

**教学上机实验报告**

**课程名称：**   **数据结构**

**任课教师姓名：**  **贾盼盼**

**学生学号：**  312105010207

**学生姓名：**  刘晨阳

**学生专业班级：** 计算2106

**2021 ～ 2022 学年 第一学期**

|  |
| --- |
| **河南理工大学**  **教学上机实验报告评价分值标准** |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 序号 | 评价指标 | 分值 | 评价等级及参考分值 | | | | | 评价分 | | 优 | 良 | 中 | 合格 | 差 | | 1 | 实验报告内容完整充实 | 10 | 10 | 8 | 7 | 6 | 3 |  | | 2 | 实验内容书写规范、字迹工整认真 | 10 | 10 | 8 | 7 | 6 | 3 |  | | 3 | 实验过程叙述详细、概念正确，语言表达准确，结构严谨，调理清楚，逻辑性强，自己努力完成，没有抄袭。 | 30 | 30 | 26 | 23 | 20 | 10 |  | | 4 | 对实验过程中存在的问题分析详细透彻、深刻、全面、规范、，结合实验内容，有自己的个人见解和想法，并能结合该实验提出相关问题，给出解决方法。 | 30 | 30 | 26 | 23 | 20 | 10 |  | | 5 | 实验结果、分析和结论正确无误 | 20 | 20 | 17 | 15 | 13 | 6 |  | | 总得分 | | | | | | | |  |   签名（签章）：  日期： 年 月 日 |
|  |

