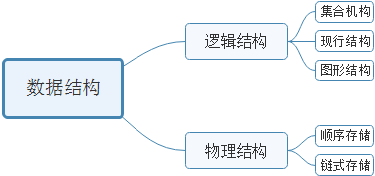
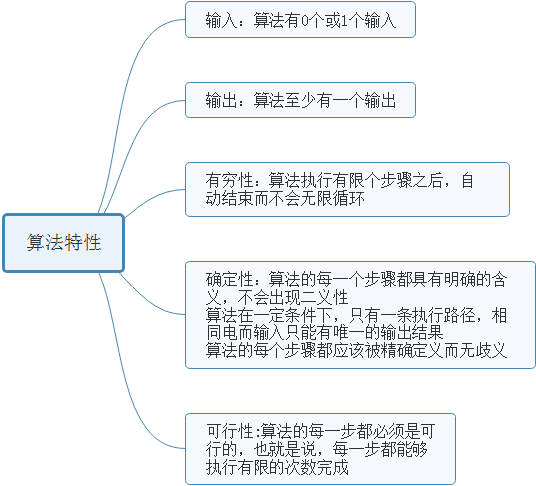
## 01数据结构和算法绪论



## 02谈谈算法

算法五个基本特性:输入、输出、又穷性、确定性、可行性



## 03时间复杂度和空间复杂度1

**算法的度量方法**

(1)事后统计方法:缺点很大

(2)事前分析估算方法：

算法的度量方法

--算法采用的策略，方案

--编译产生的代码质量

--问题的输入规模

--机器执行指令的速度

**函数的渐进增长**:给定两个函数f（n）和g（n）,如果存在一个整数N，在n>N时，f(n)总是比g（n）大，那么我们说f(n)的增长渐进快鱼g(n)

在算法中，随着n的增多，加的整数不影响最终曲线变化，所欲我们可以忽略这些加法常数

## 04时间复杂度和空间复杂度2

线性阶o(n)

平方阶0(n2)

对数阶o(logn)

## 05时间复杂度和空间复杂度3

函数调用的时间复杂度分析

最坏情况与平均情况

平均运行时间是期望的运行时间，最坏运行时间是一种保证。在应用中，这是一种最重要的需求，通常除非特别指定，我们提到的运行时间都是最坏情况的运行时间。

算法的空间复杂度

要判断某年是不是闰年，可以写一个算法计算，也可以建立一个2050个元素的数组，然后把所有年份按下标的数字对应，如果是闰年，则元素值为1，这样判断某一年是否为闰年就变成了查找这个数字某一个元素值的问题

## 06线性表1

线性表的定义 (list)

由零个或多个数据元素组成的有限序列,当n=0时，称为空表

问题:公司的组织架构受属于线性关系

答案:不是，其为一对多的关系

数据类型

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

抽象数据类型Abstract data type

对数据类型进行分类，分出多种数据类型来适应各种不同计算条件的差异,例如在C语言中，按照取值的不同，数据类型可以分为两类。

--原子类型：不可再分解的基本类型

--结构类型：有若干个类型组合而成

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。

抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关。例如1+1=2在不同cpu的处理上可能不一样，但由于其定义的数学特征相同，所以在计算机编程者来看，他们都是相同的。

抽象

抽象是指抽取出事务具有的普遍的本质。它要求抽取出问题的特征而忽略非本质的细节，是对具体事物的一个概况。抽象是一种思考问题的方式，它隐藏了繁杂的细节。

抽象的意义在于数据类型的数据抽象特征，而且，抽象数据类型不止已经定义并实现的数据类型，还可以是计算机编程者在设计软件程序时定义的数据类型。

## 07线性表2

线性表的相关操作

初始化

判断是否为空

清空

将线性表第i个元素返回

查找与e相等的元素

插入i

删除i

返回个数

## 08线性表3---线性表的顺序存储结构

线性表有两种物理存储结构:顺序存储结构和链式存储结构

顺序存储结构需要三个属性

--起始位置

--最大存储容量:数组的长度

--当前长度

线性表的顺序存储结构在存读数据时，不管在哪个位置，时间复杂度都是0（1），在插入或删除时，时间复杂度是o（n），这说明，它比较适合元素个数比较稳定，不经常插入和删除元素，而更多是**存取数据**的应用

优点:

--无需为表示表中元素之间的逻辑关系而增加额外的存储空间

--可以快速地存取表中任意位置元素

缺点：

--插入和删除操作需要移动大量元素

--当元素变化比较大时，难以确定存储空间的容量

--容易造成存储空间的碎片

## 09线性表4---线性表的链式存储结构

定义：线性表的链式存储结构的特点是用一组任意的存储单元存储线性表的数据元素，这组存储单元可以在内存中未被占用的任意位置

链表的第一个节点存储位置叫头指针，最后一个节点指针为空。

## 10线性表5---头指针与头节点的异同

头指针

---头指针是指链表指向第一个节点的指针，若链表有头结点，则是指向头结点的指针

---头指针具有标识作用，所以常用头指针冠以链表的名字

---无论链表是否为空，头指针均不为空

---头指针是链表的必要元素

头节点

---头节点是为了操作的统一和方便而设立的，放在第一个元素的节点之前，其数据域一般无异议

---有了头节点，对在第一元素节点前插入节点和删除第一节点起操作与其它节点的操作就统一了

---头节点不一定是链表的必须要素

单链表示意图



空链表示意图



由于单链表中没有定义表长，所以不能实现知道要循环多少次，因此也就不方便使用for来控制循环，其核心思想叫做 **工作指针后移**

## 11线性表6---单链表的插入删除

## 12线性表7---头插法，尾插法建立单链表

## 13线性表8---单链表的整表删除

## 14线性表9---静态链表

## 15线性表10---静态链表操作

## 16线性表11---循环链表

## 17线性表12---循环链表

## 18约瑟夫问题

## 19线性表14

## 20魔术师发牌问题

## 21线性表16

## 22线性表17

## 23栈和队列

队列先进先出，栈先进后出。

栈与队列的相同点：

1.都是线性结构。

2.插入操作都是限定在表尾进行。

3.都可以通过顺序结构和链式结构实现。、

4.插入与删除的时间复杂度都是O（1），在空间复杂度上两者也一样。

5.多链栈和多链队列的管理模式可以相同。

栈与队列的不同点：

1.删除数据元素的位置不同，栈的删除操作在表尾进行，队列的删除操作在表头进行。

2.应用场景不同；常见栈的应用场景包括括号问题的求解，表达式的转换和求值，函数调用和递归实现，深度优先搜索遍历等；常见的队列的应用场景包括计算机系统中各种资源的管理，消息缓冲器的管理和广度优先搜索遍历等

3.顺序栈能够实现多栈空间共享，而顺序队列不能。

## 24栈和队列

## 25\_进制转换

## 26\_栈和队列4

## 27\_逆波兰计算器

## 28\_中缀表达式转换为后缀表达式01

## 29\_中缀表达式转换为后缀表达式02

## 30\_栈和队列7

## 31\_栈和队列8

## 32\_递归和分治思想

## 33\_递归和分治思想2

## 34\_汉诺塔

## 35\_八皇后问题

## 36\_字符串

## 37\_KMP算法

## 38\_KMP算法2

## 39\_KMP算法之NEXT数组代码原理分析

## 40\_KMP算法之实现及优化

## 41\_树

## 42\_树的存储结构

## 43\_树的存储结构2

## 44\_二叉树

## 45\_二叉树2

## 46\_二叉树的存数结构

## 47\_二叉树的遍历

## 48\_二叉树的建立和遍历算法

## 49\_线索二叉树

## 50\_线索二叉树代码实现

## 51\_树、森林及二叉树的相互转换

## 52\_赫夫曼树

## 53\_赫夫曼编码

## 54\_赫夫曼编码C语言实现

## 55\_图

## 56\_图的定义与术语2

## 57\_图的存储结构

## 58\_图的存储结构（邻接表）

## 59\_图的存储结构（十字链表、邻接多重表、边集数组）

## 60\_图的遍历（深度优先遍历）

## 61\_马踏棋盘算法（骑士周游问题）

## 62\_图的遍历（广度优先遍历）

## 63\_最小生成树（普里姆算法）

## 64\_最小生成树（克鲁斯卡尔算法）

## 65\_最短路径（迪杰斯特拉算法）

## 66\_最短路径（弗洛伊德算法）

## 67\_拓扑排序

## 68\_关键路径

## 69\_查找算法

## 69\_关键路径（代码讲解）

## 70\_插值查找（按比例查找）

## 71\_斐波那契查找（黄金分割法查找）

## 72\_线性索引查找

## 73\_二叉排序树

## 74\_二叉排序树的查找、插入和删除

## 75\_二叉排序树的查找、插入和删除2

## 76\_平衡二叉树

## 77\_平衡二叉树的实现原理

## 78\_平衡二叉树的实现原理（代码实现）

## 79\_多路查找树之2-3树

## 80\_多路查找树之2-3树的插入原理

## 81\_多路查找树之2-3树的删除原理

## 82\_多路查找树之2-3-4树和B树

## 83\_散列表（哈希表）查找

## 84\_散列函数的构造方法

## 85\_处理散列冲突的方法

## 86\_散列表查找的代码实现

## 87\_排序算法

## 88\_冒泡排序

## 89选择排序

## 90直接插入排序

## 91希尔排序

## 92堆排序

## 93堆排序的代码实现

## 94归并排序

## 95归并排序（迭代实现）

## 96快速排序

## 97快速排序的优化

## 98总结回顾