《数据结构》实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 李爽 | 班级 | 22级软件工程专升本1班 | 学号 | 2206831522 |
| 实验名称 | **实验六 图** | | | | |
| 实验时间 | 2023年5月5日 | | 成绩 | |  |
| 1. **实验目的：**   1.理解图的逻辑结构及存储结构；  2.掌握图的基本操作；  3.掌握图的深度优先和广度优先遍历算法。  **二、实验工具：**  软件平台：Windows7或以上版本，Visual Studio 2019  **三、实验原理：**  1、算法描述：  （1）深度优先遍历函数：在dfs函数中，我们先将当前节点标记为已访问，并输出该节点的编号，然后遍历该节点的所有相邻节点。对于每个相邻节点，如果它还未被访问，则递归调用dfs函数，访问该相邻节点。当所有与起始节点直接或间接相连的节点都被访问过后，dfs函数结束。  （2）广度优先遍历函数：BFS算法从给定的源节点开始遍历图，首先将该节点加入到队列中。然后，从队列中弹出一个节点并访问该节点，将其所有未被访问过的相邻节点加入到队列中。重复执行上述过程，直到队列为空。  **四、实验步骤和内容：**  1.建立任意无向图，采用邻接矩阵存储，完成以下操作：  （1）对该图进行深度优先遍历，并输出遍历结果；  （2）对该图进行广度优先遍历，并输出遍历结果。  #include <iostream>  #include <cstring>  using namespace std;  const int MAXN = 100;  // 邻接矩阵存储图  bool graph[MAXN][MAXN];  // 标记节点是否被访问过  bool visited[MAXN];  // n 表示节点数，m 表示边数  int n, m;  // 深度优先遍历  void dfs(int u) {  // 标记当前节点已被访问  visited[u] = true;  // 输出当前节点  cout << u << " ";  // 枚举与当前节点相邻的节点  for (int v = 0; v < n; v++) {  // 如果与当前节点相邻的节点未被访问过  if (graph[u][v] && !visited[v]) {  // 继续深度优先遍历该节点  dfs(v);  }  }  }  // 不使用 queue 的广度优先遍历  void bfs(int u) {  // 定义一个队列 q  int q[MAXN];  // head 和 tail 分别指向队首和队尾  int head = 0, tail = 0;  // 标记起始节点已被访问  visited[u] = true;  // 将起始节点加入队列  q[tail++] = u;  // 当队列非空时  while (head < tail) {  // 取出队首元素  int x = q[head++];  // 输出当前节点  cout << x << " ";  // 枚举与当前节点相邻的节点  for (int v = 0; v < n; v++) {  // 如果与当前节点相邻的节点未被访问过  if (graph[x][v] && !visited[v]) {  // 标记该节点已被访问  visited[v] = true;  // 将该节点加入队列  q[tail++] = v;  }  }  }  }  int main() {  // 输入节点数和边数  cin >> n >> m;  // 初始化邻接矩阵和标记数组  memset(graph, false, sizeof(graph));  memset(visited, false, sizeof(visited));  // 输入边的信息，并在邻接矩阵中添加边  for (int i = 0; i < m; i++) {  int u, v;  cin >> u >> v;  graph[u][v] = graph[v][u] = true;  }  // 深度优先遍历  cout << "DFS: ";  for (int i = 0; i < n; i++) {  if (!visited[i]) {  dfs(i);  }  }  cout << endl;  // 广度优先遍历  // 重置标记数组  memset(visited, false, sizeof(visited));  cout << "BFS: ";  for (int i = 0; i < n; i++) {  if (!visited[i]) {  bfs(i);  }  }  cout << endl;  return 0;  }    **五、实验总结：**  本次实验主要是通过使用 C++ 语言来建立任意无向图，并采用邻接矩阵的方式进行存储。实现了深度优先遍历和广度优先遍历，通过输出遍历结果进行了验证。  在实验过程中，我们首先输入了图的节点数和边数，然后通过循环输入每条边的信息，并在邻接矩阵中添加边。接着，我们实现了深度优先遍历和广度优先遍历两个函数。深度优先遍历采用递归的方式实现，从任意一个节点开始，按照深度优先的顺序访问所有节点，记录下访问的节点顺序，并输出遍历结果。广度优先遍历则采用队列的方式实现，从任意一个节点开始，按照广度优先的顺序访问所有节点，记录下访问的节点顺序，并输出遍历结果。  通过本次实验，我们掌握了图的存储方式——邻接矩阵，以及实现深度优先遍历和广度优先遍历的算法。同时，我们也学会了如何使用 C++ 语言来实现这些功能，提高了我们的编程技能和实践能力。  **六、教师评语：** | | | | | |