|  |
| --- |
| **河南理工大学教学上机实验报告** |
| 上机时间 2022 年 11月20 日 |
| **实验题目：**  图结构基本算法的实现 |
| **实验目的和要求：**  **实验目的**：掌握图的存储结构及深度优先、广度优先遍历算法,理解图在实际应用中的经典算法如：最小生成树算法、最短路径算法、拓扑排序算法等。  **实验内容：**实现邻接链表和逆邻接链表两种求顶点入度的算法，并在拓扑排序算法及求关键路径算法中应用。 |
| **实验过程：**  **程序主要功能源代码：**  **实验一：拓扑排序及关键路径： int CreateUDG(ALGraph& G) {**  **//创建有向图G的邻接表、逆邻接表**  **int i, k;**  **cout << "请输入总顶点数，总边数，以空格隔开:";**  **cin >> G.vexnum >> G.arcnum; //输入总顶点数，总边数**  **cout << endl;**  **cout << "输入点的名称，如a" << endl;**  **for (i = 0; i < G.vexnum; ++i) { //输入各点，构造表头结点表**  **cout << "请输入第" << (i + 1) << "个点的名称:";**  **cin >> G.vertices[i].data; //输入顶点值**  **G.converse\_vertices[i].data = G.vertices[i].data;**  **//初始化表头结点的指针域为NULL**  **G.vertices[i].firstarc = NULL;**  **G.converse\_vertices[i].firstarc = NULL;**  **}//for**  **cout << endl;**  **cout << "输入边依附的顶点及其权值，如a b 3" << endl;**  **for (k = 0; k < G.arcnum; ++k) { //输入各边，构造邻接表**  **VerTexType v1, v2;**  **int i, j, w;**  **cout << "请输入第" << (k + 1) << "条边依附的顶点及其权值:";**  **cin >> v1 >> v2 >> w; //输入一条边依附的两个顶点**  **i = LocateVex(G, v1); j = LocateVex(G, v2);**  **//确定v1和v2在G中位置，即顶点在G.vertices中的序号**  **ArcNode\* p1 = new ArcNode; //生成一个新的边结点\*p1**  **p1->adjvex = j; //邻接点序号为j**  **p1->nextarc = G.vertices[i].firstarc; G.vertices[i].firstarc = p1;**  **p1->weight = w;**  **//将新结点\*p1插入顶点vi的边表头部**  **ArcNode\* p2 = new ArcNode; //生成一个新的边结点\*p1**  **p2->adjvex = i; //逆邻接点序号为i**  **p2->nextarc = G.converse\_vertices[j].firstarc; G.converse\_vertices[j].firstarc = p2;**  **p2->weight = w;**  **//将新结点\*p1插入顶点vi的边表头部**  **}//for**  **return OK;**  **}//CreateUDG**  **void FindInDegree(ALGraph G) {**  **//求出各顶点的入度存入数组indegree中**  **int i, count;**  **for (i = 0; i < G.vexnum; i++) {**  **count = 0;**  **ArcNode\* p = G.converse\_vertices[i].firstarc;**  **if (p) {**  **while (p) {**  **p = p->nextarc;**  **count++;**  **}**  **}//if**  **indegree[i] = count;**  **}//for**  **}//FindInDegree**  **/\*void FindInDegree(ALGraph G)**  **{ // 正邻接表求顶点的入度**  **int i;**  **ArcNode \*p;**  **for(i=0;i<G.vexnum;i++)**  **indegree[i]=0; // 赋初值**  **for(i=0;i<G.vexnum;i++)**  **{**  **p=G.vertices[i].firstarc;**  **while(p)**  **{**  **indegree[p->adjvex]++;**  **p=p->nextarc;**  **}**  **}**  **}**  **\*/**  **int TopologicalOrder(ALGraph G, int topo[]) {**  **//有向图G采用邻接表存储结构**  **//若G无回路，则生成G的一个拓扑序列topo[]并返回OK，否则ERROR**  **int i, m;**  **FindInDegree(G); //求出各顶点的入度存入数组indegree中**  **InitStack(S); //栈S初始化为空**  **for (i = 0; i < G.vexnum; ++i)**  **if (!indegree[i]) Push(S, i); //入度为0者进栈**  **m = 0; //对输出顶点计数，初始为0**  **while (!StackEmpty(S)) { //栈S非空**  **Pop(S, i); //将栈顶顶点vi出栈**  **topo[m] = i; //将vi保存在拓扑序列数组topo中**  **++m; //对输出顶点计数**  **ArcNode\* p = G.vertices[i].firstarc; //p指向vi的第一个邻接点**  **while (p) {**  **int k = p->adjvex; //vk为vi的邻接点**  **--indegree[k]; //vi的每个邻接点的入度减1**  **if (indegree[k] == 0) Push(S, k); //若入度减为0，则入栈**  **p = p->nextarc; //p指向顶点vi下一个邻接结点**  **}//while**  **}//while**  **if (m < G.vexnum) return ERROR; //该有向图有回路**  **else return OK;**  **}//TopologicalOrder**  **int CriticalPath(ALGraph G) {**  **//G为邻接表存储的有向网，输出G的各项关键活动**  **int n, i, k, j, e, l;**  **if (!TopologicalOrder(G, topo)) return ERROR;**  **//调用拓扑排序算法，使拓扑序列保存在topo中，若调用失败，则存在有向环，返回ERROR**  **n = G.vexnum; //n为顶点个数**  **for (i = 0; i < n; i++) //给每个事件的最早发生时间置初值0**  **ve[i] = 0;**  **/\*――――――――――按拓扑次序求每个事件的最早发生时间－――――－―――――\*/**  **for (i = 0; i < n; i++) {**  **k = topo[i]; //取得拓扑序列中的顶点序号k**  **ArcNode\* p = G.vertices[k].firstarc; //p指向k的第一个邻接顶点**  **while (p != NULL) { //依次更新k的所有邻接顶点的最早发生时间**  **j = p->adjvex; //j为邻接顶点的序号**  **if (ve[j] < ve[k] + p->weight) //更新顶点j的最早发生时间ve[j]**  **ve[j] = ve[k] + p->weight;**  **p = p->nextarc; //p指向k的下一个邻接顶点**  **} //while**  **} //for**  **for (i = 0; i < n; i++) //给每个事件的最迟发生时间置初值ve[n-1]**  **vl[i] = ve[n - 1];**  **/\*――――――――――按逆拓扑次序求每个事件的最迟发生时间－――――－―――――\*/**  **for (i = n - 1; i >= 0; i--) {**  **k = topo[i]; //取得拓扑序列中的顶点序号k**  **ArcNode\* p = G.vertices[k].firstarc; //p指向k的第一个邻接顶点**  **while (p != NULL) { //根据k的邻接点，更新k的最迟发生时间**  **j = p->adjvex; //j为邻接顶点的序号**  **if (vl[k] > vl[j] - p->weight) //更新顶点k的最迟发生时间vl[k]**  **vl[k] = vl[j] - p->weight;**  **p = p->nextarc; //p指向k的下一个邻接顶点**  **}//while**  **}//for**  **/\*――――――――――――判断每一活动是否为关键活动－――――――－―――――\*/**  **cout << endl;**  **cout << "关键活动路径为:";**  **for (i = 0; i < n; i++) { //每次循环针对vi为活动开始点的所有活动**  **ArcNode\* p = G.vertices[i].firstarc; //p指向i的第一个邻接顶点**  **while (p != NULL) {**  **j = p->adjvex; //j为i的邻接顶点的序号**  **e = ve[i];**  **l = vl[j] - p->weight;**  **if (e == l) //若为关键活动，则输出<vi, vj>**  **cout << G.vertices[i].data << "-->" << G.vertices[j].data << " ";**  **p = p->nextarc; //p指向i的下一个邻接顶点**  **} //while**  **} //for**  **return OK;**  **}//CriticalPath**  **int main() {**  **cout << "\*\*\*\*\*\*\*\*\*\*\*\*算法6.13　关键路径算法\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;**  **ALGraph G;**  **CreateUDG(G);**  **int\* topo = new int[G.vexnum];**  **cout << endl;**  **cout << "有向图创建完成!" << endl << endl;**  **if (!CriticalPath(G))**  **cout << "网中存在环，无法进行拓扑排序！" << endl << endl;**  **cout << endl;**  **return OK;**  **}//main**  **实验二：Floyed算法：**    **实验三：Kruskal**    **实验四：prim** |
| **实验结果：**  **实验一：拓扑排序及关键路径：**    **实验二：**  **最短路径Floyed**    **实验三：kruskal:**    **实验四：Prim:** |
| **实验分析：**   1. **拓扑排序及关键路径** 2. **拓扑排序**   **求出各顶点的入度4存入数组indegree[i] 使入度为0的顶点入栈，若不空则一直循环让顶点出栈且保存在拓扑数组里，让每个顶点连接点的入度减一，变成0了就入栈。如果输出的数量小于顶点个数，说明有环。**   1. **关键路径**   **对图中顶点排序，按拓扑序列求出每个时间最早发生的时间ve[i]，按逆拓扑序列求出每个事件发生的最迟时间vl[i]。如果ve[i]==vl[i] 的活动a[i]就是关键路径**   1. **最短路径Floyd**   **算法实现：先初始化所有最短路径长度，D[i][j]=G.arcs[i][j]然后进行n次比较和更新。**  **每次找i,j 之间最短的中间点，寻找最小的路径**   1. **最小生成树Kruskla**   **算法实现：将所有边的信息按权值从小到大，每次循环从排好序的边找出一条，判断他们是不是再一个Vexset中在就舍去大的边，若不是，合并这两个连通分量。**   1. **最小生成树Prim**   **算法实现：初始化所有顶点，对其余的每一个顶点v，将closedge[j] 均初始化到u的边信息，循环n-1次，每次选出最小的closedge[k] 输出此边，将k加入，更新剩余的最小边信息，再次循环。** |
| **实验成绩：**  日期： 年 月 日 |