

**课 程 设 计 报 告**

**题目： 中国行政区域图染色与信息查询**

**课程名称： 数据结构**

**专业班级：计算机卓越工程师1501**

**学 号： U201514482**

**姓 名： 魏硕**

**指导教师： 许贵平**

**报告日期： 2017.2.18**

**计算机科学与技术学院**

**任务书**

* **设计目的：**掌握图的存储结构与基本算法，通过解决较复杂的基于图模型的实际问题，提高学生对数据结构知识综合运用的技能与实践能力。
* **设计内容：**设计有效的逻辑数据结构与存储结构表示中国各行政区域的有关信息（如省会城市名，电话区号，人口数，地理位置等）及行政区域间的相邻关系、省会城市间的距离；分析与设计有效的算法对行政区域图进行染色，使每个行政区域染一种颜色且相邻的省份染不同颜色，而总的颜色数最少；另外如在全国省城之间建立通信网，构造费用最低的通信线路铺设方案。
* **设计要求：**

⑴从互联网或相关资料获取可靠的行政区域及其地理数据，有关数据与信息以文件形式存储，用无向网建模上述问题并以文件保存。

⑵界面上能够显示与输出求解结果，具有对各省份相关信息的查询功能。对主要算法进行理论复杂度分析，并实测其执行效率。

⑶在界面设计与其他功能上可自由发挥，行政区划母图如图1，可以供界面设计处理之用。

* **设计提示：**每个行政区域作为一个顶点，邻接矩阵作为主要存储结构，边的权值及信息设置兼顾染色与通信网构建需求；用回溯法设计染色算法，用典型求解最小生成树的算法解决最小费用通信网规划问题。顶点信息在涵盖上述要求信息之外还可作适当补充，通过输入顶点与边的信息建立无向连通网。
* **参考文献：**

[1] 严蔚敏, 吴伟民. 数据结构（C语言版）. 北京: 清华大学出版社,1997

[2] 王晓东. 计算机算法设计与分析. 北京: 电子工业出版社, 2007

[3] 严蔚敏, 吴伟民, 米宁. 数据结构题集（C语言版）. 北京: 清华大学出版社,1999

[4] 谢力军,李晓梅,何佳等. 基于模糊最小生成树的通信网络架设模型.吉首大学学报（自然科学版），2010,31(4):43~46

**目录**

任务书

[1引言 3](#_Toc476335268)

[1.1课题背景与意义 3](#_Toc476335269)

[2系统需求分析与总体设计 4](#_Toc476335270)

[2.1系统需求分析 4](#_Toc476335271)

[2.2系统总体设计 4](#_Toc476335272)

[3系统详细设计 6](#_Toc476335273)

[3.1有关数据结构的定义 6](#_Toc476335274)

[3.2主要算法设计 7](#_Toc476335275)

[4系统实现与测试 15](#_Toc476335276)

[4.1系统实现 15](#_Toc476335277)

[4.2系统测试 16](#_Toc476335278)

[5总结与展望 24](#_Toc476335279)

[5.1全文总结 24](#_Toc476335280)

[5.2工作展望 24](#_Toc476335281)

[6体 会 25](#_Toc476335282)

[7参考文献 26](#_Toc476335283)

[8附录 27](#_Toc476335284)

# 1引言

# 1.1课题背景与意义

在数据结构学习图后，没有进行大量的训练来巩固，对于图的认识还是停留在书本上比较浅显的位置。希望通过这个课题来掌握图的存储结构与基本算法，深入了解图的结构特点以及实现实际应用，通过解决较复杂的基于图模型的实际问题，对于图形成一个完整的认识。

实际问题即通过以中国各省份为图的顶点，以省份之间的交通网络为图的弧，交通联系代价为弧的权值构造一个图的模型。然后运用书上的最短路径的知识来构建代价最小的交通网络，从而以最小的代价生成一个能够联通全国的交通网络。关于该实现，目前最为广泛的算法主要是普里姆和克利斯卡尔算法，根据实际问题特点（为稠密图，3个顶点，100多条边），选择的最优算法为普里姆算法。通过染色问题实现地图着色，且相邻地点不可着相同颜色，求最少的着色方法，并简单验证四色猜想。该算法实现主要是回溯法解决染色问题，也可以通过这个桥梁来进一步了解回溯法的具体思想，了解其他典型回溯问题（比如八皇后问题）。

该课题主要是想通过以解决简单的实际问题模型来反过来理解图的一系列算法及图的实际意义，用理论知识来指导实际应用，并用实际应用反过来巩固理论知识。

# 2系统需求分析与总体设计

# 2.1系统需求分析

设计有效的逻辑数据结构与存储结构表示中国各行政区域的有关信息（如省会城市名，电话区号，人口数，地理位置等）及行政区域间的相邻关系、省会城市间的距离；分析与设计有效的算法对行政区域图进行染色，使每个行政区域染一种颜色且相邻的省份染不同颜色，而总的颜色数最少；另外如在全国省城之间建立通信网，构造费用最低的通信线路铺设方案。

# 2.2系统总体设计

本系统主要存储运算结构为一个存储图指针的数组，该数组可以同时操作20个图。图的结点信息包括图当前的结点数和弧数，以及一个结点数组的基地址。结点的基本信息包括顶点的值和指向第一条依附该顶点的弧的指针。而弧的信息包括弧指向的顶点的位置，弧的权值以及指向下一条弧的指针。

结点信息包括省名，省会城市名，人口数以及连接相关省的指针，弧信息包括弧尾以及弧的权值。

系统下的子菜单（即具体的实现函数）有：造图，销毁图，返回结点值，增加顶点或弧等20个子功能，并额外增加了文件的读入与写出功能。在这些图的基本操作外新增加生成最小生成树以及判断当前最少色数的函数。

主要模块为生成最小生成树模块和最少染色数模块。最小生成树模块希望通过输入图，返回图的最小生成树信息及执行时间，而最少染色数模块希望通过输入图，返回最少需要的染色数及每一个顶点的着色情况。

本系统提供一个简易的文本界面，在输入相应的数字后调用相应的函数，并执行下一步操作。具体菜单图及流程图如下：

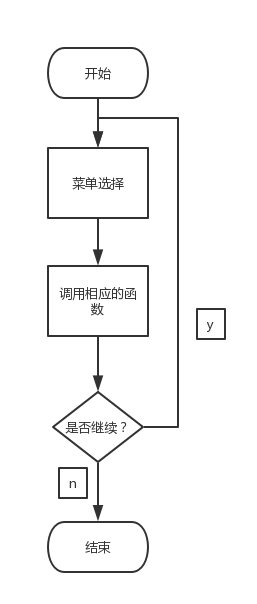


图2.1.1



图2.1.2

# 3系统详细设计

# 3.1有关数据结构的定义

系统需要处理的数据即图的数据，结点数据和弧相关数据：

（1）数据元素类型的定义：

typedef int status;

