绪论

目 录

Contents

程序=数据结构+算法

- 绪论
  - 数据结构(三要素)
    - 逻辑结构
      - 线性结构：线性表、栈、队列
      - 非线性结构：树、图、集合
    - 存储结构(物理结构)
    - 数据的运算
  - 五个特征
    - 算法定义
    - 五个特性：有穷性、确定性、可行性、输入、输出
    - 效率的度量
      - 时间复杂度
      - 空间复杂度

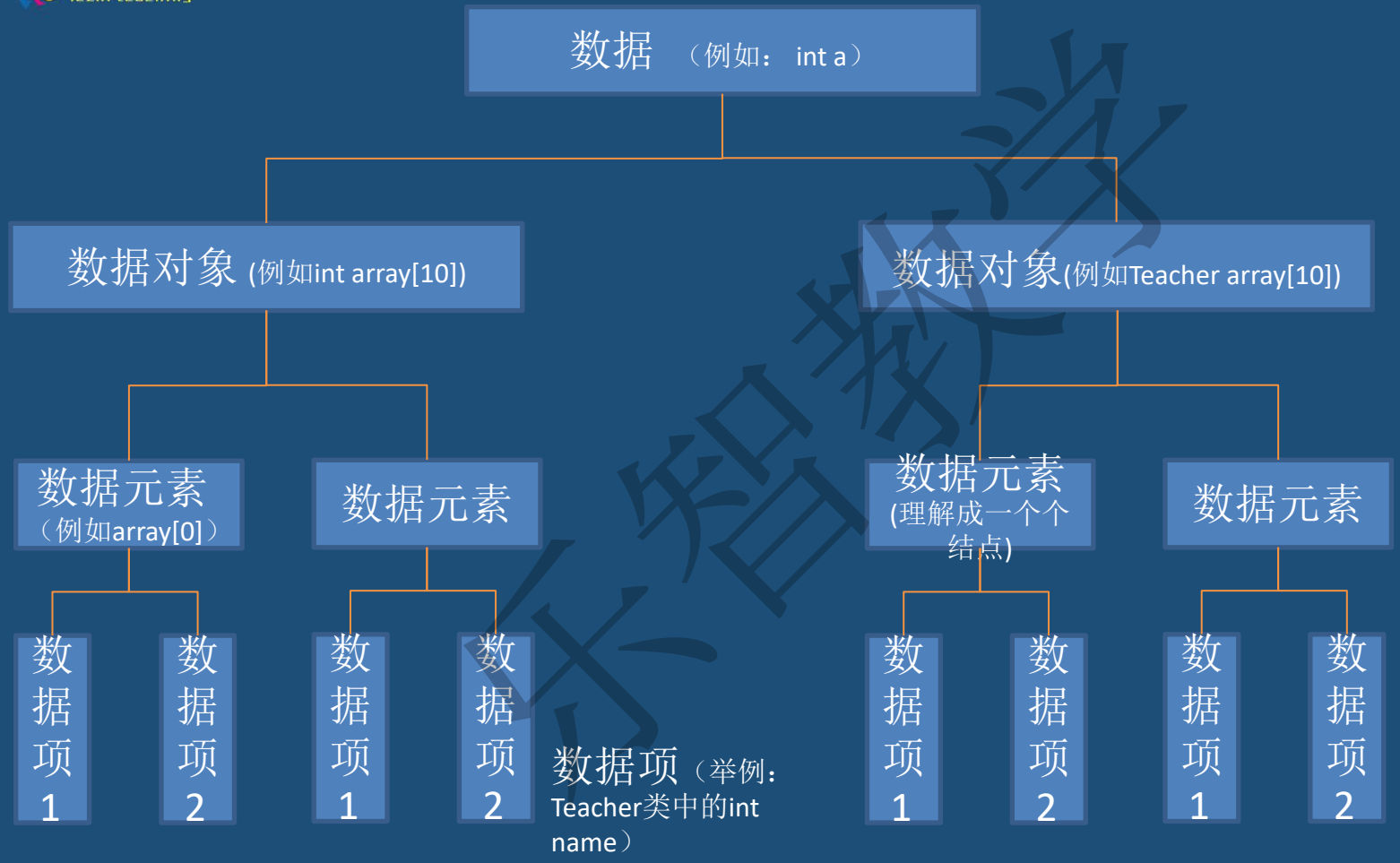
基本概念和术语

数据(Data)：是客观事物的符号表示。在计算机科学中指的是所有能输入到计算机中并被计算机程序处理的符号的总称。

数据元素(Data Element)：是数据的基本单位，在程序中通常作为一个整体来进行考虑和处理。有时，一个数据元素可由若干数据项组成。

