Độ phức tạp của thuật toán là gì?

Các ký pháp

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

Phân tích độ phức tạp thuật toán

Nhóm 7

Đại học Công nghệ Thông tin

Ngày 27 tháng 6 năm 2021

Khoa Khoa học Máy tính

Độ phức tạp của thuật toán là gì?
Khái niệm
Độ phức tạp về thời gian
Độ phức tạp về không gian
Tại sao phải đánh giá thuật toán?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Độ phức tạp của thuật toán là gì?

Khái niệm

Độ phức tạp của thuật toán là gì?
Khái niệm
Độ phức tạp về thời gian
Độ phức tạp về không gian
Tại sao phải đánh giá thuật toán?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- Một thuật toán được xây dựng phải kèm theo những đánh giá mang tính định lượng về nó.
- Để so sánh với các thuật toán liên quan khác.
- Có *hai* cách tiếp cận, tương ứng với mỗi tiêu chí đánh giá:
  - Độ phức tạp về thời gian.
  - Độ phức tạp về không gian.

Độ phức tạp về thời gian

Nhóm 7

Độ phức tạp của thuật toán là gì?

Khái niệm

Độ phức tạp về thời gian

Độ phức tạp về không gian

Tại sao phải đánh giá thuật toán?

Các ký pháp

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

- Trực quan nhất để lượng hóa tính hiệu quả của một thuật toán.
- Trong cùng một điều kiện hoạt động, thuật toán nào cho ra kết quả sớm nhất sẽ là tốt nhất.

Độ phức tạp về thời gian

Độ phức tạp của thuật toán là gì?
Khái niệm
Độ phức tạp về thời gian
Độ phức tạp về không gian
Tại sao phải đánh giá thuật toán?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

```
int Fib1(int n) {
    int a, b, c;
    if(n <= 1)
        return n;
    a = 0; b = 1; c = 0;
    for(int k = 2; k <= n; ++k) {
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}
```

Độ phức tạp về thời gian

Độ phức tạp của thuật toán là gì?
Khái niệm
Độ phức tạp về thời gian
Độ phức tạp về không gian
Tại sao phải đánh giá thuật toán?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

```
int Fib2(int n){
    if(n <= 1)
        return n;
    return Fib2(n - 1)
                + Fib2(n - 2);
}
```

Độ phức tạp về thời gian

Độ phức tạp của thuật toán là gì?
Khái niệm
Độ phức tạp về thời gian
Độ phức tạp về không gian
Tại sao phải đánh giá thuật toán?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

N	Fib1	Fib2
40	400 *ns*	1048 μs
60	61 *ns*	1s
80	81 *ns*	18 phút
100	101 *ns*	13 ngày
120	121 *ns*	36 năm
160	161 *ns*	$3.8 * 10^7$ năm
200	201 *ns*	$4 * 10^{13}$ năm

Bảng 1: So sánh thời gian thực hiện của hai thuật toán cho dãy Fibonacci

Độ phức tạp về không gian

Nhóm 7

Độ phức tạp của thuật toán là gì?

Khái niệm

Độ phức tạp về thời gian

Độ phức tạp về không gian

Tại sao phải đánh giá thuật toán?

Các ký pháp

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

- Dựa trên mức độ tiêu thụ tài nguyên của hệ thống.
- Dựa vào cấu trúc dữ liệu được sử dụng.
- Độ phức tạp về không gian không được chú ý nhiều.

Độ phức tạp về không gian

Thuật toán thứ nhất	**Thuật toán thứ hai**
$temp \leftarrow a;$	$a \leftarrow a + b;$
$a \leftarrow b;$	$b \leftarrow a - b;$
$b \leftarrow temp;$	$a \leftarrow a - b;$

Bảng 2: Hai thuật toán hoán chuyển giá trị lưu trong hai biến.

Tại sao phải đánh giá thuật toán?

Độ phức tạp của thuật toán là gì?
Khái niệm
Độ phức tạp về thời gian
Độ phức tạp về không gian
Tại sao phải đánh giá thuật toán?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- Giúp ta chọn thuật toán phù hợp nhất.
- Đơn giản mà vẫn tổng quát, có thể áp dụng trong nhiều vấn đề khác nhau.
- Tiết kiệm thời gian, không bị giới hạn bởi các yếu tố như cấu hình máy tính, ngôn ngữ sử dụng, trình biên dịch, dữ liệu đầu vào, ...

