|  |  |  |  |
| --- | --- | --- | --- |
| **《数据结构与算法》实验报告** | | | |
| **学生姓名** |  | **院（系）** | 计算学部 |
| **学 号** |  | **专 业** | 计算机类 |
| **实验时间** | 2021年10 月27日（周三） | **实验地点** | 格物213室 |
| **实验项目** | **实验3-2：图型结构的建立与搜索（3学时）** | | |
| **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：  ①图的邻接表定义。使用邻接表完成图的存储，邻接表结构体AdjGraph包含的内容有：  Int型变量n,e，其中n为图的顶点数，e为图的边数。  VertexNode型变量vexlist[Maxnum]，存储图的顶点表。  结构体VertexNode包含的内容有：  Char型变量element，存储每个结点的元素值。  Edgenode型指针变量firstedge，指向顶点的第一个邻接点。  结构体Edgenode包含的内容有：  Int型变量adjvex，存储结点的下标。  node型指针变量next，指向结点的邻接点  ②图的邻接矩阵定义。使用邻接矩阵完成图的存储，邻接矩阵结构体MTGraph包含的内容有：  Int型二维数组mat，用于存储矩阵。  Char型一维数组element，用于存储顶点的元素值，  Int型变量n,e，其中n为图的顶点数，e为图的边数。  ②栈的数据结构定义：在实现图的深度优先遍历非递归算法时，引入了栈这一类型的数据结构，载入头文件#include <stack>。  ③队列的数据结构定义：在实现图的广度优先遍历算法时，需要引入队列这一类型的数据结构，载入头文件#include <queue>。 | | | |
| 算法设计与分析（要求画出核心内容的程序流程图）：  **①任务一（图的两种存储结构的建立）**  一.邻接表的建立  邻接表建立的流程整体上分为三步：  1)读入图的顶点数n，边数e  2)根据n，读入每个顶点对应的元素值  3)根据e，以输入构成边的两个顶点的形式将图中的边读入到邻接表中，**由于程序需要同时兼顾有向图和无向图的建立，因此无向图的每条边需要输入两次，有向图的边的方向需要由第一个顶点指向第二个顶点。**  核心步骤算法设计：  步骤2)算法设计：设计一个n次的for循环，每次读入对应顶点的element（元素值）并在将该顶点的指向第一个邻接点的指针firstedge指向NULL，方便下一步边的输入。  步骤3)算法设计：设计一个e次的for循环，在读入构成边的两个顶点后，到顶点表中访问对应顶点，以头插法的方式，将邻接点（即输入的另一个顶点）插入到该顶点的邻接链表中（有向图规定边的方向为由第一个输入的顶点指向第二个输入的顶点）。  二.邻接矩阵的建立  邻接矩阵建立的流程整体上分为三布：  1)读入图的顶点数n，边数e  2)根据n，读入每个顶点对应的元素值  3)根据e，以输入构成边的两个顶点的形式将图中的边读入到邻接矩阵中，**由于程序需要同时兼顾有向图和无向图的建立，因此无向图的每条边需要输入两次，有向图的边的方向需要由第一个顶点指向第二个顶点。**  核心步骤算法设计：  步骤2)算法设计：设计一个n次的for循环，每次读入对应顶点的element值（元素值）。  步骤3)算法设计；首先将邻接矩阵（二维数组mat）初始化为0，再设计一个e次的for循环，每次读入构成边的两个顶点，将邻接矩阵中对应位置的值赋为1，代表这两个点之间存在一条边。  三.建立算法的时间复杂度及空间占用情况分析   1. 时间复杂度分析。   I邻接表：邻接表建立过程中读入顶点信息的过程需要使用一个n次的for循环，读入边信息的过程需要使用一个e次的for循环，故时间复杂度为O(n+e)  II邻接矩阵：邻接矩阵建立过程中读入顶点信息的过程需要使用一个n次的for循环，邻接矩阵初始化为0的过程需要使用两个嵌套的n次for循环，读入边信息需要一个e次的for循环，故时间复杂度为O()  2)空间复杂度分析。  