第一章 数据结构概论

- 什么是数据结构
- 抽象数据类型及面向对象概念
- 算法定义
- 性能分析与度量

什么是数据结构？

数据结构的概念

- 数据 (Data)：是信息的载体，它能够被计算机识别、存储和加工处理。
- 数据项 (Data Item):有独立含义的数据最小单位，也称域(Field)
- 数据元素 (Data Element)：是数据的基本单位。也称为元素、结点、顶点、记录。
- 数据对象 (Data Object)：是具有相同特性的数据元素的集合，是数据的一个子集。
- 数据结构(Data Structure)：是指数据之间的相互关系，即数据的组织形式。

数据类型

- 高级语言中指数据的取值范围及其上可进行的操作的总称
- 例：C语言中，提供int，char，float，double等基本数据类型，数组、结构体、共用体、枚举等构造数据类型，还有指针、空(void)类 型等。用户也可用typedef 自己定义数据类型

数据结构包括的内容

- 数据元素之间的逻辑关系，也称为数据的逻辑结构(Logical Structure)。
- 数据元素及其关系在计算机存储器内的表示，称为数据的存储结构(Storage Structure)。
- 数据的运算，即对数据施加的操作。

数据结构的定义

- 按某种逻辑关系组织起来的一批数据，应用计算机语言，按一定的存储表示方式把它们存储在计算机的存储器中，并在这些数据上定义了一个运算的集合。

数据的逻辑结构

- 线性结构的逻辑特征：有且仅有一个开始结点和一个终端结点，并且所有结点都最多只有一个直接前趋和直接后继。
- 非线性结构的逻辑特征：一个结点可能有多个直接前趋和直接后继。

数据的存储结构

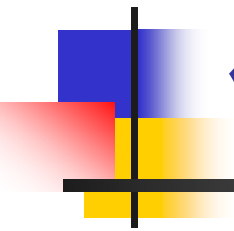
- 顺序存储方法：把逻辑上相邻的结点存储在物理位置上相邻的存储单元里，结点间的逻辑关系由存储单元的邻接关系来体现。
- 链接存储方法：该方法不要求逻辑上相邻的结点在物理位置上也相邻，结点间的逻辑关系由附加的指针字段表示。
- 索引存储方法：在存储结点信息的同时，还建立附加的索引表。
- 散列存储方法：根据结点的关键字直接计算出该结点的存储地址。

同一逻辑结构可有不同的存储结构

- 例如，线性表是一种逻辑结构，若采用顺序方法的存储表示，则为顺序表；若采用链接方法的存储表示，则为链表；若采用散列方法的存储表示，则为散列表；若对线性表上的插入、删除运算限制在表的一端进行，则为栈；若对插入限制在表的一端进行，而删除限制在表的另一端进行，则为队列。

数据结构的三个方面：

- 数据的逻辑结构
  - 线性结构
    - 线性表
    - 栈
    - 队
  - 非线性结构
    - 树形结构
    - 图形结构
- 数据的存储结构
  - 顺序存储
  - 链式存储
  - 索引存储
  - 散列存储
- 数据的运算：检索、排序、插入、删除、修改等



什么是算法？

算法描述

- 算法：解决某一特定问题的具体步骤的描述，是指令的有限序列。
- 算法特性：
  - 输入性：具有零个或多个输入的外界量。
  - 输出性：至少产生一个输出。
  - 有穷性：每一条指令的执行次数必须是有限的。
  - 确定性：每条指令的含义都必须明确，无二义性。
  - 可行性：每条指令的执行时间都是有限的。

算法评价

- 评价一个算法的好坏一般从4个方面进行：
  - 正确性：是指算法是否正确；
  - 运行时间：执行算法所耗费的时间；
  - 占用空间：执行算法所耗费的存储空间；
  - 简单性：是指算法的易读性等。

算法性能分析与度量

- 算法的性能标准
  - 正确性，可使用性，可读性，效率，健壮性
- 算法的后期测试
  - 在算法中的某些部位插装时间函数 time ( )
  - 测定算法完成某一功能所花费时间
- 算法的事前估计
  - 空间复杂度
  - 时间复杂度

空间复杂度度量

- 存储空间的固定部分
  - 程序指令代码的空间，常数、简单变量、定长成分(如数组元素、结构成分、对象的数据成员等)变量所占空间
- 可变部分
  - 尺寸与实例特性有关的成分变量所占空间、引用变量所占空间、递归栈所用空间、通过new和delete命令动态使用空间

时间复杂度度量

- 编译时间
- 运行时间
  - 程序步
    - 语法上或语义上有意义的一段指令序列
    - 执行时间与实例特性无关
    - 例如：声明语句：程序步数为0; 表达式：程序步数为1

例：求两个 n 阶方阵的乘积 C=A×B， 其算法如下：

```
#define  n 自然数
MATRIXMLT(A，B，C)
float A[ ][n]，B[ ] [n]，C [ ][n] ；
{int  i，j，k ；
(1)    for ( i=0 ; i<n ; i++)                    n+1
(2)      for (j=0 ; j<n ; j++)                   n(n+1)
(3)        {C [i][j]=0  ;                        n^2
(4)          for (k=0 ; k<n ; k++)               n^2 (n+1)
(5)          C [i][j]= C [i][j]+ A [i][k]*B      n^3
[k][j]  ;
            }
 }/* MATRIXMLT */
```

该算法中所有语句的频度之和（即算法的时间耗费）为：

- $T(n)= n+1+n(n+1)+n^2+n^2(n+1)+n^3$
- $=2\ n^3+3\ n^2+2n+1$
- 我们将算法求解问题的输入量（或初始数据量）称为问题的规模（Size，大小），并用一个整数表示。例如，矩阵乘积问题的规模是矩阵的阶数n。

算法MATRIXMLT的时间复杂度

$T(n)= 2\ n^3 +3\ n^2+2n+1$

$\lim_{n \to \infty} T(n)/\ n^3 = \lim (2\ n^3 +3\ n^2+2n+1)/\ n^3 = 2$，

这表明，当n充分大时，T(n)和n^3的数量级相同，可记作$T(n)=O(n^3)$。我们称

$$T(n)=O(n^3)$$

是算法MATRIXMLT的时间复杂度。

时间复杂度相应的数量级(阶)按递增排列有:

常数阶$O(1)$

对数阶$O(\log_2 n)$

线性阶$O(n)$

线性对数阶$O(n\log_2 n)$

平方阶$O(n^2)$

立方阶$O(n^3)$

……

k次方阶$O(n^k)$

指数阶$O(2^n)$

- 例：分析以下程序段的时间复杂度。
- i=1；(1)
- while (i<=n)
- i=i*2；(2)
- 其中语句(1) 的频度是1，设语句(2) 的频度是f(n)，则有：
- $2^{f(n)-1} \leq n$
- 即$f(n) \leq \log_2 n+1$，取最大值$f(n) = \log_2 n +1$
- 则该程序段的时间复杂度
- $T(n) = 1+ f(n) = 1+1+\log_2 n = O(\log_2 n)$