数据结构课程设计

项目说明文档

电网建设造价模拟

软件工程

张靖凯

2151396

目录

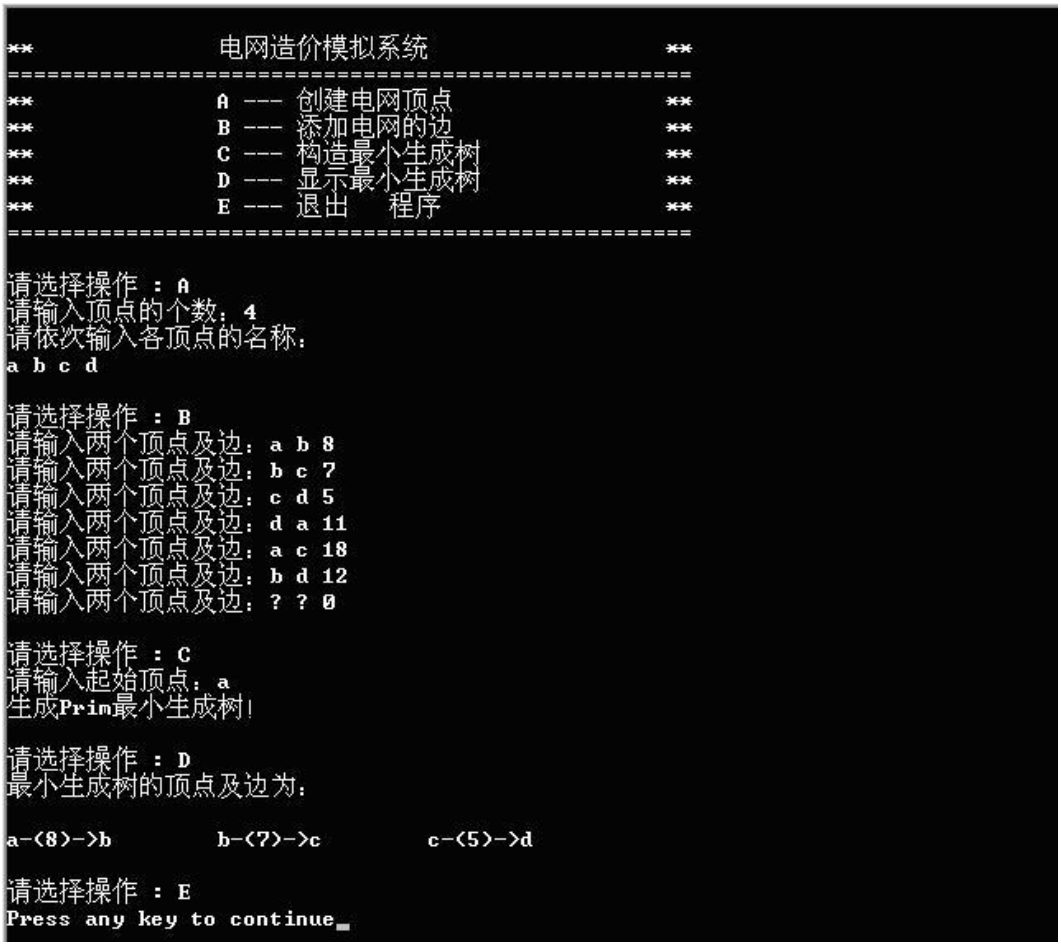
项目简介

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。

项目功能要求

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。

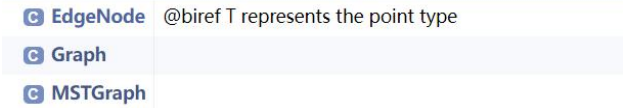
项目示例



项目设计

数据结构设计

总共设计了三个类，分别为 EdgeNode 类、Graph 类、MSTGraph 类，MSTGraph 继承自 Graph 类，EdgeNode 是上述两类的成员。



EdgeNode 类

Public Member Functions

	EdgeNode ()
	EdgeNode (T rstart, T rend, int rweight)

