第零章 算法

1. 算法的概念与历史

1.1算法的概念

算法的概念：
算法，英文名称叫做Algorithm，非常古老的概念，最早来自数学领域。
在数学领域里，算法适用于解决某一类问题的公式和思想。
计算机科学领域内的算法，它的本质是一些列程序指令，用于解决特定的运算和逻辑问题，从宏观上看，数学领域的算法和计算机领域的算法有很多想通之处。
算法既不是剪辑专用术语也不是数学专用，它可以应用到很多不同的领域中，其应用场景更是多种多样的，例如，运算，查找，排序，最优决策。

1.2 算法的历史

算法的概念自古就有，公元前2500年，古巴比伦数学家就用到了除法这样的算术算法，到了9世纪，阿拉伯数学家肯迪则利用基于频率分析的密码学算法破译密码。
“算法”即演算法的大陆中文名称出自《周髀算经》；而英文名称 Algorithm 来自于 9世纪波斯数学家阿尔·花拉子密，因为阿尔·花拉子密在数学上提出了算法这个概念。“算法”原为 " algorism "，意思是阿拉伯数字的运算法则，在18世纪演变为 " algorithm "。欧几里得算法被人们认为是史上第一个算法。

2. 算法的特征

一个合格的算法应该具备以下五个重要的特征：

1. 有穷性：算法的有穷性是指算法必须能在执行有限个步骤之后终止；
2. 确切性：算法的每一步骤必须有确切的定义；
3. 输入项：一个算法有零个或多个输入，以刻画运算对象的初始情况，所谓零个输入是指算法本身指定了初始条件；
4. 输出项：一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的；
5. 可行性：算法中执行的任何计算步骤都是可以被分解为基本的可执行的操作步骤，即每个计算步骤都可以在有限时间内完成（也称之为有效性）。

3. 算法的评定

同一问题可用不同算法解决，而一个算法的质量优劣将影响到算法乃至程序的效率 算法分析的 目的在于选择合适算法和改进算法 一个算法的评价主要从以下几点来考虑：

1. 时间复杂度，是指执行算法所需要的计算工作量
2. 空间复杂度，是指算法需要消耗的内存空间
3. 正确性， 算法的正确性是评价一个算法优的最重要的标准
4. 可读性， 是指一个算法可供人们阅读的容易程度；
5. 容错性， 是指一个算法对不合理输入数据的反应能力和处理能力。

