



西安电子科技大学
XIDIAN UNIVERSITY

程序设计竞赛实训基地
Programming Contest Training Base

复杂度理论初步

冯畅

西安电子科技大学计算机科学与技术学院

2021 年 11 月 24 日



计算机的能力是有限的
可计算 \neq 一定时间内能计算出来

处理器

Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50
GHz

1.5GHz 也就是 1s CPU 的时钟周期数，1s 内能执行的指令数和这个是同数量级的
在程序设计竞赛里大多是要在一定时间内、空间内解决问题，空间问题一般不大



这确实是个方法，但可能存在以下情况

1. 不一定能覆盖所有的数据集，得到的不一定是最坏情况需要的时间
2. 难以较准确得出当数据规模增长时需要运行时间的变化
3. 假如实现这个算法需要 500 行甚至更多



西安电子科技大学
XIDIAN UNIVERSITY

程序设计竞赛实训基地
Programming Contest Training Base

最坏时间复杂度



调用 `foo(n)` 后 `cnt` 与 `n` 的关系

```
1 | int cnt = 0;
2 | void F() {
3 |     cnt += 1;
4 | }
5 | void foo(int n) {
6 |     for (int i = 1; i <= n; ++i) {
7 |         F();
8 |     }
9 | }
1 2
```

¹本文代码均由 WangJJ++ 编写，WangJJ++ 语言的语法与 C++ 一样，但语义上所有的变量相当于 C++ 的 `volatile` 变量

²后序代码忽略 `F()` 的定义



- Θ 渐进紧确界** 对于函数 $f(n)$ 和 $g(n)$, $f(n) = \Theta(g(n))$, 当且仅当
 $\exists c_1, c_2, n_0 > 0$, 使得
 $\forall n \geq n_0, 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ 。
- O 渐进上界** $f(n) = O(g(n))$, 当且仅当 $\exists c, n_0$, 使得
 $\forall n \geq n_0, 0 \leq f(n) \leq c \cdot g(n)$ 。
- Ω 渐进下界** $f(n) = \Omega(g(n))$, 当且仅当 $\exists c, n_0$, 使得
 $\forall n \geq n_0, 0 \leq c \cdot g(n) \leq f(n)$ 。



与硬件有关，大多情况分析时可忽略
常数太大， $\log_2 10^6 \approx 20$
题目卡常情况



```
1 void foo(int n) {  
2     for (int i = 1; i <= n; ++i)  
3         for (int j = 1; j <= i; ++j)  
4             F();  
5 }
```




```
1 void foo(int n) {  
2     for (int i = 1; i <= n; ++i)  
3         for (int j = 1; j <= i; ++j)  
4             F();  
5 }
```

$$\begin{aligned} T(n) &= \sum_{i=1}^n \sum_{j=1}^i 1 \\ &= \sum_{i=1}^n i \\ &= \frac{n(n+1)}{2} \\ &= \Theta(n^2) \end{aligned}$$



```
1 void foo(int n) {  
2     for (int i = 1; i <= n; ++i)  
3         for (int j = i; j <= n; j += i)  
4             F();  
5 }
```



```
1 void foo(int n) {  
2     for (int i = 1; i <= n; ++i)  
3         for (int j = i; j <= n; j += i)  
4             F();  
5 }
```

$$\begin{aligned} T(n) &= \sum_{i=1}^n \sum_{i|j}^n 1 \\ &= \sum_{i=1}^n \left\lfloor \frac{n}{i} \right\rfloor \\ &\approx \int_1^n \frac{n}{x} dx \\ &= n \ln(n) \\ &= \Theta(n \log n) \end{aligned}$$



```
1 void foo(int n) {  
2     for (int i = 0; i < n; ++i)  
3         for (int j = 0; j < n; ++j)  
4             for (int j = 0; j < n; ++j)  
5                 for (int k = 0; k < n; ++k)  
6                     F();  
7     for (int i = 0; i < (1 << n); ++i)  
8         for (int j = 0; j < n; ++j)  
9             F();  
10 }
```



```
1 void foo(int n) {  
2     for (int i = 0; i < n; ++i)  
3         for (int j = 0; j < n; ++j)  
4             for (int j = 0; j < n; ++j)  
5                 for (int k = 0; k < n; ++k)  
6                     F();  
7     for (int i = 0; i < (1 << n); ++i)  
8         for (int j = 0; j < n; ++j)  
9             F();  
10 }
```

