Tongji University

Data Structure Course Curriculum Design Document

Task8: Grid Construction Cost Simulation System

Author:
嘉杰 李
Jiajie Li

Supervisor:
颖 张
Prof. Ying Zhang

A companion use documentation for a program submitted in fulfillment of the requirements for the Data Structure curriculum design

in the

School of Software Engineering, Tongji

December 27, 2019

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”

Dave Barry

Contents

Chapter 1

Analysis

1.1 Background Analysis

The laying of power grid lines is a problem that power companies pay great attention to. The cost of wires between two places at different distances is different. A reasonable laying of power grid lines can save a lot of costs for power companies. Therefore, it is very difficult to solve the problem of power grid simulation systems. Great significance.

1.2 Functional Analysis

Assume that there are n communities in a city. To realize that the power grids between the n communities can be connected to each other, the power grid between the n communities in this city is constructed so as to minimize the total project cost. Please design a cost solution that can meet the requirements.

As a simple grid cost simulation system, a grid line can be set up between each district, and corresponding economic costs must be paid. There can be at most n (n-1) / 2 lines between n cells, and choosing n-1 of them can minimize the total cost.

The user is required to input the number of vertices and the names of the vertices, input the weights of the edges of each set of vertices in turn, and the system generates the minimum spanning tree. The user controls the output of the minimum spanning tree and the minimum cost.

Chapter 2

Design

2.1 Data structure design

As described in the functional analysis above, this problem is a typical minimum spanning tree problem. The Prim algorithm is used to solve the minimum spanning tree. The adjacency matrix is used as the graph storage method. During the construction process, two auxiliary arrays are also set: lowcost [] to store The current minimum weight of each vertex in the spanning tree vertex set to each edge outside the spanning tree; nearvex [] records which vertex in the spanning tree vertex set is closest to the vertex in the set (that is, the smallest weight). Stores the minimum spanning tree in an array of boundary values.

2.2 Member function design

Vector Class:

```
template <class T>
class Vector {
public:
   Vector(int size=0):theSize(size),theCapacity(size+SPACE_CAPACITY){ data = new
       T[theCapacity];} // 构造函数
   ~Vector(void) { delete[] data;} // 析构函数
   Vector& operator=(Vector& other){
      // 判断是否为自身赋值
      if (this == &other)
         return *this;
      else {
         delete[]data;
         theSize = other.theSize;
         theCapacity = other.theCapacity;
         this->data = new T[theCapacity];

         for (int i = 0; i < theSize; ++i) {
            data[i] = other.data[i];
         }
         return *this;
```

```
            }
        } // 重载赋值函数
        Vector(Vector& other) :theSize(0),theCapacity(0),data(NULL){ *this=other; }//
            复制构造函数

        // 重新分配空间大小函数
        void reServe(int newCapacity) {
            if (newCapacity <= theCapacity)
                return;

            T *temp = data;
            data = new T[newCapacity];
            theCapacity=newCapacity;
            //std::cout << newCapacity << std::endl;
            for (int i = 0; i < theSize; ++i)
                data[i] = temp[i];
            delete[] temp;
        }
        // push_back函数
        void push_back(T val) {
            if (theSize == theCapacity)
                reServe(2 * theCapacity + 1);
            data[theSize++] = val;
        }
        bool pop_back(){
            if (!theSize) {
                std::cerr << "Fail to pop back, the vector is empty!\n";
                return false;
            }
            theSize--;
            return true;
        }
        bool Delete(int ind){
            if (ind<0||ind>=theSize){
                std::cerr << "Fail to delete, the ind is out of range!\n";
                return false;
            }
            for (int i=ind;i<theSize-1;++i) data[i]=data[i+1];
            return pop_back();
        }
        T& operator[] (int n) {return data[n];}
        T *data;
        int size() const {return theSize;}
    private:
        const int SPACE_CAPACITY = 16;
        int theCapacity;
        int theSize;
    };
```

Heap Class:

```cpp
template <typename T>
class Heap{
private:
    Vector<T> data;
public:
    Heap(){};
    Heap(Vector<T> v);
    void adj(int curNode=0);
    void push_heap(T val);
    void push_heap(T val, int pos);
    void ergodic();
    T pop_heap();
    void swap(T &x,T &y){
        T temp;
        temp=x;
        x=y;
        y=temp;
    }
    int size() {return data.size();}
};

template <typename T>
Heap<T>::Heap(Vector<T> v){
    data = v;
    for (int i=size()-1; i>=0; --i) adj(i);
}

template <typename T>
T Heap<T>::pop_heap(){
    T temp=data[0];
    swap(data[0], data[size()-1]);
    data.Delete(size()-1);
    adj();
    return temp;
}

template <typename T>
void Heap<T>::adj(int curNode){
    int left=2*curNode+1;
    int right=2*curNode+2;
    int min=curNode;
    if (left<data.size()) min=data[min]<data[left]?min:left;
    if (right<data.size()) min=data[min]<data[right]?min:right;
    if (min!=curNode) swap(data[min],data[curNode]), adj(min);
}

template <typename T>
```

```
void Heap<T>::push_heap(T val){
    data.push_back(val);
    push_heap(val, size()-1);
}

template <typename T>
void Heap<T>::push_heap(T val, int pos){
    if (pos==0) return;
    int father = (pos-1)/2;
    if (data[father]>data[pos]) swap(data[father],data[pos]), push_heap(val,
        father);
}
```

