# 算法

1.算法的五个特性

输入：以待解决的问题的信息作为输入

输出：输入对应指令集处理后得到的信息

可行性：算法是可行的，即算法中的每一条指令都是可以实现的，均能在有限的时 间内完成。

有穷性：算法执行的指令个数是有限的

确定性：算法对于特定的合法输入，其对应的输出是唯一

2.评价算法性能的好坏：**时间复杂度**(运行时间)**、空间复杂度**(占用的存储空间)

**算法=控制结构+原操作(固有数据类型的操作)**

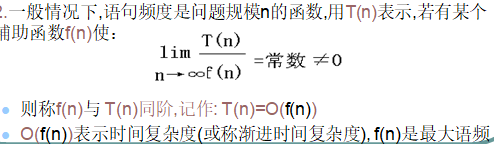
**算法的执行时间=原操作(i)的执行次数×原操作(i)的执行时间**。算法的执行时间与原操作执行次数之和成正比

算法和程序是相互独立的。

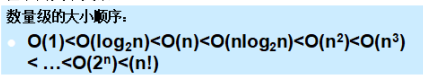
3.时间复杂度

通常利用语句频度和渐进时间复杂度衡量时间复杂度

语句频度：该语句被执行的次数



执行一些简单的输入,输出或赋值语句时,可近似认为需要时间。



## 一、递归

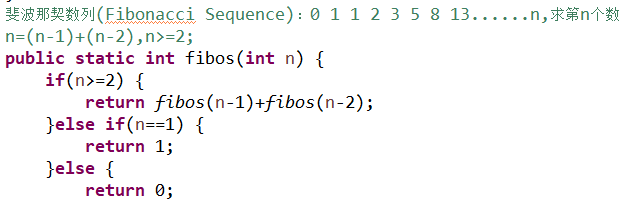
递归：在定义自身的同时又出现了对自身的引用。

如果一个算法直接或间接地调用自己，则称这个算法是一个递归算法。

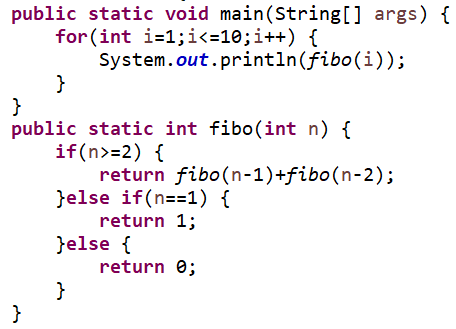
递归的两个组成部分：自身调用和结束条件

在使用递归时必须注意结束条件，否则特别容易出现栈溢出。

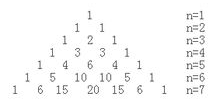
**斐波那契数列**：(递归方法)



使用递归打印出前10个斐波那契数列：

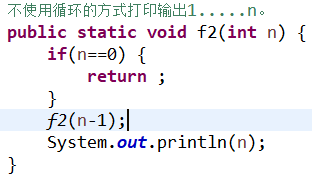


杨辉三角：



汉诺塔：

实例：理解下面的代码，注意方法的调用是栈类型的。



## 二、排序

内排序：若待排序记录都在内存中，称为内部排序

外排序：若待排序记录一部分在内存，一部分在外存，则称为外部排序

排序的时间开销：是衡量算法好坏的最重要的标志。通常用算法执行中的**数据比较次数**和**数据移动次数**来衡量

**排序**：冒泡排序、选择排序、插入排序、快速排序。

**选择排序和插入排序的代码必须会手写，这四种排序方法都必须会描述出来**。

### 1.冒泡排序(bubble sort)

**基本原理**：对要排序的原始数组进行排序，若发现相邻数据的大小次序与要求排序的大小次序不一致时，则将这两个数据的次序交换一下。较小的数据就会逐个向前移动，好象气泡向上浮起一样。

冒泡排序是一种稳定的排序。

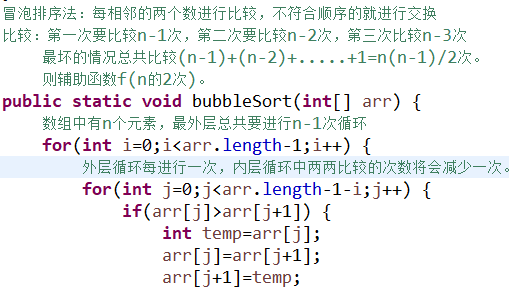
**冒泡法升序排序**：数组中元素的个数为n个

第一次：对数组中的元素从前往后，每每比较相邻两个元素之间的大小，若这两个元素的大小次序是降序的状态，则将这两个数据进行交换。执行完第一次之后，数组的最后面的元素是最大的。两两比较的次数为n次。