Độ phức tạp của thuật toán là gì?

Các ký pháp

Tham số quyết định và phép toán cơ sở

Tỉ suất tăng

Worst-case, Best-case, Average-case

Các ký pháp

Dùng lim để so sánh tỉ suất tăng

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

Các ký pháp

Tham số quyết định và phép toán cơ sở

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- Chiều dài của một list.
- Bậc của một đa thức hoặc số lượng các hệ số.
- Số lượng bit(b) trong biểu diễn nhị phân của n.

$$b = \log_2 n + 1$$

Tham số quyết định và phép toán cơ sở

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- Để đo được độ hiệu quả của thuật toán thì phải có một phép đo không dựa trên các yếu tố không liên quan.
- Một cách tiếp cận khả thi là *đếm số lần thực thi* của mỗi phép toán trong một giải thuật.

$\Rightarrow$ *Phép toán cơ sở:* phép toán mà đóng góp thời gian chạy nhiều nhất trong một giải thuật.

- Điển hình là các phép toán số học: $+, -, *, /, \ldots$ Thứ tự thời gian để thực hiện các phép toán số học là: $/, *, (+, -)$.

Tham số quyết định và phép toán cơ sở

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j \leftarrow 2$ **to** *length*$[A]$	c_1	n
2 **do** $key \leftarrow A[j]$	c_2	$n-1$
3 ▷ Insert $A[j]$ into the sorted		
▷ sequence $A[1..j-1]$.	0	$n-1$
4 $i \leftarrow j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 **do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] \leftarrow key$	c_8	$n-1$

Hình 1: Số lượng các phép toán cơ sở trong Insertion Sort

Tham số quyết định và phép toán cơ sở

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Problem	**Tham số quyết định**	**Phép toán cơ sở**
Tìm giá trị k trong mảng có n phần tử	Số lượng phần tử n	Phép so sánh
Phép nhân hai ma trận	Số chiều hoặc số lượng các phần tử	Phép nhân
Kiểm tra số n có phải là số nguyên tố	n'size = số chữ số (biểu diễn nhị phân)	Phép chia

Tham số quyết định và phép toán cơ sở

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- Giả sử c_{op} là thời gian thực thi của một phép toán cơ sở.
- $C(n)$ là số lần thực hiện phép toán cơ sở.

$\Rightarrow$ Công thức ước lượng thời gian $T(n)$ thực hiện giải thuật:

$$T(n) \approx c_{op} C(n)$$

Nhóm 7

Độ phức tạp của thuật toán là gì?

Các ký pháp

Tham số quyết định và phép toán cơ sở

Tỉ suất tăng

Worst-case, Best-case, Average-case

Các ký pháp

Dùng lim để so sánh tỉ suất tăng

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

Giả sử: $C(n) = \frac{1}{2}n(n-1)$

(?) Thời gian chạy giải thuật sẽ lâu hơn bao nhiêu nếu kích cỡ input lớn gấp đôi?

$$C(n) = \frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \approx \frac{1}{2}n^2$$

$$\Rightarrow \frac{T(2n)}{T(n)} \approx \frac{c_{op}C(2n)}{c_{op}C(n)} \approx \frac{\frac{1}{2}(2n)^2}{\frac{1}{2}n^2} = 4$$

Tỉ suất tăng

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- Kích thước đầu vào nhỏ thông thường được tính một cách ngay lập tức, do đó chúng ta chỉ quan tâm thuật toán hoạt động ra sao khi $n \longrightarrow \infty$.
- Thực tế, với n là giá trị nhỏ thì phần lớn các giải thuật đều cho ra thời gian đều như nhau. Chỉ khi $n \longrightarrow \infty$ thì sự khác biệt ngày càng rõ.

