算法常识

【什么是算法】

算法（Algorithm）是解决特定问题的准确而完整的描述。算法是指令（规则）的有限序列。算法就是一种策略机制，能够对一定规范的输入，在有限时间内获得所要求的输出。

【算法的特征】

有穷性（Finiteness）；确定性（Definiteness）；输入项（Input）；输出项（Output）；可行性（Effectiveness）

【算法的评价标准】

在正确性的前提下，评价算法的最主要因素时时间复杂度和空间复杂度。此外还有可读性和健壮性。

【时间复杂度】

算法的时间复杂度（Time Complexity）是指执行算法所需要的计算工作量，一般来说是问题规模n的函数$f(n)$。

算法的时间复杂度记做

$$T(n) = \mathrm{O}\big(f(n)\big)$$

常见的时间复杂度有：常数阶$O(1)$；对数阶$O(\log_2 n)$；线性阶$O(n)$；线性对数阶$O(n\log_2 n)$；平方阶$O(n^2)$；立方阶$O(n^3)$；……；指数阶$O(2^n)$。

问题的规模n越大，算法执行的时间也就越长。对于相同规模的问题，时间复杂度越高，算法的执行效率越低。

在计算时间复杂度的时候，先找出算法的基本操作，然后确定它的执行次数，最高阶的执行次数就是时间复杂度。

例如下面程序片段的时间复杂度就是$O(n^3)$。

```
for(int i=1; i<=n; i++)
{
    for(int j=1; j<=n; j++)
    {
        c[i][j] = 0; //基本操作，执行次数：n 的平方次
        for(int k=1; k<=n; k++)
        {
            c[i][j] += a[i][k] * b[k][j]; //基本操作，执行次数：n 的三次方次
        }
    }
}
```

【空间复杂度】

算法执行期间所需要的存储空间主要包括三部分：输入数据所占的存储空间、程序本身所占的空间、算法执行过程中需要临时占用的存储空间。

算法的空间复杂度（Space Complexity）是对一个算法在运行过程中临时占用存储空间大小的量度，记作

$$S(n) = O(f(n))$$

【算法复杂度】

算法的时间复杂度和空间复杂度合称为算法的复杂度。对于一个算法，其时间复杂度和空间复杂度往往是相互影响的。当追求一个较好的时间复杂度时，可能会使空间复杂度的性能变差。当追求一个较好的空间复杂度时，可能会使时间复杂度的性能变差。

【算法的基本结构】

一个算法可以用顺序、选择、循环三种基本结构组合而成。

【算法思想】

模拟；枚举；递推；递归；迭代；贪心；分治；试探。

【数据结构】

向量；栈；队列；链表；树；图；堆（优先队列）；森林；集合；映射。

【问题的难度】

P（Deterministic Polynomial，确定多项式）问题是可以在多项式时间内被确定机解决的问题。

NP（Non-Deterministic Polynomial，非确定多项式）问题是可以在多项式时间内被非确定机解决的问题。

所有的 P 问题都是 NP 问题。P 问题是否等于 NP 问题，尚不得而知。

NPC（NP Complete，NP 完全）问题是最有可能不是 P 问题的问题类型。如果 NPC 问题可以在多项式时间内求解，那么 P 问题就等于 NP 问题。

NPH（NP-Hard，NP 难）问题：至少难度是 NP 问题的问题，也就是可能比 NP 问题还难的问题。