数据结构与算法（Python版）

大O表示法

陈斌 北京大学 gischen@pku.edu.cn

算法时间度量指标

- **一个算法所实施的操作数量或步骤数可作为独立于具体程序/机器的度量指标**

  哪种操作跟算法的具体实现无关？

  需要一种通用的基本操作来作为运行步骤的计量单位

- **赋值语句是一个合适的选择**

  一条赋值语句同时包含了（表达式）计算和（变量）存储两个基本资源

  仔细观察程序设计语言特性，除了与计算资源无关的定义语句外，主要就是三种控制流语句和赋值语句，而控制流仅仅起了组织语句的作用，并不实施处理。

赋值语句执行次数

❖ 分析SumOfN的赋值语句执行次数

对于“问题规模”n

赋值语句数量T(n)=1+n

那么，什么是问题规模？

```
31  def sumOfN(n):
32      theSum = 0          # 1
33      for i in range(1, n + 1):   # n
34          theSum = theSum + i     # 1
35      return theSum
```

问题规模影响算法执行时间

- 问题规模：影响算法执行时间的主要因素
- 在前n个整数累计求和的算法中，需要累计的**整数个数**合适作为**问题规模**的指标

  前100,000个整数求和对比前1,000个整数求和，算是同一问题的更大规模
- 算法分析的目标是要找出**问题规模**会怎么影响一个算法的**执行时间**

数量级函数 Order of Magnitude

- 基本操作数量函数T(n)的精确值并不是特别重要，重要的是T(n)中起决定性因素的**主导**部分

  用动态的眼光看，就是当问题规模增大的时候，T(n)中的一些部分会盖过其它部分的贡献

- 数量级函数描述了T(n)中随着n增加而增加速度**最快**的主导部分

  称作“大O”表示法，记作O(*f*(*n*))，其中*f*(*n*)表示T(n)中的主导部分

确定运行时间数量级大O的方法

- 例1：T(n)=1+n

  当n增大时，常数1在最终结果中显得越来越无足轻重

  所以可以去掉1，保留n作为主要部分，运行时间数量级就是O(n)

