**数据结构实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号-姓名 |  | 实验时间 | 2023 年12 月22 日 |
| 诚信声明 | 手写，签名 | | |
| 实验题目 | 图的深度优先遍历 | | |
| 实验过程中遇到的主要问题 | 主要问题是理解和实现邻接矩阵和邻接表的数据结构，并使用这些数据结构进行深度优先遍历操作。 | | |
| 实验小结 | 本次实验中，我学习了图的深度优先遍历，并使用邻接矩阵和邻接表两种数据结构实现了该算法。通过实验，我了解了深度优先遍历的原理和实现方法，以及如何使用邻接矩阵和邻接表来存储图的信息。 | | |
| 数据结构  （自定义数据类型） | // 邻接矩阵的数据结构  typedef int AdjMatrix[MAXV][MAXV];  struct GraphMatrix {  int vexnum, arcnum;  AdjMatrix arcs;  };  // 邻接表的数据结构  struct ArcNode {  int adjvex;  double weight;  ArcNode \*nextarc;  };  struct VertexNode {  ArcNode \*firstarc;  };  typedef VertexNode AdjList[MAXV];  struct GraphList {  int vexnum, arcnum;  AdjList vertices;  }; | | |
| 主要算法  （或算法说明） | 深度优先遍历算法：   1. 从给定的起始节点开始，标记当前节点为已访问并进行访问操作。 2. 对当前节点的所有未访问的相邻节点执行操作，查找相邻节点在邻接矩阵中为遍历一整行并查找非0元素，在邻接表中为遍历链表。对于节点递归地调用深度优先遍历算法。 3. 还需处理非连通图，如果图不是连通的，即还有未访问的顶点，那么选择一个未访问的顶点，回到步骤1 | | |

**数据结构实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号-姓名 |  | 实验时间 | 2023 年12 月22 日 |
| 诚信声明 | 手写，签名 | | |
| 实验题目 | 图的广度优先遍历 | | |
| 实验过程中遇到的主要问题 | 主要问题是需要注意在处理不连通图时，需要在每个未访问的顶点上重新开始广度优先搜索，以确保访问到所有顶点。 | | |
| 实验小结 | 通过这个实验，我了解了如何使用邻接矩阵和邻接表来表示图，以及如何实现广度优先搜索算法。虽然两种数据结构都可以有效地表示图，但它们各有优劣，适用于不同的场景。邻接矩阵在稠密图中更有效，而邻接表在稀疏图中更有效。我也学会了广度优先搜索这种重要的图遍历算法，它的应用广泛，可以用于解决各种问题，如寻找最短路径，检查图的连通性等。 | | |
| 数据结构  （自定义数据类型） | // 邻接矩阵的数据结构  typedef int AdjMatrix[MAXV][MAXV];  struct GraphMatrix {  int vexnum, arcnum;  AdjMatrix arcs;  };  // 邻接表的数据结构  struct ArcNode {  int adjvex;  double weight;  ArcNode \*nextarc;  };  struct VertexNode {  ArcNode \*firstarc;  };  typedef VertexNode AdjList[MAXV];  struct GraphList {  int vexnum, arcnum;  AdjList vertices;  }; | | |
| 主要算法  （或算法说明） | 广度优先遍历算法：   1. 首先创建一个队列，用于存储待访问的顶点。同时，使用一个全局数组，用于记录每个顶点是否已被访问，所有顶点初始状态都设为未访问。 2. 选择一个起始顶点，执行访问操作，将其标记为已访问，并将其加入队列。 3. 当队列不为空时，执行以下操作：  * 从队列中取出队首顶点，称为当前顶点。 * 查找当前顶点的所有未访问的邻接顶点。 * 访问找到的所有未访问邻接顶点，并标记为已访问，并将它们加入队列。  1. 重复步骤3，直到队列为空，即所有可达的顶点都已访问。 2. 还需处理非连通图，如果图不是连通的，即还有未访问的顶点，那么选择一个未访问的顶点，回到步骤2。 | | |