# 数据结构和算法导论

1. 数据结构：程序设计=数据结构+算法

分为逻辑结构，物理结构

1. 逻辑结构：

集合结构：元素同属一个集合

线性结构：一对一

树形结构：一对多

图形结构：多对多

1. 物理结构：

顺序存储：地址连续，逻辑关系和物理关系一致

链式存储：地址不连续，逻辑连续

1. 算法：解决特定问题求解步骤的描述
2. 五个基本特征：

输入：零个或多个输入

输出：至少有一个或多个输出

有穷性：执行有限步骤后自动结束

确定性：每一个步骤都具有确定含义，在一定的条件下，只有一条执行路径，相同的输入只能有唯一的输出结果

可行性：每一步都必须可行的，每一步都能够执行有限次数完成

1. 设计的要求：

正确性：

* 算法程序没有语法错误
* 算法程序对于合法输入能够产生满足要求的输出
* 算法程序对于非法输入能够产生满足规格的说明
* 算法程序对于故意刁难的测试输入都有满足要求的输出结果

可读性：

* 方便阅读，理解和交流

健壮性：

* 输入数据不合法时，算法也能够作出相关处理

高效性：

* 时间效率高和存储量低

# 算法效率的度量方法

1. 事后统计方法：一般不使用
2. 事前分析估算方法：编写程序前，依据统计方法对算法进行估算
3. 算法采用的策略，方案
4. 不同编译器编译产生的代码质量
5. 问题的输入规模
6. 机器执行指令的速度

# 时间复杂度 空间复杂度

1. 算法时间复杂度的定义：在进行算法分析时，语句总的执行次数T（n）是关于问题规模n的函数，进而分析T（n）随n的变化情况并确定T（n）的数量级。T（n）= O（f(n)）,它表示随问题规模n的增大，算法执行时间的增长率和f(n)的增长率相同，称作算法的渐进时间复杂度，简称时间复杂度。
2. 分析算法的时间复杂度
3. 用常数1取代运行时间中的所有加法常数
4. 在修改后的运行次数函数中，只保留最高阶项
5. 如果最高阶项存在且不是1，则去除与这个项相乘的常数
6. 时间复杂度分类
7. 常数阶:O(1)
8. 对数阶:O(log(n))
9. 线性阶:O(n)
10. nlog阶:O(nlog(n))
11. 平方阶:O(n^2)
12. 立方阶:O(n^3)
13. 算法空间复杂度：通过计算算法所需的存储空间实现，算法的空间复杂度的计算公式记作：S(n)=O(f(n))，其中，n为问题的规模，f(n)为语句关于n所占存储空间的函数。用“时间复杂度”指运行时间的需求，用“空间复杂度”指空间需求。

# 线性表

1. 定义：由零个或多个数据元素组成的有限序列
2. 首先它是一个序列，元素之间有先后顺序
3. 若元素存在多个，第一个元素无前驱，最后一个无后继，其他元素有且只有一个前驱和后继
4. 元素是有限的
5. 无元素为空表
6. 数据类型定义：是指一组性质相同的值的集合及定义在此集合上的一些操作的总称
7. 原子类型：不可以再分解的基本类型，例如整型，浮点型，字符型
8. 结构类型：由若干个类型组合而成，是可以再分解的，例如整型数组是由若干整型数据组成的
9. 抽象：是指去除事物具有的普遍性的本质，它要求抽出问题的特征而忽略非本质的细节
10. 抽象数据类型：是指一个数学模型及定义在该模型上的一组操作。
11. 线性表的抽象数据类型

ADT 线性表（list）

Data

{a1, a2, ..., an};

Operation

InitList(\*L)：初始化操作;

ListEmpty(L)：判断是否为空表;

...

end ADT

1. 顺序储存结构：用一段地址连续的存储单元一次存储线性表的数据元素

封装的三个属性：

1. 存储空间的起始位置
2. 最大存储容量
3. 当前的长度
4. 地址计算

LOC(ai) = LOC(a1) + (i-1)\*c

取地址时间为O(1)，称为随机存储结构

1. 线性表顺序存储结构：

结构：typedef struct{

Element [maxSize];

int length;

}sqList

优点：

1. 无须为表示表中元素之间的逻辑关系而增加额外的存储空间
2. 可以快速的存取表中任意位置的元素

缺点：

1. 插入和删除操作需要移动大量元素
2. 当线性表长度变化较大时，难以确定存储空间的容量
3. 容易造成存储空间的“碎片”
4. 线性表的链式存储结构

单链表

1. 结构：

typedef struct Node{

ElemType data; // 数据域

struct Node\* Next;// 指针域

}Node；

Typedef struct Node\* = LinkList;

1. 优点：
   * 1. 插入，删除时间复杂度O(1)
     2. 无需分配空间
2. 缺点：查找时间复杂度O(n);

循环链表