Tỉ suất tăng

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

n	$log_2 n$	n	$nlog_2 n$	n^2	n^3	$2n$	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	10^6
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	10^{157}
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^5	$2.0 \cdot 10^7$	10^{12}	10^{18}		

Bảng 3: Ví dụ tỉ suất tăng

Worst-case

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Trường hợp xấu nhất (hiệu quả thấp nhất)

$$C_{worst}(n) = n$$

- Giải thuật chạy **lâu nhất** trong số các input phù hợp.
- Cách xác định: phân tích giải thuật để xem loại input nào cho ra số lần đếm $C_{worst}(n)$ là **lớn nhất** giữa những input phù hợp.

Best-case

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Trường hợp tốt nhất (hiệu quả nhất)

$$C_{best}(n) = 1$$

- Giải thuật chạy **nhanh nhất** trong số các input phù hợp.
- Cách xác định: phân tích giải thuật để xem loại input nào cho ra số lần đếm $C_{best}(n)$ là **nhỏ nhất** giữa những input phù hợp.

Average-case

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Trường hợp trung bình = trung bình cộng

- Chạy giải thuật nhiều lần bằng những input cùng size n (dùng một số hàm phân phối để tạo ra các input đó).
- Tính tổng thời gian chạy và chia cho số lần thử.

Các ký pháp

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Các ký hiệu trong phần này:

- $t(n)$: thời gian chạy của giải thuật.
  (thông thường là được chỉ ra qua phép đếm $C(n)$).
- $g(n)$: là hàm được dùng để so sánh với $t(n)$

Big-O

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- $O(g(n))$ là tập của tất cả các hàm có tỉ suất tăng **thấp hơn hoặc bằng** với $g(n)$ (với bội số không đổi và $n \to \infty$).
- Ví dụ:

$$n \in O(n^2) \tag{1}$$

$$100n + 5 \in O(n^2) \tag{2}$$

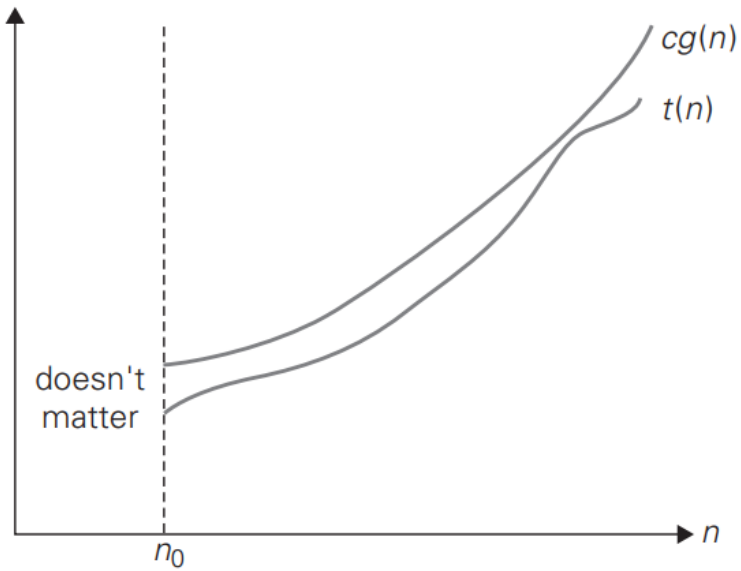
$$\frac{1}{2}n(n-1) \in O(n^2) \tag{3}$$

Big-O

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Định nghĩa toán học:

$$t(n) \in O(g(n)) \iff t(n) \leqslant cg(n) \quad (\exists c > 0, n \geqslant n_0)$$



Hình 2: Ký pháp Big-O:
$t(n) \in O(g(n))$

Nhóm 7

Độ phức tạp của thuật toán là gì?

Các ký pháp

Tham số quyết định và phép toán cơ sở

Tỉ suất tăng

Worst-case, Best-case, Average-case

Các ký pháp

Big-O

Big-Omega

Big-Theta

Dùng lim để so sánh tỉ suất tăng

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

Chứng minh: $100n + 5 \in O(n^2)$

$$\begin{aligned} & 100n + 4 \leqslant 100n + n \quad (\forall n \geqslant 5) \\ = \; & 101n \leqslant 100n^2 \end{aligned}$$

Chứng minh xong với $c = 101, n_0 = 5$.