Public Attributes

T	**start**
T	**end**
int	**weight**

模板参数为图节点类型，如 char，int 型。

Graph 类

Public Member Functions

	Graph ()
virtual	**~Graph** ()
int	**convertToIndex** (T nodeName)
void	**createNode** ()
void	**createEdge** ()

Protected Attributes

int	**verticesNum** 定点数（只能赋值一次） More...
int	**edgesNum** 边数 More...
int **	**edgeMatrix** 邻接矩阵 More...
T *	**verticesList** T类型与数组下标对应 More...
bool	**haveCreateNodeOnce**

Static Protected Attributes

static const int	**maxWeight** = INT_MAX >> 1 最大权值 More...

Private Member Functions

void	**_addNode** ()
void	**_initMatrix** ()
bool	**_judgeNodeRepeat** ()

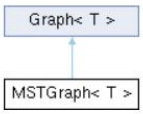
供外部调用的函数只有两个，createNode 函数和 createEdge 函数，分别为图创造节点和边。

内部成员主要为邻接矩阵和 T 类型与 int 的映射数组。

HaveCreateNodeOnce 是让 createNode 函数只建立一次动态数组，防止多次创建后析构时有内存泄漏，作用类似 C++11 中的 call_once 函数。

MSTGraph 类

Inheritance diagram for MSTGraph< T >:



Public Types

using MyBase = Graph< T >
using EdgeNode = EdgeNode< T >

Public Member Functions

MSTGraph ()
virtual ~MSTGraph ()
void createPrimTree ()
void printPrimTree ()
Public Member Functions inherited from Graph< T >

Private Member Functions

void __createPrimTree ()

Private Attributes

EdgeNode * edges
bool edgesCanBePrinted

Additional Inherited Members

Protected Attributes inherited from Graph< T >
Static Protected Attributes inherited from Graph< T >

本类继承自 Graph 类，其中派生类的数据成员 edges 是储存了构建 prim 最小生成树的所有边;根据题意设置了两个 public 函数分别为 createPrimTree 和 printPrimTree 函数。

数据结构关键代码

在该问题中最重要的就是动态内存申请时容易造成内存泄漏，因为添加节点时节点数不在 Graph 类中的构造函数进行初始化。

```
void Graph<T>::createEdge()
{
    if (verticesNum == 0)
    {
        cout << "无顶点!请先创建顶点\n";
        return;
    }
    // 如果已经动态创建就略过
    if (edgeMatrix == nullptr)
        __initMatrix();
       ...
}
```

addNode 同理：

```
template<typename T>
void Graph<T>::__addNode()
{
    cout << "请输入各顶点的名称(顶点个数为"<< verticesNum <<"): ";
    for (int i = 0; i < verticesNum; i++)
        cin >> verticesList[i];

    if (__judgeNodeRepeat()) // 有重复顶点就可以重新输入
    {
        haveCreateNodeOnce = false;
    }
}
```

解题关键代码

dist 数组记录该节点到已经构建好的树的最小路径值；st 数组记录是否该节点已经被加入树；pre 数组记录该节点的前一个节点；edges 数组记录所有树中的边；res 记录树的所有权值。

第一次 for 循环找出不在树中的节点到树最小的节点为 t，然后遍历所有不在树中的节点找出 dist 最小的值，加入树中，遍历 n 次即可构建出最小生成树。

```
    int* dist = new (std::nothrow)int[n];
    int* st   = new (std::nothrow)int[n];
    int* pre  = new (std::nothrow)int[n];
    edges = new EdgeNode[n];
    int res = 0;
    int cnt = 0; // edges 的下标
    if (dist != nullptr && st != nullptr && pre != nullptr)
    {
        for (int i = 0; i < n; i++)
        {
            dist[i] = MyBase::maxWeight;
            st[i] = 0;
            pre[i] = 0;
        }
        dist[k] = 0;
        for (int i = 0; i < n; i++)
        {
            int t = -1;
            for (int j = 0; j < n; j++)
```

```
            {
                if (!st[j] && (t == -1 || dist[j] < dist[t]))
                    t = j;
            }
            if (t != k)
                edges[cnt++] = EdgeNode(MyBase::verticesList[pre[t]],
MyBase::verticesList[t], MyBase::edgeMatrix[pre[t]][t]);
            st[t] = 1;
            res += dist[t];
            for (int j = 0; j < n; j++)
            {
                if (dist[j] > MyBase::edgeMatrix[t][j] && !st[j])
                {
                    dist[j] = MyBase::edgeMatrix[t][j];
                    pre[j] = t;
                }
            }
        }
        delete[] dist;
        delete[] st;
        delete[] pre;
```