I邻接表：存储n个顶点的元素值，存储e条边的信息，空间复杂度为O(n+e)  II邻接矩阵：存储n个顶点的元素值，大小的邻接矩阵存储图的邻接信息，故空间复杂度为O()  **②任务二（邻接矩阵、邻接表两种存储结构的相互转换算法）**   1. **邻接矩阵转为邻接表的算法**   **邻接矩阵转为邻接表的流程整体上分为两步：**   1. **将邻接矩阵存储的顶点信息传入邻接表** 2. **将邻接矩阵存储的边信息传入邻接表**   **核心步骤算法设计：**  **步骤1)算法设计：首先将邻接矩阵存储的图顶点数n读入到邻接表；再设计一个n次的for循环，依次将邻接矩阵存储的顶点的元素值赋给邻接表，即可完成顶点信息的读入。**  **步骤2)算法设计：对邻接矩阵进行遍历，当访问到数据为1的位置时，说明该位置的两个顶点之间存在一条边，在邻接表中访问对应顶点，在其邻接链表中插入这条边即可。**   1. **邻接表转为邻接矩阵的算法**   **邻接表转为邻接矩阵的流程整体上分为两步：**  **1)将邻接表存储的顶点信息传入邻接矩阵**  **2)将邻接表存储的边信息传入邻接矩阵**  **核心步骤算法设计：**  **步骤1)算法设计：首先将邻接表存储的图顶点数n读入到邻接矩阵；再设计一个n次的for循环依次将邻接表存储的顶点的元素值赋给邻接矩阵，即可完成顶点信息的读入。**  **步骤2)算法设计：首先将邻接矩阵初始化为0，然后设置一个n次的for循环，依次访问邻接表的顶点，将顶点的邻接链表存储的边信息传入到邻接矩阵，即将邻接矩阵中对应位置的数据值赋为1，最终可完成边信息的传入。**  **③任务三（两种搜索算法的实现）**  **i深度优先搜索**  **算法的设计思想：首先以一个未被访问过的顶点作为起始顶点，沿当前顶点的边走到未访问过的顶点；当没有未访问过的顶点时，则回到上一个顶点，继续试探访问别的顶点，直到所有的顶点都被访问。搜索结果采用生成森林的方式进行显示。**  **算法设计的流程描述：**  1)访问顶点v；  2)依次从v的未被访问的邻接点出发，对图进行深度优先遍历；直至图中和v有路径相通的顶点都被访问；  3)若此时图中尚有顶点未被访问，则从一个未被访问的顶点出发，重新进行深度优先遍历，得到生成森林中的另一个生成树，直到图中所有顶点均被访问过为止。  **一.深度优先搜索的递归实现**  I邻接表的深度优先搜索（递归）  算法设计（设搜索的起始顶点为v）：   1. 初始化指针p，访问顶点v，输出顶点的元素值并标记未已访问 2. 指针p指向当前结点的firstedge 3. 当p不为空时，重复4)操作 4. 若p未访问过，递归调用该函数访问顶点p；p指向p的next顶点 5. 再依次以其他顶点为搜索起始点进行搜索，得到生成森林中的另一个生成树，确保图的所有顶点均被访问   程序流程图：    **II邻接矩阵的深度优先搜索（递归）**  **算法设计**（设搜索的起始顶点为v）**：**   1. **访问顶点v，输出v的元素值并将其标记为已访问** 2. **初始化j=0，当j小于图的顶点数n时，执行步骤3)** 3. **如果邻接矩阵的（v，j）位置元素值为1且顶点j未被访问过，递归调用函数访问顶点j，j的值加1** 4. 再依次以其他顶点为搜索起始点进行搜索，得到生成森林中的另一个生成树，确保图的所有顶点均被访问   **程序流程图：**    **二.深度优先搜索的非递归实现**  I邻接表的深度优先搜索（非递归）  算法设计（设搜索的起始顶点为v）：   1. **初始化栈s，指针p，选定第一个访问的顶点v压栈** 2. **当栈不满时，重复执行3),4)操作** 3. **取栈顶元素w，若其未被访问过，则输出其元素值并标记为已访问。