## 1.1.3

计时功能：

Clock\_t start,stop;

Clock():捕捉程序开始运行到clock()被调用时所耗费的时间，这个时间单位是clock tick，即时钟打点；

常数CLK\_TCK：机器时钟每秒所走德时钟打点数。

有的时候如果时间太少的话，跑一次的时间不一定能够被clock()函数所检测到，因此，可以考虑使用循环来解决这个问题；比如让一个函数重复运行一万次，这算是一个普遍的测试方法

多项式求乘法：

1. 系数乘以x^n
2. 分解成x\*pn+….

空间复杂度S(n)

表示程序占用的机器的内存空间，通常用来和变量进行比较！

时间复杂度T(n)

表示程序占用的机器所需要消耗的时间。在计算时间复杂度的时候，计算机计算加减法的速度比乘除法要块很多，因此我们**基本上只计算机器计算乘除法的时间，加减法基本不算**。

时间复杂度有平均和最差，**我们一般关心的是最差的时间复杂度**！

讨论1.5 分析“二分法”

老师参与

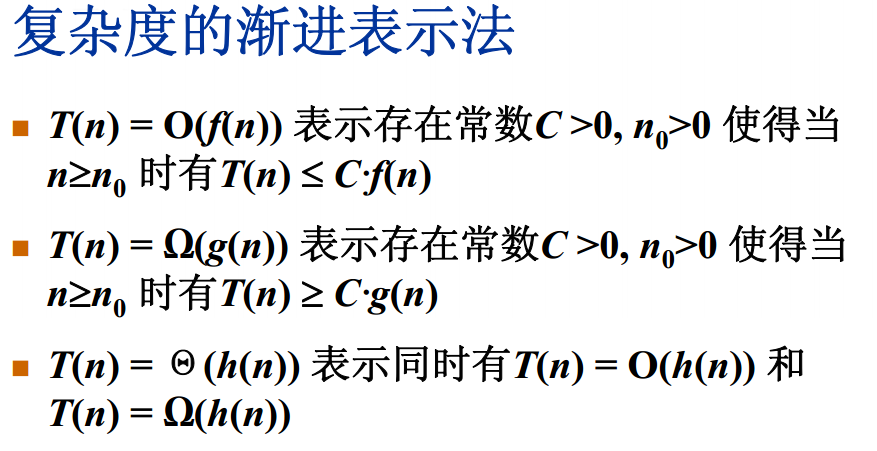
    查找算法中的“二分法”是这样定义的：

    给定N个**从小到大排好序的整数序列List[]**，以及某待查找整数X，我们的目标是找到X在List中的下标。即若有List[i]=X，则返回i；否则返回-1表示没有找到。

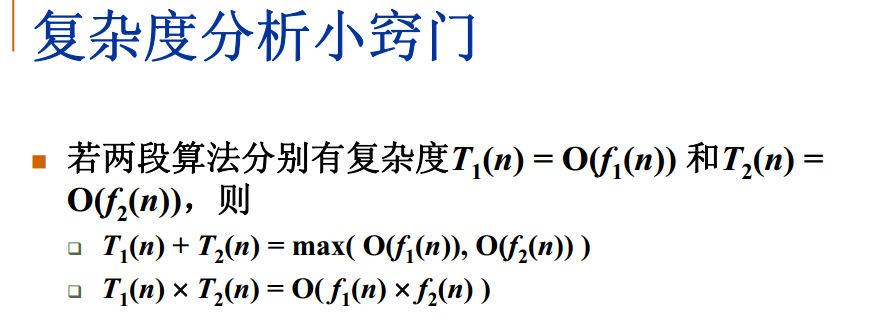
    二分法是先找到序列的中点List[M]，与X进行比较，若相等则返回中点下标；否则，若List[M]>X，则在左边的子系列中查找X；若List[M]<X，则在右边的子系列中查找X。

    试写出算法的伪码描述，并分析最坏、最好情况下的时间、空间复杂度。

### 复杂度的渐进表示法



复杂度比较：logn<n<**nlogn<n^2**<2^n



如果有两段代码拼凑在一起的时候，那么整个代码的复杂度就是两段的和，取更大的那一段，即T1+T2=max(O(f1(n), O(f2(n))。如果两端代码是嵌套在一起的话，那么代码的复杂度就是两段的乘积,即T1\*T2=O（f1(n)\*f2(n)）。

因此对于for循环基本上是乘积

If\_else的话，基本上是3个部分的复杂度，首先是判断if部分的判断的复杂度，这个是肯定有的，然后进入的分支分别进行判断复杂度，然后取分支里面最差的那个就是最差的复杂度了。

## 1.3 应用例子最大子列和

作业题：

1. 自己实现了4种求最大子列和的方法，一种是全部遍历，一种是部分遍历，一种是分治，还有一种是在线扫描。其中比较好理解的是部分遍历和在线扫描。
2. 实现了部分遍历和在线扫描的记录开头和末尾元素的功能。其中有一些技巧和细节。比如说1、要求全部为负数的时候输出开头和结尾的元素，还有最大子列和记为0 2、如果是间隔的话，那么最大子列和记为0，并且记录下0的位置

这个时候，首先应该实现基本的正常的框架，也就是先实现基本的记录开头和结尾的功能，这个时候发现右端可以随时随着求和来进行更新，左端的话会有一定的延迟，因此我们需要多设置一个变量。然后对于特定的情况，我们可以进行特殊的修改初始值来解决，也可以在外面添加一个循环，判断所有元素都为负数（由于只是循环了N次，所以复杂度其实还算OK）