# 排序

## JDK排序接口

Comparable

# 常见排序



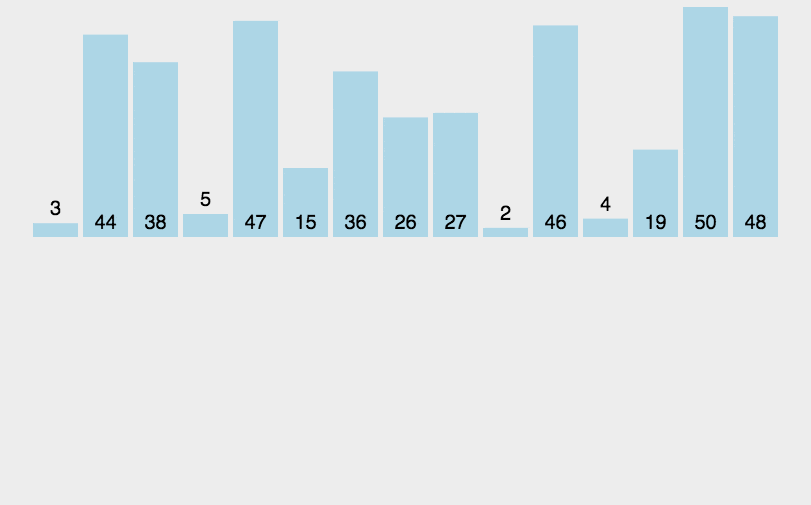


## 插入排序

插入排序，一般也被称为直接插入排序。对于少量元素的排序，它是一个有效的算法。插入排序是一种最简单的排序方法，它的基本思想是将一个记录插入到已经排好序的有序表中，从而一个新的、记录数增1的有序表。在其实现过程使用双层循环，外层循环对除了第一个元素之外的所有元素，内层循环对当前元素前面有序表进行待插入位置查找，并进行移动。

算法步骤

1. 将第一待排序序列第一个元素看做一个有序序列，把第二个元素到最后一个元素当成是未排序序列。
2. 从头到尾依次扫描未排序序列，将扫描到的每个元素插入有序序列的适当位置。（如果待插入的元素与有序序列中的某个元素相等，则将待插入元素插入到相等元素的后面。）



## 希尔排序

希尔排序（Shellsort），也称递减增量排序算法，是插入排序的一种更高效的改进版本。希尔排序是非稳定排序算法。

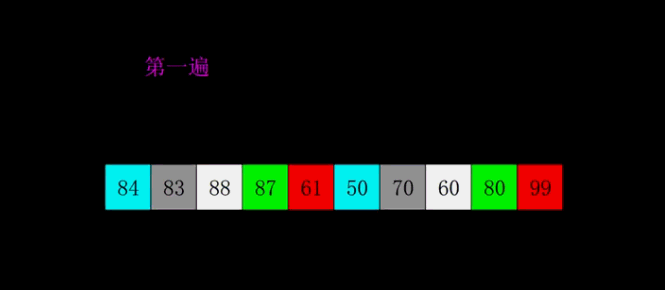
希尔排序是基于插入排序的以下两点性质而提出改进方法的：

1.插入排序在对几乎已经排好序的数据操作时，效率高，即可以达到线性排序的效率

2.但插入排序一般来说是低效的，因为插入排序每次只能将数据移动一位

算法步骤

1. 选择一个增量序列 t1，t2，……，tk，其中 ti > tj, tk = 1；
2. 按增量序列个数 k，对序列进行 k 趟排序；
3. 每趟排序，根据对应的增量 ti，将待排序列分割成若干长度为 m 的子序列，分别对各子表进行直接插入排序。仅增量因子为 1 时，整个序列作为一个表来处理，表长度即为整个序列的长度。

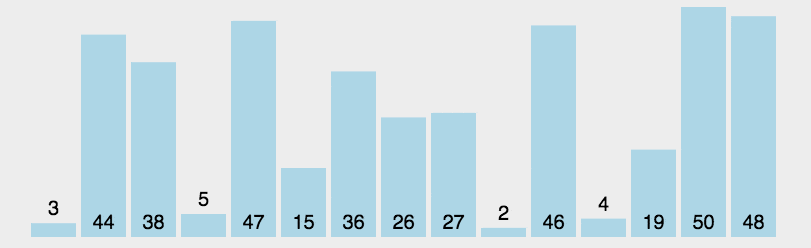


## 选择排序

选择排序是一种简单直观的排序算法，无论什么数据进去都是 O(n²) 的时间复杂度。

算法步骤

1. 首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置。
2. 再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。
3. 重复第二步，直到所有元素均排序完毕。



