**新疆政法学院**

**数据结构与算法**

**实验报告**

班 级：计算机科学与技术4班

学 号： 2124030160

姓 名： 孙久猛

指导老师： 张家琪

学 期： 2022-2023第一学期

|  |  |  |  |
| --- | --- | --- | --- |
| **实验题目** | 实验六：邻接矩阵存储图的深度优先遍历 | | |
| **实验时间** | **10:00** | **实验地点** | **东训0114** |
| **实验成绩** |  | **实验性质** | **□验证性 √设计性 □综合性** |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | |
| 一、实验目的  1.掌握图的矩阵存储过程；  2.理解矩阵存储和邻接表存储的异同；  3.掌握图的深度优先遍历规则；  4. 加深图的创建与表示、图的邻接矩阵表示的理解与实现，逐步培养解决实际问题的能力。 | | | |
| **二、实验项目内容（实验题目）**  （一）图的矩阵存储及表示  1.初始化一个图，依次输入该图的顶点数、边数、每个顶点以及每条边所对应的两个顶点的下标；  2.根据以上信息创建一个无向图；  3.根据创建的无向图输出该图的邻接矩阵；  4.并将该过程代码化  （二）图的深度优先遍历  1.在第一步的基础上，对创建好的图，采用深度优先遍历的方法，对该图进行深度优先遍历，并输出遍历结果；  2.在第一步代码基础上，对代码进行丰富，执行图的深度优先遍历，并输出遍历结果。  **三、源程序（实验步骤/实验过程/算法）**  //邻居矩阵存储图的深度优先遍历  //Deom\_06.c  #include <stdio.h>  #include <stdlib.h>  #define MAX\_VERTEX\_NUM 100 // 图中最大节点数  typedef char VertexType; // 顶点类型设置为字符型  typedef int EdgeType; // 边上权值类型设置为整型  //访问数组  int visited[MAX\_VERTEX\_NUM];  typedef struct // 边表节点  {  VertexType vex[MAX\_VERTEX\_NUM]; // 顶点表  EdgeType edges[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM]; // 邻接矩阵  int vexnum; // 节点的数目  int edgenum; // 边的数目  } MGraph;  void CreateMG(MGraph \*MG); // 邻接表法创建无向图  void PrintMG(MGraph MG); // 邻接矩阵形式输出图MG  void CreateMG(MGraph \*MG)  {  int i = 0, j, k, w; // w：权值  char ch;  printf("请依次输入顶点数、边数：");  scanf("%d %d", &(MG->vexnum), &(MG->edgenum));  printf("请依次输入顶点（以回车结束输入）：");  getchar();  while ((ch = getchar()) != '\n') // 输入顶点信息  MG->vex[i++] = ch;  /\*  \* for (i = 0; i < MG->vexnum; i++)  \* scanf("\n%c", &(MG->vex[i]));  \*/  // 初始化邻接矩阵  for (i = 0; i < MG->vexnum; i++)  for (j = 0; j < MG->vexnum; j++)  MG->edges[i][j] = 0;  printf("顶点 | 下标\n");  // 显示图中顶点及其对应下标  for (i = 0; i < MG->vexnum; i++) {  printf("%3c%6d\n", MG->vex[i], i);  }  printf("请输入每条边对应的两个顶点下标（格式：i,j）：\n");  // 建立邻接矩阵  for (k = 0; k < MG->edgenum; k++) {  scanf("\n%d,%d", &i, &j);  MG->edges[i][j] = 1;  MG->edges[j][i] = 1;  }  }  void PrintMG(MGraph MG)  {  int i, j;  // 输出邻接矩阵  for (i = 0; i < MG.vexnum; i++) {  for (j = 0; j < MG.vexnum; j++)  printf("%2d", MG.edges[i][j]);  printf("\n");  }  }  //递归函数  void DFS(MGraph G, int i) {  int j; //数组下标j，用来遍历图中顶点  visited[i] = 1; //标志下标为i的顶点已经被访问  printf("%c", G.vex[i]); //打印该结点  for (j = 0; j < G.vexnum; j++) {  if (G.edges[i][j] == 1 && !visited[j]) {  DFS(G, j);//如果下标ij的顶点邻接且j没有被访问过  }  }  }  //深度优先遍历  void DFSTraverse(MGraph G) {  int i;  for (i = 0; i < G.vexnum; i++) {  visited[i] = 0; //初始化所有结点为未访问  }  for (i = 0; i < G.vexnum; i++) {  if (!visited[i]) {  DFS(G, i); //选用下标j对应的顶点作为新的起始点，递归直到图中所有顶点都被访问过为止  }  }  }  int main(void)  {  MGraph g;  //int n;  CreateMG(&g);  printf("该图的顶点数和边数依次为：\n");  printf("顶点数： %d ; 边数： %d\n", g.vexnum, g.edgenum);  printf("该图的邻接矩阵表示为：\n");  PrintMG(g);  printf("该图的深度优先遍历序列为：\n");  DFSTraverse(g);  //scanf("%d",&n);  return 0;  }  **四、运行结果**  //Deom\_06.c  //Picture\_01    //Picture\_02 | | | |

日 期： 2022年11月2日