力。在构建最小生成树的过程中，我深入理解了克鲁斯卡尔算法的实现细节，并成功应用于电网造价模拟系统中。

## 8.3 问题与挑战

性能优化：在处理大量数据时，如何优化算法和数据结构以提高系统性能是一个挑战。我通过优化优先队列操作和算法逻辑来解决这一问题，例如，通过heapify\_up和heapify\_down方法来维护堆的性质，确保优先队列的操作效率。

错误处理：在用户输入错误或系统异常情况下，如何保证系统的健壮性是一个难题。我通过增加输入验证和异常捕获机制来提高系统的稳定性，例如，在CreatGraph和AddEdges方法中添加了对用户输入的验证。

用户界面设计：设计一个用户友好的界面也是项目中的一个挑战。我通过不断迭代和改进用户界面，使其更加直观和易用，例如，通过清晰的提示信息和菜单选项来引导用户进行操作。

## 8.4 改进方向

功能扩展：未来可以增加更多功能，如数据导出、数据导入、成绩查询等，使其更符合实际应用需求。这将使系统更加灵活，能够适应不同的使用场景。

性能提升：可以探索更高效的数据结构和算法，进一步提高系统的处理速度和响应能力。例如，可以考虑使用并查集数据结构来优化最小生成树的构建过程。

用户体验优化：可以进一步优化用户界面和交互设计，提供更加友好和直观的用户体验。例如，可以引入图形化界面，使用户操作更加直观。

安全性增强：考虑到系统处理的是电网建设的敏感信息，未来可以加入更多的安全特性，如数据加密、访问控制等，以保护数据的安全性和完整性。

## 8.5 结论

总的来说，这次课程设计是一次宝贵的学习经历。它不仅让我将所学知识应用于实践，还让我学会了如何分析和解决实际问题。通过团队合作，我克服了各种挑战，最终完成了一个功能完善、性能良好的电网造价模拟系统。我相信，这次经历将对我未来的学习和工作产生积极的影响。