其中最重要的算法衡量标准*是时间和空间*

4. 时间复杂度

一般情况下，集算法程序中基本操作的执行次数， 即时间度量是问题规模n的一个函数 f（n）。

它表示随看问题规模 n的增大， 程序执行时间的增长情况，简称时间复杂度 通常使用大 表示法来表示算法的时间复杂度，记作T=o（f（n））

代码中的基本操作是指最内层循环中的语句。

4.1 常见的复杂度计算

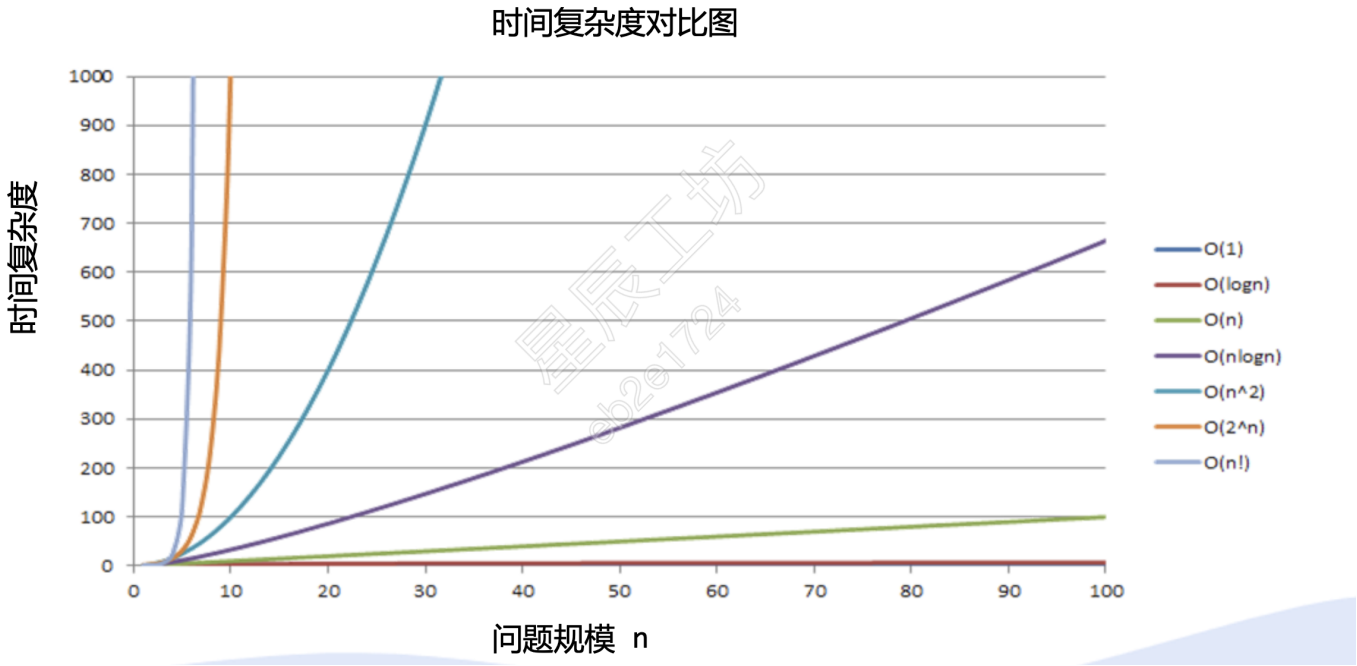
```
//线性时间复杂度O（n）
for(int i = 0; i < n; ++i) {
    // 执行一些常数时间操作
}
//二次时间复杂度（O(n²)）
for(int i = 0; i < n; ++i) {
    for(int j = 0; j < n; ++j) {
        // 执行一些常数时间操作
    }
}
// 对数时间复杂度（O(log n)）
for(int i = 1; i < n; i = i * 2) {
    // 执行一些常数时间操作
}
// 线性对数时间复杂度（O(n log n)）
for(int i = 0; i < n; ++i) {
    for(int j = 1; j < n; j = j * 2) {
        // 执行一些常数时间操作
    }
}
//常数时间复杂度（O(1)）
int i = 0;
i = i + 5;
```

4.2 时间复杂度的比较

n	logn	n	n^2	2^n	n!
10	3.32	10	100	1024	3628800
30	4.91	30	900	1073741824	/
100	6.64	100	10000	/	/
10000	13.29	10000	100000000（亿）	/	/

常见的时间复杂度排序：

O(1) < O(log n) < O(n) < O(n log n) < O(n²) < O(n³) < O(2^n) < O(n!)



实际问题示例： 问题: 查找数组中的第k小元素。

假设我们有一个未排序的数组，我们需要找到该数组中第k小的元素。例如，对于数组 [7, 10, 4, 3, 20, 15] 和 k = 3，第三小的元素是 7。

5. 空间复杂度

空间复杂度是指一个算法在运行过程中占用内存空间大小的度量，记作S=o（f（n））； 它表示随看问题规模 n 的增大， 程序占用内存空间的增长情况

5.1 常见的空间复杂度计算

```
int j = 0;
for(int i =0;i<n;i++)
{
    j++;
}

//由上述程序可知，随着n的变化，程序所需的存储空间并不会发生变化，所以算法的空间复杂度为常量 S = 0（1）


int *a = new int(n);
for (int i = 0; i <n;++i)
{
    a[i] = i;
}

// 上述程序中利用new关键字动态常见了一个大小为n的数组，随着n的增大，开辟的数组空间也会线性增长，即S=0（n）
```

6. 常见的排序算法及其时间复杂度

排序算法	最好情况时间复杂度	平均情况时间复杂度	最坏情况时间复杂度	空间复杂度
冒泡排序	O(n)	$O(n^2)$	$O(n^2)$	O(1)
插入排序	O(n)	$O(n^2)$	$O(n^2)$	O(1)
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
快速排序	O(n log n)	O(n log n)	$O(n^2)$	O(log n)
归并排序	O(n log n)	O(n log n)	O(n log n)	O(n)
堆排序	O(n log n)	O(n log n)	O(n log n)	O(1)
希尔排序	O(n log n)	$O(n(\log n)^2)$	$O(n(\log n)^2)$	O(1)
计数排序	O(n + k)	O(n + k)	O(n + k)	O(k)
桶排序	O(n + k)	O(n + k)	$O(n^2)$	O(n + k)
基数排序	O(nk)	O(nk)	O(nk)	O(n + k)