Thực tế là định nghĩa phía trên cho ta rất nhiều cách chọn khác nhau với hai hằng số c và n_0.

Ví dụ: $100n + 5 \leqslant 100n + 5n \quad (\forall n \geqslant 1)$

Big-Omega (Ω)

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

- $\Omega(g(n))$ là tập của tất cả các hàm có tỉ suất tăng **cao hơn hoặc bằng** với $g(n)$ (với bội số không đổi và $n \to \infty$).
- Ví dụ:

$$n^3 \in \Omega(n^2) \tag{1}$$

$$\frac{1}{2}n(n-1) \in \Omega(n^2) \tag{2}$$

nhưng

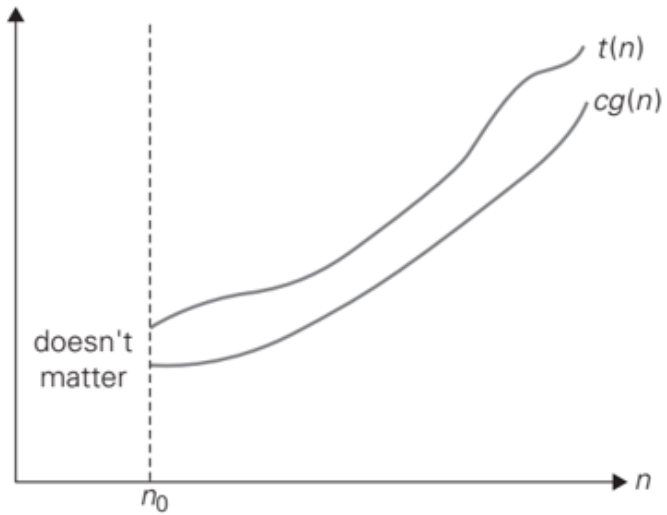
$$100n + 5 \notin \Omega(n^2) \tag{3}$$

Big-Omega (Ω)

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Định nghĩa toán học:

$$t(n) \in \Omega(g(n)) \iff t(n) \geqslant cg(n) \quad (\exists c > 0, n \geqslant n_0)$$



Hình 3: Ký pháp Big-Ω:
$t(n) \in \Omega(g(n))$

Big-Omega (Ω)

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Ví dụ chứng minh: $n^3 \in \Omega(g(n))$

$$\iff t(n) \geqslant cg(n) \quad (\forall n \geqslant 0)$$

Ở đây chúng ta chọn: $c = 1, n_0 = 0$

Big-Theta (Θ)

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

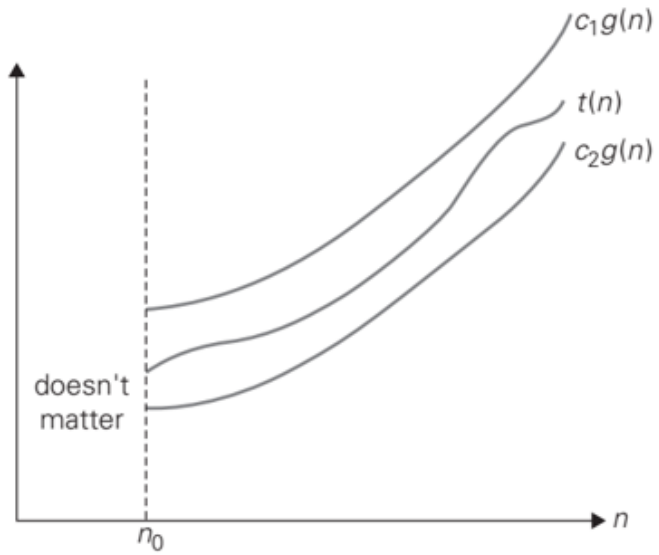
- $\Theta(g(n))$ là tập của tất cả các hàm có tỉ suất tăng **bằng** với $g(n)$ (với bội số không đổi và $n \to \infty$).
- Do đó mọi hàm bậc 2 "$an^2 + bn + c$", $\forall a > 0$ đều thuộc $\Theta(n^2)$

Big-Theta (Θ)