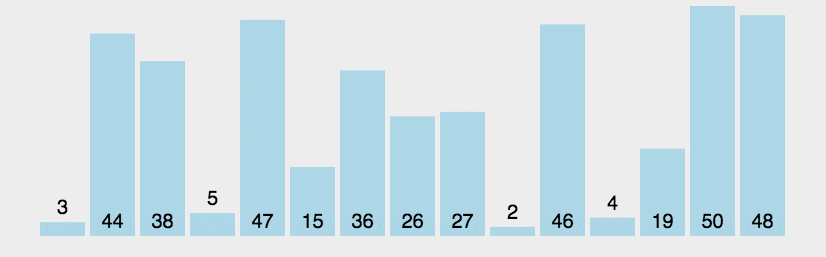
## 堆排序

## 冒泡排序

一种简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。

算法步骤：

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。这步做完后，最后的元素会是最大的数。
3. 针对所有的元素重复以上的步骤，除了最后一个。
4. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

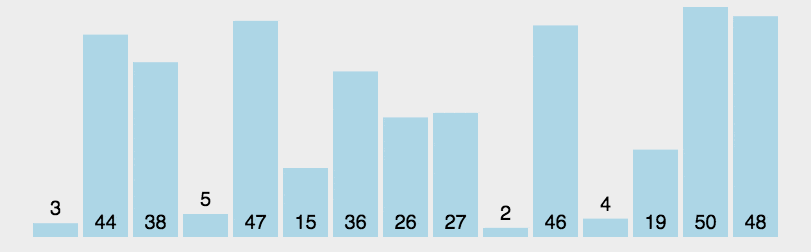


## 快速排序

快速排序是对冒泡排序的一种改进。它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一 部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序 过程可以递归进行，以此达到整个数据变成有序序列。

算法步骤：

1. 首先设定一个分界值，通过该分界值将数组分成左右两部分。
2. 将大于或等于分界值的数据放到到数组右边，小于分界值的数据放到数组的左边。此时左边部分中各元素都小于或等于分界值，而右边部分中各元素都大于或等于分界值。
3. 然后，左边和右边的数据可以独立排序。对于左侧的数组数据，又可以取一个分界值，将该部分数据分成左右两部分，同样在左边放置较小值，右边放置较大值。右侧的数组数据也可以做类似处理。
4. 重复上述过程，可以看出，这是一个递归定义。通过递归将左侧部分排好序后，再递归排好右侧部分的顺序。当左侧和右侧两个部分的数据排完序后，整个数组的排序也就完成了。



## 归并排序

归并排序，是创建在归并操作上的一种有效的排序算法。算法是采用分治法（Divide and Conquer）的一个非常典型的应用，且各层分治递归可以同时进行。归并排序思路简单，速度仅次于快速排序，为稳定排序算法，一般用于对总体无序，但是各子项相对有序的数列。

