

# Độ Phức Tạp Thuật Toán

Nguyễn Văn Huy

[huyite.vn@gmail.com](mailto:huyite.vn@gmail.com)

Ngày 22 tháng 2 năm 2024

- 1 Định nghĩa và ký hiệu
- 2 Các phương pháp phân tích
  - Đếm các phép toán cơ bản

# Giới thiệu

| Input size | Algorithm A     | Algorithm B            |
|------------|-----------------|------------------------|
| $n$        | $5000\ n$       | $1.2^n$                |
| 10         | 50,000          | 6                      |
| 100        | 500,000         | 2,817,975              |
| 1,000      | 5,000,000       | $1.5 \times 10^{79}$   |
| 100,000    | $5 \times 10^8$ | $1.3 \times 10^{7918}$ |

- Phần lớn các bài toán thường có nhiều giải thuật khác nhau để giải một bài toán.
- Lựa chọn giải thuật phù hợp với ngữ cảnh và tài nguyên hệ thống.
- So sánh các giải thuật được giải trên một bài toán.
- Thời gian tính toán là tài nguyên quan trọng nhất.

## Ví dụ

Tính tổng  $S = 1 + 2 + \dots + n$

# Độ phức tạp thuật toán

- ❶ Thời gian chạy trong trường hợp tốt nhất(Best-case). Thời gian chạy ít nhất của thuật toán trên tất cả các tập dữ liệu cùng cỡ.
- ❷ Thời gian chạy trung bình. Là thời gian chạy trung bình cộng trên tất cả các tập dữ liệu.
- ❸ Thời gian chạy trong trường hợp xấu nhất(Worse-case). Thời gian chạy lớn nhất của thuật toán trên cùng tập dữ liệu

# Đánh giá thời gian chạy thuật toán

- ➊ Hầu hết các giải thuật thường có một thông số chính,  $N$ , số mẫu dữ liệu input.
- ➋  $T(n)$  số lượng các phép toán sơ cấp phải thực hiện (phép toán số học, logic, so sánh).
- ➌ Quan tâm tốc độ tăng của hàm  $T(n)$

## Ví dụ

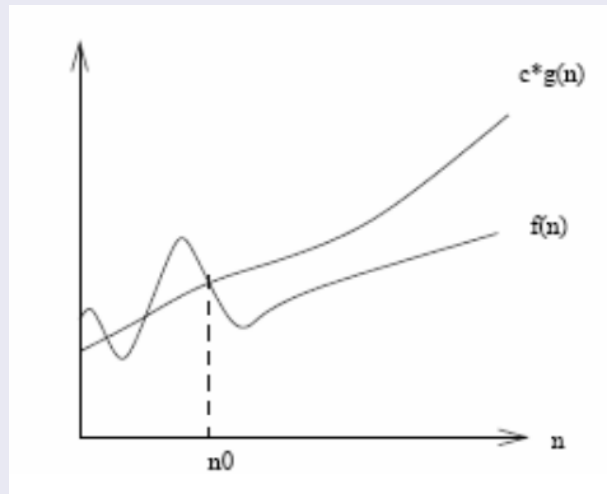
$$T(n) = 3 \times n^4 + 5 \times n$$

# Big-O

Định nghĩa: Giả sử  $f(n)$  và  $g(n)$  là các hàm số thực không âm. Ta nói:

$$f(n) = O(g(n))$$

Nếu tồn tại hằng số  $c$  và  $n_0$  sao cho  $f(n) \leq c \times g(n), \forall n \geq n_0$



## Ví dụ

Let  $T(n) = 2n + 3n^3 + 5$ .  $T(n)$  is in  $O(n^3)$  with:

- ( $c = 8$  and  $n_0 = 1$ ) or ( $c = 5$  and  $n_0 = 2$ )

# Complexity classes - a small vocabulary

- Constant:  $O(1)$  (independent on the input size)
- Sub-linear or logarithmic:  $O(\log n)$
- Linear:  $O(n)$
- Quasi-linear:  $O(n \log n)$
- Quadratic:  $O(n^2)$
- Cubic:  $O(n^3)$
- Polynomial:  $O(n^p)$  ( $O(n^2)$ ,  $O(n^3)$ , etc)
- Quasi-polynomial:  $O(n^{\log(n)})$
- Exponential:  $O(2^n)$
- Factorial:  $O(n!)$