（2）有关图的定义：

typedef struct {//图信息

AdjList vertices;

int vexnum, arcnum; //图的当前顶点数和弧数

int kind; //图的种类标志

}aLGraph, \*ALGraph;

图结点包含当前定点数，图的种类和图的顶点数组的基地址。

typedef struct VNode { //结点信息

int num;//顶点编号

char name[20];//顶点指代地名

char captial[20];//顶点省会

int people;//顶点人数

char phone[15];

ArcNode \*firstarc; //指向第一条依附该顶点的弧的指针

}VNode, AdjList[MAX\_VERTEX\_NUM];

顶点包含顶点名称，省会名称，人口数和指向第一条依附该顶点的弧的指针。

typedef struct ArcNode { //弧的信息

int adjvex; //该弧所指向的顶点的位置

struct ArcNode \*nextarc; //指向下一条弧的指针

int Weight; //用来记录网的权值

}ArcNode;

弧包含弧指向的顶点的位置，弧的权值以及指向下一条弧的指针。

数据关联即图是由顶点和弧构成的，顶点存储在一个数组中，且结点信息中包括指向相邻结点的弧的头指针，方便遍历相邻结点，而弧信息中包括尾结点和弧的权值，它们的关系应该是自上而下，相互关联的。

# 3.2主要算法设计

（1）CreateGraph(ALGraph G)

**算法输入**：需要创建的图的指针

**函数调用**：定位函数

**算法输出**：创建好的图的指针

**算法处理步骤描述**：

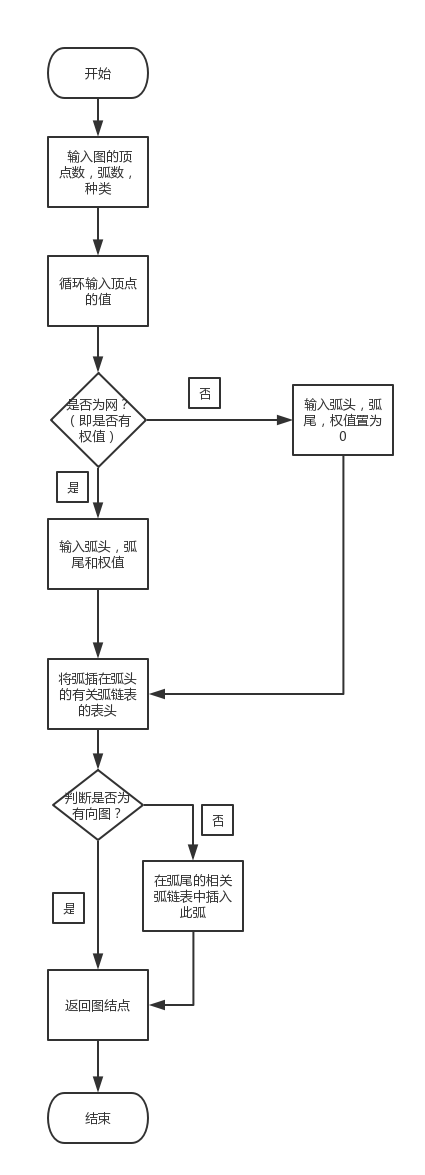


图3.2.1

**算法时间复杂度分析**：T（n）=O(n)

（2）DestroyGraph(ALGraph G)

**算法输入**：需要摧毁的图的指针

**算法输出**：修改后指针

**算法处理步骤描述**：

1. 首先判断传递的参数是不是空指针，为空指针直接返回。
2. 从第一个节点开始，对每一个结点进行销毁，并沿着与其有关的弧的指针将每一个弧结点都销毁。
3. 最后销毁掉最开始传入的指针，将头指针置为空，然后返回头指针。

**算法时间复杂度分析**：T（n）=O(n)

（3）PutVex(ALGraph G, VertexType v, VertexType ch)

**算法输入**：需要改变的图的指针，需要改变的顶点的值，改变的值

**函数调用**：定位函数

**算法输出**：实现状态（完成返回OK）

**算法处理步骤描述**：

1. 首先定位需要修改的结点的值，看这个节点是不是图中的，如果是，找出其位置，如果不是，直接返回错误。
2. 依据返回的位置，将修改的值赋给相应结点的数据域，并且返回OK。

**算法时间复杂度分析**：T（n）=O(n)

（4）NextAdjVex(ALGraph G, VertexType v, VertexType ch)

**算法输入**：需要查找的图的指针，指定顶点的值，指定顶点的值

**函数调用**：定位函数

**算法输出**：实现状态（完成返回OK）

**算法处理步骤描述**：

1. 首先定位传入的两个结点的值的位置，如果两个结点有一个不在图中，就返回错误。
2. 然后沿着与指定结点的有关的弧的指针搜索，直到找到第二个指定结点。
3. 如果没有找到，返回错误；如果找到且其后继不为空，则返回其后继的值。

**算法时间复杂度分析**：T（n）=O(n)

（5）DeleteVex(ALGraph G, VertexType ch)

**算法输入**：需要改变的图的指针，需要删除顶点的值

**函数调用**：定位函数

**算法输出**：实现状态（完成返回OK）

**算法处理步骤描述**：

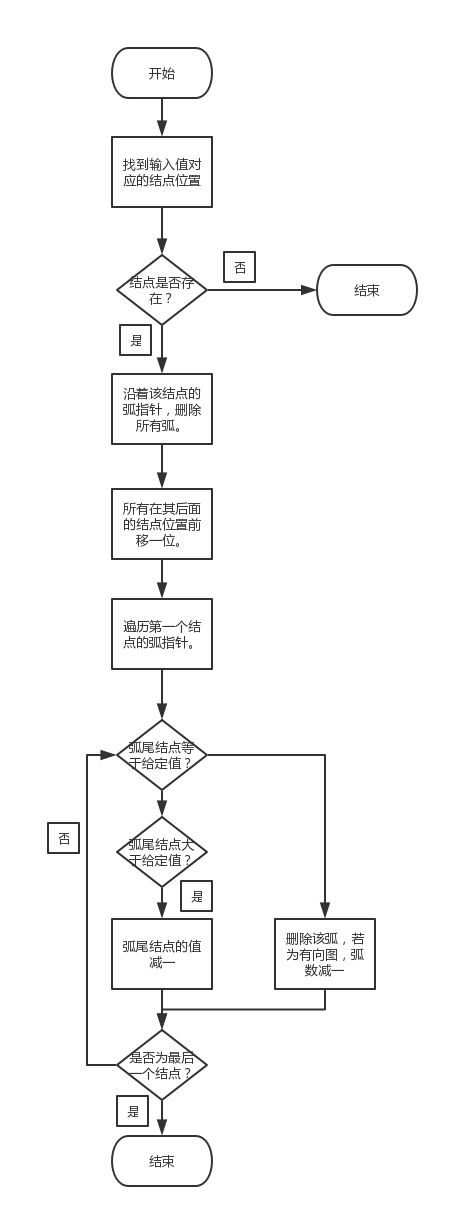


图3.2.2

需要注意结点是邻接表方式存储，所以要逐个前移。

**算法时间复杂度分析**：T（n）=O(n)

（6）InsertArc(ALGraph G, VertexType v1, VertexType v2)

