第9课 时间复杂度与空间复杂度

时间复杂度概念：

一般情况下，算法中基本操作重复执行的次数是问题规模n的某个函数f(n),算法的时间度量记作 T(n)=O(f(n))，它表示随问题规模n的增大，算法执行时间的增长率和f(n)的增长关系，称作算法的渐进时间复杂度(asymptotic time complexity),简称时间复杂度。

时间复杂度概念：

时间复杂度T(n)按数量级递增顺序为：

常数阶	对数阶	线性阶	线性对数阶	平方阶	立方阶	……	K次方阶	指数阶
$O(1)$	$O(\log_2 n)$	$O(n)$	$O(n\log_2 n)$	$O(n^2)$	$O(n^3)$		$O(n^k)$	$O(2^n)$

复杂度低 ---->---->---->---->---->---->---->---->---->---->---->----> 复杂度高

计算时间复杂度：

步骤：

1、找到执行次数最多的语句

2、语句执行语句的数量级

3、用O表示结果

以上是计算时间复杂度的3个出发点。

然后：

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，只保留最高阶项

3、如果最高阶项存在且不是1，那么我们就去除于这个项相乘的常数。比如3n^2我们取n^2

最后就可以得到你们想要的结果了。

例1:

```
print("111")
print("111")
print("111")
print("111")
print("111")
print("111")
print("111")
print("111")
```

打印8条语句，问这个程序的时间复杂度是多少？

O（8）？ 当然不是！！！按照时间复杂度的概念“T（n）是关于问题规模为n的函数”，这里跟问题规模有关系吗？没有关系，用我们的第一个方法，时间复杂度为O（1）。

例2:

```
sum=0
for i in range(101):
    sum+=i
```

线性阶

时间复杂度为O（n）。

例3:

```
sum=0
for i in range(100):
    for j in range(100):
        sum+=j
```

平方阶

外层i的循环执行一次，内层j的循环就要执行100次，所以外层执行100次，那么总的就需要执行100*100次，那么n次呢？就是n的平方次了。所以时间复杂度为：O（n^2）。

例4：

```
sum=0
for i in range(100):
    for j in range(i,100):
        sum+=j
```

平方阶

当i=1的时候执行n次，当i=2的时候执行（n-1）次，......一直这样子下去就可以构造出一个等差数列：n+（n-1）+（n-2）+......+2+1

根据等差数列的求和公式：或者求和易得：n+n*(n-1)/2，整理一下就是n*(n+1)/2，然后我们将其展开可以得到n^2/2+n/2。根据我们的步骤走，保留最高次项，去掉相乘的常数就可以得到时间复杂度为：O（n^2）

例4：

```
i=1
n=100
while i<n:
    i=i*2
```

对数阶

2^x = n,所以时间复杂度为O(log2n)。

时间复杂度小结：

最坏情况与平均情况：
平均运行时间是期望的运行时间。

最坏的运行时间是一种保证。我们提到的运行时间都是最坏的运行时间。

空间复杂度概念：

空间复杂度（Space complexity）：是指算法编写成程序后，在计算机中运行时所需存储空间大小的度量。记作：S(n) = O(f(n))，其中：n为问题的规模或大小。

空间复杂度概念：

空间存储空间一般包括三个方面：

1.输入数据所占用的存储空间；

2.指令常数变量所占用的存储空间；

3.辅助（存储）空间。

一般地，算法的空间复杂度指的是辅助空间。如：

一维数组a[n]：空间复杂度 O(n)

二维数组a[n][m]：空间复杂度 O(n*m)