数据结构小结

葛乾

February 9, 2023

1 简介

- 数据结构的基本概念
  - 数据结构是指所涉及的数据元素以及数据元素之间的关系，可以看作是相互之间存在着特定关系的数据元素的集合
  - 数据结构中讨论的元素关系主要是指相邻关系或邻接关系
  - 数据元素之间的逻辑关系 → 数据的**逻辑结构**；数据元素及其关系在计算机存储器中的存储方式 → 数据的**存储结构**（或物理结构）；施加在该数据上的操作 → **数据运算**
  - 表示形式：序偶，前驱元素，后继元素，开始元素，终端元素
  - 逻辑结构的类型
    - 线性结构：若结构是非空的，则有且仅有一个开始元素和终端元素，并且所有元素最多只有一个前驱元素和一个后继元素
    - 树形结构：若结构是非空的，则有且仅有一个元素为开始元素（也称为根结点），可以有多个终端元素，每个元素有零个或多个后继元素，除开始元素外每个元素有且仅有一个前驱元素。
    - 树形结构：若结构是非空的，则有且仅有一个元素为开始元素（也称为根结点），可以有多个终端元素，每个元素有零个或多个后继元素，除开始元素外每个元素有且仅有一个前驱元素
  - 存储结构的类型
    - 顺序存储结构: 所有元素存放在一片地址连续的存储单元中；逻辑上相邻的元素在物理位置上也是相邻的，所以不需要额外空间表示元素之间的逻辑关系。存储密度大，存储空间利用率高。但插入删除需要移动元素，适用于查找操作。
    - 链式存储结构：数据元素存放在任意的存储单元中，这组存储单元可以是连续的，也可以是不连续的（注意，每个存储单元内部是连续的）；通过指针域来反映数据元素的逻辑关系。插入删除方便，但存储利用率低。适用于插入与删除操作
    - 索引存储结构
    - 哈希（散列）存储结构
- 抽象数据类型 (ADT) 的写法
- 算法
  - 对特定问题求解步骤的一种描述，它是指令的有限序列
  - 算法的特性：有穷性，确定性，可行性，输入，输出
  - 算法分析：事前估算（算法的执行时间是问题规模 n 的函数，与编写程序的语言不同，执行程序的环境不同等因素无关），事后分析
  - **算法的时间复杂度分析**：算法中执行时间 $T(n)$ 是问题规模 n 的某个函数 $f(n)$，记作：$T(n) = O(f(n))$。大 O 标记的涵义是，存在一个正的常数 c，使得当 $n \geqslant n_0$ 时都满足：$|T(n)| \leqslant c|f(n)|$。这种上界可能很多，通常取最接近的上界，即紧凑上界

* 一个没有循环的算法的执行时间与问题规模 n 无关，记作 $O(1)$，也称作常数阶。
* 一个只有一重循环的算法的执行时间与问题规模 n 的增长呈线性增大关系，记作 $O(n)$，也称线性阶。
* 其余常用的算法时间复杂度还有平方阶 $O(n^2)$、立方阶 $O(n^3)$、对数阶 $O(log_2 n)$、指数阶 $O(2^n)$ 等。

$$O(1) < O(log_2 n) < O(n) < O(nlog_2 n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$$

2 线性表

- 顺序表
  - 插入元素的平均时间复杂度为 $O(n)$
  - 删除元素的平均时间复杂度为 $O(n)$
- **二路归并算法**
- 链表
  - 在给定结点前后插入元素的时间复杂度？在链表任意位置插入元素的时间复杂度？
- 应用：掌握二路归并算法

