时间复杂度为一个算法流程中，常数操作数量的指标。常用O（读作big O）来表示。具体来说，在常数操作数量的表达式中，只要高阶项，不要低阶项，也不要高阶项的系数，剩下的部分如果记为f(N)，那么时间复杂度为O(f(N))。

**例一：一个简单的理解时间复杂度的例子**

一个有序数组A，另一个无序数组B，请打印B中的所有不在A中的数，A数

组长度为N，B数组长度为M。

算法流程1：对于数组B中的每一个数，都在A中通过遍历的方式找一下；

算法流程2：对于数组B中的每一个数，都在A中通过二分的方式找一下；

算法流程3：先把数组B排序，然后用类似外排的方式打印所有在A中出现

的数；

三个流程，三种时间复杂度的表达...如何分析好坏？

* 算法流程1：所需要的时间复杂度为O(M\*N)
* 算法流程2：二分的方式：首先将数组遍历，然后取中间数跟要取的数比较，依次进行。需要的次数为log2N。时间复杂度为O(log2N)，总共时间复杂度为O(M\*log2N)。
* 算法流程3：外排将两个有序数组依次比较，谁小动哪个数组，直到两数组相等打印。排序时间复杂度为O(N\*log2N)，第二步比较时间复杂度O(N+M),总的时间复杂度为：O(N\*log2N)+O(N+M)根据样本量来化简。

比较：算法流程1直接排除，算法流程2和算法流程3确定N、M数据量来比较。

**注**：评价一个算法流程的好坏，先看时间复杂度的指标，再分析不同数据样本下的实际运行时间，也就是常数项时间。

**例二：对数器的概念和使用**

0、有一个你想要测的方法a，

1、实现一个绝对正确但是复杂度不好的方法b，

2、实现一个随机样本产生器------样本的长度尽量小一点，出现错误之后方便查看

3、实现比对算法a和b的方法

4、把方法a和方法b比对很多次来验证方法a是否正确。

5、如果有一个样本使得比对出错，打印样本分析是哪个方法出

错

6、当样本数量很多时比对测试依然正确，可以基本判断a正确。

**例三：冒泡排序细节的讲解与复杂度分析**

原理：比较两个相邻的元素，将值大的元素交换至右端。

时间复杂度O(N^2)，额外空间复杂度O(1)

**例四：选择排序的细节讲解与复杂度分析**

原理：每一趟从待排序的记录中选出最小的元素，顺序放在已排好序的序列最后，直到全部记录排序完毕。也就是：每一趟在n-i+1(i=1，2，…n-1)个记录中选取关键字最小的记录作为有序序列中第i个记录。基于此思想的算法主要有简单选择排序、树型选择排序和堆排序。（这里只介绍常用的简单选择排序）

简单选择排序的基本思想：给定数组：int[]arr={里面n个数据}；第1趟排序，在待排序数据arr[1]~arr[n]中选出最小的数据，将它与arrr[1]交换；第2趟，在待排序数据arr[2]~arr[n]中选出最小的数据，将它与r[2]交换；以此类推，第i趟在待排序数据arr[i]~arr[n]中选出最小的数据，将它与r[i]交换，直到全部排序完成。

时间复杂度O(N^2)，额外空间复杂度O(1)

**例五：插入排序的细节讲解与复杂度分析**

所谓插入排序法，就是检查第i个数字，如果在它的左边的数字比它大，进行交换，这个动作一直继续下去，直到这个数字的左边数字比它还要小，就可以停止了。插入排序法主要的回圈有两个变数：i和j，每一次执行这个回圈，就会将第i个数字放到左边恰当的位置去。

如果数组有序，时间复杂度为O(N)；

如果数组无需，时间复杂度O(N^2)，额外空间复杂度O(1)。

按数据状况最差来估算时间复杂度。O(N^2)

**例六：剖析递归行为和递归行为时间复杂度的估算**

递归的系统栈解释：

1 2 3 4 5 6

1 3 6 10 15 21

int f(x)

{

int b;

if(x==1) return 1;

b=f(x-1);

return x+b;

}

系统栈：求f(4)

x=1,return 1

——————————

x=2,b=f(1),return b+x

——————————

x=3,b=f(2),return b+x

——————————

x=4,b=f(3),return b+x

——————————

从上往下弹栈，得到结果f(4)=4+6.

master公式的使用

master公式（也称主方法）是用来利用分治策略来解决问题经常使用的时间复杂度的分析方法，（补充：分治策略的递归解法还有两个常用的方法叫做代入法和递归树法）。

T(N)=a\*T(N/b)+O(N^d)

N为行为的总样本量，N/b为子行为的样本量，a为子行为发生的次数，O(N^d)表示分解和合并所要花费的时间之和。

解法：  
①当d<logb a时，时间复杂度为O(N^(logb a))  
②当d=logb a时，时间复杂度为O((N^d)\*logN)  
③当d>logb a时，时间复杂度为O(N^d)

**例七：归并排序的细节讲解与复杂度分析**

归并排序的基本思想：

归并排序法是将两个或以上的有序表合并成一个新的有序表，即把待排序序列分成若干个子序列，每个子序列是有序的。然后再把有序子序列合并为整体有序序列。

注意：一定要是有序序列！

T(N)=2\*T(N/2)+O(N)

a=2,b=2,d=1 -> d=logb a -> O((N^d)\*logN)

时间复杂度O(N\*logN)，额外空间复杂度O(N)

**例八：小和问题和逆序对问题**

（1）小和问题

在一个数组中，每一个数左边比当前数小的数累加起来，叫做这个数组的小和。求一个数组的小和。

例子：

[1,3,4,2,5]

1左边比1小的数，没有；

3左边比3小的数，1；

4左边比4小的数，1、3；

2左边比2小的数，1；

5左边比5小的数，1、3、4、2；

所以小和为1+1+3+1+1+3+4+2=16

1. 归并方法

思想：从第一个元素p起向后看，如果有n个元素都比p大，说明n个元素的小和中都有p，因此，第一个元素对小和的贡献是n\*p,遍历完整个数组后，就可以求出全部小和了。1 + 1 + 1 + 1 + 3 + 3 + 4 + 2 = 16

已知归并过程如下：

1、首先划分划分划分，一直划分到不能划分，即每个组都只有一个数值。

2、然后合并，合并的过程就是每个二划分排序的过程。

3、在合并的时候，开辟一个辅助数组，其大小等于这两个合并数列的大小。

4、设置两个指针分别指向每个数列的首部，然后比较得到其中较小的值，并将这个值放入辅助数组中。然后取出小值的那个数列的指针可以继续向前走，与另一个数列的指针所指向的值继续比较。

5、这样比较完成后，如果两个数列中有个数列的数值有剩余，即其指针没有走到末尾，则将这个数列直接赋到辅助数组末尾即可。

6、然后将辅助数组中的值拷贝回原数组中刚才合并的那两个数列的位置上。

1,3,4 2,5

1,3 4 2 5

1 3

合并：1+1+3+2+1+1+3+4

（2）逆序对问题

在一个数组中，左边的数如果比右边的数大，则折两个数构成一个逆序对，请打印所有逆序

对。