

实验七 无向图及其应用

一、实验目的

1. 掌握基于邻接矩阵存储结构的图的定义与实现。
2. 掌握图的 Prim 算法和 Kruskal 算法的实现以及应用

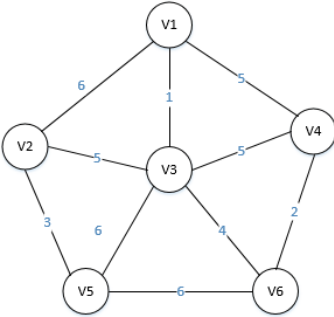
二、实验内容

项目名称：通信网构建

项目内容：在 n 个城市之间建立通信联络网，则连通 n 个城市只需要 $n-1$ 条线路。要求在最节省经费的前提下建立这个通信网。

- (1) 完成城市信息的输入。

表 1 城市信息

城市间关系图	城市编号	城市名称
	V1	北京
	V2	成都
	V3	武汉
	V4	上海
	V5	广州
	V6	深圳

- (2) 完成城市信息的编辑，包括城市以及城市间距离的增加，删除，信息修改等。
- (3) 允许用户指定下列两种策略进行通信网的构建
 - 1) 采用 Prim 算法进行通信网的构建；
 - 2) 采用 Kruskal 算法进行通信网的构建；

测试数据不限于此。

三、实验要求

1、线性表的应用

可以直接调用在前面课程中实现的线性表的类，该类中包含线性表的初始化操作，线性表中元素的增、删、改、查等功能。该线性表主要用来存放城市以及城市间距离信息，完成城市以及城市间距离的编辑功能，包括城市以及城市间距离的增加，删除，信息修改。

线性表的定义参考如图 1 所示。

```

template <class T> <T>
class SqList
{
private:
    T *elem; //保持不变, NULL 不存在
    int length; //实际存放元素的个数
    int listsize; //可以容纳的最大元素的个数
public:
    SqList();
    ~SqList();
    void InputList();
    void OutputList();
    Status Insert(int i, T e); //在i位置插入一个元素
    Status Delete(int i, T &e); //删除i位置的元素
    Status Update(int i, T e); //在i位置更新一个元素
    int Locate(T e); //根据元素查找在线性表中的位置
};

```

图 1 线性表的定义参考图

2、基于邻接矩阵存储结构的图结构的定义与实现

设计并实现基于邻接矩阵的图存储结构，需要包括顶点增删改，边增删改等方法。图的定义参考如图 2，图 3 所示。

```

#pragma once
#define numMAX 20
#define StrMAX 100
#define MAX 10000
#include "SqList.h"

//顶点信息
struct Vex
{
    char Code[StrMAX];
    char Name[StrMAX];
};

//边的信息
struct Edge
{
    Vex vex1;
    Vex vex2;
    int weight;
};

```

图 2 结构体定义

```

class Graph
{
private:
    int AdjMatrix[numMAX][numMAX]; //邻接矩阵
    SqList<Vex> Vexs; //点的集合
    SqList<Edge> Edges; //边的集合
    int VexNum; //点的个数
public:
    Graph();
    ~Graph();

    bool InsertVex(Vex svex);
    bool DeleteVex(Vex svex);
    bool UpdateVex(Vex svex);
    bool InsertEdge(Edge sedge);
    bool DeleteEdge(Edge sedge);
    bool UpdateEdge(Edge sedge);

    Edge GetEdge(char *vex1Code, char *vex2Code);
    Vex GetVex(char *vex1Code);
    void SetVexNum(int);

    int PrimMinTree(Edge aPath[]);
    int KruskalMinTree(Edge aPath[]);
};

```

图 3 类定义

3、Prim 算法的实现与应用

假设 $G=(V, E)$ 是一个具有 n 个顶点的带权无向连通图， $T(U, TE)$ 是 G 的最小生成树，其中 U 是 T 的顶点集， TE 是 T 的边集，则构造 G 的最小生成树 T 的步骤如下：

- 1) 初始状态， TE 为空， $U=\{v_0\}$ ， $v_0 \in V$ ；
 - 2) 在所有 $u \in U$ ， $v \in V-U$ 的边 $(u, v) \in E$ 中找一条代价最小的边 (u', v') 并入 TE ，同时将 v' 并入 U ；
- 重复执行步骤 (2) $n-1$ 次，直到 $U=V$ 为止。

4、Kruskal 算法的实现与应用

假设 $G=(V, E)$ 是一个具有 n 个顶点的带权无向连通图， $T(U, TE)$ 是 G 的最小生成树，其中 U 是 T 的顶点集， TE 是 T 的边集，则构造 G 的最小生成树 T 的步骤如下：

- 1) 置 U 的初值等于 V ， TE 的初值为空
- 2) 将图 G 中的边按权值从小到大的顺序依次选取：若选取的边未使生成树 T 形成回路，则加入 TE ，否则舍弃，直到 TE 中包含 $(n-1)$ 条边为止。

5、主函数的实现

主函数要求控制良好的界面操作，提示用户进行各种不同功能操作的选择。

四、实验注意事项

1. 使用 c++语言的模板类实现;
2. 设计具有通用性、可复用性,代码可读性强;
3. 实验分析和设计要有详尽的描述;
4. 在完成基本功能的基本上可以自己扩展其他功能。