**算法输入**：需要改变的图的指针，需要插入的弧的弧头和弧尾

**函数调用**：定位函数

**算法输出**：实现状态（完成返回OK）

**算法处理步骤描述**：

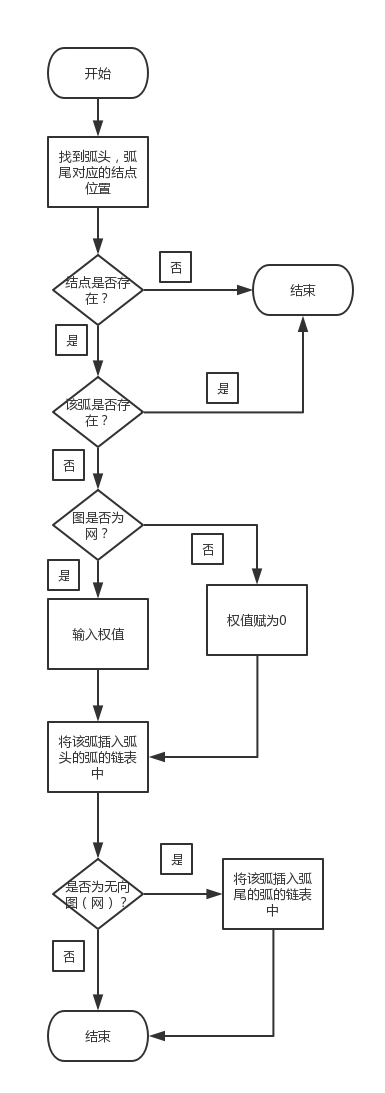


图3.2.3

注意到图的类型，采取不同处理

**算法时间复杂度分析**：T（n）=O(n)

（7）Minspantree(ALGraph T,char\* v)

**算法输入**：需要改变的图的指针，最小生成树的起始顶点

**函数调用**：无

**算法输出**：实现状态（完成返回OK）

**算法处理步骤描述**：

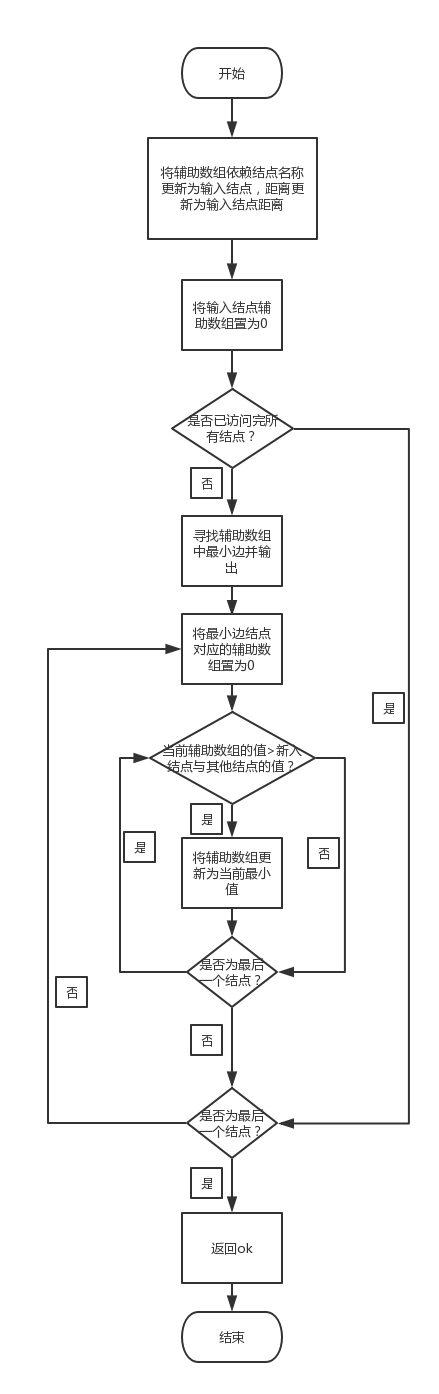
****

图3.2.4

**算法时间复杂度分析**：T（n）=O(n2)

（8）graphcolor(ALGraph T,int m)

**算法输入**：需要改变的图的指针，所需色数（由1开始递增）

**函数调用**：定位函数，判断是否有相邻结点同色函数

**算法输出**：需要最多色数

**算法处理步骤描述**：

1. 将颜色辅助数组置为0（0表示没有染色）
2. 从第一个顶点开始，用最小色数开始着色。判断有没有相邻结点同色，若有，则换下一种颜色。
3. 判断是否处理完所有结点，如果没有，判断当前着色是否符合要求（即是否超过色数m），超过则回到上一结点，没有超过则继续下一结点。
4. 处理完所有结点后，选择所有色素中最大的返回即为最小染色数。

**算法时间复杂度分析**：在最坏情况下，检查一个顶点是否与相邻结点同色并将其染色的复杂度为O（mn），共有n个结点，即总的复杂度为O（m\*n2）

T（n）=O(m\*n2)

4系统实现与测试

# 4.1系统实现

电脑系统：windows10 专业版

编程环境：Dev c++

编程语言：C语言

函数说明及调用关系：

（1）CreateCraph(&G,V,VR)

初始条件：V是图的顶点集，VR是图的关系集。

操作结果：按V和VR的定义构造图G。

调用函数：无

（2）DestroyCraph(&G)

初始条件：图G存在。

操作结果：销毁图G。

调用函数：无

（3）LocateVex(G,u)

初始条件：图G存在，u和G中的顶点具有相同特征。

操作结果：若u在图G中存在，返回顶点u的位置信息，否则返回其它信息。

调用函数：无

（4）FirstAdjVex(&G, v)

初始条件：图G存在，v是G的一个顶点。

操作结果：返回v的第一个邻接顶点，如果v没有邻接顶点，返回空。

调用函数：调用LocateVex定位传入的地点名

（5）NextAdjVex(&G, v, w)

初始条件：图G存在，v是G的一个顶点,w是v的邻接顶点。

操作结果：返回v的（相对于w）下一个邻接顶点，如果w是最后一个邻接

顶点，返回空。

调用函数：调用LocateVex定位传入的地点名

（6）InsertVex(&G,v)

初始条件：图G存在，v和G中的顶点具有相同特征。

操作结果：在图G中增加新顶点v。

调用函数：无

（7）DeleteVex(&G,v)

初始条件：图G存在，v是G的一个顶点。

操作结果：在图G中删除顶点v和与v相关的弧。

调用函数：无

（8）InsertArc(&G,v,w)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中增加弧<v,w>，如果图G是无向图，还需要增加<w,v>。

调用函数：调用LocateVex定位传入的地点名

（9）DeleteArc(&G,v,w)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

调用函数：调用LocateVex定位传入的地点名

（10）Arcweight(ALGraph T,char\* v1,char\* v2)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

