时间复杂度与空间复杂度（算法效率）

分析估算方法：在程序编写前,依据统计方法对算法进行估算

耗时取决因素：

```
1.算法采用的策略与方案
2.编译产生的代码质量（编译器优略与编译器采用优化措施的优略）
3.问题的输入规模
4.机器执行指令的速度
总结：抛开与计算机硬件，软件的因素，一个程序的运行时间依赖于算法的好坏和问题的输入规模。
```

因此分析一个算法的运行时间是，重要的是把基本操作的数量和输入模式关联起来

判断一个算法的效率时，函数中的常熟和其他次要项常常可以忽略，而更应关注主项（最高项）的阶数

测试数据时数据越多判断越精确

例如：

```
1.循环时间长度为n，嵌套循环为n^x;
2.单个公式时间长的为1;
```

算法时间复杂度

定义：

再进行算法分析时，语句总的执行次数T(n)是关于问题规模n的函数，而分析T(n)随n的变化情况并确定T(n)的数量级。算法的时间复杂度，也就是算法的时间量度，记作：T(n) = O(f(n))。他表示随问题规模n的增大，算法执行时间的增长率和f(n)的增长率相同，称作算法的渐近时间复杂度，简称为时间复杂度。其中f(n)是问题规模n的某个函数。

关键需要知道执行次数等于时间

记法：

这样用大写O()来体现算法时间复杂度的记法，称之为大O记法。

一般情况下，随着输入规模n的增大，T(n)增长最慢的算法为最优算法。

推到大O阶的方法

```
1.用常数1取代运行时间中的所有加法常数。
2.在修改后的运行次数函数中只保留最高项。
3.得到的最后结果就是大O阶。
```

常数阶

无非语句多少，只与n有关，所有加法常数记作O(1)即可。例：10

线性阶

一般含有非嵌套循环涉及线性阶，线性阶就是随着问题规模n的扩大，对计算次数呈直线增长，记作O(n)。例：3n+4

平方阶

含有嵌套循环，记作O(n^x)。例：3n^2+4n+5

总结：时间复杂度等于循环体复杂度乘以该循环体的运行次数

对数阶

记作O(logn)。例：3log(2)n+14

nlogn阶

记作O(nlogn)。例：2n+3nlog(2)n+14

立方阶

记作O(n^3)。例：n^3+2n^2+4n+6

指数阶

记作O(2^n)。例：2^n

大小顺序

O(n^n) > O(n!) > O(2^n) > O(n^3) > O(n^2) > O(nlogn) > O(n) > O(logn) > O(1)

最坏情况与平均情况

查找一个有n个随机数字数组中的某个数字，最好的情况是第一个数字就是，那么该算法的时间复杂度就是O(1)，但也有可能这个数字就在最后一个位置，那么时间复杂度为O(n)。

平均运行时间就是期望时间。

最坏运行时间是一种保证。在应用中，这是一种最重要的需求，通常除非特别指定，我们提到的运行时间都是最坏情况的运行时间。

算法空间复杂度 （写代码时可以用空间换取时间）

算法的空间复杂度计算通过计算算法所需的存储空间实现，算法的空间复杂度的计算公式记作：S(n) = O(f(n))，其中，n为问题的规模，f(n)为语句关于n所占存储空间的函数

注意

1. 通常我们都是用时间复杂度来指运行时间的需求，是用空间复杂度指空间需求。
2. 当直接要求让我们求复杂度时，通常指时间复杂度。