算法和算法评价

算法的基本概念

算法（Algorithm）是对特定问题求解步骤的一种描述，它是指令的有限序列，其中的每条指令表示一个或多个操作。此外，一个算法还具有下列五个重要特性：

有穷性	一个算法必须总在执行有穷步之后结束，且每一步都可在有穷时间内完成。
确定性	算法中每条指令必须有确切的含义，对于相同的输入只能得出相同的输出。
可行性	算法中描述的操作都可以通过已实现的基本运算执行有限次来实现。
输入	一个算法有零个或多个输入，这些输入取自于某个特定的对象的集合。
输出	一个算法有一个或多个输出，这些输出是与输入有着某种特定关系的量。

通常，设计一个“好”的算法应考虑达到以下目标：

正确性	算法应能够正确地解决求解问题。
可读性	算法应具有良好的可读性，以帮助人们理解。
健壮性	算法能对输入的非法数据做出反应或处理，而不会产生莫名其妙的输出。
高效率与低存储量需求	效率是指算法执行的时间，存储量需求是指算法执行过程中所需要的最大存储空间，这两者都与问题的规模有关。

算法效率的度量

算法效率的度量是通过时间复杂度和空间复杂度来描述的。

时间复杂度

一个语句的频度是指该语句在算法中被重复执行的次数。算法中所有语句的频度之和记为T(n)，它是该算法问题规模n的函数，时间复杂度主要分析T(n)的数量级。算法中基本运算（最深层循环中的语句）的频度与T(n)同数量级，因此通常将算法中基本运算的执行次数的数量级作为该算法的时间复杂度。于是，算法的时间复杂度记为

$$T(n)=O(f(n))$$

式中，O的含义是T(n)的数量级，其严格的数学定义是：

$$\text{若}T(n)\text{和}f(n)\text{是定义在正整数集合上的两个函数，则存在正常数}C\text{和}n_0\\ \text{使得当}n\geq n_0\text{时，都满足}0\leq T(n)\leq Cf(n)$$

算法的时间复杂度不仅依赖于问题的规模n，也取决于待输入数据的性质（如输入数据元素的初始状态）。例如，在数组A[0...n-1]中，查找给定值k的算法大致如下：

```
i=n-1;
while(i>=0&&(A[i]!=k))
    i--;
return i;
```

该算法中语句3（基本运算）的频度不仅与问题规模n有关，而且与下列因素有关：

$$①\text{若}A\text{中没有与}k\text{相等的元素，则语句}3\text{的频度}f(n)=n\\ ②\text{若}A\text{的最后一个元素等于}k\text{，则语句}3\text{的频度}f(n)\text{是常数}0$$

最坏时间复杂度是指在最坏情况下，算法的时间复杂度。

平均时间复杂度是指所有可能输入实例在等概率出现的情况下，算法的期望运行时间。

最好时间复杂度是指在最好情况下，算法的时间复杂度。

一般总是考虑在最坏情况下的时间复杂度，以保证算法的运行时间不会比它更长。

在分析一个程序的时间复杂性时，有以下两条规则：

$$1)\text{加法规则}：T(n)=T_1(n)+T_2(n)=O(f(n))+O(g(n))=O(\max(f(n),g(n)))$$
$$2)\text{乘法规则}：T(n)=T_1(n)\times T_2(n)=O(f(n))\times O(g(n))=O(f(n)\times g(n))$$

例如，设a{},b{},c{}三个语句块的时间复杂度分别为O(1),O(n),O(n^2)，则

```
a{
    b{}
    c{}
}//时间复杂度为O(n^2)，满足加法规则
a{
    b{
        c{}
    }
}//时间复杂度为O(n^3)，满足乘法规则
```

常见的渐进时间复杂度为

$$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)<O(n!)<O(n^n)$$

空间复杂度

算法的空间复杂度定义为该算法所需的存储空间，它是问题规模n的函数，记为

$$S(n)=O(g(n))$$

一个程序在执行时除需要存储空间来存放本身所用的指令、常数、变量和输入数据外，还需要一些对数据进行操作的工作单元和存储一些为实现计算所需信息的辅助空间。若输入数据所占空间只取决于问题本身，和算法无关，则只需分析除输入和程序之外的额外空间。例如，若算法中新建了几个与输入数据规模n相同的辅助数组，则空间复杂度为O(n)。

算法原地工作是指算法所需的辅助空间为常量，即O(1)。

归纳总结

本章的重点是分析程序的时间复杂度。一定要掌握分析时间复杂度的方法和步骤，很多读者在做题时一眼就能看出程序的时间复杂度，但就是无法规范地表述其推导过程。为此，编者查阅众多资料，总结出了此类题型的两种形式，供大家参考。

循环主体中的变量参与循环条件的判断

在用于递推实现的算法中，首先找出基本运算的执行次数x与问题规模n之间的关系式，解得x=f(n)，f(n)的最高次幂为k，则算法的时间复杂度为O(n^k)。

```
//1
int i=1;
while(i<=n)
    i=i*2;
//2
int y=5;
while((y+1)*(y+1)<n)
    y=y+1;
```

在例1中，设基本运算$i=i*2$的执行次数为t，则$2^t\le n$，解得$t\le\log_2 n$，故$T(n)=O(\log_2 n)$
在例2中，设基本运算$y=y+1$得执行次数为t，则$t=y-5$，且$(t+5+1)(t+5+1)<n$，解得$t<\sqrt{n}-6$，即$T(n)=O(\sqrt{n})$

循环主体中的变量与循环条件无关

此类题可采用数学归纳法或直接累计循环次数。多层循环时从内到外分析，忽略单步语句、条件判断语句，只关注主体语句的执行次数。此类问题又可分为递归程序和非递归程序：

- 递归程序一般使用公式进行递推。时间复杂度的分析如下：

  $$T(n) = 1 + T(n-1) = 1 + 1 + T(n-2) = \ldots = n - 1 + T(1)$$

  即T(n)=O(n)。

- 非递归程序的分析比较简单，可以直接累计次数。

zengbaocheng zengbaocheng zengbaocheng zengbaocheng