调用函数：无

（11）CorrectArc(ALGraph G, char\* v1,char\* v2)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

调用函数：调用LocateVex定位传入的地点名

（12）Minspantree(ALGraph T,char\* v)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

调用函数：调用LocateVex定位传入的地点名

（13）graphcolor(ALGraph T,int m)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

调用函数：调用Iscolorok判断是否有颜色相同的相邻顶点

# 4.2系统测试

本次测试采用的测试方法：

1. **用户界面测试-UI测试**

用户界面测试是指测试用户界面的风格是否满足客户要求，文字是否正确，页面是否美观，文字，图片组合是否完美，操作是否友好等等。UI 测试的目标是确保用户界面会通过测试对像的功能来为用户提供相应的访问或浏览功能。确保用户界面符合公司或行业的标准。包括用户友好性、人性化、易操作性测试。

　　 用户界面测试用户分析软件用户界面的设计是否合乎用户期望或要求。它常常包括菜单，对话框及对话框上所有按钮，文字，出错提示，帮助信息 (Menu 和Help content)等方面的测试。比如，测试Microsoft Excel中插入符号功能所用的对话框的大小，所有按钮是否对齐，字符串字体大小，出错信息内容和字体大小，工具栏位置/图标等等。

1. **随机测试**

随机测试没有书面测试用例、记录期望结果、检查列表、脚本或指令的测试。主要是根据测试者的经验对软件进行功能和性能抽查。随机测试是根据测试说明书执行用例测试的重要补充手段，是保证测试覆盖完整性的有效方式和过程。随机测试主要是对被测软件的一些重要功能进行复测，也包括测试那些当前的测试样例(Test Case)没有覆盖到的部分。另外，对于软件更新和新增加的功能要重点测试。重点对一些特殊点情况点、特殊的使用环境、并发性、进行检查。尤其对以前测试发现的重大Bug，进行再次测试，可以结合回归测试 (Regressive testing)一起进行。

