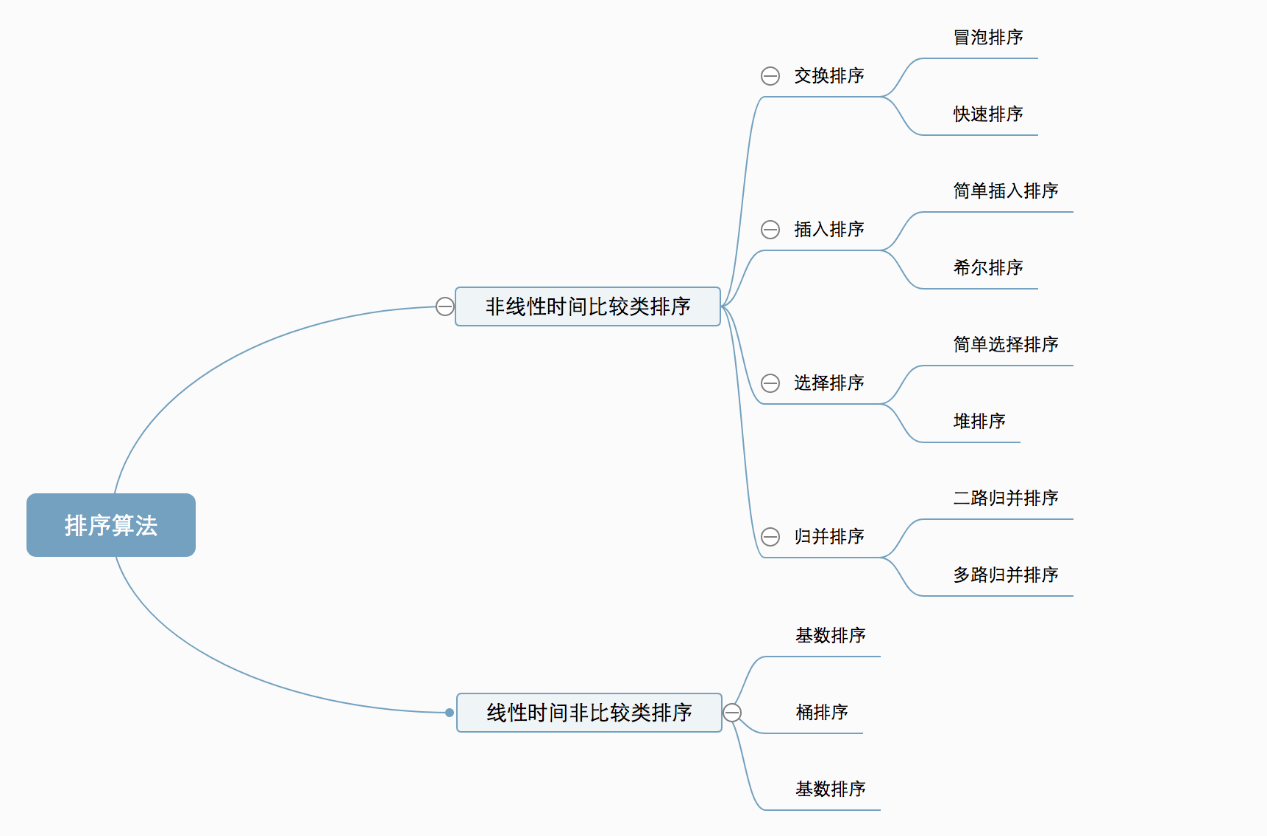
**十种常见排序算法可以分为两大类：**

**非线性时间比较类排序**：通过比较来决定元素间的相对次序，由于其时间复杂度不能突破O(nlogn)，因此称为非线性时间比较类排序。

**线性时间非比较类排序**：不通过比较来决定元素间的相对次序，它可以突破基于比较排序的时间下界，以线性时间运行，因此称为线性时间非比较类排序。



**各排序算法复杂度及稳定性：**



**相关概念**

**稳定**：如果a原本在b前面，而a=b，排序之后a仍然在b的前面。

**不稳定**：如果a原本在b的前面，而a=b，排序之后 a 可能会出现在 b 的后面。

**时间复杂度**：

1.时间复杂度  
一般情况下，算法中基本操作重复执行的次数是问题规模n的某个函数，用T(n)表示，若有某个辅助函数f(n),使得当n趋近于无穷大时，T（n)/f(n)的极限值为不等于零的常数，则称f(n)是T(n)的同数量级函数。记作T(n)=Ｏ(f(n)),称Ｏ(f(n)) 为算法的渐进时间复杂度，简称时间复杂度。  
上面这一段解释是很规范的，但是对于非专业性的我们来说并不是那么好理解，说白了时间复杂度就是时间复杂度的计算并不是计算程序具体运行的时间，而是算法执行语句的次数。通常我们计算时间复杂度都是计算最坏情况 。

最坏时间复杂度和平均时间复杂度  
　最坏情况下的时间复杂度称最坏时间复杂度。一般不特别说明，讨论的时间复杂度均是最坏情况下的时间复杂度。  
　这样做的原因是：最坏情况下的时间复杂度是算法在任何输入实例上运行时间的上界，这就保证了算法的运行时间不会比任何更长。  
　平均时间复杂度是指所有可能的输入实例均以等概率出现的情况下，算法的期望运行时间。设每种情况的出现的概率为pi,平均时间复杂度则为sum(pi\*f(n))

**空间复杂度：**

一个程序的空间复杂度是指运行完一个程序所需内存的大小。利用程序的空间复杂度，可以对程序的运行所需要的内存多少有个预先估计。一个程序执行时除了需要存储空间和存储本身所使用的指令、常数、变量和输入数据外，还需要一些对数据进行操作的工作单元和存储一些为现实计算所需信息的辅助空间。程序执行时所需存储空间包括以下两部分。　　  
