线性复杂度 O(n)-表示每个元素要被处理一次

平方复杂度 O(n²)-表示每个元素要被处理n次

**大O推导法：**

1、用常数1取代运行时间中的所有加法常数

2、在修改后的运行函数中，只保留最高阶项

3、如果最高阶项存在且不是1，则去除与这个项相乘的常数

**常数阶：**

int sum = 0 ; n = 100; /\*执行一次\*/

sum = (1+n)\*n/2; /\*执行一次\*/

printf("%d",sum); /\*执行一次\*/

这个算法的运行次数f(n) = 3,根据推导大O阶的方法，第一步是将3改为1，在保留最高阶项是，它没有最高阶项，因此这个算法的时间复杂度为O(1);

另外，

[复制代码](javascript:void(0);)

int sum = 0 ; n = 100; /\*执行一次\*/

sum = (1+n)\*n/2; /\*执行第1次\*/

sum = (1+n)\*n/2; /\*执行第2次\*/

sum = (1+n)\*n/2; /\*执行第3次\*/

sum = (1+n)\*n/2; /\*执行第4次\*/

sum = (1+n)\*n/2; /\*执行第5次\*/

sum = (1+n)\*n/2; /\*执行第6次\*/

sum = (1+n)\*n/2; /\*执行第7次\*/

sum = (1+n)\*n/2; /\*执行第8次\*/

sum = (1+n)\*n/2; /\*执行第9次\*/

sum = (1+n)\*n/2; /\*执行第10次\*/

printf("%d",sum); /\*执行一次\*/

[复制代码](javascript:void(0);)

上面的两段代码中，其实无论n有多少个，本质是是3次和12次的执行差异。这种与问题的大小无关，执行时间恒定的算法，成为具有O(1)的时间复杂度，又叫做常数阶。

注意：不管这个常数是多少，3或12，都不能写成O(3)、O(12)，而都要写成O(1)

此外，对于分支结构而言，无论真假执行的次数都是恒定不变的，不会随着n的变大而发生变化，所以单纯的分支结构（不在循环结构中），其时间复杂度也是O(1)。

**线性阶：**

线性阶的循环结构会复杂一些，要确定某个算法的阶次，需要确定特定语句或某个语句集运行的次数。因此要分析算法的复杂度，关键是要分析循环结构的运行情况。

int i;

for(i = 0 ; i < n ; i++){

/\*时间复杂度为O(1)的程序\*/

}

 O(n)

**对数阶：**

int count = 1;

while(count < n){

count = count \* 2;

/\*时间复杂度为O(1)的程序\*/

}

因为每次count\*2后，距离结束循环更近了。也就是说有多少个2 相乘后大于n，退出循环。

数学公式：2x = n    -->     x = log2n

因此这个循环的时间复杂度为O(logn)

**平方阶：**

int i;

for(i = 0 ; i < n ; i++){

for(j = 0 ; j < n ; j++){

/\*时间复杂度为O(1)的程序\*/

}

}

上面的程序中，对于对于内层循环，它的时间复杂度为O(n)，但是它是包含在外层循环中，再循环n次，因此这段代码的时间复杂度为O(n2)。

int i;

for(i = 0 ; i < n ; i++){

for(j = 0 ; j < m ; j++){

/\*时间复杂度为O(1)的程序\*/

}

但是，如果内层循环改成了m次，时间复杂度就为O(n\*m)

再来看一段程序：

int i;

for(i = 0 ; i < n ; i++){

for(j = i ; j < n ; j++){

/\*时间复杂度为O(1)的程序\*/

}

}

注意：上面的内层循环j = i ;而不是0

因为i = 0时，内层循环执行了n次，当i=1时，执行了n-1次……当i=n-1时，执行了1次，所以总的执行次数为：

n+(n-1)+(n-1)+...+1 =n(n+1)/2  = n2/2 + n/2

根据大O推导方法，保留最高阶项，n2/2 ，然后去掉这个项相乘的常数，1/2

因此，这段代码的时间复杂度为O(n2)

再来看一个比价复杂的语句:

[复制代码](javascript:void(0);)

n++; /\*执行次数为1\*/

function(n); /\*执行次数为n\*/

int i,j;

for(i = 0 ; i < n ; i++){ /\*执行次数为nXn\*/

function(i);

}

for(i = 0 ; i < n ; i++){ /\*执行次数为n(n+1)/2\*/

for(j = i ; j < n ; j++){

/\*时间复杂度为O(1)的程序\*/

}

}

[复制代码](javascript:void(0);)

它的执行次数f(n) = 1 + n + n2+ n(n+1)/2 = 3/2n2+3/2n+1,

根据推导大O阶的方法，最终它的时间复杂度为：O(n2)

**常见的时间复杂度：**

| **执行次数函数** | **阶** | **术语描述** |
| --- | --- | --- |
| 12 | O(1) | 常数阶 |
| 2n+3 | O(n) | 线性阶 |
| 3n2+2n+1 | O(n2) | 平方阶 |
| 5log2n+20 | O(log2n) | 对数阶 |
| 2n+3nlog2n+19 | O(nlogn) | nlog2n阶 |
| 6n3+2n2+3n+4 | O(n3) | 立方阶 |
| 2n | O(2n) | 指数阶 |

时间复杂度所耗费的时间是：

O(1) < O(logn) < O(n) < O(nlogn) < O(n2) < O(n3) <O(2n) < O(n!) <O(nn)