Python 数据结构与算法分析（第二章 算法分析）

1. 基本概念

- **算法分析**：基于所使用的计算资源比较算法。如甲算法相较于乙算法具有更高的资源利用率或使用更少的资源，则算法甲优于算法乙。
- **计算资源**：①考虑算法在解决问题时要占用的空间或内存；②考虑算法的执行时间或运行时间。
- **问题规模**：如果将每一步看成基本计算单位，那么可以将算法的执行时间描述成解决问题所需的步骤数。例如，对于累加算法，初始化赋值执行步骤数为1，加和运算执行步骤数为n，参数n即为问题规模。则累加算法的运算时间为$T(n) = n + 1$，即当问题规模为n时，解决问题所需的时间是$T(n)$，即需要$1 + n$步。
- **数量级（大O记法）**：数量级函数描述的就是，当n增长时，$T(n)$增长最快的部分。数量级又被称为大O记法，表示为$O(f(n))$，其中$f(n)$为$T(n)$函数中起决定性作用的部分。$O(f(n)$提供了步骤数$T(n)$的有效近似。

  常见的大O函数

f(n)	名称	算法
1	常数	一元运算、二元运算
$logn$	对数	二分法查找
n	线性	累加求和
$nlogn$	对数线性	快速排序
n^2	平方	排序
2^n	指数	动态规划解TSP问题
$n!$	阶乘	暴力枚举

2. Python数据结构的性能

（1）列表

表 2-2　Python 列表操作的大 O 效率

操　　作	大 O 效率
索引	$O(1)$
索引赋值	$O(1)$
追加	$O(1)$
`pop()`	$O(1)$
`pop(i)`	$O(n)$
`insert(i, item)`	$O(n)$
删除	$O(n)$
遍历	$O(n)$
包含	$O(n)$
切片	$O(k)$
删除切片	$O(n)$
设置切片	$O(n+k)$
反转	$O(n)$
连接	$O(k)$
排序	$O(n \log n)$
乘法	$O(nk)$

（2）字典

表 2-3　Python 字典操作的大 O 效率

操　　作	大 O 效率
复制	$O(n)$
取值	$O(1)$
赋值	$O(1)$
删除	$O(1)$
包含	$O(1)$
遍历	$O(n)$

相较于列表，字典的取值和赋值操作均更高效。

3. 参考文献

- Python数据结构与算法分析（第2版）