设计亮点

代码注释规范

采用了 doxygen 注释规范，对类、函数等有简要说明，命名采用驼峰命名法，类内的各个声明规范，方便本人回顾之前写过的代码和 code reviewer 查看，使 API 规范，帮助这个开发流程高效、规范地进行。

```
template<typename T = char>
class Graph
{
protected:
    int verticesNum;                          ///< 定点数（只能赋值一次）
    int edgesNum;                             ///< 边数
    static const int maxWeight = INT_MAX >> 1; ///< 最大权值
    int** edgeMatrix;                         ///< 邻接矩阵
    T* verticesList;                          ///< T类型与数组下标对应
```

输入错误处理

```
请选择操作：  A
请输入顶点的个数：  aaaaaa
顶点输入错误
```

```
请选择操作：  C
还未创建结点，请先创建结点

请选择操作：  D
还未创建结点，请先创建结点

请选择操作：  B
无顶点!请先创建顶点
```

```
请选择操作：  A
请输入顶点的个数：  C
顶点输入错误

请选择操作：  B
无顶点!请先创建顶点

请选择操作：  C
还未创建结点，请先创建结点
```

项目测试

```
请选择操作：  A
请输入顶点的个数：  4
请输入各顶点的名称(顶点个数为4)：a b c d

请选择操作：  B
请输入两个顶点及边（结束请输入0 0 0）：  a b 1
请输入两个顶点及边（结束请输入0 0 0）：  a c 6
请输入两个顶点及边（结束请输入0 0 0）：  a d 4
请输入两个顶点及边（结束请输入0 0 0）：  b c 9
请输入两个顶点及边（结束请输入0 0 0）：  b d 2
请输入两个顶点及边（结束请输入0 0 0）：  c d 3
请输入两个顶点及边（结束请输入0 0 0）：  0 0 0

请选择操作：  C
请输入最小生成树的起始顶点：  c
生成Prim最小生成树！

请选择操作：  D
最小生成树的顶点及边为：
c--<3>-->d    d--<2>-->b    b--<1>-->a
```

Linux 测试运行（Ubuntu）

```
kk@LAPTOP-UJDPHKT8:~/dataStruct/homework8$ make
g++    -c -o h8.o h8.cpp
g++ -o run h8.o
kk@LAPTOP-UJDPHKT8:~/dataStruct/homework8$ ./run
**********************************************
**                                          **
**              电网造价模拟系统            **
**                                          **
**********************************************
**              A --- 创建电网顶点          **
**              B --- 添加电网的边          **
**              C --- 构造最小生成树        **
**              D --- 显示最小生成树        **
**              E --- 退出程序              **
**                                          **
**********************************************


请选择操作：  A
请输入顶点的个数：  3
请输入各顶点的名称(顶点个数为3)：a b c

请选择操作：  B
请输入两个顶点及边（结束请输入0 0 0）：  a b 4
请输入两个顶点及边（结束请输入0 0 0）：  a c 3
请输入两个顶点及边（结束请输入0 0 0）：  b c 7
请输入两个顶点及边（结束请输入0 0 0）：  0 0 0

请选择操作：  C
请输入最小生成树的起始顶点：  b
生成Prim最小生成树！

请选择操作：  D
最小生成树的顶点及边为：
b--<4>-->a    a--<3>-->c
```