

# 数据结构项目八文档

同济大学 软件学院 15级2班 1552651 王依睿

---

- [使用说明](#)
  - [操作手册](#)
    - [开始](#)
    - [创建电网顶点](#)
    - [添加电网的边](#)
    - [构造最小生成树](#)
    - [显示最小生成树](#)
    - [退出程序](#)
    - [整体预览](#)
  - [注意事项](#)
- [- 将图中输入两个顶点及边全部输入后，输入“??0”表示输入结束。](#)
- [概述](#)
  - [项目功能要求](#)
  - [程序设计目的](#)
  - [算法思路](#)
  - [数据结构](#)
  - [文件目录](#)
- [实现](#)
  - [Prim算法](#)
    - [描述](#)
    - [时间复杂度](#)
    - [Prim算法实现最小生成树代码实现](#)

## 使用说明

---

### 操作手册

# 开始

运行程序后，显示电网造价模拟系统的操作说明。



## 创建电网顶点

- 选择操作A。
- 输入顶点个数。
- 依次输入顶点名称。



## 添加电网的边

- 选择操作B。
- 输入两个顶点及边。



## 构造最小生成树

- 选择操作C。
- 输入起始顶点。



- 输出“生成prim最小生成树！”字样，表示程序已经用prim方法生成最小生成树。



## 显示最小生成树

- 选择操作D。



- 输出最小生成树的顶点及边。



## 退出程序

选择操作E。



## 整体预览



## 注意事项

- 输入的图必须是连通图。
- 顶点个数最多不能超过100人。
- 顶点名称在程序内的存储类型为string。

- 将图中输入两个顶点及边全部输入后，输入“?? 0”表示输入结束。

## 概述

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

提示：输入的图必须是连通图，采用邻接矩阵或邻接表表示法表示图，定义一个最小堆用来得到权值最小的边，构造最小生成树(Prim算法)实现

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。

## 程序设计目的

实现一个生成最小生成树的简单程序，练习使用邻接表实现图的存储。

## 算法思路

以Prim算法为基本算法，实现生成最小生成树。

## 数据结构

- 将prim算法中各顶点状态封装成一个结构体。
- 利用由二维数组实现的邻接表实现图的存储。
- 利用一维数组存储顶点名字、顶点是否被访问过、顶点在最小生成树中的父亲节点、prim算法中各顶点状态。
- 利用map存储最小生成树的边长度及边的两顶点信息。
- 利用priority\_queue（优先队列）存储未知顶点到已知顶点的最短相连距离。

## 文件目录

- 8\_1552651\_wangyirui.cpp（主文件）
- 8\_1552651\_wangyirui.exe（可执行文件）
- 8\_1552651\_wangyirui.pdf（项目文档）

---

## 实现

### Prim算法

#### 描述

从单一顶点开始，普里姆算法按照以下步骤逐步扩大树中所含顶点的数目，直到遍及连通图的所有顶点。

1. 输入：一个加权连通图，其中顶点集合为 $V$ ，边集合为 $E$ ；
2. 初始化： $V_{\text{new}} = \{x\}$ ，其中 $x$ 为集合 $V$ 中的任一节点（起始点）， $E_{\text{new}} = \{\}$ ；
3. 重复下列操作，直到 $V_{\text{new}} = V$ ：
  1. 在集合 $E$ 中选取权值最小的边 $(u, v)$ ，其中 $u$ 为集合 $V_{\text{new}}$ 中的元素，而 $v$ 则是 $V$ 中没有加入 $V_{\text{new}}$ 的顶点（如果存在有多条满足前述条件即具有相同权值的边，则可任意选取其中之一）；
  2. 将 $v$ 加入集合 $V_{\text{new}}$ 中，将 $(u, v)$ 加入集合 $E_{\text{new}}$ 中；
4. 输出：使用集合 $V_{\text{new}}$ 和 $E_{\text{new}}$ 来描述所得到的最小生成树。

