1. 绪论

1.1 数据结构基本概念

1.1.1 基本概念和术语

术语	定义
数据	数据是信息的载体，是描述客观事物属性的数、 字符及所有能输入到计算机中并被计算机程序识别和处理的符号的集合
数据元素	数据元素是数据的基本单位，通常作为一个整体进行考虑和处理，含有多个数据项
数据项	是构成数据元素的不可分割的最小单位
数据对象	具有相同性质的数据元素的集合，是数据的一个子集
数据类型	一个值的集合和定义在此集合上的一组操作的总称 {原子类型、结构类型、抽象数据类型}
数据结构	相互之间存在一种或多种特定关系的数据元素的集合

数据类型

数据类型	定义
原子类型	其值不可再分的数据类型
结构类型	其值可以再分解为若干成分的数据类型
抽象数据类型 ADT	抽象数据组织及与之相关的操作

抽象数据类型的定义格式

```
ADT 抽象数据类型名{//抽象数据类型定义格式
        数据对象:<数据对象的定义>          //自然语言
        数据关系:<数据关系的定义>          //自然语言
        基本操作:<基本操作的定义>
}ADT 抽象数据类型名;

//基本操作的定义格式
基本操作名(参数表)        //参数表中赋值参数只提供输入值，引用参数以&打头，可提供输入值和返回操作结果
    初始条件:<初始条件描述>
    操作结果:<操作结果描述>
```

1.1.2 数据结构三要素

逻辑结构

定义：逻辑结构是指数据元素之间的逻辑关系，即从逻辑关系上描述数据。

分类：

- 线性结构：一般线性表、受限线性表（栈和队列）、线性表推广（数组）
- 非线性结构：集合结构、树结构、图结构

存储结构

定义：存储结构是指数据结构在计算机中的表示，也称物理结构

分类：

存储结构	定义	优点	缺点
顺序存储	把逻辑上相邻的元素存储在物理位置上也相邻的存储单元中，元素之间的关系由存储单元的邻接关系来体现	随机存取，占用空间少	使用一整块相邻的存储单元，产生较多碎片
链式存储	不要求逻辑上相邻的元素在物理位置上也相邻，借助指示元素存储地址的指针来表示元素之间的逻辑关系	不会出现碎片，充分利用所有存储单元	需要额外空间，只能顺序存取
索引存储	在存储元素信息的同时，还建立附加的索引表。	检索速度快	附加的索引表需要额外空间。增删数据修改索引表时花费时间
散列存储	根据元素的关键字直接计算出该元素的存储地址，又称哈希(Hash)存储。	检索、增加和删除结点的操作很快	可能出现元素存储单元的冲突，解决冲突会增加时间和空间开销

数据的运算

定义：施加在数据上的运算包括运算的定义和实现。

- 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。

1.2 算法和算法评价

1.2.1 算法的定义、特性和评价标准

定义：算法是针对特定问题求解步骤的一种描述，它是指令的有限序列，其中的每条指令表示一个或多个操作。

特性：

- 输入，零个或多个
- 输出，一个或多个
- 确定性，每条指令含有确定，相同的输入得出相同的结果
- 有穷性
- 可行性，所有可以通过已经实现的基础运算操作有限次来实现

评价标准:

- 正确性：正确结果
- 可读性
- 健壮性：输入数据非法时，能够适当的作出反应或相应处理，不会产生莫名其妙的输出结果
- 高效性：时间和空间

1.2.2 算法效率的度量

- 算法效率分为时间效率和空间效率
- 时间复杂度定义:

$$T(n) = O(f(n)$$

- 空间复杂度定义:

$$S(n) = O(g(n))$$

- 函数渐近的界$O(g(n))$,存在正数c和n_0使得对于一切$n \geq n_0, 0 \leq f(n) \leq cg(n)$

注意:

- 原地工作的算法的空间复杂度为O(1);
- 算法的时间复杂度不仅仅依赖于数据的规模，也取决于待输入数据的性质，如数据的初始状态;

算法复杂度分析步骤

- 确定表示输入规模的参数
- 找出算法的基本操作
- 检查基本操作的执行次数是否只依赖于输入规模。这决定是否需要考虑最差、平均以及最优情况下的复杂性
- 对于非递归算法，建立算法基本操作执行次数的求和表达式；对于递归算法，- 建立算法基本操作执行次数的递推关系及其初始条件
- 利用求和公式和法则建立一个操作次数的闭合公式，或者求解递推公式，确定增长的阶

加法法则:

$$T(n) = T_1(n) + T_2(n) = O(f(n)) + O(g(n)) = O(max(f(n), g(n)))$$

乘法法则:

$$T(n) = T_1(n) \times T_2(n) = O(f(n)) \times O(g(n)) = O(f(n)) \times O(g(n))$$

常见的复杂度:

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

两类递归算法问题的复杂度求解:

- 线性分解

$$T(n) = \begin{cases} O(1) & n = 1 \\ aT(n-1) + f(n) & n > 1 \end{cases}$$

$$T(n) = a^{n-1}T(1) + \sum_{i=2}^{n} a^{n-i} f(i)$$

- 指数分解

$$T(n) = \begin{cases} O(1) & n = 1 \\ aT(\frac{n}{b}) + f(n) & n > 1 \end{cases}$$

$$T(n) = n^{log_b a} T(1) + \sum_{j=0}^{log_b n - 1} a^j f(\frac{n}{b^j})$$