# Example

1. **Var** int:  $d = 0$
2. **For**  $i$  from 1 to  $n$  **do**
  1.  $d = d + 1$
  2.  $a[i] = a[i] \times a[i] + d \times d$
3. **Endfor**

# Example

1. (1) **Var** int:  $d = 0$
2. **For**  $i$  from 1 to  $n$  **do**
  1.  $d = d + 1$
  2.  $a[i] = a[i] \times a[i] + d \times d$
3. **Endfor**

# Example

1. (1) **Var** int:  $d = 0$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1.  $d = d + 1$
  2.  $a[i] = a[i] \times a[i] + d \times d$
3. **Endfor**

# Example

1. (1) **Var** int:  $d = 0$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1)  $d = d + 1$
  2.  $a[i] = a[i] \times a[i] + d \times d$
3. **Endfor**

# Example

1. (1) **Var** int:  $d = 0$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1)  $d = d + 1$
  2. (1)  $a[i] = a[i] \times a[i] + d \times d$
3. **Endfor**

# Example

1. (1) **Var** int:  $d = 0$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1)  $d = d + 1$
  2. (1)  $a[i] = a[i] \times a[i] + d \times d$
3. **Endfor**

Number of elementary operations:  $1 + n \times (1 + 1) = 2n + 1$ .

# Linear loop example

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. Write “Bonjour”
  2.  $i = i + 1$
3. **EndWhile**

# Linear loop example

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. Write “Bonjour”
  2.  $i = i + 1$
3. **EndWhile**

1. **Var** int:  $i = n$
2. **While**  $i \geq 1$  **do**
  1. Write “Bonjour”
  2.  $i = i - 1$
3. **EndWhile**



# Linear loop example

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. Write “Bonjour”
  2.  $i = i + 1$
3. **EndWhile**

1. **Var** int:  $i = n$
2. **While**  $i \geq 1$  **do**
  1. Write “Bonjour”
  2.  $i = i - 1$
3. **EndWhile**

Number of elementary operations:  $2n + 1$ .

# Logarithmic loop example

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. Write “Bonjour”
  2.  $i = i \times 2$
3. **EndWhile**

# Logarithmic loop example

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. Write “Bonjour”
  2.  $i = i \times 2$
3. **EndWhile**

1. **Var** int:  $i = n$
2. **While**  $i \geq 1$  **do**
  1. Write “Bonjour”
  2.  $i = i/2$
3. **EndWhile**

# Logarithmic loop example

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. Write “Bonjour”
  2.  $i = i \times 2$
3. **EndWhile**

1. **Var** int:  $i = n$
2. **While**  $i \geq 1$  **do**
  1. Write “Bonjour”
  2.  $i = i/2$
3. **EndWhile**

Number of elementary operations:  $1 + \log_2(n)$ .

# Nested loop example

Nb of iterations = nb of iterations of external loop  $\times$  nb of iterations of internal loop

# Nested loop example

Nb of iterations = nb of iterations of external loop  $\times$  nb of iterations of internal loop

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. **Var** int:  $j = 1$
  2. **While**  $j \leq n$  **do**
    1. Write “Bonjour”
    2.  $j = j \times 3$
  3. **EndWhile**
  4.  $i = i + 1$
3. **EndWhile**

# Nested loop example

Nb of iterations = nb of iterations of external loop  $\times$  nb of iterations of internal loop

1. **Var** int:  $i = 1$
2. **While**  $i \leq n$  **do**
  1. **Var** int:  $j = 1$
  2. **While**  $j \leq n$  **do**
    1. Write “Bonjour”
    2.  $j = j \times 3$
  3. **EndWhile**
  4.  $i = i + 1$
3. **EndWhile**

Number of elementary operations:  
 $1 + 2n + n \times \log_3(n)$ .

# Exercise

Function XYZ(array:  $a[]$ )

1. **Var** int:  $i$
2. **For**  $i$  from 1 to  $n$  **do**
  1. **Var** int:  $t = a[i]$
  2. **Var** int:  $j$
  3. **For**  $j$  from  $i - 1$  to 0 **do**
    1.  $a[j + 1] = a[j]$
  4. **EndFor**
  5.  $a[j + 1] = t$
3. **EndFor**



# Exercise

Function XYZ(array:  $a[]$ )