2.3 System design

The system first calls the InitScreen () function to initialize the screen, and then builds a graph and a minimum spanning tree according to the code entered by the user.

Chapter 3

Implementation

3.1 Insert edges into graph Function

3.1.1 Insert edges into graph core code

```
cout << "Please input the name to establish family: ";
          cin >> name;
          if (geneTree.find(name)== nullptr) {
             cerr << "The name '" << name << "' doesn't exist! try again plz" <<
                 endl;
             break;
          }
          __treenode<string>* father = geneTree.find(name);
          cout << "Please input the number of " << name << "'s offsprings: ";
          cin >> n;
          Vector<__treenode<string> *> temp;
          cout << "Please input the names of " << name << "'s offsprings in order
              : ";
          for (int i=0; i<n; ++i) {
             cin >> name_t;
             __treenode<string> *son = new __treenode<string>(name_t, father);
             temp.push_back(son);
          }
          geneTree.build(name, temp);
          cout << name << "'s first generation offsprings are: ";
          geneTree.outSpring(name);
          cout << endl;
```

3.2 Prim Function

3.2.1 Prim code

```
cout << "Please input the name to add offspring: ";
          cin >> name;
          if (geneTree.find(name)== nullptr) {
```

```
                    cerr << "The name '" << name << "' doesn't exist! try again plz" <<
                        endl;
                    break;
                }
                cout << "Please input the names of " << name << "'s offspring: ";
                cin >> name_t;
                geneTree.add(name, name_t);
                cout << name << "'s first generation offsprings are: ";
                geneTree.outSpring(name);
                cout << endl;
                break;
```

3.2.2 Prim flowchart

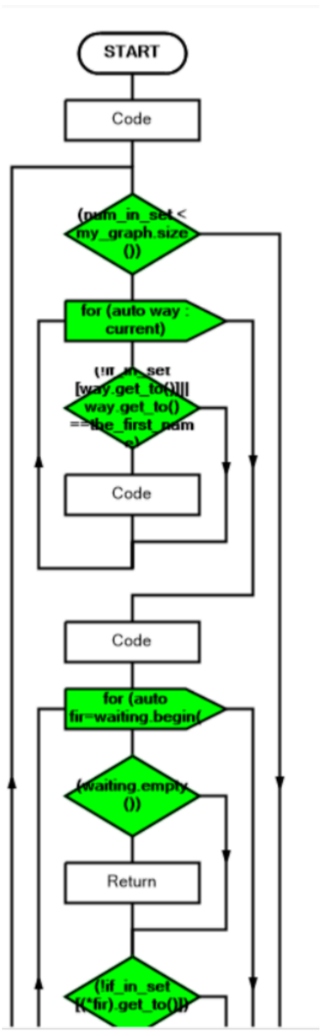

START
Code
(num_in_set < my_graph.size())
for (auto way : current)
(!if_in_set[way.get_to()]|| way.get_to()==the_first_name)
Code
Code
for (auto fir=waiting.begin(
(waiting.empty())
Return
(!if_in_set[(*fir).get_to()])

Figure 3.1: Prim flowchart

Chapter 4

Test

4.1 Function Tests

4.1.1 Normal

Test Case:

4

a b c d

a b 8

b c 7

c d 5

d a 11

a c 18

b d 12

Expected Result:

