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

王孟晞

西安电子科技大学计算机科学与技术学院

2021 年 11 月 24 日

Why
为什么需要复杂度分析?

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

Why

我把算法实现出来运行?

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

最坏时间复杂度

感性理解

调用 foo(n) 后 cnt 与 n 的关系

```
int cnt = 0;
void F() {
    cnt += 1;
}
void foo(int n) {
    for (int i = 1; i <= n; ++i) {
        F();
    }
}
```

[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

渐进符号

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

例 1

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

例 1

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

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

例 2

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

例 2

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

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

例 3

```
void foo(int n) {
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            for (int j = 0; j < n; ++j)
                for (int k = 0; k < n; ++k)
                    F();
    for (int i = 0; i < (1 << n); ++i)
        for (int j = 0; j < n; ++j)
            F();
}
```

例 3

```
void foo(int n) {
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            for (int j = 0; j < n; ++j)
                for (int k = 0; k < n; ++k)
                    F();
    for (int i = 0; i < (1 << n); ++i)
        for (int j = 0; j < n; ++j)
            F();
}
```

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

主定理

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：

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

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 $T(n) = \Theta(n^{\log_b a} lgn)$.
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 $T(n) = \Theta(f(n))$.

拓展阅读

- 算法导论　第三章　函数的增长
- 具体数学　第九章　渐进式

网络赛最后一题

- 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，$T_1, T_2, \ldots, T_n$，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
n2^n
```

对照表

$\Theta(n^2)$	`n^2`
$\Theta(n2^n)$	`n2^n`
$\Theta(n\log n)$	`n\log n`
$\Theta(n^{\frac{8}{7}}2^n)$	`n^\frac872^n`
$\Theta(n^{\frac{11}{13}}2^n)$	`n^\frac{11}{13}2^n`

T1

```
void foo(int n, int *a, int b) {
    int l = 0, r = n;
    while (l < r) {
        int m = (l + r) / 2;
        if (a[m] >= b)
            r = m;
        else
            l = m + 1;
    }
}
```

T2

```
void foo(int n, int *a) {
    bool flag;
    do {
        flag = false;
        for (int i = 1; i < n; ++i) {
            if (a[i] > a[i + 1]) {
                flag = true;
                int t = a[i];
                a[i] = a[i + 1];
                a[i + 1] = t;
            }
        }
    } while(flag);
}
```

T3

```c
#define maxn 100011
void foo(int n, int *a) {
    if (n <= 1) return;
    int m = n / 2;
    foo(m, a);
    foo(n - m, a + m);
    static int b[maxn];
    for (int i = 0; i < n; ++i) b[i] = a[i];
    int p = 0, q = m, i = 0;
    while (p < m && q < n)
        a[i++] = b[p] < b[q] ? b[p++] : b[q++];
    while (p < m) a[i++] = b[p++];
    while (q < n) a[i++] = b[q++];
}
```

T4

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

T5

```
void foo(int n, int *a) {
    for (int i = 0; i < n; ++i) F();
    if (n <= 1) return;
    int m = n / 2;
    foo(m, a);
    foo(n - m, a + m);
    foo(m, a);
    foo(n - m, a + m);
}
```

T6

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

GL and HF