1 线性表

26 页“时间复杂度却不同”

2 串

3 树

3.1 树的存储结构

3.1.1 双亲表示法

如果需要访问双亲的情况比较多，可以用这种方法。
不仅可以用来表示数，也可以用来表示森林。（显然的。不连通即可。）

3.1.2 孩子表示法

容易访问孩子，不容易访问双亲。需要增加头结点去表示森林

```
//孩子链表节点
class CTNode{
    int child;
    CTNode *next;
};
tpyedef CTNode* ChildPtr;

//孩子链表
class CTBox{
    TElemType data;
    //孩子链表头指针
    ChildPtr firstchild;
};

//孩纸表示法树
class CTree{
    CTBox nodes[MAX_TREE_SIZE];
    int n, r;
}
```

3.1.3 兄弟表示法

可以把所有根节点当作兄弟。此时可以对森林作先根遍历（对应二叉树先序遍历）和后根遍历（对应二叉树中序遍历）。

```
//孩子兄弟链表节点
class CSNode{
    ElemType data; //节点的数据
    CSNode *firstchild; //节点的第一个孩子
    CSNode *nextsibling; //节点的兄弟
};
```

3.1.4 例 8

统计树（森林）中叶子节点个数。即二叉链中结点的 firstchild 为空的数目。

```
int n = 0;
void CSleaf(CSTree T){
    if(T != nullptr]) {
        if(T->firstchild != nullptr) n++;
        CSleaf(T->firstchild);
        CSleaf(T->nextsibling);
    }
}
```

3.1.5 例 10

求树的度（最大分支数）

```
void SCDegree(CSTree T, int &degree) {
    if(T == NULL) degree = 0;
    else {
        d = 1;p = T->firstchild;
        // 遍历兄弟来统计孩子数
        while(p->nextsibling != NULL) {
            p = p->nextsibling; d++;
        }
    }
    CSDegree(T->firstchild , d1); CSDegree(T->nextsilbling , d2);
    degree = max(d1, d2, d);
```

```
12 }
```

3.2 最优树，最优二叉树——赫夫曼树

3.2.1 一些概念

路径：从树中一个结点到另一个结点之间的分支。
路径长度：路径分支数目。
树的路径长度：根到每一结点路径长度之和
结点的带权路径长度：该结点到树根之间的路径长度与结点上权的乘积
树的带权路径长度：加起来。$WPL = \sum_{i=1}^{n} w_i * l_i$

3.2.2 赫夫曼树的应用

最佳判定算法。

3.3 树与等价问题

- 等价类划分
- MFSet 树型结构，Find(S, x),Merge(&S, i, j)

3.4 回溯法与树的遍历

3.4.1 求含 n 个元素的集合的幂集

算法基本描述如下：
- n 个元素 $\rightarrow$ 高度为 n 的满二叉树
- 根节点：空集
- 叶子：形成幂集中的一个元素
- 左分支：取
- 右分支：舍

3.4.2 树的计数

具有 n 个节点的不同形态的树有多少棵？
值相同则称**等价**，不同则称**相似**

给定先序序列，寻找中序遍历的所有可能性。

4 图

- 图在邻接矩阵和邻接表上的遍历算法及应用。
- 求图的最小生成树，最短路径，拓扑排序，关键路径等算法及其应用，性能分析。

一些定义：- 无向图：
  - 无向图边：边 $(v,w) \in E$，则 v 与 w 为邻接点
  - 无向图度：$TD(v)$
  - 连通分量：极大连通子图
- 有向图：
  - 有向图弧：弧 $\langle v,w \rangle \in E$，$w$ 为弧头，v 为弧尾。顶点 v 邻接到顶点 w，顶点 w 邻接自顶点 v
  - 有向图度：入度：$ID(v)$；出度 $OD(v)$
  - 简单路径：没有环路的路径
  - 简单环：没有重复绕圈圈
  - 强连通图：可以有向连通的有向图
  - 强连通分量：有向图中的极大强连通子图
- 稀疏图：$e < nlogn$
- 网：图中边或弧上有权。有向图，有向网，无向图，无向网

4.1 关系集存储方法

教材算法 7.1-7.3

4.1.1 数组表示法邻接矩阵

```
typedef struct {
    // 顶点向量
    VertexType vexs[MAX_VERTEX_NUM];
    AdjMatrix arcs;
    int vexnum, arcnum;
    GraphKind kind;
}
```

4.1.2 邻接表

```
typedef struct ArcNode { //表结点
    int adjvex; //弧指向的顶点的位置
    struct ArcNode *nextarc; 指向下一条弧的指针
    InfoType *info;
} ArcNode;
```

4.2 有向无环图及其应用

- **表达式共享子式**
- **描述一项工程** 工程由若干**活动**组成

4.2.1 拓扑排序

- **顶点表示活动的网 (AOV-网)** 用弧表示活动间有限关系的有向图
- **拓扑排序算法**

1. 在图中找一个没有前驱的顶，输出;
2. 删除以其为尾的弧, 直至所有顶已输出. 可建立一个 0 度顶栈来实现.

时间复杂度为 o(n+e)