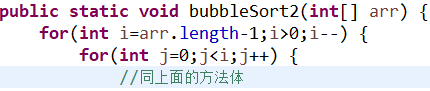
第二次：对数组中的元素重复第一次操作。最后一个元素不再需要比较，两两比较的次数为n-1次。

每重复一次上次的操作，两两比较的次数就-1。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。

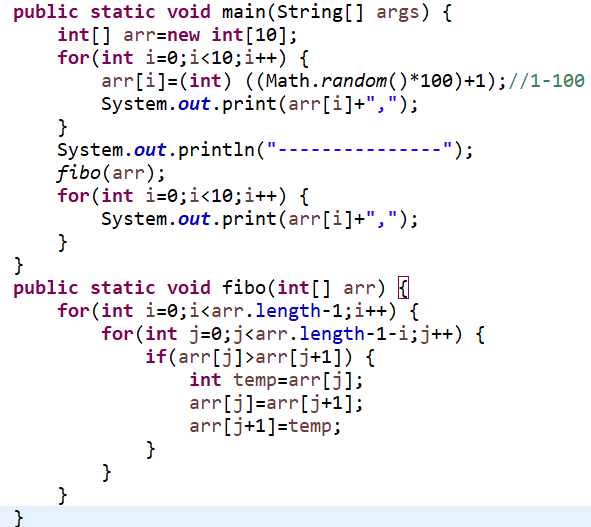
**法一**



**法二**



**完整例子**：



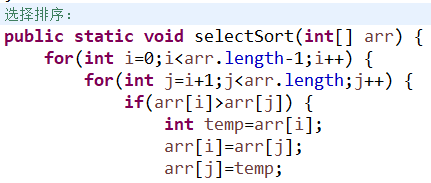


### 2.选择排序

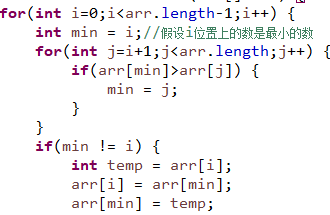
**基本原理**：数组中有n个元素，首先从数组中的第(1～n)个元素中找出最大或最小的元素，然后将该元素放到数组的第一个元素。然后再从数组中的第(2～n)个元素中找出最大或最小的元素，然后放到数组的第二个位置。以此类推，直到所有元素均排序完毕。

简单选择排序是一种**不稳定**的排序。

自己写的选择排序：



下面是老师的选择排序



算法稳定：对于待排序列中相同项的原来次序不能被算法改变则称该算法稳定。排序算法是否为稳定的是由具体算法决定的，不稳定的算法在某种条件下可以变为稳定的算法，而稳定的算法在某种条件下也可以变为不稳定的算法

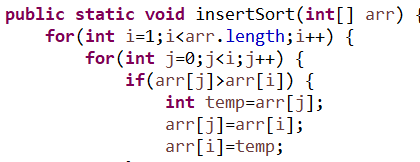
### 3.插入排序

直接插入排序是一种**稳定**的排序方式。

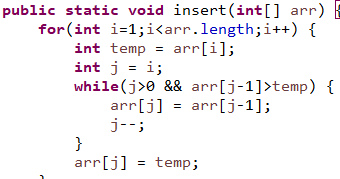
**基本原理**：对于一个待排序的序列，假设前面的序列是有序的，依次将待排序的序列中的每个元素按大小插入到有序序列的适当位置，直到全部元素插入完成。

有空自己写一个用链表实现插入排序

自己写的插入排序：



老师的插入排序：(没明白)



### 4.快速排序

快速排序是一种**不稳定**的排序方式。

**基本原理**：一般选取数组的第一个元素作为枢纽，然后将数组分成两部分，比枢纽小的元素放在数组的左半部分，比枢纽大的元素放在数组的右半部分。然后可以采用递归的方法，再次按照同样的思路将数组分割排序。

快速排序(下面代码没弄明白，自己单独写不出)