（1）固定部分。这部分空间的大小与输入/输出的数据的个数多少、数值无关。主要包括指令空间（即代码空间）、数据空间（常量、简单变量）等所占的空间。这部分属于静态空间。  
（2）可变空间，这部分空间的主要包括动态分配的空间，以及递归栈所需的空间等。这部分的空间大小与算法有关。  
一个算法所需的存储空间用f(n)表示。S(n)=O(f(n))　　其中n为问题的规模，S(n)表示空间复杂度。

**稳定性：**

所谓稳定性是指待排序的序列中有两元素相等,排序之后它们的先后顺序不变.假如为A1,A2.它们的索引分别为1,2.则排序之后A1,A2的索引仍然是1和2.

稳定也可以理解为一切皆在掌握中,元素的位置处在你在控制中.而不稳定算法有时就有点碰运气,随机的成分.当两元素相等时它们的位置在排序后可能仍然相同.但也可能不同.是未可知的.

另外要注意的是:算法思想的本身是独立于编程语言的,所以你写代码去实现算法的时候很多细节可以做不同的处理.采用不稳定算法不管你具体实现时怎么写代码,最终相同元素位置总是不确定的(可能位置没变也可能变了).而稳定排序算法是你在具体实现时如果细节方面处理的好就会是稳定的,但有些细节没处理得到的结果仍然是不稳定的.

比如冒泡排序,直接插入排序,归并排序虽然是稳定排序算法,但如果你实现时细节没处理好得出的结果也是不稳定的.

**稳定性的用处**

我们平时自己在使用排序算法时用的测试数据就是简单的一些数值本身.没有任何关联信息.这在实际应用中一般没太多用处.实际应该中肯定是排序的数值关联到了其他信息,比如数据库中一个表的主键排序,主键是有关联到其他信息.另外比如对英语字母排序,英语字母的数值关联到了字母这个有意义的信息.

可能大部分时候我们不用考虑算法的稳定性.两个元素相等位置是前是后不重要.但有些时候稳定性确实有用处.它体现了程序的健壮性.比如你网站上针对最热门的文章或啥音乐电影之类的进行排名.由于这里排名不会像我们成绩排名会有并列第几名之说.所以出现了元素相等时也会有先后之分.如果添加进新的元素之后又要重新排名了.之前并列名次的最好是依然保持先后顺序才比较好.