一个数据元素可由若干个数据项(Data Item)组成。数据项是数据的不可分割的最小单位。是数据记录中最基本的。不可分的数据单位。

数据对象(Data Object)：是性质相同的数据元素的集合，是数据的一个子集。如数组、链表等。



数据结构(Data Structure)：是指相互之间存在一种或多种特定关系的数据元素的集合。数据结构包括3个方面：逻辑结构、存储结构和数据的运算。

逻辑结构是对数据之间关系的描述，与数据的存储结构无关。分为两大类：线性和非线性。

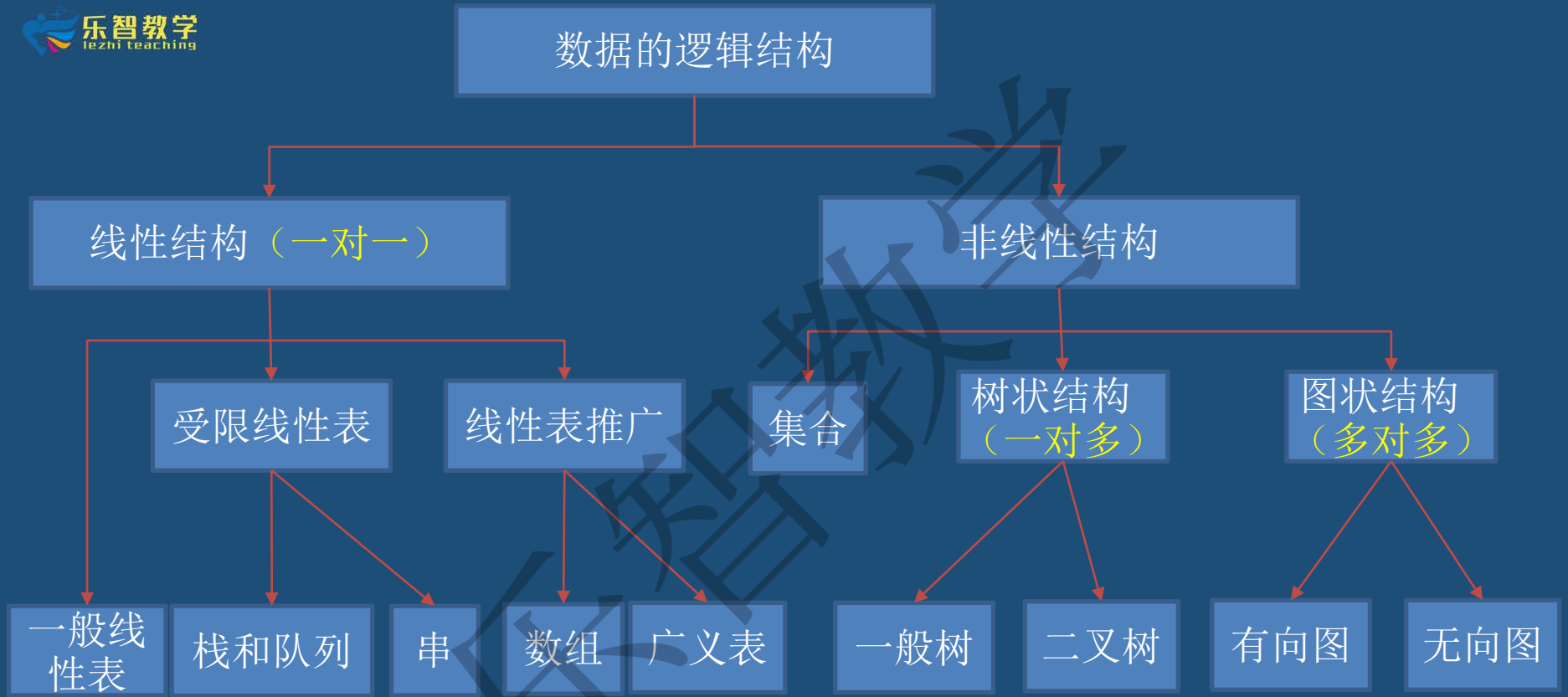
数据元素之间的逻辑结构有四种基本类型：

① 集合：结构中的数据元素除了“同属于一个集合”外，没有其它关系。

② 线性结构：结构中的数据元素之间存在一对一的关系。

③ 树型结构：结构中的数据元素之间存在一对多的关系。

④ 图状结构或网状结构：结构中的数据元素之间存在多对多的关系。



物理结构/存储结构：是数据的逻辑结构在计算机中的表示（又称映像）。
物理结构是描述数据具体在内存中的存储
数据结构中常见的4种存储方式：
顺序存储：顺序存储方法是存储结构类型中的一种,该方法是把逻辑上相邻的结点存储在物理位置上相邻的存储单元中,结点之间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储结构称为顺序存储结构,通常顺序存储结构是借助于计算机程序设计语言(如C/C++)的数组来描述的。
链式存储：链式存储方法不要求逻辑上相邻的结点在物理位置上也相邻,结点间的逻辑关系是由附加的指针字段表示的。由此得到的存储结构表示称为链式存储结构,通常借助于计算机程序设计语言(如C/C++)的指针类型来描述它。

索引存储：索引存储方法在存储结点信息时除建立存储结点信息外,还建立附加的索引表来标识结点的地址。索引项的一般形式是<关键字,地址>关键字标识唯一一个结点,地址作为指向结点的指针

