“大O记法”： 对于单调的证书函数f，如果存在一个整数g和实常数c>0使得对于充分大的n总有f(n)<=c\*g(n), 就说函数g是f的一个渐进函数（忽略常数），记为f(n)=O(g(n))。也就是说在趋近无穷的极限意义下，函数f的增长速度受到函数g的约束，亦函数f与函数g的特征相似。

时间复杂度： 假设存在函数g，使得算法A处理规模n的问题示例所用时间T(n)=O(g(n))，则称O(g(n))为算法A的渐进时间复杂度（asymptomatic time complexity），简称时间复杂度，记为T（n）。

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

import timeit

class timeit.Timer(stmt='pass', setup='pass', timer=<timer function>)

Timer是测量小段代码执行速度的类

stmt参数是要测试的代码语句（statement）

setup参数是运行代码时需要的设置

timer参数是一个定时器函数，与平台有关

timeit.Timer.timeit(number = 1000000)

Timer类中测试语句执行速度的对象方法

方法返回执行代码的平均耗时，一个float类型的秒数

timer1 = Timer("test()", "from \_\_main\_\_ import t1")

print("append", timer1.timeir(1000))

list = list + [i] 是把list和[i]先生成一个新的列表再传给list

list += [i] 和extend类似，直接在list上操作

少用前者加法

range()： 在python2生成一个列表 srange()生成一个可迭代的对象

在python3返回一个可迭代对象

Python列表与字典操作的时间复杂度

List：

index[] O(1)

index assignment O(1) 赋值

append O(1)

pop() O(1)

pop(i) O(n) 最坏

insert(i,item) O(n)

del O(n)

iteration O(n)

contains(in) O(n)

get slice[x:y] O(k) 先找x位置再找k个

del slice O(n) 删之后要移位

set slice O(n+k) li[0:3] = [2,3,4,5,6,7] 先删O()再补充O(k)

reverse O(n)

concatenate O(k) 第二个list补充k个在第一个后面

sort O(nlogn)

multiply O(nk) k次

Dict:

copy O(n)

get item O(1)

set item O(1)

delete item O(1)

contains(in) O(1)

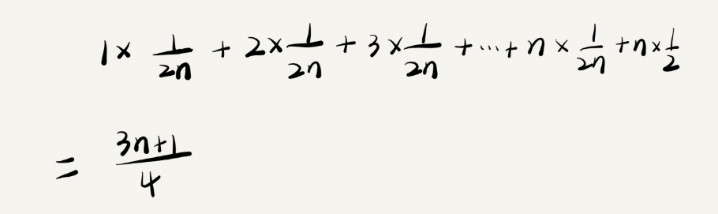
iteration O(n)

抽象数据类型 Abstract Data Type （ADT）

平均情况时间复杂度（average case time complexity）

也叫加权平均时间复杂度，期望时间复杂度

例：要查找的变量 x 在数组中的位置，有 n+1 种情况：在数组的 0～n-1 位置中和不在数组中。我们知道，要查找的变量 x，要么在数组里，要么就不在数组里。这两种情况对应的概率统计起来很麻烦，为了方便你理解，我们假设在数组中与不在数组中的概率都为 1/2。另外，要查找的数据出现在 0～n-1 这 n个位置的概率也是一样的，为 1/n。所以，根据概率乘法法则，要查找的数据出现在 0～n-1 中任意位置的概率就是 1/(2n)。如果我们把每种情况发生的概率也考虑进去，那平均时间复杂度的计算过程就变成了这样：



均摊时间复杂度（amortized time complexity）



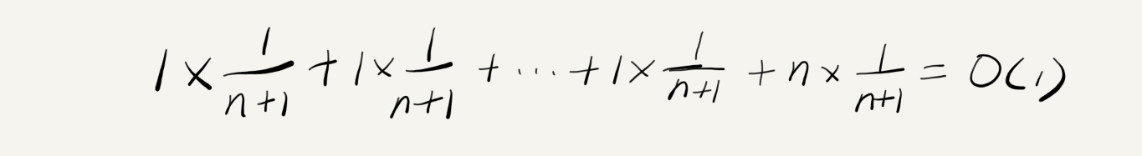
我先来解释一下这段代码。这段代码实现了一个往数组中插入数据的功能。当数组满了之后，也就是代码中的 count == array.length 时，我们用 for 循环遍历数组求和，并清空数组，将求和之后的 sum 值放到数组的第一个位置，然后再将新的数据插入。但如果数组一开始就有空闲空间，则直接将数据插入数组。那这段代码的时间复杂度是多少呢？

你可以先用我们刚讲到的三种时间复杂度的分析方法来分析一下。

最理想的情况下，数组中有空闲空间，我们只需要将数据插入到数组下标为 count 的位置就可以了，所以最好情况时间复杂度为 O(1)。

最坏的情况下，数组中没有空闲空间了，我们需要先做一次数组的遍历求和，然后再将数据插入，所以最坏情况时间复杂度为 O(n)。

那平均时间复杂度是多少呢？答案是 O(1)。我们还是可以通过前面讲的概率论的方法来分析。假设数组的长度是 n，根据数据插入的位置的不同，我们可以分为 n 种情况，每种情况的时间复杂度是 O(1)。除此之外，还有一种“额外”的情况，就是在数组没有空闲空间时插入一个数据，这个时候的时间复杂度是 O(n)。而且，这 n+1 种情况发生的概率一样，都是 1/(n+1)。所以，根据加权平均的计算方法，我们求得的平均时间复杂度就是：



对于均摊时间复杂度，我们还是继续看在数组中插入数据的这个例子。每一次 O(n) 的插入操作，都会跟着 n-1 次 O(1) 的插入操作，所以把耗时多的那次操作均摊到接下来的 n-1 次耗时少的操作上，均摊下来，这一组连续的操作的均摊时间复杂度就是 O(1)。