** 4. **使用指针p遍历顶点w的邻接点链表，遇到未访问过的顶点就将其压栈** 5. 再依次以其他顶点为搜索起始点进行搜索，**得到生成森林中的另一个生成树，确保图的所有顶点均被访问**   **程序流程图：**    **II邻接矩阵的深度优先搜索（非递归）**  算法设计（设搜索的起始顶点为v）：   1. **初始化栈s，指针p，选定第一个访问的顶点v压栈** 2. **当栈不满时。重复3)4)步骤** 3. **取栈顶元素w，若其未被访问过，则输出其元素值并标记为已访问。** 4. **遍历邻接矩阵中顶点w的这一行，若访问到值为1的位置（说明w与该位置的另一个顶点邻接）且该位置的另一个顶点未被访问过，将此顶点入栈** 5. 依次以其他顶点为搜索起始点进行搜索，得到生成森林中的另一个生成树，确保图的所有顶点均被访问   **程序流程图：**    **ii广度优先搜索**  **算法的设计思想：当到达某一顶点v时，依次考察与v相邻的全部顶点，并访问其中没有访问过的顶点；然后，从顶点v沿着一条弧(v,w)到达下一个顶点w，再考察与w相邻的全部顶点。搜索结果以生成森林的形式进行显示**  I邻接表的广度优先搜索  算法设计（设搜索的起始顶点为v）：   1. **初始化队列Q，指针P** 2. **访问起始顶点v，输出其元素值，进队，并标记为已访问** 3. **当队列不空时，重复执行4)，5)步骤** 4. **取队列头顶点w，并将其出队，指针P指向w的第一个临界点firstedge** 5. **当P不为空时，重复执行6)，7)步骤** 6. **若顶点P未被访问过，则输出其元素值，标为已访问并进队** 7. **P指向P的next结点** 8. 依次以其他顶点为搜索起始点进行搜索，得到生成森林中的另一个生成树，确保图的所有顶点均被访问   **程序流程图：**    **II**邻接矩阵的广度优先搜索  **算法设计：**   1. **初始化队列Q** 2. **访问起始顶点v，进队。** 3. **当队列不为空时，重复执行4),5)步骤** 4. **取队列头顶点w，并将其出队，若其为被访问过则输出其元素值并标记为已访问** 5. **遍历邻接矩阵中顶点w的这一行，对于对应矩阵位置值为1且未被访问过的顶点，将其入队** 6. 依次以其他顶点为搜索起始点进行搜索，得到生成森林中的另一个生成树，确保图的所有顶点均被访问   **程序流程图：** | | | |
| 实验测试结果及结果分析：  ①文件输入样例1：    生成邻接表（由于程序是按0开始生成的序号，所以会和图上的序号差一位，但不影响相对的顺序）：    生成邻接矩阵：    深度优先搜索递归（在这里选择测试初始搜索顶点为0的结果）：    深度优先搜索非递归：    广度优先搜索：    邻接矩阵转为邻接表：    邻接表转为邻接矩阵：    结果分析：  实验任务①，②，③均正确完成。程序能够实现文件读入图的信息；能够生成邻接表，邻接矩阵；能够完成深度优先搜索的递归，非递归实现，广度优先搜索的实现；能够完成邻接表转邻接矩阵，邻接矩阵转邻接表。 | | | |
| 问题及解决方法：  ①问题1：不知道如何实现非递归实现深度优先算法  解决方法：经过对课本及上课所讲的知识进行复习总结，对栈的使用有了更加深入的理解，使用栈这一数据结构，将顶点按照一定的顺序入栈，出栈，输出，理论上是可以完成深度优先搜索的非递归实现，最终通过思考与练习，成功设计出了算法，完成了深度优先算法的非递归实现。  ②问题2：对邻接表的建立方法不太清晰  解决方法：通过对课程知识的再复习，了解到可以通过依次读入顶点的元素值，以及读入构成边的两个顶点，并使用头插法在对应顶点的邻接点链表中插入边信息的方式来完成邻接表的建立，最终成功设计出对应算法，实现邻接表的建立。 | | | |
| 源程序名称：实验2.cpp | | | |

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。