|  |  |  |  |
| --- | --- | --- | --- |
| **《数据结构与算法》实验报告** | | | |
| **学生姓名** | 曾钰城 | **院（系）** | 计算机科学与技术 |
| **学 号** | 1173710105 | **专 业** | 软件工程 |
| **实验时间** | 2018年12月5日（周三） | **实验地点** | 格物213室 |
| **实验项目** | **实验3/4：图型结构的建立与搜索** | | |
| **实验目的：**将课程的基本原理、技术和方法与实际应用相结合，训练和提高学生组织、存储和处理信息的能力，以及复杂问题的数据结构设计能力和程序设计能力，培养软件设计与开发所需要的实践能力。  **实验要求：**灵活运用基本的数据结构和算法知识，对实际问题进行分析和抽象；结合程序设计的一般过程和方法为实际问题设计数据结构和有效算法；用高级语言对数据结构和算法进行编程实现、调试，测试其正确性和有效性。 | | | |
| **实验内容：**  **图的搜索（遍历）算法是图型结构相关算法的基础，本实验要求编写程序演示图两种典型存储结构的建立和搜索（遍历）过程。**   1. **分别实现图的邻接矩阵、邻接表存储结构的建立算法，分析和比较各建立算法的时间复杂度以及存储结构的空间占用情况；** 2. **实现图的邻接矩阵、邻接表两种存储结构的相互转换算法；** 3. **在上述两种存储结构上，分别实现图的深度优先搜索（递归和非递归)和广度优先搜索算法。并以适当的方式存储和显示相应的搜索结果（深度优先或广度优先生成森林（或生成树）、深度优先或广度优先序列和编号）；** 4. **分析搜索算法的时间复杂度；** 5. **以文件形式输入图的顶点和边，并显示相应的结果。要求顶点不少于10个，边不少于13个；** 6. **软件功能结构安排合理，界面友好，便于使用。** | | | |
| 数据结构定义：  //图的类型  typedef enum {DG, DN, AG, AN} GraphKind;  //邻接矩阵  typedef struct ArcCell  {  int adj; //单元之间的关系  int \*info; //弧/边附加信息  }ArcCell,AdjMatrix[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];  typedef struct  {  int vex[MAX\_VERTEX\_NUM] ; //**顶点信息**  AdjMatrix arcs; // 连接矩阵  int vexnum, arcnum; //分别为顶点个数，弧的个数  GraphKind kind; //图的类型  }MGraph;  //邻接表  typedef struct Vnode  {  int data; //自身节点数据  struct Vnode \*firstarc;// 顶点之间的关系  }Vnode, AdjList[MAX\_VERTEX\_NUM];  typedef struct  {  AdjList vertices; // 邻接链表  int vexnum; //顶点个数  int kind; //图的类型  }ALGraph;  //队列  typedef struct queue  {  int data[MAX\_VERTEX\_NUM];  int front;  int rear;  }QUEUE;  // 栈  typedef struct  {  int data[MAX\_VERTEX\_NUM];  int top;  }STACK; | | | |
| 算法设计与分析（要求画出核心内容的程序流程图）：  建立邻接矩阵：    建立邻接链表：    邻接链表与连接矩阵的相互转换：  连接链表 to 连接矩阵：    连接矩阵 to 邻接链表：    邻接矩阵 和 邻接链表 深度优先搜索算法（递归）：  首先访问起点，然后依次访问与该起点相关联的每一个顶点，并以该关联顶点为新的起点，继续深度优先遍历。若图中还有未被访问的顶点，则换一个起点，继续深度优先遍历，直到所有的顶点都被访问。  深度优先搜索算法（非递归 邻接链表与邻接矩阵）：    广度优先搜索算法（连接链表 与 邻接矩阵）： | | | |
| 实验测试结果及结果分析：  图：    程序的功能：    图的表示：  邻接矩阵表示：    邻接链表表示：    两种结构的空间复杂度分析：    两种建立算法的时间复杂度：  从文件之中读取或者键盘输入：  邻接链表 建立算法时间复杂度： n为节点个数，e为弧的条数  邻接矩阵 建立算法时间复杂度： n为节点个数  若果通关先建立连接矩阵（连接链表），然后转换为邻接链表（邻接矩阵），则建立算数的时间复杂度都为： n为节点个数  深度优先搜索算法：  深度优先搜索生成森林，其中一种:            广度优先搜索算法：  广度优先搜索生成森林，其中一种：        深度优先搜索算法时间复杂度分析：  邻接链表 时间复杂度为： n为节点个数，e为弧的条数  连接矩阵 时间复杂度为： n为节点个数  广度优先搜索算法的时间复杂度：  邻接链表 时间复杂度为： n为节点个数，e为弧的条数  邻接矩阵 时间复杂度为： n为节点个数 | | | |
| 问题及解决方法：  问题：深度优先非递归算法中用到栈，广度优先搜索算法中用到队列，导致一个函数内代码非常多且乱，不易debug；解决方法：加入队列以及栈的ADT，简化搜索算法中的代码。  问题：搜索算法之中的树边表示问题；解决算法，创建一个长度为n的数组，初始化元素都为-1，下标为i数组的元素表示节点i在生成树之中的父节点信息，这样在搜索完成之后，数组中每一个元素都能位置确定其在生成树中的父节点，（注：-1代表根节点），这样就可以表示出树边以及生成树。  问题：计算存储结构空间复杂度，由于使用了宏定义#define MAX\_VERTEX\_NUM ，所以有可能导致用sizeof（结构体）表示的空间大小并不是真正占用空间的大小，因为vexnum小于MAX\_VERTEX\_NUM，数组中有一些元素未曾用到；故邻接矩阵空间复杂度=结构体MGraph大小-未曾占用空间大小  此外,用于连接链表还申请动态内存空间，故连接链表空间复杂度=结构体ALGraph大小-未曾占用空间大小+节点大小\*动态申请节点个数  问题：由于邻接链表申请了动态内存空间，必须释放这些动态申请的内存空间，故编写一个destroy\_ALG函数，在不再用到该邻接链表的时候，删除该邻接链表申请的内存空间。  问题：在为曾构建邻接链表或邻接矩阵的时候就使用其他功能会导致程序异常终止，解决方法：加入标志ALG\_FLAG,MG\_FLAG,来标志ALG和MG是否已经建立，若果未曾构建存储结构，就使用其他功能，则给出错误提示。 | | | |
| 源程序名称：GRAPH.C | | | |

注意：正文文字为宋体小4号，图中文字为宋体5号。行距为多倍行距1.25。

源程序与此报告打包提交，压缩包采用学号命名。