

绪论

渐近复杂度：多项式

$\Theta(1) - O(2)$

Computational problems can be feasibly computed on some computational device only if they can be computed in polynomial time.

- A. Cobham & J. Edmonds

邓俊辉

deng@tsinghua.edu.cn

$O(1)$: constant

❖ 常数: $2 = 2023 = 2023 \times 2023 = O(1)$

//含RAM的所有基本操作, 甚至

$$2023^{2023} = O(1)$$

❖ 从渐近的角度来看, 再大的常数, 也要小于递增的变数

//尽管实际并非如此

❖ [General Twin Prime Conjecture, de Polignac 1849]

For every natural number k , there are infinitely many prime pairs p and q
such that $p - q = 2k$

❖ [Yitang Zhang, April 2013] $k \leq 35,000,000$

❖ [Terence Tao, May 2013] $k \leq 6,500,000$

❖ [Polymath Project, April 2014] $k \leq 123$

$O(1)$: constant

❖ 这类算法的效率最高

//总不能奢望不劳而获吧

❖ 什么样的代码段对应于常数执行时间?

//应具体分析...

❖ 一定不含循环?

```
for ( i = 0; i < n; i += n/2023 + 1 );
```

//2023, 常数

```
for ( i = 1; i < n; i = 1 << i );
```

//log*n, 几乎常数

❖ 一定不含分支转向?

```
if ( (n + m) * (n + m) < 4 * n * m ) goto UNREACHABLE;
```

//不考虑溢出

❖ 一定不能有 (递归) 调用?

```
if ( 2 == (n * n) % 5 ) O1op(n);
```

// $O(1)$ -time Operation

...

$\mathcal{O}(\log^c n)$: poly-log

❖ 对数 $\mathcal{O}(\log n)$: $\ln n$ $\lg n$ $\log_{100} n$ $\log_{2022} n$ // 为何不注明底数?

❖ 常底数无所谓: $\forall a, b > 1, \log_a n = \boxed{\log_a b} \cdot \log_b n = \Theta(\log_b n)$

❖ 常数次幂无所谓: $\forall c > 0, \log n^c = c \cdot \log n = \Theta(\log n)$

❖ 对数多项式: $123 \cdot \log^{321} n + \log^{205} (7 \cdot n^2 - 15 \cdot n + 31) = \Theta(\log^{321} n)$

❖ 这类算法非常有效, 复杂度无限接近于常数: $\forall c > 0, \log n = \mathcal{O}(n^c)$

$\mathcal{O}(n^c)$: polynomial

❖ **多项式**: $a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \dots + a_2 \cdot n^2 + a_1 \cdot n + a_0 = \mathcal{O}(n^k), \quad a_k > 0$

$$100 \cdot n + 2023 = \mathcal{O}(n) \quad \sqrt{23 \cdot n - 472} \times \sqrt{101 \cdot n + 2023} = \mathcal{O}(n)$$

$$(100 \cdot n - 532) \cdot (20 \cdot n^2 - 445 \cdot n + 2023) = \mathcal{O}(n^3) \quad (2023 \cdot n^2 - 129) / (1911 \cdot n - 37) = \mathcal{O}(n)$$

$$\sqrt[3]{2 \cdot n^3 - \sqrt[3]{3 \cdot n^4 - \sqrt{4 \cdot n^5 + \sqrt{5 \cdot n^6 + \sqrt{6 \cdot n^7 + \sqrt{7 \cdot n^8 + \sqrt{8 \cdot n^9 + n^{2023} / \sqrt{n^{14} - 5 \cdot n^9 + 2023}}}}}}}}}} = \mathcal{O}(n^7)$$

❖ **线性 (linear function)** : 所有 $\mathcal{O}(n)$ 类函数

❖ 从 $\mathcal{O}(n)$ 到 $\mathcal{O}(n^2)$: 本课程编程习题主要覆盖的范围

❖ 这类算法的效率通常认为**已可**令人满意, 然而...这个标准是否**太低**了?

//P难度!