# ****选择排序、快速排序、希尔排序、堆排序不是稳定的排序算法，而冒泡排序、插入排序、归并排序和基数排序是稳定的排序算法选择排序、****

# 数据的逻辑结构

概念：数据元素之间的逻辑关系。

类型：

集合结构，数据元素（即一个节点）之间同属于一个集合的关系外，没有其他的关系（节点之间存在一对一的关系）

线性结构，数据元素之间存在着一对一的层次关系（线性表，栈，队列，字符串，数组和广义表）

树形结构，数据元素之间存在着一对多的层次关系

图状结构。数据元素之间存在着多对多的层次关系

# 数据的存储（或物理）结构

概念：研究数据元素之间的关系如何在计算机中表示

类型：

顺序存储：借助元素在存储器中的相对位置来表示数据元素时间的逻辑关系。

链式存储：借助指针表示数据元素之间的逻辑关系

# 算法的性能分析

## 时间复杂度

A:时间复杂度：

大O推导法：

1：用常数1 取代运行时间中的所有加法常数

2：在修改后的运行函数中只保留最高阶的项

3：如果最高阶存在且不是1，则去除这个项相乘的常数

B:大小

常数<对数《（线性《线性对数》）多项式《指数

C:一个算法的执行时间是：所有原操作的执行时间之和。

多个加（原操作的执行次数\*原操作的执行时间）

近似值：取原操作的重复执行次数为依据。--频度

时间之和：频度之和

## 空间复杂度

算法执行期间所需要的存储空间包括：

算法程序所占的空间

输入的初始数据所占的存储空间

算法执行过程中所需要的额外空间

# 线性表

## 逻辑关系：

表中有且仅有一个开始节点，有且仅有一个终端节点，出开始节点外，表中的每个节点均只有一个前驱节点。，出终端节点外每个节点只有一个后继节点

## 特点

同一性：线性表由同类数据元素组成，每个ai必须属于同一数据对象

有穷性：由有限个数据元素组成。

有序性：线性表中相邻数据元素之间存在着序偶关系。

## 顺序表

### 存储结构：

顺序存储：是在内存中用一块地址连续的存储空间按顺序存储线性表中的各个数据元素。

### 特点：

在逻辑上相邻的数据元素在物理存储位置上也是相邻的

随机存取

### 基本运算

1:插入：

时间复杂度：o(n)

2:删除

时间复杂度：o(n)

3:按值查找

时间复杂度：o(n)

## 链表

### 存储结构：

链式存储：通过一组任意的存储单元来存储线性表的数据元素，这组存储单元可以是连续的，也可以是不连续的。

### 特点：

插入删除容易

### 基本运算

1:插入：

时间复杂度：o(n)

2:删除

时间复杂度：o(n)

3:按值查找

时间复杂度：o(n)

# 栈

## 顺序栈

## 链式栈

# 队列

## 顺序队列

## 链式队列

### 数据类型

Struct node

{

Datatype date;

Struct node \*next;

};

Struct queue{

Struct node \*front;

Struct node \*rear;

}；

带头结点的

Struct node \*front指向：头结点

Struct node \*rear 指向：最后一个元素节点

# 串

# 数组与广义表

# 树

## 术语：

树的度：树中所有结点的度的最大值

树的深度：树中所有结点层次的最大值

## 二叉树

### 定义

每个结点的度不大于2

结点每颗子树的位置是区分左右的，不能随意改变

总结：二叉树中每个结点只能由0，1，2个孩子。即使只有一个孩子也是区分左右的。，

### 性质

1：二叉树第i层至多有2（i-1次方）个结点

2：深度为k的二叉树至多有2（k次方）-1个结点

3：若终端结点数为n0，度为2的结点数为n2则n0=n2+1

4；满二叉树：深度为k的二叉树至多有2（k次方）-1个结点

5：完全二叉树：深度为k的二叉树至多有n<2（k次方）-1个结点,且当其n个结点与满二叉树中连续编号为1-n的结点位置一一对应

6：具有n个结点的完全二叉树的深度是log2（n次方）+1

7：对于完全二叉树连续编号后。对于任意序号i

（1）：i=1，则结点i为根，其无双亲结点，若i.>1则i的双亲结点为i/2

（2）：2i<=n :i的左孩子是2i,否则没有做孩子

（3）：2i+1<=n ：i的右孩子是2i+1,否则没有右孩子

### 存储

1：顺序存储

和线性表的一样

2：链式存储

Struct node{

Datatype data;

Struct node \*left;

Struct node \*right;

}

3；三叉链表：

加一个parent域指向双亲结点

### 遍历

1：深度遍历（递归定义的可以用栈实现）

先序

后序

中序

2;层次遍历（队列实现）

自上而下逐层遍历，同层内按从左到右

队列实现的算法是：

首先根节点入队，队列非空时执行下面

对头结点出队，并访问出队结点

出队结点的非空左，右孩子依次入队

### 遍历算法的应用

1. 统计二叉树的节点数
2. 输出二叉树中的叶子节点
3. 统计叶子节点的数目
4. 求二叉树的高度
5. 求节点的双亲
6. 建立二叉链表存储的二叉树

### 有遍历序列确定二叉树

1. 先序+中序
2. 中序+后序

## 哈弗曼树

1.概念

带权路径长度：节点到树根间的路径长度与结点的权的乘积，为该结点的带全路径长度

树的带全路径长度：树中所有叶子结点的带全路径长度之和。

最优二叉树：树的带全路径最小的二叉树。

1. 哈弗曼树的建立

静态三叉链表来存储

哈弗曼树中没有度为1的结点，n个叶子结点，n-1个度为2的结点，总共2n-1个结点。存放在大小为2n-1的一维数组中。

Struct hf{

Int weight;

Int parent,lchild,rchild;

}

1. 哈弗曼编译码

编码是从根-----叶子

左分支表示符号0，右分支表示符号1，用根节点到 叶子结点路径上的分支符号串作为叶子结点的编码。

# 图

# 查找

## 基于线性表的查找

### 顺序查找

1. 数据元素存在线性表中（数组或链表中）
2. 数据元素是无序的
3. 用所给关键字与线性表中的数据逐个的比较。（从前到后或者从后到前
4. 时间复杂度：

### 折半查找

1. 分治类算法
2. 线性表是有序表。
3. 基本的查找过程：

每次讲待查范围中间位置的数据元素的关键字与给定值比较，

若相等成功，

否则利用中间位置将表分成两半，，如果中间位置的关键字》则，在前半部分查找，否则在后半部分查找。

## 基于树的查找

### 二叉排序树=二叉查找树（搜索树）

1. 特点：

若它的左子树不空，则左子树上所有结点的值均小于根结点的值

若它的右子树不空，则右子树上所有结点的值均大于根结点的值

它的左，右子树也都分别是二叉排序树。

对这个二叉排序树进行中序遍历，可以得到一个递增的有序序列，后序遍历则从大到小。

用二叉链表作为存储结构

1. 二叉排序树的查找

由于其是有序表所以在二叉排序树上进行查找类似于折半查找。

1. 二叉排序树的插入

不允许插入重复关键字

### 平衡二叉树

### B树和B+树

### 伸展树

### 红黑树

## 散列

# 排序