散列存储：散列存储方法的基本思想是根据结点的关键字通过散列函数直接计算出该结点的存储地址。这种存储方法本质上是顺序存储方法的扩展。

数据的运算：

施加在数据上的运算包括运算的定义和实现。运算的定义是针对逻辑结构的，指出运算的功能；运算的实现是针对存储结构的，指出运算的具体操作步骤。

数据结构的主要运算包括：（可以简单记：增删改查）

(1) 建立(Create)一个数据结构；

(2) 消除(Destroy)一个数据结构；

(3) 从一个数据结构中删除(Delete)一个数据元素；

(4) 把一个数据元素插入(Insert)到一个数据结构中；

(5) 对一个数据结构进行访问(Access)；

(6) 对一个数据结构(中的数据元素)进行修改(Modify)；

(7) 对一个数据结构进行排序(Sort)；

(8) 对一个数据结构进行查找(Search)。

1.数据的最小单位是（）
A.数据项　　B.数据类型
C.数据元素　　D.数据变量

2.数据元素是数据的最小单元（）

3.数据结构是（）
A.一种数据类型　　B.数据的存储结构
C.一组性质相同的数据元素的集合
D.相互之间存在一种或多种特定关系的数据元素的集合

答案：A　错　D

1.在存储数据时，通常不仅要存储各数据元素的值，而且还要存储（）

A.数据的处理方法　　B.数据元素的类型
C.数据元素之间的关系　　D.数据的存储方法

2.数据结构中，与所使用的计算机无关的是数据的（ ）结构。
A.存储　　B.物理　　C.逻辑　　D.物理和存储

3.下列4种基本逻辑结构中，数据元素之间关系最弱的是　（）
A.集合　　B.线性结构　　C.树形结构　　D.图形结构

答案：C　C　A

1.可以用（）定义一个完整的数据结构
A.数据元素　B.数据对象　C. 数据关系　　D.抽象数据类型

2.链式存储设计时，节点内的存储单元地址　（）
A.一定连续　　B.一定不连续
C.不一定连续　D.部分连续，部分不连续

3.对于两种不同的数据结构，逻辑结构或物理结构一定不相同吗？

答案：D A
答：应该注意到，数据的运算也是数据结构的一个方面。
对于两种不同的数据结构，他们的逻辑结构和物理结构完全有可能相同。

算法与算法评价

算法是由基本运算及规定的运算顺序所构成的完成的解题步骤，或者看成按照要求设计好的有限的确切的计算序列。

算法的五个特征：

有穷性：指算法在执行有限的步骤之后，自动结束而不会出现无限循环，并且每一个步骤在可接受的时间内完成。

确定性：算法的每一步骤都具有确定的含义，不会出现二义性。

可行性：算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现。

输入：一个算法有零个或多个输入。

输出：一个算法有一个或多个输出。

算法与算法评价（算法的设计要求）

一个算法可以用多种方法描述，主要有：使用自然语言描述；使用形式语言描述；使用计算机程序设计语言描述。

算法和程序是两个不同的概念。一个计算机程序是对一个算法使用某种程序设计语言的具体实现。算法必须可终止意味着不是所有的计算机程序都是算法。

算法与算法评价（算法的设计要求）

评价一个好的算法有以下几个标准：

正确性(Correctness)： 算法应满足具体问题的需求。

可读性(Readability)： 算法应容易供人阅读和交流。可读性好的算法有助于对算法的理解和修改。

健壮性(Robustness)： 算法应具有容错处理。当输入非法或错误数据时，算法应能适当地作出反应或进行处理，而不会产生莫名其妙的输出结果。

通用性(Generality)： 算法应具有一般性，即算法的处理结果对于一般的数据集合都成立。

算法与算法评价（算法的设计要求）

1、事后统计法

比较不同算法对同一组输入数据的运行处理时间

缺陷：为了获得不同算法的运行时间必须编写相应程序。

运行时间严重依赖硬件以及运行时的环境因素；算法的测试数据的选取相当困难。

事后统计法虽然直观，但是实施困难且缺陷多。

2、事前分析估算

依据统计的方法对算法效率进行估算

影响算法效率的主要因素：

算法采用的策略和方法；问题的输入规模；

编译器所产生的代码；计算机执行速度。

算法与算法评价（算法的设计要求）

算法效率的度量是通过时间复杂度和空间复杂度来描述。

算法中基本操作重复执行的次数是问题规模n的某个函数，其时间量度记作T(n)=O(f(n))，称作算法的渐近时间复杂度(Asymptotic Time complexity)，简称时间复杂度。

一般地，常用最深层循环内的语句中的原操作的执行频度(重复执行的次数)来表示。

表示时间复杂度的阶有：

O(1)：常量时间阶　　O (n)：线性时间阶

O(logn)：对数时间阶　O(nlogn)：线性对数时间阶

定理：若$A(n)=a_m n^m+a_{m-1} n^{m-1}+\ldots+a_1 n+a_0$是一个m次多项式，

则$A(n)=O(n^m)$

算法与算法评价（算法的设计要求）

最坏时间复杂度是指在最坏情况下,算法的时间复杂度。

平均时间复杂度是指所有可能输入实例在等概率出现的情况下,算法的期望运行时间。
最好时间复杂度是指在最好情况下,算法的时间复杂度。

一般总是考虑在最坏情况下的时间复杂度,以保证算法的运行时间不会比它更
在分析一个程序的时间复杂性时,有以下两条规则:
a) 加法规则
T(n) = T1(n)+T2(n) = O(f(n))+O(g(n)) = O(max(f(n),g(n)))
b) 乘法规则
T(n) = T1(n) * T2(n) = O(f(n))*O(g(n)) = O(f(n)*g(n))