归并排序是用分治思想，分治模式在每一层递归上有三个步骤：

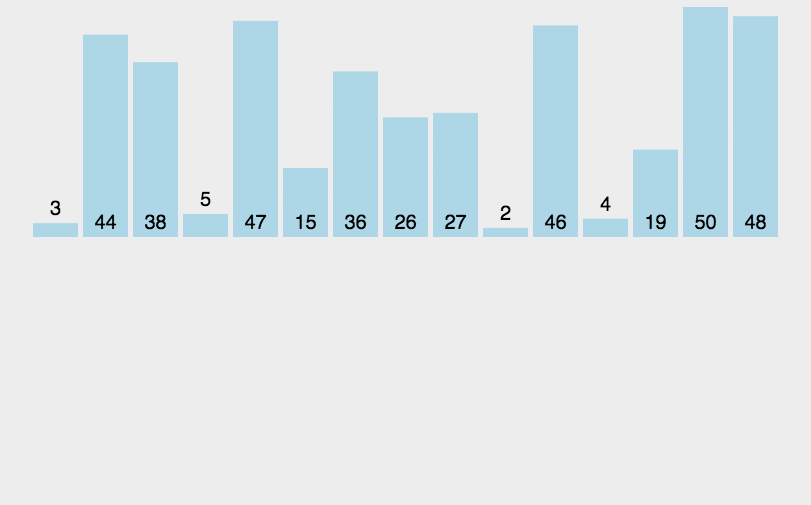
分解（Divide）：将n个元素分成个含n/2个元素的子序列。

解决（Conquer）：用合并排序法对两个子序列递归的排序。

合并（Combine）：合并两个已排序的子序列已得到排序结果。

算法步骤：

1. 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列；
2. 设定两个指针，最初位置分别为两个已经排序序列的起始位置；
3. 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置；
4. 重复步骤 3 直到某一指针达到序列尾；
5. 将另一序列剩下的所有元素直接复制到合并序列尾。



## 计数排序

## 桶排序

## 基数排序

# 线性表

顺序表是在计算机内存中以数组的形式保存的线性表，线性表的顺序存储是指用一组地址连续的存储单元依次存储线性表中的各个元素、使得线性表中在逻辑结构上相邻的数据元素存储在相邻的物理存储单元中，即通过数据元素物理存储的相邻关系来反映数据元素之间逻辑上的相邻关系，采用顺序存储结构的线性表通常称为顺序表。顺序表是将表中的结点依次存放在计算机内存中一组地址连续的存储单元中。

链表（Linked list）是一种常见的基础数据结构，是一种线性表，是一种物理存储单元上非连续、非顺序的存储结构。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括存储数据元素的数据域和存储下一个结点地址的指针域两个部分。 相比于线性表顺序结构，操作复杂。数据元素的逻辑顺序也是通过链表中的指针链接次序实现的。

# 栈

堆栈（英语：stack）又称为栈或堆叠，是计算机科学中的一种抽象资料类型，只允许在有序的线性资料集合的一端（称为堆栈顶端，英语：top）进行加入数据（英语：push）和移除数据（英语：pop）的运算。因而按照后进先出（LIFO, Last In First Out）的原理运作。

# 队列

队列，又称为伫列（queue），计算机科学中的一种抽象资料型别，是先进先出（FIFO, First-In-First-Out）的线性表。在具体应用中通常用链表或者数组来实现。队列只允许在后端（称为rear）进行插入操作，在前端（称为front）进行删除操作。