2.测试过程

（1）DestroyGraph（摧毁图测试）

用例一：选择图一（该图为空图无法摧毁，应该返回“该图为空图”）

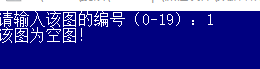


图4.2.1

用例二：选择图二（图二为存在的图，应该返回“该图销毁成功”）

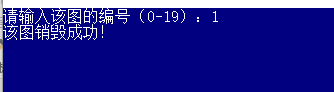


图4.2.2

测试结果：运行结果与期望结果相同，该模块功能正常

（2）DeleteVex（删除节点测试）

用例一：删除北京（北京存在，当该节点删除，其他节点相应位置应发生改变，与之相连的弧也应该删除，应该显示“删除成功”）

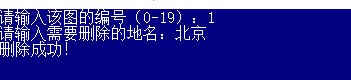


图4.2.3

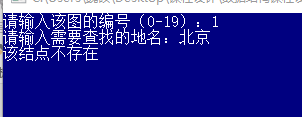


图4.2.4

用例二：新疆（该节点不存在，无法完成删除操作，应该返回“该节点不存在”）

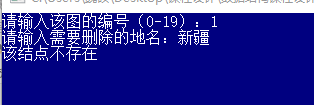


图4.2.5

测试结果：运行结果与期望结果相同，该模块功能正常

（3）DeleteArc（删除弧测试）

用例一：上海-武汉（武汉不存在于省份中，无法完成此操作，应该返回“节点不存在”）

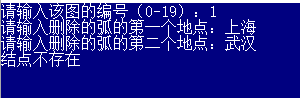


图4.2.6

用例二：北京-湖北（这条弧不存在，无法完成此操作，应该返回“该弧不存在”）

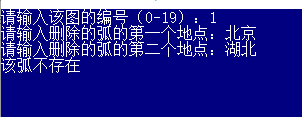


图4.2.7

用例三：上海—北京（该弧存在，应该返回“删除成功”）

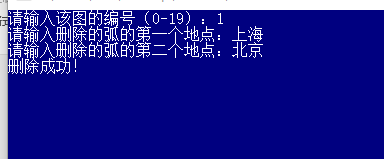


图4.2.8

测试结果：运行结果与期望结果相同，该模块功能正常

（4）Minspantree（最小生成树测试）

用例一：起始顶点设为新疆（不存在该节点，应该返回“该节点不存在”）

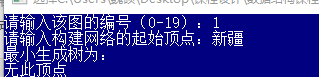


图4.2.9

用例二：起始顶点设为北京（结点存在，应该有最小生成树和运行时间）

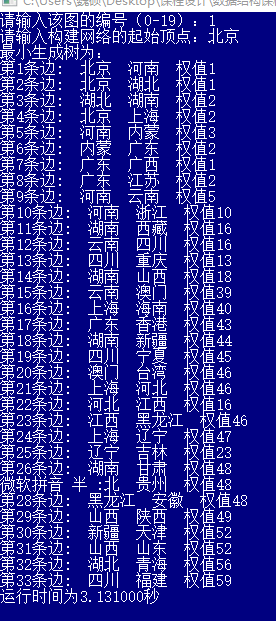


图4.2.10

用例三：起始顶点设为湖北，生成树应该与上相同，但是顺序不同



图4.2.11

测试结果：运行结果与期望结果相同，该模块功能正常

（5） graphcolor（图着色测试）

用例一：给图二着色（图二为空图，无法完成操作，应该返回“该图为空图”）

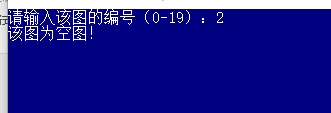


图4.2.12

用例二：给顶点较少图着色，最少染色数为3，返回所有顶点颜色及用时

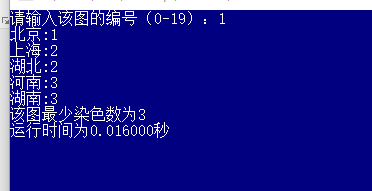


图4.2.13

用例三：给顶点与弧较多顶点着色，最少染色数为4，返回所有顶点颜色及用时

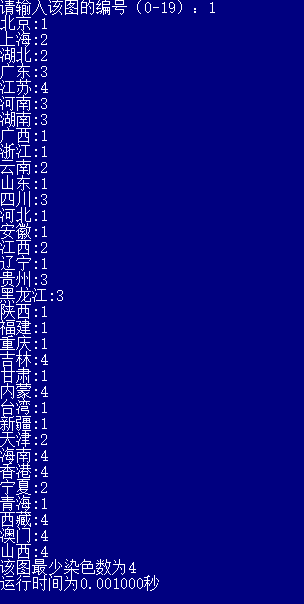


图4.2.14

测试结果：运行结果与期望结果相同，该模块功能正常

# 5总结与展望

# 5.1全文总结

主要工作如下：

（1）构建相应的图结构模型，分析完成相应任务所需要的数据类型以及细分的函数块。

（2）完成图的基本操作，并在实现中发现所需要的其他细节，进一步修改细化，并增添修改相应的函数。

（3）独立思考并查阅资料，了解最小生成树以及最少染色数背景资料及其算法思想，理解其内涵及中心思想。

（4）分析各个函数性能，查看是否还有值得提升的地方，了解相关经典回溯问题，思考相应的解决办法。

（5）完成报告。

# 5.2工作展望

在今后的研究中，围绕着如下几个方面开展工作：

（1）将一个具体问题抽象化，寻找相应的计算机模型，并将其对应，尽可能的将问题转化为计算机能够解决的问题。

（2）将一个宏观问题分解成多个小部分，细化问题，找出大量出现的重复操作并将其做成一个函数，简化代码。

（3）了解算法背后的思想方法及其背景和实际应用，深入挖掘一个问题背后的一类问题，尽量将其与实际结合，做到学以致用。

（4）提高自己的动手能力，在面对一个问题时，不应该一味空想，尽量自己动手实践，以期达到深入理解问题并解决问题。

（5）善于分析总结，在完成任务后应该进一步分析做法的完成度和性能，尽可能简化问题，精炼思路，达到尽可能简单的解决问题的目的。

# 6体 会

这个课程设计属于比较简单的一个设计，在图的基本操作上进行新增并且修改相应的信息即可。主要是在做的过程中，更加深入地体会到图结构在现实生活中的广泛应用，将主要实物表示成结点，相关信息用结点信息表示出来，然后将实物之间的联系用弧来抽象，即可构成一个图结构。而着色问题可以用于排课，安排见面会等一系列不可以冲突的或者相关事件不可以同时发生的情况，而最少着色数就是最少需要几块时间。最小生成树问题可以用于一些线路安排设计等等，不过在实际应用的过程中需要考虑的因素更加多，不过这些因素也可以通过抽像加入一些限制条件然后转化为代码，让计算机去实现。

在这些比较繁琐的事件设计中，人脑远不及电脑，通过一系列抽象分析，让电脑不可知的事物变为电脑可以通过它已知的定式去求解那些极其繁琐，人脑难以实现或者说无法实现的东西就是我们现在做的工作。

很多工作做出来和看着分析还是有比较大的差别，课程设计的意义大概就在此。

# 7参考文献

[1] 严蔚敏, 吴伟民. 数据结构（C语言版）. 北京: 清华大学出版社,1997

[2] 王晓东. 计算机算法设计与分析. 北京: 电子工业出版社, 2007

[3] 严蔚敏, 吴伟民, 米宁. 数据结构题集（C语言版）. 北京: 清华大学出版社,1999

[4] 谢力军,李晓梅,何佳等. 基于模糊最小生成树的通信网络架设模型.吉首大学学报（自然科学版），2010,31(4):43~46

8附录（源代码）

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include<time.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define MAX\_VERTEX\_NUM 40

typedef int status;

typedef struct ArcNode { //弧的信息

int adjvex; //该弧所指向的顶点的位置

struct ArcNode \*nextarc; //指向下一条弧的指针

int Weight; //用来记录网的权值

}ArcNode;

typedef struct VNode { //结点信息

int num;//顶点编号

char name[20];//顶点指代地名

char captial[20];//顶点省会

int people;//顶点人数

char phone[15];//电话区号

char where[50];//地理位置

ArcNode \*firstarc; //指向第一条依附该顶点的弧的指针

}VNode, AdjList[MAX\_VERTEX\_NUM];

typedef struct {//图信息

AdjList vertices;

int vexnum, arcnum; //图的当前顶点数和弧数

int kind; //图的种类标志

}aLGraph, \*ALGraph;

typedef struct Queue {

int \*base;

int capacity; //最大容量

int front;

int rear;

int now; //当前个数

}\*Queue;

typedef struct lowtree{//最小生成树辅助数组

char adjvex[20];

int lowcost;

}closedge[MAX\_VERTEX\_NUM];

ALGraph G[MAX\_VERTEX\_NUM];//存储多个图

int visited[MAX\_VERTEX\_NUM];//访问标记，0表示未访问过，1表示访问过

//图

int InitGraphs(void);//图置为空

ALGraph CreateGraph(ALGraph G);//创建图

ALGraph DestroyGraph(ALGraph T);//销毁图

status LocateVex(ALGraph G, char\* ch);//给定数据，判定该节点位置

status CorrectVex(ALGraph G, char\* ch);//修改指定顶点的值

VNode\* GetVex(ALGraph G, char\* ch);//获得指定顶点指针

status CorrectArc(ALGraph G, char\* v1,char\* v2);//替换指定弧的值

status FirstAdjVex(ALGraph G, char\* ch);//获得指定顶点第一个邻接点位置

status NextAdjVex(ALGraph G, char\* v);//显示所有邻接点

status Arcweight(ALGraph T,char\* v1,char\* v2);//查询弧的权值

ALGraph InsertVex(ALGraph G);//插入顶点

status DeleteVex(ALGraph G, char\* ch);//删除顶点

status InsertArc(ALGraph G, char\* v1, char\* v2,int weight);//新增一条弧

status DeleteArc(ALGraph G, char\* v1, char\* v2);//删除一条弧

int Save(ALGraph G);//保存到文件

ALGraph Load(ALGraph G);//从文件中读出

int Minspantree(ALGraph G,char\* v);//从顶点v出发构造最小生成树

int Iscolorok(ALGraph G,int vex,int color[]);//判断该顶点有没有相同颜色的邻接点

int graphcolor(ALGraph G,int m);//用m种颜色着色

//队列操作函数

int InitQueue(Queue Q);//创建队列

int EnQueue(Queue Q, int e);//入队

int DeQueue(Queue Q);//出队

int IsQueueEmpty(Queue Q); //判空

int DestroyQueue(Queue Q);//摧毁队列

int main(void)

{

int i,op=1,back,weight,m;

clock\_t start,finish;

double duration;

char e1[20],e2[20];

VNode\* p=NULL;

InitGraphs();

while (op)

{

system("cls");

system("color 1F");//系统命令，调颜色

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\* \*\n");

printf("\t\t\* ◆图的功能实现菜单 ◆ \*\n");

printf("\t\t\* \*\n");

printf("\t\t\* 0.退出系统 \*\n");

printf("\t\t\*图操作： \*\n");

printf("\t\t\* 1.创建图 \*\n");

printf("\t\t\* 2.销毁图 \*\n");

printf("\t\t\*顶点操作： \*\n");

printf("\t\t\* 3.修改顶点的值 \*\n");

printf("\t\t\* 4.获得顶点值 \*\n");

printf("\t\t\* 5.插入顶点 \*\n");

printf("\t\t\* 6.删除顶点 \*\n");

printf("\t\t\* 7.获得第一邻接点 \*\n");

printf("\t\t\* 8.获得所有邻接点 \*\n");

printf("\t\t\*弧操作： \*\n");

printf("\t\t\* 9.修改弧的权值 \*\n");

printf("\t\t\* 10.查询弧的权值 \*\n");

printf("\t\t\* 11.插入弧 \*\n");

printf("\t\t\* 12.删除弧 \*\n");

printf("\t\t\*附加操作： \*\n");

printf("\t\t\* 13.存入文件 \*\n");

printf("\t\t\* 14.从文件中读数据 \*\n");

printf("\t\t\* 15.生成最小生成树 \*\n");

printf("\t\t\* 16.最少染色数 \*\n");

printf("\t\t\* \*\n");

printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("\t\t 请输入你的选择：");

scanf("%d",&op);getchar();

switch(op)