3 栈与队列

- 栈（stack）是一种只能在同一端进行插入或删除操作的线性表。表中允许进行插入、删除操作的一端称为栈顶（top），表的另一端称为栈底（bottom）。栈的插入操作通常称为进栈或入栈（push），栈的删除操作通常称为退栈或出栈（pop）。
  - 后进先出，即后进栈的元素先出栈。每次进栈的元素都作为新栈顶元素，每次出栈的元素只能是当前栈顶元素。
  - **给定进栈序列，其输出序列的判断**
- 队列（queue）是一种只能在不同端进行插入或删除操作的线性表。进行插入的一端称做队尾（rear），进行删除的一端称做队头或队首（front）。队列的插入操作通常称为进队或入队（push），队列的删除操作通常称为出队或离队（pop）
  - 先进先出，即先进队的元素先出队。每次进队的元素作为新队尾元素，每次出队的元素只能是队头的元素
  - 进出队序列的判断

4 串与数组

- 串的匹配法（不要求掌握）：暴力算法与 KMP 算
- 二维数组按行优先与按列优先存储地址的计算

5 递归

- 在定义一个过程或函数时出现调用本过程或本函数的成分，称之为递归。一般地，一个递归模型是由递归出口和递归体两部分组成
  - 递归出口确定递归到何时结束，即指出明确的递归结束条件。
  - 递归体确定递归求解时的递推关系。
- 优缺点：优点是程序结构简单、清晰，易证明其正确性；缺点是执行中占内存空间较多，运行效率低。
- **递归算法的实现**

6 树与二叉树

- 树相关的一些基本概念
  - 结点的度。树中每个结点具有的子树数或者后继结点数称为该结点的度
  - 树的度。树中所有结点的度的最大值称之为树的度。
  - 分支结点。度大于 0 的结点称为分支结点或非终端结点。度为 1 的结点称为单分支结点，度为 2 的结点称为双分支结点，依次类推。
  - 叶子结点（或叶结点）。度为零的结点称为叶子结点或终端结点。
  - 孩子结点。一个结点的后继称之为该结点的孩子结点。
  - 双亲结点（或父亲结点）。一个结点称为其后继结点的双亲结点。
  - 子孙结点。一个结点的子树中除该结点外的所有结点称之为该结点的子孙结点。
  - 祖先结点。从树根结点到达某个结点的路径上通过的所有结点称为该结点的祖先结点（不含该结点自身）。
  - 兄弟结点。具有同一双亲的结点互相称之为兄弟结点。
  - 结点层次。树具有一种层次结构，根结点为第一层，其孩子结点为第二层，如此类推得到每个结点的层次。
  - 树的高度。树中结点的最大层次称为树的高度或深度。
  - 森林。零棵或多棵互不相交的树的集合称为森林。
- 树的遍历方式
  - 先根遍历
  - 后根遍历
  - 层次遍历
- 二叉树：是有限的结点集合，这个集合或者是空，或者由一个根结点和两棵互不相交的称为左子树和右子树的二叉树组成
  - 树与二叉树的区别：度为 2 的树至少有 3 个结点，而二叉树的结点数可以为 0；度为 2 的树不区分子树的次序，而二叉树中的每个结点最多有两个孩子结点，且必须要区分左右子树，即使在结点只有一棵子树的情况下也要明确指出该子树是左子树还是右子树。二叉树允许为空，树一般不允许为空
  - 在一棵二叉树中，如果所有分支结点都有左孩子结点和右孩子结点，并且叶子结点都集中在二叉树的最下一层，这样的二叉树称为满二叉树。叶子结点都在最下一层; 只有度为 0 和度为 2 的结点; 含 n 个结点的满二叉树的高度为 $log_2(n+1)$，叶子结点个数为 $\lfloor n/2 \rfloor + 1$，度为 2 的结点个数为 $\lfloor n/2 \rfloor$。

  - 若二叉树中最多只有最下面两层的结点的度数可以小于 2，并且最下面一层的叶子结点都依次排列在该层最左边的位置上，则这样的二叉树称为完全二叉树。

- 二叉树的构造
  - 由先序遍历序列和中序遍历序列能够唯一确定一棵二叉树。
  - 由后序遍历序列和中序遍历序列能够唯一确定一棵二叉树。
  - 由先序遍历序列和后序遍历序列不能唯一确定一棵二叉树

- 应用：掌握二叉树的递归遍历算法，以及由先（后）序遍历序列和中序遍历序列构造二叉树的方法。