# 符号表

符号表是一种存储键值对的数据结构，支持插入（put）和查找（get）操作，即给定建得到相应的值。

# 树

树是我们计算机中非常重要的一种数据结构，同时使用树这种数据结构，可以描述现实生活中的很多事物，例如家 谱、单位的组织架构、等等。

树是由n（n>=1）个有限结点组成一个具有层次关系的集合。把它叫做“树”是因为它看起来像一棵倒挂的树，也就 是说它是根朝上，而叶朝下的。

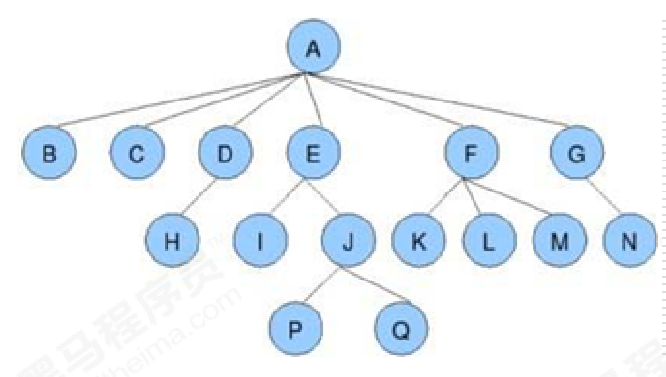
树具有以下特点：

1.每个结点有零个或多个子结点；

2.没有父结点的结点为根结点；

3.每一个非根结点只有一个父结点；

4.每个结点及其后代结点整体上可以看做是一棵树，称为当前结点的父结点的一个子树



## 结点的度

一个结点含有的子树的个数称为该结点的度；

## 叶结点

度为0的结点称为叶结点，也可以叫做终端结点

## 分支结点

度不为0的结点称为分支结点，也可以叫做非终端结点

## 结点的层次

从根结点开始，根结点的层次为1，根的直接后继层次为2，以此类推

## 结点的层序编号

将树中的结点，按照从上层到下层，同层从左到右的次序排成一个线性序列，把他们编成连续的自然数。