{

case 1:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

G[i]=CreateGraph(G[i]);

printf("成功创建图！\n");

getchar();

break;

case 2:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

G[i]=DestroyGraph(G[i]);

printf("该图销毁成功！\n");

getchar();

break;

case 3:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入需要修改顶点的地名：");

scanf("%s",e1);getchar();

back=CorrectVex(G[i],e1);

if(back == -1)

printf("该结点不存在\n");

else

printf("修改成功\n");

getchar();

break;

case 4:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入需要查找的地名：");

scanf("%s",e1);getchar();

p=GetVex(G[i],e1);

if(p == NULL)

printf("该结点不存在\n");

else

printf("信息为\n名称：%s 省会：%s 编号：%d 人口数：%d 电话区号：%s 地理位置：%s",p->name,p->captial,p->num,p->people,p->phone,p->where);

getchar();

break;

case 5:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

G[i]=InsertVex(G[i]);

printf("插入成功！\n");

getchar();break;

case 6:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入需要删除的地名：");

scanf("%s",e1);getchar();

back=DeleteVex(G[i],e1);

if(back == 0)

printf("该结点不存在\n");

else

printf("删除成功！\n");

getchar();break;

case 7:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入需要查找的地名：");

scanf("%s",e1);getchar();

back=FirstAdjVex(G[i],e1);

if(back == -1)

printf("该第一邻接点不存在\n");

getchar();break;

case 8:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入需要查找的地名：");

scanf("%s",e1);getchar();

back=NextAdjVex(G[i],e1);

getchar();break;

case 9:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入需要改变的第一个地名：");

scanf("%s",e1); getchar();

printf("请输入需要改变的第二个地名：");

scanf("%s",e2); getchar();

if(CorrectArc(G[i],e1,e2) != -1)

printf("修改成功！\n");

else

printf("该弧不存在!\n");

getchar();break;

case 10:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入需要查询权值的第一个地名：");

scanf("%s",e1);getchar();

printf("请输入需要查询权值的第二个地名：");

scanf("%s",e2);getchar();

back=Arcweight(G[i],e1,e2);

if(back == -1)

printf("该弧不存在\n");

else

printf("该弧权值为%d",back);

getchar();break;

case 11:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入新增的弧的第一个地点：");

scanf("%s",e1);getchar();

printf("请输入新增的弧的第二个地点：");

scanf("%s",e2);getchar();

printf("请输入新增的弧的权值：");

scanf("%d",&weight);getchar();

back=InsertArc(G[i],e1,e2,weight);

if(back == 0)

printf("结点不存在\n");

else

printf("插入成功！\n");

getchar();break;

case 12:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

printf("请输入删除的弧的第一个地点：");

scanf("%s",e1);getchar();

printf("请输入删除的弧的第二个地点：");

scanf("%s",e2);getchar();

back=DeleteArc(G[i],e1,e2);

if(back == 0)

printf("该弧不存在\n");

else

printf("删除成功！\n");

getchar();break;

case 13:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

if(Save(G[i]))

printf("文件保存成功！\n");

else

printf("文件保存失败！\n");

getchar();break;

case 14:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

G[i]=Load(G[i]);

printf("文件读取成功！\n");

getchar();break;

case 15:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

start=clock();

printf("请输入构建网络的起始顶点：");

scanf("%s",e1);getchar();

back=Minspantree(G[i],e1);

if(back == -1)

printf("无此顶点\n");

finish=clock();

duration=(double)(finish-start)/CLOCKS\_PER\_SEC;

printf("运行时间为%f秒\n",duration);

getchar();break;

case 16:

system("cls");

printf("请输入该图的编号（0-19）：");

scanf("%d",&i);getchar();

if(G[i] == NULL)

{

printf("该图为空图！\n");

getchar();

break;

}

back=-1;

m=1;

start=clock();

while(back < 0)

{

back=graphcolor(G[i],m);

m++;

}

printf("该图最少染色数为%d\n",back);

finish=clock();

duration=(double)(finish - start)/CLOCKS\_PER\_SEC;

printf("运行时间为%f秒\n",duration);

getchar();break;

}

}

printf("欢迎下次再使用本系统！\n");

}

int InitQueue(Queue Q)

{

Q->base = (int \*)malloc(sizeof(int) \* MAX\_VERTEX\_NUM);

Q->capacity = MAX\_VERTEX\_NUM;

Q->front = 0;

Q->rear = 0;

Q->now = 0;

return OK;

}

int EnQueue(Queue Q, int e)

{

Q->now++;

Q->base[Q->rear] = e;

Q->rear++;

return OK;

}

int DeQueue(Queue Q)

{

int e;

if (IsQueueEmpty(Q))

return -1;

else

{

e = Q->base[Q->front];

Q->now--;

Q->front++;

return e;

}

}

int IsQueueEmpty(Queue Q)

{

if (Q->now == 0)

return TRUE;

else

return FALSE;

}

int DestroyQueue(Queue Q)

{

free(Q->base);

free(Q);

Q = NULL;

return OK;

}

int InitGraphs(void)

{

int i;

for (i = 0; i<MAX\_VERTEX\_NUM; i++)

G[i] = NULL;

return OK;

}

ALGraph CreateGraph(ALGraph T)

{

int i, j, k, weight;

char tou[20], wei[20];

ArcNode \*p,\*q;

T = (ALGraph)malloc(sizeof(aLGraph));

i=3;

if(i<1 || i>4)

{

printf("类型错误！\n");

return NULL;

}

T->kind=i;

printf("请输入图的顶点数:");

scanf("%d", &T->vexnum); getchar();

printf("请输入图的边数:");

scanf("%d", &T->arcnum); getchar();

for (i = 0; i < T->vexnum; i++)

{

printf("输入第%d个结点的信息\n", i+1);

T->vertices[i].num=i;

printf("名称：");

scanf("%s",T->vertices[i].name);getchar();

printf("省会：");

scanf("%s",T->vertices[i].captial);getchar();

printf("人口数：");

scanf("%d",&T->vertices[i].people);getchar();

printf("电话区号：");

scanf("%s",T->vertices[i].phone);getchar();

printf("地理位置：");

scanf("%s",T->vertices[i].where);getchar();

T->vertices[i].firstarc = NULL;

}

for (k = 0; k < T->arcnum; k++)

