1.度量一个程序(算法)执行时间的两种方法

1)事后统计的方法

这种方法可行，但是有两个问题：一是要想对设计的算法的运行性能进行评测，需要实际运行该程序；二是所得时间的统计量依赖于计算机的硬件、软件等环境因素，**这种方式，要在同一台计算机的相同状态下运行，才能比较那个算法速度更快。**

2)事前估算的方法

通过分析某个算法的**时间复杂度**来判断哪个算法更优.

2.时间频度

1.基本介绍

时间频度：一个算法花费的时间与算法中语句的执行次数成正比例，哪个算法中语句执行次数多，它花费时间就多。一个算法中的语句执行次数称为语句频度或时间频度。记为T(n)。

2.算法的时间复杂度

忽略常数

忽略低次项

忽略系数

3.常见时间复杂度

1)常数阶O(1)

无论代码执行了多少行，只要是没有循环等复杂结构，那这个代码的时间复杂度就都是O(1)

2)对数阶O(log2n)

3)线性阶O(n)

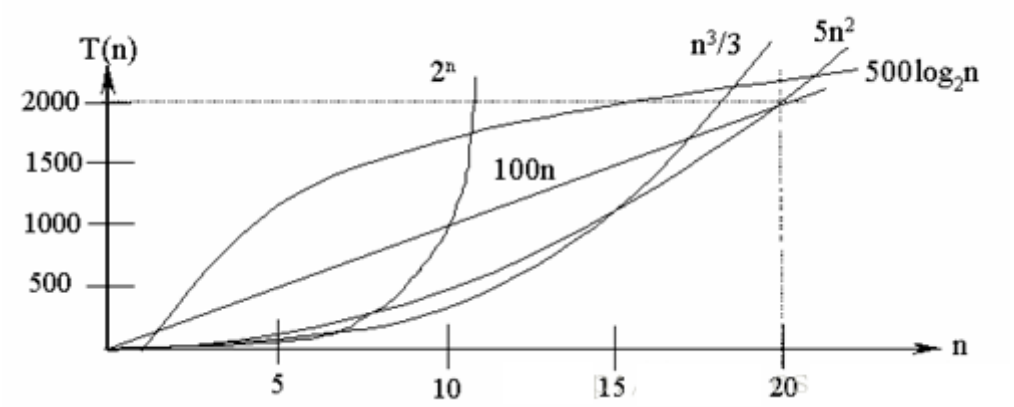
4)线性对数阶O(nlog*2n*)

5)平方阶O(n^2)

6)立方阶O(n^3)

7)k次方阶O(n^k)

8)指数阶O(2^n)



•常见的算法时间复杂度由小到大依次为：O(1)＜O(log*2n*)＜O(n)＜O(nlog*2n*)＜O(*n*^2)＜O(*n*^3)＜ O(*n*^k) ＜O(*2*^n) ，随着问题规模n的不断增大，上述时间复杂度不断增大，算法的执行效率越低

•从图中可见，我们应该尽可能避免使用指数阶的算法

4.平均时间复杂度和最坏时间复杂度

1)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，该算法的运行时间。

2)最坏情况下的时间复杂度称最坏时间复杂度。一般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界限，这就保证了算法的运行时间不会比最坏情况更长。

3)平均时间复杂度和最坏时间复杂度是否一致，和算法有关(如图:)

排序法	平均时间	最差情形	稳定度	额外空间	备注
冒泡	$O(n^2)$	$O(n^2)$	稳定	O(1)	n小时较好
交换	$O(n^2)$	$O(n^2)$	不稳定	O(1)	n小时较好
选择	$O(n^2)$	$O(n^2)$	不稳定	O(1)	n小时较好
插入	$O(n^2)$	$O(n^2)$	稳定	O(1)	大部分已排序时较好
基数	$O(\log_R B)$	$O(\log_R B)$	稳定	O(n)	B是真数(0-9), R是基数(个十百)
Shell	O(nlogn)	$O(n^s)$ $1<s<2$	不稳定	O(1)	s是所选分组
快速	O(nlogn)	$O(n^2)$	不稳定	O(nlogn)	n大时较好
归并	O(nlogn)	O(nlogn)	稳定	O(1)	n大时较好
堆	O(nlogn)	O(nlogn)	不稳定	O(1)	n大时较好

3.算法的空间复杂度简介

1)类似于时间复杂度的讨论，一个算法的空间复杂度(Space Complexity)定义为该算法所耗费的存储空间，它也是问题规模n的函数。

2)空间复杂度(Space Complexity)是对一个算法在运行过程中临时占用存储空间大小的量度。有的算法需要占用的临时工作单元数与解决问题的规模n有关，它随着n的增大而增大，当n较大时，将占用较多的存储单元，例如快速排序和归并排序算法就属于这种情况

3)在做算法分析时，主要讨论的是时间复杂度。从用户使用体验上看，更看重的程序执行的速度。一些缓存产品(redis, memcache)和算法(基数排序)本质就是用空间换时间.

算法很多情况下是空间换时间