#### 时间复杂度

最小边、权的数据结构	时间复杂度（总计）
邻接矩阵、搜索	$O(V^2)$
二叉堆（后文伪代码中使用的数据结构）、邻接表	$O((V + E) \log(V)) = O(E \log(V))$
斐波那契堆、邻接表	$O(E + V \log(V))$

本程序利用邻接矩阵实现，故时间复杂度为 $O(V^2)$

## Prim算法实现最小生成树代码实现

```
#include<iostream>
#include<string>
#include<map>
#include<queue>
using namespace std;
const int maxn = 100;

#define INF 1 << 30

struct node {
    int vnum;
    int key;//到已经访问过的顶点边的长度最小值
    friend bool operator<(node a, node b) {
        return a.key > b.key;
    }
};

int n;//顶点个数
int rootnum;//开始点，即最小生成树的根节点
int graph[maxn][maxn]);//邻接表，存储两点间是否相连以及两点间边的长度的信息
string vname[maxn]);//顶点名字对应节点序号
bool visited[maxn]);//顶点是否访问过
int dad[maxn]);//顶点在最小生成树中的父亲节点
node v[maxn]);//prim算法中各顶点状态
map<int, int> tree;//存储最小生成树的边及边的两顶点
priority_queue <node> q;//选择已知顶点

void prim(int beginnum){

    for (int i = 0; i < n; ++i) {
        v[i].vnum = i;
        v[i].key = INF;
        dad[i] = -1;
        visited[i] = false;
```

```

}

v[beginnum].key = 0;
q.push(v[beginnum]);

while (!q.empty()) {

    node cur_nd = q.top();
    q.pop();
    int cur_vnum = cur_nd.vnum;
    int dad_vnum = dad[cur_vnum];
    if (dad_vnum != -1)
        tree.insert(make_pair(dad_vnum, cur_vnum));
    if (visited[cur_vnum])
        continue;
    visited[cur_vnum] = true;

    for (int i = 0; i < n; ++i) {

        if (i != cur_vnum && !visited[i] && graph[cur_vnum][i] < v[i].key) {

            dad[i] = cur_vnum;
            v[i].key = graph[cur_vnum][i];
            q.push(v[i]);
        }
    }
}
}

```

```

int main() {

    printf("\n**          电网造价模拟系统          **\n");
    printf("=====\n");
    printf("**          A---创建电网顶点          **\n");
    printf("**          B---添加电网的边          **\n");
    printf("**          C---构造最小生成树          **\n");
    printf("**          D---显示最小生成树          **\n");
    printf("**          E---退出程序          **\n");
    printf("=====\n");

    char op;
    while (true) {

        printf("\n请选择操作: ");
        scanf("%c", &op);

        if (op == 'A') {

```

```

printf("请输入顶点个数: ");
scanf("%d", &n);
printf("请依次输入个顶点名称: \n");
for (int i = 0; i < n; ++i)
    cin >> vname[i];
for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
        graph[i][j] = INF;
getchar();
}

if (op == 'B') {
    string v1, v2;
    int v1num, v2num;
    int len;
    while (true) {
        printf("请输入两个顶点及边: ");
        cin >> v1 >> v2 >> len;
        if (v1 == "?")
            break;
        for (int i = 0; i < n; ++i) {
            if (vname[i] == v1)
                v1num = i;
            if (vname[i] == v2)
                v2num = i;
        }
        graph[v1num][v2num] = len;
        graph[v2num][v1num] = len;
    }
    getchar();
}

if (op == 'C') {
    string root;
    printf("请输入起始顶点: ");
    cin >> root;
    getchar();
    for (int i = 0; i < n; ++i) {
        if (vname[i] == root) {
            rootnum = i;
            break;
        }
    }
    prim(rootnum);
    printf("生成prim最小生成树! \n");
}

if (op == 'D') {
    printf("最小生成树的顶点及边为: \n");

```

```

        for (auto vpair : tree) {
            int v1num = vpair.first;
            int v2num = vpair.second;
            cout << vname[v1num] << "-<" << graph[v1num][v2num] << ">->" << vname[v2num] << " ";
        }
        cout << endl;
        getchar();
    }
    if (op == 'E')
        break;
}
}

```