算法与算法评价（算法的设计要求）

时间复杂度：算法的执行时间与原操作执行次数之和成正比。

常见的渐进时间复杂度为：

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

空间复杂度：算法的空间复杂度通过计算算法所需的存储空间实现。

算法原地工作是指算法所需辅助空间，是常量，即O(1)。

算法与算法评价（算法的设计要求）

空间复杂度(Space complexity)：是指算法编写成程序后，在计算机中运行时所需存储空间大小的度量。记作：S(n)=O(f(n)) 其中：n为问题的规模(或大小)
该存储空间一般包括三个方面：

1、指令常数变量所占用的存储空间;
2、输入数据所占用的存储空间;
3、辅助(存储)空间。

一般地，算法的空间复杂度指的是辅助空间。

一维数组a[n]：空间复杂度 O(n)
二维数组a[n][m]：空间复杂度 O(n*m)

算法与算法评价（真题检测）

1.一个算法应该是　　（ ）
A.程序　　B.问题求解步骤的描述
C.要满足五个基本特征　　D. A和C

2.某算法的时间复杂度为$O(n^2)$，表明该算法的（ ）
A.问题规模是n^2　　B.执行时间等于n^2
C.执行时间与n^2成正比　　D. 问题规模与n^2成正比
3.以下算法的时间复杂度为（ ）

```
void fun(int n){
    int i=1;
    while(i<=n)
        i=i*2;
}
```

A.O(n)　　B.$O(n^2)$　　C.$O(n\log_2 n)$　　D.$O(\log_2 n)$

答案：B　C　D

算法与算法评价（真题检测）

1.下面程序片段的时间复杂度是　　（ ）

```
x=2;
while(x<n/2)
    x=2*x;
```

A.$O(\log_2 n)$　　B.$O(n)$　　C.$O(n\log_2 n)$　　D.$O(n^2)$

2.下面程序段的时间复杂度为　　（ ）

```
a=0;
for (k=1;k<=n;k*=2)
    for (j=1;j<=n;j++)
        a++;
```

A.$O(\log_2 n)$　　B.$O(n)$　　C.$O(n\log_2 n)$　　D.$O(n^2)$

答案：A　C

递归

程序调用自身的编程技巧称为递归（ recursion）。

递归在计算机中是借助栈来实现的

递归可以通过简单的函数调用来完成，如计算**阶乘**的程序在数学上可以定义为：

$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0, \\ n \cdot \text{fact}(n-1) & \text{if } n > 0. \end{cases}$$

递归（举例）

斐波那契数列

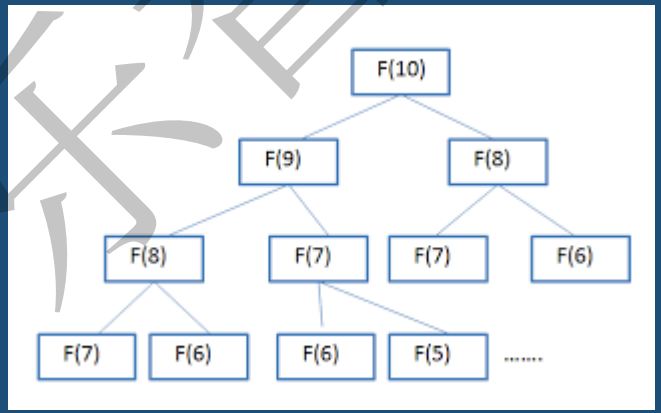
0，1，1，2，3，5，8，13，21，34，55，89，144……依次类推下去，你会发现，它后一个数等于前面两个数的和。

递归思想：一个数等于前两个数的和。

递归表达式：

$$f(n)=\begin{cases} n, & n<=1 \\ f(n-1)+f(n-2), & n>1 \end{cases}$$

斐波那契执行过程：



谢 谢 观 看