$$\begin{aligned} T(n) &= n^4 + n2^n \\ &= \Theta(n2^n) \end{aligned}$$



新生赛网络赛最后一题可能会用到，比赛用的极少

$a \geq 1, b > 1$ 是常数， $f(n)$ 是一个函数， $T(n)$ 是定义在非负整数上的递归式：

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

其中我们将 $\frac{n}{b}$ 解释为 $\lfloor \frac{n}{b} \rfloor$ 或 $\lceil \frac{n}{b} \rceil$ 。那么 $T(n)$ 有如下渐进界：

1. 若对于某个常数 $\varepsilon > 0$ 有 $f(n) = O(n^{\log_b a - \varepsilon})$ ，则 $T(n) = \Theta(n^{\log_b a})$.
2. 若 $f(n) = \Theta(n^{\log_b a})$ ，则 $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. 若对于某个常数 $\varepsilon > 0$ 有， $f(n) = \Omega(n^{\log_b a + \varepsilon})$ ，且对于某个常数 $c < 1$ 和足够大的 n 有 $af(\frac{n}{b}) \leq cf(n)$ ，则 $T(n) = \Theta(f(n))$.



- ▶ 算法导论 第三章 函数的增长
- ▶ 具体数学 第九章 渐进式



- ▶ 本题大家可以讨论一下，无查重，但是开赛后的其它题目禁止讨论
- ▶ 不一定有 $F()$ ；，如果有还是一样

输出格式

假如后面有 n 个 T , T_1, T_2, \dots, T_n , 则输出为 n 行, 为 Θ 括号里的表达式的最短 \LaTeX 公式形式, 多个相乘按照对照表的顺序, 可通过 <https://katex.org/> 测试, 如例 1-3 应输出

n^2
 $n \log n$
 n^{2^n}

对照表

$\Theta(n^2)$	n^2
$\Theta(n^{2^n})$	n^{2^n}
$\Theta(n \log n)$	$n \log n$
$\Theta(n^{\frac{8}{7}} 2^n)$	$n^{\frac{8}{7}} 2^n$
$\Theta(n^{\frac{11}{13}} 2^n)$	$n^{\frac{11}{13}} 2^n$



```
1 void foo(int n, int *a, int b) {  
2     int l = 0, r = n;  
3     while (l < r) {  
4         int m = (l + r) / 2;  
5         if (a[m] >= b)  
6             r = m;  
7         else  
8             l = m + 1;  
9     }  
10 }
```



```
1 void foo(int n, int *a) {  
2     bool flag;  
3     do {  
4         flag = false;  
5         for (int i = 1; i < n; ++i) {  
6             if (a[i] > a[i + 1]) {  
7                 flag = true;  
8                 int t = a[i];  
9                 a[i] = a[i + 1];  
10                a[i + 1] = t;  
11            }  
12        }  
13    } while(flag);  
14 }
```



```
1  #define maxn 100011
2  void foo(int n, int *a) {
3      if (n <= 1) return;
4      int m = n / 2;
5      foo(m, a);
6      foo(n - m, a + m);
7      static int b[maxn];
8      for (int i = 0; i < n; ++i) b[i] = a[i];
9      int p = 0, q = m, i = 0;
10     while (p < m && q < n)
11         a[i++] = b[p] < b[q] ? b[p++] : b[q++];
12     while (p < m) a[i++] = b[p++];
13     while (q < n) a[i++] = b[q++];
14 }
```



```
1 void foo(int n) {  
2     for (int i = 0; i < (1 << n); ++i)  
3         for (int j = i; j; j = (j - 1) & i)  
4             F();  
5 }
```



```
1 void foo(int n, int *a) {  
2     for (int i = 0; i < n; ++i) F();  
3     if (n <= 1) return;  
4     int m = n / 2;  
5     foo(m, a);  
6     foo(n - m, a + m);  
7     foo(m, a);  
8     foo(n - m, a + m);  
9 }
```



```
1 void foo(int n) {  
2     for (int i = 1; i <= n; ++i)  
3         for (int j = 1; j * j <= i; ++j)  
4             F();  
5 }
```



西安电子科技大学
XIDIAN UNIVERSITY

程序设计竞赛实训基地
Programming Contest Training Base

GL and HF