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Định nghĩa toán học: $t(n) \in \Theta(g(n))$

$$\iff c_2 g(n) \leqslant t(n) \leqslant c_1 g(n) \quad (\exists c_1, c_2 > 0, n \geqslant n_0)$$



Hình 4: Ký pháp Big-Θ: $t(n) \in \Theta(g(n))$

Big-Theta (Θ)

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Big-O
Big-Omega
Big-Theta
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

Chứng minh: $\frac{1}{2}n(n-1) \in \Theta(n^2)$

- Chứng minh cận trên, bất đẳng thức về phải:

$$\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \leqslant \frac{1}{2}n^2 \quad (\forall n \geqslant 0) \tag{1}$$

- Chứng minh cận dưới, bất đẳng thức về trái:

$$\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \geqslant \frac{1}{2}n^2 - \frac{1}{2}n\frac{1}{2}n = \frac{1}{4}n^2 \quad (\forall n \geqslant 2) \tag{2}$$

- Do đó, chúng ta có thể chọn: $c_2 = \frac{1}{4}, c_1 = \frac{1}{2}$ và $n_0 = 2$

Dùng lim để so sánh tỉ suất tăng

Độ phức tạp của thuật toán là gì?
Các ký pháp
Tham số quyết định và phép toán cơ sở
Tỉ suất tăng
Worst-case, Best-case, Average-case
Các ký pháp
Dùng lim để so sánh tỉ suất tăng
Cách tính độ phức tạp của thuật toán
Tổng quan
Tài liệu tham khảo

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \begin{cases} 0 \\ c \\ \infty \end{cases}$$

Trong đó:

- $lim = 0$: $t(n)$ có tỉ suất tăng **nhỏ** hơn $g(n)$.
- $lim = c$: $t(n)$ có **cùng** tỉ suất tăng với $g(n)$.
- $lim = \infty$: $t(n)$ có tỉ suất tăng **lớn** hơn $g(n)$.

Độ phức tạp của thuật toán là gì?

Các ký pháp

Cách tính độ phức tạp của thuật toán

Các độ phức tạp thường gặp

Các quy tắc

Tổng quan

Tài liệu tham khảo

Cách tính độ phức tạp của thuật toán

Các độ phức tạp thường gặp

- Hằng số: $O(1)$
- Logarith: $O(\log_2 n)$
- Tuyến tính: $O(n)$
- Đa thức: $O(P(n))$
- Hàm mũ: $O(2^n)$

Các độ phức tạp thường gặp

- $O(1)$: số phép tính/thời gian chạy/dung lượng bộ nhớ không phụ thuộc vào độ lớn đầu vào.
- $O(n)$: số phép tính/thời gian chạy/dung lượng bộ nhớ có xu hướng tỉ lệ thuận với độ lớn đầu vào.
- $O(P(n))$: với P là đa thức bậc cao (từ 2 trở lên).
- $O(2^n)$: trường hợp bất lợi nhất.

Các độ phức tạp thường gặp

Độ phức tạp của thuật toán là gì?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Các độ phức tạp thường gặp
Các quy tắc
Tổng quan
Tài liệu tham khảo

Hằng số, $O(1)$:

```
age = int(input())                  # O(1)
visitors = 0                        # O(1)
if age < 17:
    status = "Not allowed"          # O(1)
else:
    status = "Welcome!"             # O(1)
    visitors += 1                   # O(1)
```

Các độ phức tạp thường gặp

Độ phức tạp của thuật toán là gì?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Các độ phức tạp thường gặp
Các quy tắc
Tổng quan
Tài liệu tham khảo

- Logarith, $O(logn)$: thường trong các thuật toán chia nhỏ vấn đề thành các vấn đề nhỏ hơn.
- Chẳng hạn: cây nhị phân, ...