{

printf("请输入第%d条弧的权值:",k+1);

scanf("%d", &weight); getchar();

printf("请输入弧头:");

scanf("%s", tou); getchar();

printf("请输入弧尾:");

scanf("%s", wei); getchar();

p = (ArcNode\*)malloc(sizeof(ArcNode));

j = LocateVex(T, wei);

i = LocateVex(T, tou);

p->adjvex = j;

p->Weight = weight;

p->nextarc = T->vertices[i].firstarc;//插在表头

T->vertices[i].firstarc = p;

q = (ArcNode\*)malloc(sizeof(ArcNode));

q->adjvex = i;

q->Weight=weight;

q->nextarc = T->vertices[j].firstarc; //插在表头

T->vertices[j].firstarc = q;

}

return T;

}

ALGraph DestroyGraph(ALGraph T)

{

int i;

ArcNode \*p, \*pcrs;

if (T == NULL)

return NULL;

for (i = 0; i < T->vexnum; i++)

{

p = T->vertices[i].firstarc;

while (p != NULL)

{

pcrs = p->nextarc;

free(p);

p = pcrs;

}

}

free(T);

T = NULL;

return T;

}

status CorrectVex(ALGraph G, char\* ch)

{

int i,e;

i=LocateVex(G,ch);

printf("请输入需要修改的选项：1.名字 2.省会 3.人口数 4.电话区号 5.地理位置\n");

scanf("%d",&e);getchar();

if(e==1)

{

printf("请输入修改后的名字：");

scanf("%s",G->vertices[i].name);getchar();

return 1;

}

else if(e==2)

{

printf("请输入修改后的省会：");

scanf("%s",G->vertices[i].captial);getchar();

return 1;

}

else if(e==3)

{

printf("请输入修改后的人口数：");

scanf("%d",&G->vertices[i].people);getchar();

return 1;

}

else if(e==4)

{

printf("请输入修改后的电话区号：");

scanf("%s",G->vertices[i].phone);getchar();

return 1;

}

else if(e==5)

{

printf("请输入修改后的地理位置：");

scanf("%s",G->vertices[i].where);getchar();

return 1;

}

else

return -1;

}

status LocateVex(ALGraph G, char\* ch)

{

int i;

for (i = 0; i < G->vexnum; i++)

if (!strcmp(G->vertices[i].name,ch))

return i;

return -1;

}

VNode\* GetVex(ALGraph G, char\* ch)

{

int e;

e = LocateVex(G, ch);

if (e != -1)

return &G->vertices[e];

else

return NULL;

}

status CorrectArc(ALGraph G, char\* v1,char\* v2)

{

int i, j;

ArcNode \*adj;

i = LocateVex(G, v1);

j = LocateVex(G, v2);

if (i == -1 || j == -1)

return -1;

adj=G->vertices[i].firstarc;

while(adj != NULL)

{

if(adj->adjvex == j)

{

printf("请输入改变后的权值：");

scanf("%d",&adj->Weight);getchar();

return 1;

}

adj=adj->nextarc;

}

return -1;

}

status FirstAdjVex(ALGraph G, char\* ch)

{

ArcNode \*p;

int e;

e = LocateVex(G, ch);

if (e == -1)

return -1;

else

{

p = G->vertices[e].firstarc;

if (p != NULL)

{

printf("编号：%d 地名：%s 省会：%s 人口数：%d 电话区号：%s 地理位置：%s\n",G->vertices[p->adjvex].num,G->vertices[p->adjvex].name,\

G->vertices[p->adjvex].captial,G->vertices[p->adjvex].people,G->vertices[p->adjvex].phone,G->vertices[p->adjvex].where);

return 1;

}

else

return -1;

}

}

status NextAdjVex(ALGraph G, char\* v)

{

ArcNode \*p;

int e1, e2;

e1 = LocateVex(G, v);

if (e1 == -1)

return -1;

p = G->vertices[e1].firstarc;

printf("所有邻接点信息为\n");

while (p != NULL)

{

printf("编号：%d 地名：%s 省会：%s 人口数：%d 电话区号：%s 地理位置：%s\n",G->vertices[p->adjvex].num,G->vertices[p->adjvex].name,\

G->vertices[p->adjvex].captial,G->vertices[p->adjvex].people,G->vertices[p->adjvex].phone,G->vertices[p->adjvex].where);

p = p->nextarc;

}

return 1;

}

status Arcweight(ALGraph T,char\* v1,char\* v2)

{

int i, j;

ArcNode \*adj;

i = LocateVex(T, v1);

j = LocateVex(T, v2);

if (i == -1 || j == -1)

return -1;

if(i == j)

return 0;

adj=T->vertices[i].firstarc;

while(adj != NULL)

{

if(adj->adjvex == j)

return adj->Weight;

adj=adj->nextarc;

}

return -1;

}

ALGraph InsertVex(ALGraph T)//插在最后的位置

{

printf("请输入插入顶点地名:");

scanf("%s",T->vertices[T->vexnum].name);getchar();

printf("请输入插入顶点省会:");

scanf("%s",T->vertices[T->vexnum].captial);getchar();

printf("请输入插入顶点人口数:");

scanf("%d",&T->vertices[T->vexnum].people);getchar();

printf("请输入插入顶点电话区号:");

scanf("%s",T->vertices[T->vexnum].phone);getchar();

printf("请输入插入顶点地理位置:");

scanf("%s",T->vertices[T->vexnum].where);getchar();

T->vertices[T->vexnum].num = T->vexnum;

T->vertices[T->vexnum].firstarc = NULL;

T->vexnum++;

return T;

}

status DeleteVex(ALGraph G, char\* ch)

{

ArcNode \*p, \*pcrs;

int i, j;

j = LocateVex(G, ch);

if (j == -1)

return ERROR;

p = G->vertices[j].firstarc;

while (p)//释放所有以ch为头结点的弧

{

pcrs = p;

p = p->nextarc;

free(pcrs);

G->arcnum--;

}

for (i = j; i < G->vexnum - 1; i++) //所有节点前移一位

G->vertices[i] = G->vertices[i + 1];

G->vexnum--;

for (i = 0; i < G->vexnum; i++)

{

p = G->vertices[i].firstarc;

while (p != NULL)

{

if (p->adjvex != j)

{

if (p->adjvex > j)//节点序号改变

p->adjvex--;

pcrs = p;

p = p->nextarc;

}

else

{

if (G->vertices[i].firstarc == p)

{

G->vertices[i].firstarc = p->nextarc;

free(p);

p = G->vertices[i].firstarc;

}

else

{

pcrs->nextarc = p->nextarc;

free(p);

p = pcrs->nextarc;

}

}

}

}

return OK;

}

status InsertArc(ALGraph T, char\* v1, char\* v2,int weight)

{

ArcNode \*p,\*adj;

int i, j;

i = LocateVex(T, v1);

j = LocateVex(T, v2);

if (i == -1 || j == -1)

return ERROR;

adj=T->vertices[i].firstarc;

while(adj != NULL)