1. (1) **Var** int:  $i$
2. **For**  $i$  from 1 to  $n$  **do**
  1. **Var** int:  $t = a[i]$
  2. **Var** int:  $j$
  3. **For**  $j$  from  $i - 1$  to 0 **do**
    1.  $a[j + 1] = a[j]$
  4. **EndFor**
  5.  $a[j + 1] = t$
3. **EndFor**

# Exercise

Function XYZ(array:  $a[]$ )

1. (1) **Var** int:  $i$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. **Var** int:  $t = a[i]$
  2. **Var** int:  $j$
  3. **For**  $j$  from  $i - 1$  to 0 **do**
    1.  $a[j + 1] = a[j]$
  4. **EndFor**
  5.  $a[j + 1] = t$
3. **EndFor**

# Exercise

Function XYZ(array:  $a[]$ )

1. (1) **Var** int:  $i$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1) **Var** int:  $t = a[i]$
  2. **Var** int:  $j$
  3. **For**  $j$  from  $i - 1$  to 0 **do**
    1.  $a[j + 1] = a[j]$
  4. **EndFor**
  5.  $a[j + 1] = t$
3. **EndFor**

# Exercise

Function XYZ(array:  $a[]$ )

1. (1) **Var** int:  $i$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1) **Var** int:  $t = a[i]$
  2. (1) **Var** int:  $j$
  3. **For**  $j$  from  $i - 1$  to 0 **do**
    1.  $a[j + 1] = a[j]$
  4. **EndFor**
  5.  $a[j + 1] = t$
3. **EndFor**

# Exercise

Function XYZ(array:  $a[]$ )

1. (1) **Var** int:  $i$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1) **Var** int:  $t = a[i]$
  2. (1) **Var** int:  $j$
  3. (?) **For**  $j$  from  $i - 1$  to 0 **do**
    1.  $a[j + 1] = a[j]$
  4. **EndFor**
  5.  $a[j + 1] = t$
3. **EndFor**

# Exercise

Function XYZ(array:  $a[]$ )

1. (1) **Var** int:  $i$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1) **Var** int:  $t = a[i]$
  2. (1) **Var** int:  $j$
  3. (?) **For**  $j$  from  $i - 1$  to 0 **do**
    1. (1)  $a[j + 1] = a[j]$
  4. **EndFor**
  5.  $a[j + 1] = t$
3. **EndFor**

# Exercise

Function XYZ(array:  $a[]$ )

1. (1) **Var** int:  $i$
2. ( $n$ ) **For**  $i$  from 1 to  $n$  **do**
  1. (1) **Var** int:  $t = a[i]$
  2. (1) **Var** int:  $j$
  3. (?) **For**  $j$  from  $i - 1$  to 0 **do**
    1. (1)  $a[j + 1] = a[j]$
  4. **EndFor**
  5. (1)  $a[j + 1] = t$
3. **EndFor**

- ① Cài đặt các thuật toán có độ phức tạp lần lượt là  $O(n)$ ,  $O(\log(n))$ ,  $O(n^2)$ . Chạy thử nghiệm đo thời gian thực thi với tập dữ liệu lần lượt là: 10, 100, 1000, 10000, 100000, 1000000
- ② Xây dựng các tập dữ liệu sau:
  - Gồm 10 số được sinh ngẫu nhiên khác nhau trong khoảng [1..100]
  - Gồm 100 số được sinh ngẫu nhiên khác nhau trong khoảng [1..1000]
  - Gồm 1000 số được sinh ngẫu nhiên khác nhau trong khoảng [1..10000]
  - Gồm 10000 số được sinh ngẫu nhiên khác nhau trong khoảng [1..100000]

Xây dựng cây tìm kiếm nhị phân (AVL) và danh sách liên kết đơn với các tập dữ liệu đã sinh ở trên. sinh một số ngẫu nhiên thực nghiệm tìm kiếm trên các cấu trúc để đo thời gian thực thi. So sánh thực nghiệm kết quả thuật toán tìm kiếm trên 2 cấu trúc dữ liệu trên.



Give algorithms having number of elementary operations as below.

- $T_1(n) = 3 + 5n$
- $T_2(n) = n \log_2 n$
- $T_3(n) = n^3$
- $T_4(n) = (3n)!$
- $T_5(n) = \log_2(3n)$
- $T_6(n) = 2 \log_3(2n)$
- $T_7(n) = n^2 \log_4 n$
- $T_8(n) = \sqrt{n}$
- $T_9(n) = \sqrt[3]{n^2}$
- $T_{10}(n) = 2^n$
- $T_{11}(n) = n!$