$$O(n)$$

确定运行时间数量级大O的方法

- 例2：$T(n)=5n^2+27n+1005$

  当n很小时，常数1005其决定性作用

  但当n越来越大，n^2项就越来越重要，其它两项对结果的影响则越来越小

  同样，n^2项中的系数5，对于n^2的增长速度来说也影响不大

  所以可以在数量级中去掉27n+1005，以及系数5的部分，确定为$O(n^2)$

$$O(n^2)$$

影响算法运行时间的其它因素

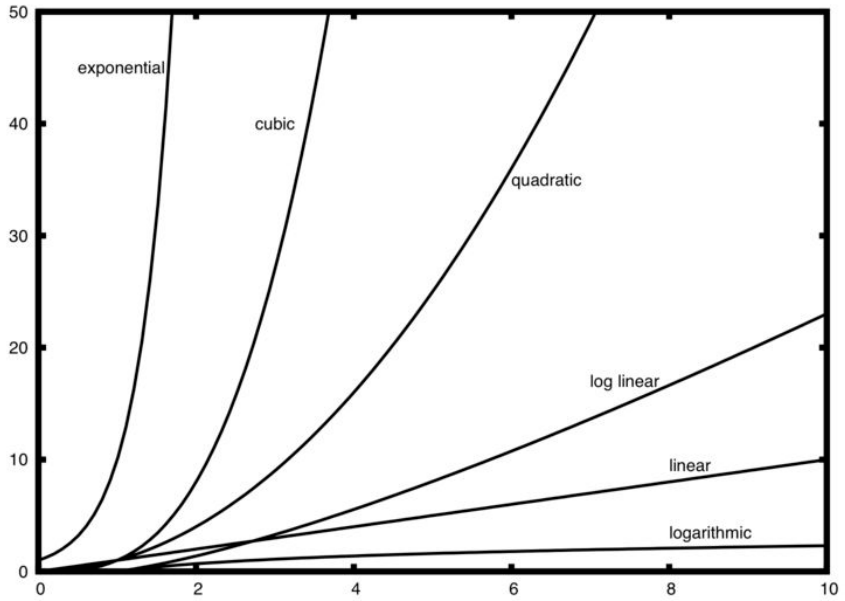
- 有时决定运行时间的不仅是问题规模
- 某些具体数据也会影响算法运行时间

  分为最好、最差和平均情况，平均状况体现了算法的主流性能

  对算法的分析要看主流，而不被某几种特定的运行状况所迷惑

常见的大O数量级函数

- 通常当n较小时，难以确定其数量级
- 当n增长到较大时，容易看出其主要变化量级



$f(n)$	名称
1	常数
$log(n)$	对数
n	线性
$n*log(n)$	对数线性
n^2	平方
n^3	立方
2^n	指数

从代码分析确定执行时间数量级函数

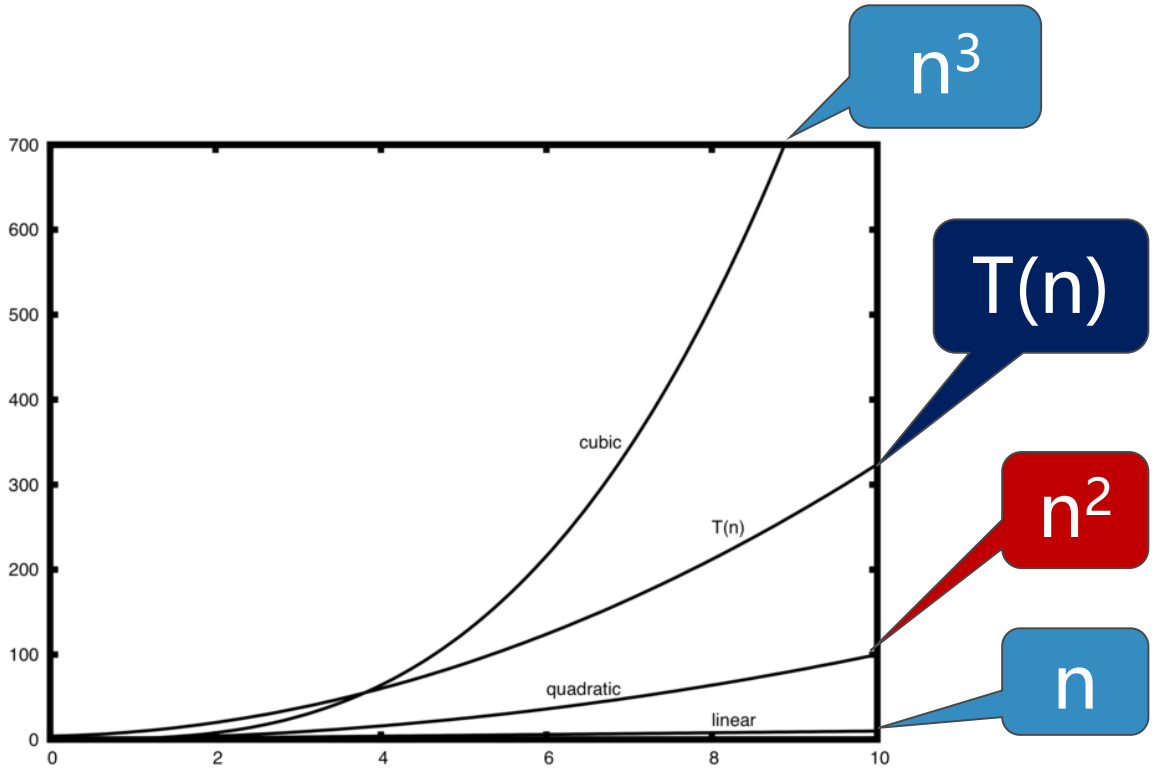
- 代码赋值语句可以分为4个部分

$T(n) = 3+3n^2+2n+1 = 3n^2+2n+4$

```
38      a = 5
39      b = 6
40      c = 10
41      for i in range(n):
42          for j in range(n):
43              x = i * i
44              y = j * j
45              z = i * j
46      for k in range(n):
47          w = a * k + 45
48          v = b * b
49      d = 33
```

从代码分析确定执行时间数量级函数

- 仅保留最高阶项n^2，去掉所有系数
- 数量级为$O(n^2)$



其它算法复杂度表示法

❖ 大O表示法

表示了所有上限中最小的那个上限。

❖ 大Ω表示法

表示了所有下限中最大的那个下限

$$f(n) = \Omega(g(n)) \text{当且仅当} g(n) = o(f(n))$$

❖ 大θ表示法

如果上下限相同，那么就可以用大**θ**表示

$$f(n) = \Theta(g(n))$$

$$\text{当且仅当} f(n) = \mathrm{O}(g(n)) \text{且} f(n) = \Omega(g(n))$$