{

if(adj->adjvex == j)

return FALSE;

adj=adj->nextarc;

}

p = (ArcNode\*)malloc(sizeof(ArcNode));

p->Weight=weight;

p->adjvex = j;

p->nextarc = T->vertices[i].firstarc;

T->vertices[i].firstarc = p;

T->arcnum++;

adj = (ArcNode\*)malloc(sizeof(ArcNode));

adj->adjvex = i;

adj->Weight=p->Weight;

adj->nextarc = T->vertices[j].firstarc;

T->vertices[j].firstarc = adj;

return OK;

}

status DeleteArc(ALGraph T, char\* v1, char\* v2)

{

int i, j;

ArcNode \*p, \*pcrs;

i = LocateVex(T, v1);

j = LocateVex(T, v2);

if (i == -1 || j == -1)

return ERROR;

p = T->vertices[i].firstarc;

if (p->adjvex == j)

{

T->vertices[i].firstarc = p->nextarc;

free(p);

}

else

{

while (p->adjvex != j)

{

pcrs = p;

p = p->nextarc;

if(p == NULL)

break;

}

}

if(p==NULL)

return FALSE;

else

{

pcrs->nextarc = p->nextarc;

free(p);

T->arcnum--;

if (T->kind >= 2)

{

p = T->vertices[j].firstarc;

if (p->adjvex == i)

{

T->vertices[j].firstarc = p->nextarc;

free(p);

}

else

{

while (p->adjvex != i)

{

pcrs = p;

p = p->nextarc;

if(p == NULL)

break;

}

}

if (p->adjvex == i)

{

pcrs->nextarc = p->nextarc;

free(p);

}

if (p == NULL)

return FALSE;

}

}

return OK;

}

int Save(ALGraph T)

{

FILE \*fp;

int i;

ArcNode\* arc;

if ((fp = fopen("shuju.txt", "w+")) == NULL)

{

printf("文件打开失败！\n");

return ERROR;

}

fprintf(fp, "结点数： %d\n", T->vexnum);

fprintf(fp, "弧数： %d\n", T->arcnum);

fprintf(fp, "图的种类为： %d\n", T->kind);

for(i = 0 ; i < T->vexnum; i++)

fprintf(fp,"%d %s %d %s %s %s\n",T->vertices[i].num,T->vertices[i].name,T->vertices[i].people,T->vertices[i].captial,T->vertices[i].phone,T->vertices[i].where);

for (i = 0; i < T->vexnum; i++)

{

fprintf(fp, "%d ", i);

arc = T->vertices[i].firstarc;

while (arc != NULL)

{

fprintf(fp, "%d ", arc->adjvex);

fprintf(fp,"%d ",arc->Weight);

arc = arc->nextarc;

}

fprintf(fp, "-1\n");

}

fclose(fp);

return OK;

}

ALGraph Load(ALGraph T)

{

FILE \*fp;

char line[256];

int i, e,weight,arc;

T = (ALGraph)malloc(sizeof(aLGraph));

fp = fopen("shuju.txt", "r");

fgets(line, 250, fp);

sscanf(line, "%\*s%d", &(T->vexnum));

fgets(line, 250, fp);//读完后续空格

sscanf(line, "%\*s%d", &arc);

fgets(line, 250, fp);

sscanf(line, "%\*s%d", &T->kind);//读入字符串但是不赋值

for (i = 0; i < T->vexnum; i++)

T->vertices[i].firstarc = NULL;

for (i = 0; i < T->vexnum; i++)

{

fscanf(fp, "%d", &T->vertices[i].num);

fscanf(fp, "%s", T->vertices[i].name);

fscanf(fp, "%d", &T->vertices[i].people);

fscanf(fp, "%s", T->vertices[i].captial);

fscanf(fp, "%s", T->vertices[i].phone);

fscanf(fp, "%s", T->vertices[i].where);

}

for (i = 0; i < T->vexnum; i++)

{

fscanf(fp, "%\*d");

fscanf(fp, "%d", &e);

while (e != -1)

{

fscanf(fp, "%d", &weight);

InsertArc(T, T->vertices[i].name, T->vertices[e].name,weight);

fscanf(fp, "%d", &e);

}

fgetc(fp);

}

T->arcnum=arc;

fclose(fp);

return T;

}

int Minspantree(ALGraph T,char\* v)

{

int i,j,k,min,flag,minsum,weight;

closedge closedges;

printf("最小生成树为：\n");

k=LocateVex(T,v);

if(k == -1)

return -1;

for(j=0;j < T->vexnum;j++)//初始化

if(j != k)

{

strcpy(closedges[j].adjvex,T->vertices[k].name);

closedges[j].lowcost=Arcweight(T,v,T->vertices[j].name);//不存在弧的话，权值是-1

}

closedges[k].lowcost=0;

for(i=1;i<T->vexnum;i++)

{

min=0;//寻找最小边

for(j=0,flag=0;j<T->vexnum;j++)

{

if(closedges[j].lowcost>0 && flag==0)

{

min=closedges[j].lowcost;

flag=1;

}

if((min>0) && (closedges[j].lowcost <= min) && (closedges[j].lowcost>0))

{

min=closedges[j].lowcost;

minsum=j;

}

}

printf("第%d条边: %s %s 权值%d\n",i,closedges[minsum].adjvex,T->vertices[minsum].name,min);

closedges[minsum].lowcost=0;

for(j=0;j < T->vexnum;j++)

{

weight=Arcweight(T,T->vertices[minsum].name,T->vertices[j].name);//同地名代表0

if(((closedges[j].lowcost < 0) && (weight > 0)) || ((closedges[j].lowcost>0) && (weight>0) && (weight < closedges[j].lowcost)))

{

strcpy(closedges[j].adjvex,T->vertices[minsum].name);

closedges[j].lowcost=weight;

}

}

}

return OK;

}

int Iscolorok(ALGraph T,int vex,int color[])

{

ArcNode \*adj;

adj=T->vertices[vex].firstarc;

while(adj != NULL)

{

if(color[adj->adjvex] == color[vex])

return 0;

adj=adj->nextarc;

}

return 1;

}

int graphcolor(ALGraph T,int m)

{

int i,k,color[40],max;

for(i=0;i < T->vexnum;i++)

color[i]=0;

k=0;

while(k>=0)

{

color[k]=color[k]+1;

while(color[k] <= m)

if (Iscolorok(T,k,color))

break;

else

color[k]=color[k]+1;

if((color[k] <= m) && (k == T->vexnum))

{

max=color[0];

for(i=0;i < T->vexnum;i++)

{

printf("%s:%d\n",T->vertices[i].name,color[i]);

if(color[i] > max)

max=color[i];

}

return max;

}

else if((color[k] <= m) && (k < T->vexnum))

k=k+1; //处理下一个顶点

else

{

color[k]=0;

k=k-1;//回溯

}

}

return -1;

}