# 附录A 实验报告模板

**数据结构实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号-姓名 | 22009200766—方子康 | 实验时间 | 2024年 5月17日 |
| 诚信声明 | 所有实验均由本人亲自完成 | | |
| 实验题目 | **题目一  深度优先遍历**  **题目二  广度优先遍历**  **题目三  求解最小生成树** | | |
| 实验过程中遇到的主要问题 | **题目一  深度优先遍历**   1. **初始化问题：在邻接表建图的时候，nextarc要提前初始化为NULL，在遍历的时候visited数组要初始化为false。邻接矩阵不需要初始化，题目直接给了完整矩阵，0代表没有边** 2. **无向图、有向图看清楚，邻接表无向图建边两次：起初忘记建两次边，导致dfs序列输出一半就没了，要切记** 3. **图连通与否：图如果不连通，需要遍历所有结点来dfs，否则只用找一个结点dfs。**   **题目二  广度优先遍历**   1. **用队列实现，问题同题目一**   **题目三  求解最小生成树**   1. **初始化问题：在邻接矩阵求解最小生成树的时候，closedge.lowcost不需要全部初始化为INF，因为更新邻接点距离的时候会遍历所有边，此时相当于初始化了一遍。但是在邻接表求最小生成树的时候，必须要提前将所有closedge[i].lowcost初始化为INF。因为更新只会涉及邻接点。** 2. **混淆：在邻接矩阵中求距离生成树最近的结点的时候，由于混淆G.arcs[i][j] 与closedge[i].lowcost,导致答案错误。G.arcs[i][j] 是用来更新closedge[i].lowcost的，找距离生成树最近的结点只用管closedge[i].lowcost** 3. **无向图、有向图：求解最小生成树，暗含是无向图，输入样例中只输入了一条边，但是建图的时候要建边两次。** | | |
| 实验小结 | **题目一二三自测数据点都大致相同，分为这几类：**  **1.有向图 2.无向图 3.连通图 4.非连通图 5.图中有无回路**  通过本次实验，我在实现图的邻接矩阵与邻接表的存储结构时，理解了这两种存储结构的适用场景，邻接矩阵使用稠密图，邻接表使用稀疏图。在实现图的遍历搜索时，体会到了递归的便利性，也再次认识到了栈与队列在求解问题时的巧妙。在实现最小生成树的时候专门引进了新的数组closedge表示了非生成树与生成树结点的最小距离，比较巧妙。在实现算法的过程中，需要逻辑清晰，注意力集中，不然代码就会写得很乱，在调试错误的时候可以print打印中间信息。通过本次实验，我再次锻炼了编程的严谨规范，通过实际操作加深了对数据结构的理解，也体会到了数据结构对于解决问题的高效。在解决错误的过程中也提升了代码调试能力。 | | |
| 数据结构  （自定义数据类型） | /\*----------------邻接矩阵-----------------\*/typedef int VertexType; //顶点类型typedef int AdjMarix[MaxVnum][MaxVnum]; //矩阵类型typedef struct{    VertexType vexs[MaxVnum]; //顶点向量（有什么用？）    AdjMarix  arcs; //邻接矩阵    int vexnum, arcnum; //顶点数与边数}Graph\_Marix;/\*-----------------邻接表-------------------\*///表结点typedef struct ArcNode{    int Adjvex; //表头结点的邻接结点的编号    int weight; //边的信息    struct ArcNode\* nextarc; //下一条边}ArcNode;//头结点typedef struct{    VertexType data; //头结点数据    ArcNode\* firstarc; //第一个表结点}AdjList[MaxVnum]; //邻接表//图类型typedef struct{    int vexnum, arcnum; //顶点数与边数    AdjList vertices; //邻接表    int kind; //图的类型标志}Graph\_List; | | |
| 主要算法  （或算法说明） | **题目一：**  **伪代码：DFS 遍历(邻接表)**  **主遍历函数 DFSTraverse\_List**    **辅助递归函数 DFS\_List**    **题目二：**  **伪代码：BFS遍历（邻接表）**    **题目三：**  **伪代码：Prim算法(邻接表)**  **主函数 PRIM\_List** | | |