Các độ phức tạp thường gặp

Độ phức tạp của thuật toán là gì?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Các độ phức tạp thường gặp
Các quy tắc
Tổng quan
Tài liệu tham khảo

Tuyến tính, $O(n)$:

```
total = 0                    # O(1)
for i in range(n)
    total += i               # O(n)
print(total)                 # O(1)
```

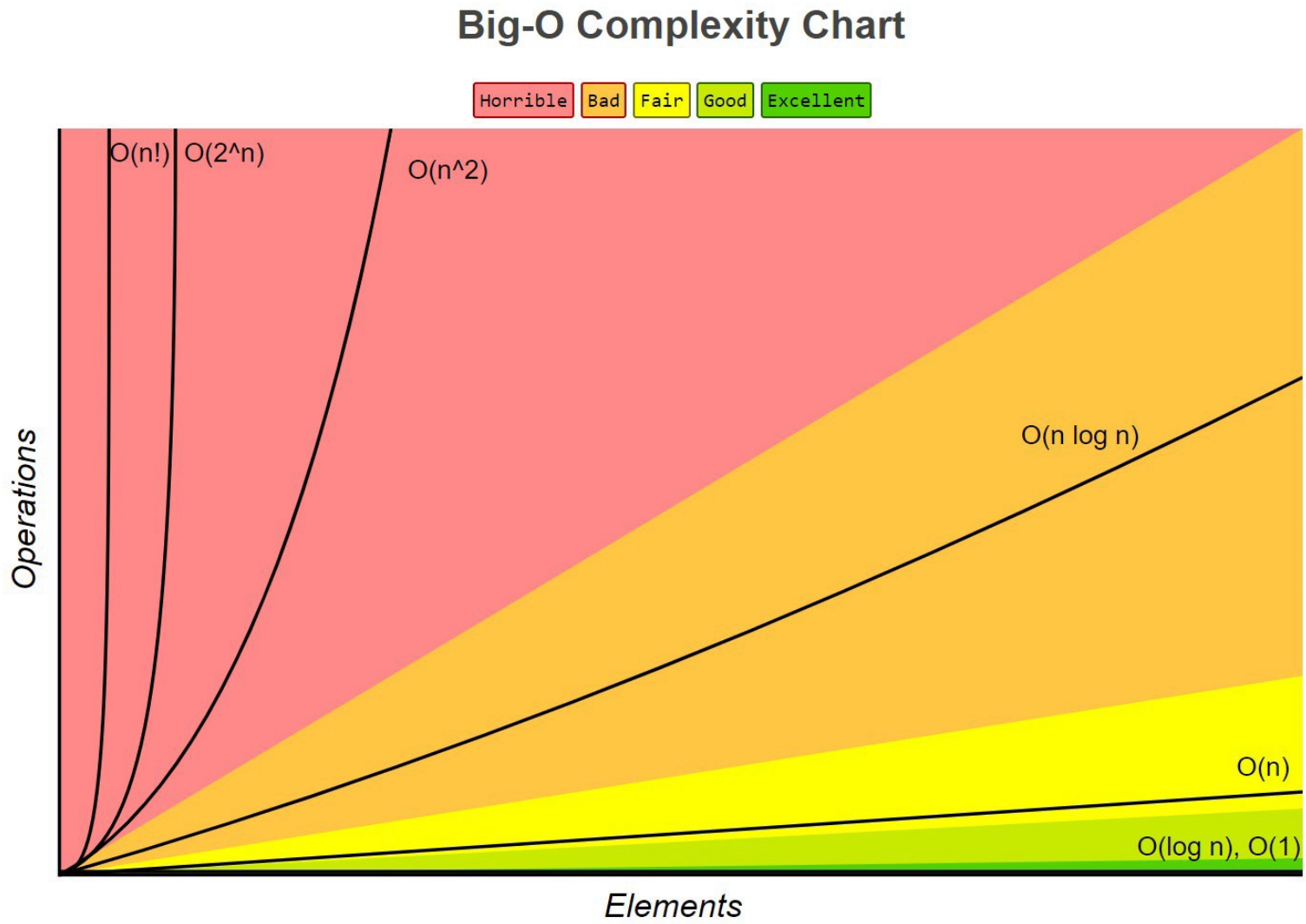
Các độ phức tạp thường gặp

Đa thức, $O(P(n))$:

```
x = 0                                   # O(1)
for i in range(n):
    for j in range(n):
        for k in range(n):
            x += 2                      # O(n^3)
```

Các độ phức tạp thường gặp

Độ phức tạp của thuật toán là gì?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Các độ phức tạp thường gặp
Các quy tắc
Tổng quan
Tài liệu tham khảo