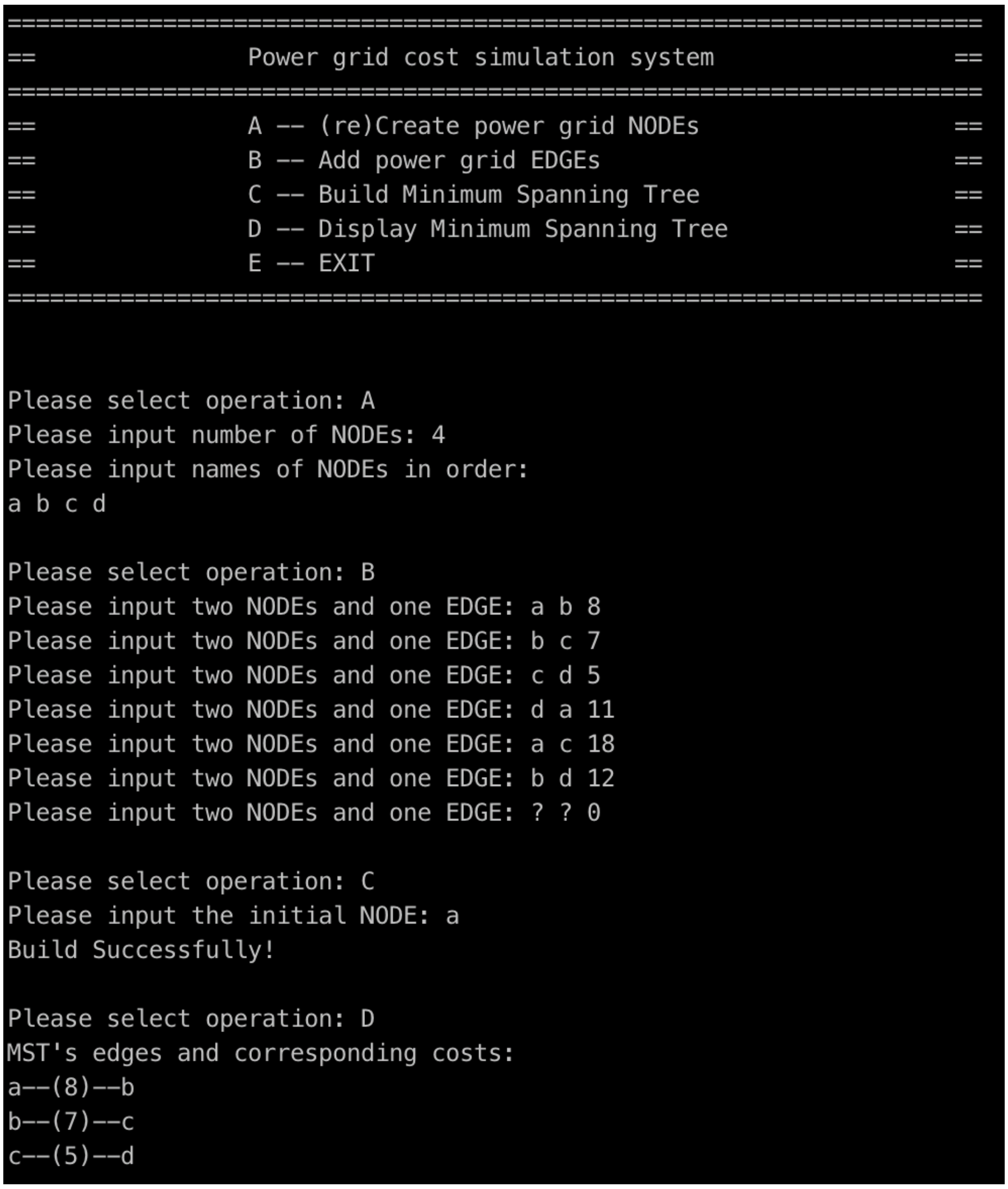
MST's edges and corresponding costs:

a–(8)–b

b–(7)–c

c–(5)–d

Program Result:



```
======================================================================
==                Power grid cost simulation system                 ==
======================================================================
==                A -- (re)Create power grid NODEs                  ==
==                B -- Add power grid EDGEs                         ==
==                C -- Build Minimum Spanning Tree                  ==
==                D -- Display Minimum Spanning Tree                ==
==                E -- EXIT                                         ==
======================================================================


Please select operation: A
Please input number of NODEs: 4
Please input names of NODEs in order:
a b c d

Please select operation: B
Please input two NODEs and one EDGE: a b 8
Please input two NODEs and one EDGE: b c 7
Please input two NODEs and one EDGE: c d 5
Please input two NODEs and one EDGE: d a 11
Please input two NODEs and one EDGE: a c 18
Please input two NODEs and one EDGE: b d 12
Please input two NODEs and one EDGE: ? ? 0

Please select operation: C
Please input the initial NODE: a
Build Successfully!

Please select operation: D
MST's edges and corresponding costs:
a--(8)--b
b--(7)--c
c--(5)--d
```

Figure 4.1: Screenshots

Appendix A

Frequently Asked Questions

A.1 Why this document looks so sparse?

This document is built using LaTeX and is arranged in book format. There may be blank pages before and after each chapter.

A.2 Why is there header files in this project?

I wrote basic stl header files, such as vector.h, etc. Except for the header files I wrote, the only header files used in all projects are iostream and string.h