## 树的度

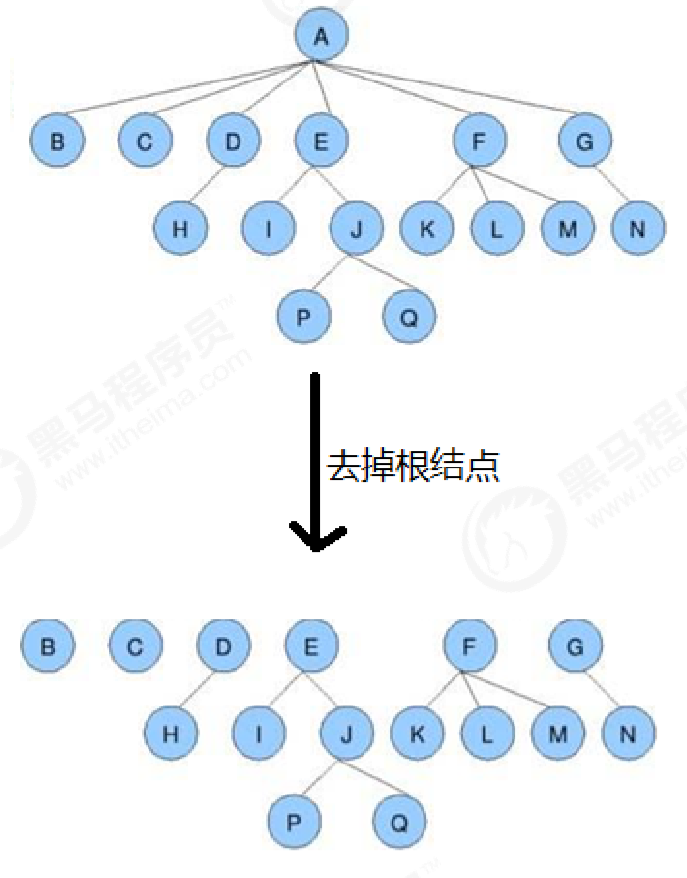
树中所有结点的度的大值

## 树的高度(深度)

树中结点的大层次

## 森林

m（m>=0）个互不相交的树的集合，将一颗非空树的根结点删去，树就变成一个森林；给森林增加一个统一的根 结点，森林就变成一棵树



## 孩子结点

一个结点的直接后继结点称为该结点的孩子结点

## 双亲结点(父结点)

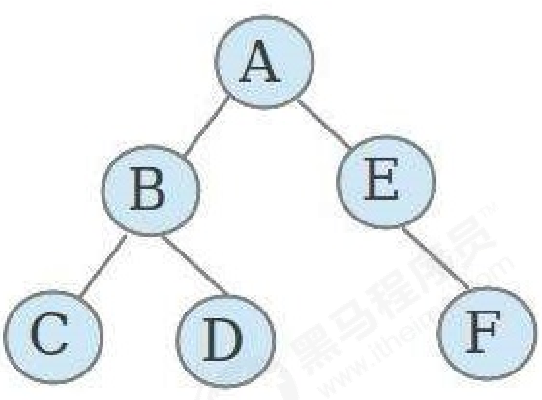
一个结点的直接前驱称为该结点的双亲结点

## 兄弟结点

同一双亲结点的孩子结点间互称兄弟结点

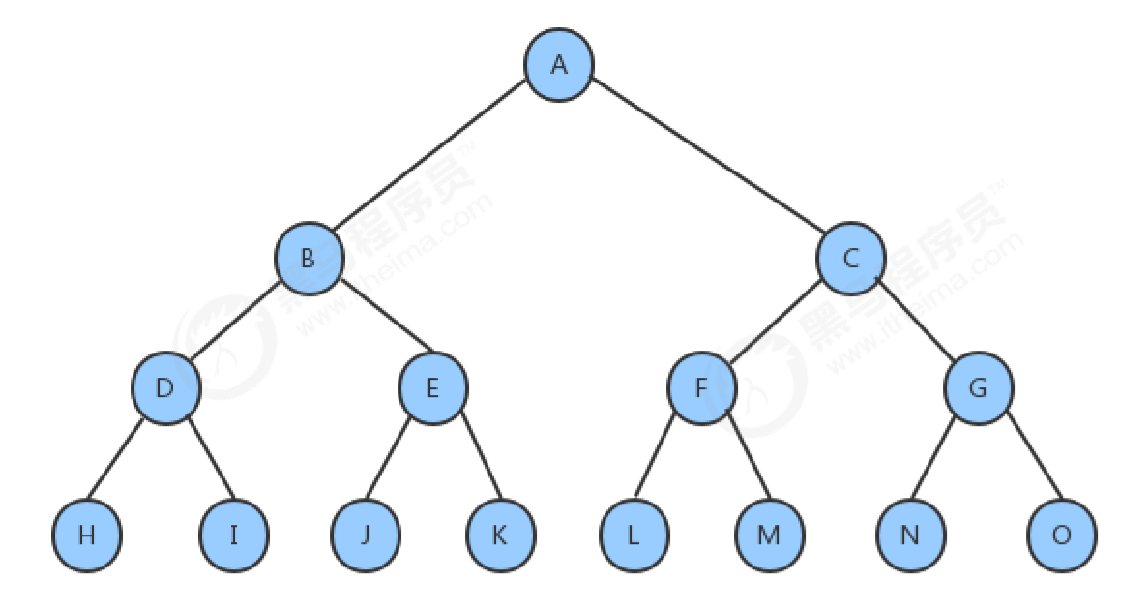
## 二叉树

二叉树就是度不超过2的树(每个结点多有两个子结点)



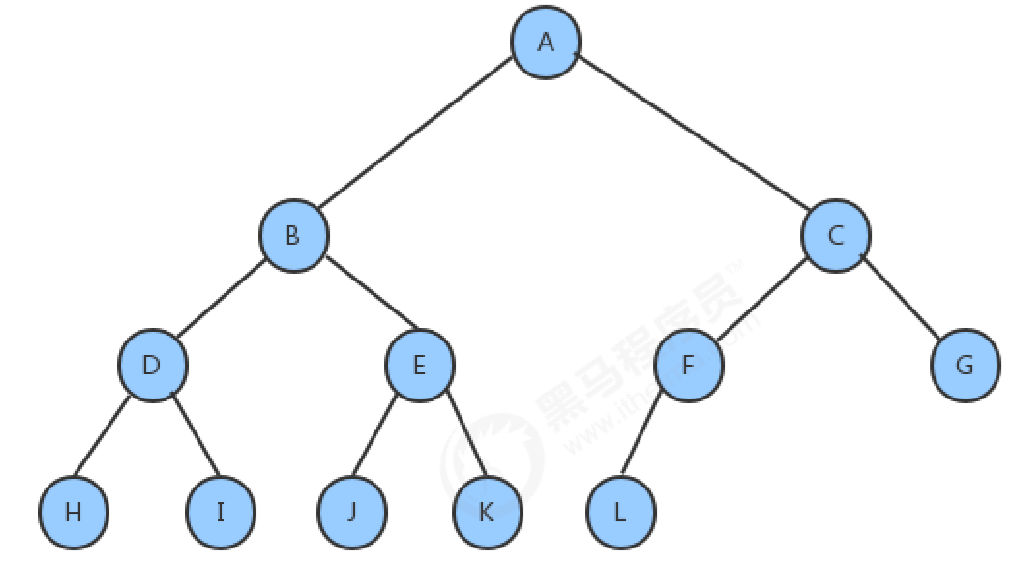
## 满二叉树

一个二叉树，如果每一个层的结点树都达到大值，则这个二叉树就是满二叉树。



## 完全二叉树

叶节点只能出现在下层和次下层，并且下面一层的结点都集中在该层左边的若干位置的二叉树



# 堆

# 优先队列

# 并查集

# 图