Các quy tắc

- Quy tắc bỏ hằng số:

$$T(n) = O(c \cdot f(n)) = O(f(n)) \quad c = const, c \geqslant 0 \qquad (1)$$

- Quy tắc lấy max:

$$O(f(n)) + O(g(n)) = O(max(f(n), g(n)) \qquad (2)$$

- Quy tắc nhân: $T(n) = O(f(n))$
  Nếu thực hiện $k(n)$ lần với $k(n) = O(g(n))$, thì độ phức tạp sẽ là $O(g(n) \cdot f(n))$.

Ví dụ

Độ phức tạp của thuật toán là gì?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Các độ phức tạp thường gặp
Các quy tắc
Tổng quan
Tài liệu tham khảo

```
1          s = 0;
2          for (int i = 0; i<= n; ++i){
3              p = 1;
4              for (int j = 1; j <= i; ++j)
5                  p = p * x / j;
6              s += p;
7          }
```

Ví dụ

Độ phức tạp của thuật toán là gì?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Các độ phức tạp thường gặp
Các quy tắc
Tổng quan
Tài liệu tham khảo

```
1        s = 0;
2        for (int i = 0; i<= n; ++i){
3            p = 1;
4            for (int j = 1; j <= i; ++j)
5                p = p * x / j;
6            s += p;
7        }
```

- Số lần thực hiện phép toán $p = p * x / j$ là $n(n+1)/2$ lần.
- Độ phức tạp của đoạn code này là:

$$O(1) + O(\frac{1}{2}(n^2 + n)) + O(1) + O(1) = O(n^2)$$

Ví dụ

Độ phức tạp của thuật toán là gì?
Các ký pháp
Cách tính độ phức tạp của thuật toán
Các độ phức tạp thường gặp
Các quy tắc
Tổng quan
Tài liệu tham khảo

```
1   def function():
2       num1 = 50000
3       n = num1 / 1000
4       m = 2
5       for i in range(n):
6           for j in range(m):
7               print("This is 1st string")
8       num2 = 10000
9       x = num2 / 1000
10      for i in range(x):
11          for j in range(x):
12              print("This is 2nd string")
13              print("This is 3rd string")
14      return "Returned string"
```

Ví dụ

- Dòng 2, 3, 4, 7, 8, 9, 12, 13, 14: mỗi dòng có $O(1)$nên đô phức tạp của 9 dòng đó là **9**.
- Dòng 5, 6: $n * m$
- Dòng 10, 11: $x * x = x^2$
- Áp dụng quy tắc cộng: $9 + n * m + x^2$
- Áp dụng quy tắc bỏ hằng số: $n * m + x^2$
- Áp dụng quy tắc lấy Max: $\max(n * m, x^2)$ là độ phức tạp của hàm trên.

Độ phức tạp của thuật toán là gì?

Các ký pháp

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

Tổng quan

Tổng quan về độ phức tạp của thuật toán

- Độ phức tạp thuật toán được dùng để đánh giá các thuật toán khác nhau từ đó chọn ra thuật toán tối ưu nhất.
- Thuật toán có độ phức tạp càng cao thông thường sẽ tốn nhiều thời gian.
- Việc đánh giá một thuật toán trước khi đem nó vào thực tiễn sẽ giúp tiết kiệm thời gian, tiền bạc.
- Có 3 nguyên tắc cần nhớ đó là **bỏ hằng số, nhân và lấy max**.

Độ phức tạp của thuật toán là gì?

Các ký pháp

Cách tính độ phức tạp của thuật toán

Tổng quan

Tài liệu tham khảo

Tài liệu tham khảo

Tài liệu tham khảo

[1] Đinh Bá Tiến Trần Đan Thư.
Nhập môn lập trình.
NXB Khoa học và Kỹ thuật, 2018.

[2] Độ phức tạp thuật toán.

[3] Tran Van Tuan.
Độ phức tạp thuật toán.

[4] Nguyễn Yên Bảo.
Time complexity.

[5] How to find time complexity of an algorithm.

[6] Philips Tel.
Determining the complexity of algorithm (the basic part).