Cấu trúc dữ liệu & Giải thuật
(Data Structures and Algorithms)

Phân tích độ phức tạp của giải thuật

Nguyễn Tri Tuấn
Khoa CNTT – ĐH.KHTN.Tp.HCM
Email: nttuan@fit.hcmus.edu.vn

Thuật ngữ

- Chi phí (cost)
- Độ phức tạp (complexity)
- Phân tích độ phức tạp (complexity analysis)

Nội dung

1. Chi phí của giải thuật
2. Độ phức tạp của giải thuật
3. Big-O, Big-Ω, Big-Θ

Chi phí của giải thuật (1)

- Tính tổng n số nguyên:

```
sum = 0;
for  (i = 0; i < n; i++)
        sum += i;
```

- Giải thuật Bubble sort:

```
for (i = n-1; i > 0; i--)
        for (j = 1; j <= i; j++)
                if (a[j-1] > a[j]) {
                        temp = a[j-1];
                        a[j-1] = a[j];
                        a[j] = temp;
                }
```

Chi phí của giải thuật (2)

- Cùng một vấn đề, có thể giải quyết bằng nhiều giải thuật khác nhau
  - VD. Sắp xếp mảng → Bubble sort, Heap sort, Quick sort,…
- Mỗi giải thuật có chi phí (cost) khác nhau
- Chi phí thường được tính dựa trên:
  - thời gian (time)
  - bộ nhớ (space/memory)
- Chi phí "*thời gian*" thường được quan tâm nhiều hơn

Chi phí của giải thuật (3)

- Tuy nhiên, việc dùng khái niệm "thời gian" theo nghĩa đen (*vd. giải thuật A chạy trong 10s*) là không ổn, vì:
  - tuỳ thuộc vào loại máy tính (vd. máy Dual-Core sẽ chạy nhanh hơn Pentium II)
  - tuỳ thuộc ngôn ngữ lập trình (vd. Giải thuật viết bằng C/Pascal có thể chạy nhanh gấp 20 lần viết bằng Basic/LISP)

- Do đó, người ta thường dùng "đơn vị đo logic" (vd. số phép tính cơ sở) thay cho đơn vị đo "thời gian thật" (mili-giây, giây,...)
  - VD. Chi phí để sắp xếp mảng n phần tử bằng giải thuật Bubble sort là n^2 (phép tính cơ sở)

Nội dung

Độ phức tạp của giải thuật (1)

VD. Tính độ phức tạp của giải thuật sau

```
sum = 0;
for (i=0; i<n; i++)
    sum += i;
```

→ số phép so sánh: n

→ số phép gán: 2n+2

- Để đơn giản, người ta xem như các phép tính cơ sở có thời gian thực hiện như nhau (vd. +,-,*,/, so sánh, if … else,…)

→ độ phức tạp của giải thuật trên: f(n) = 3n+2

Độ phức tạp của giải thuật (2)

- Thông thường, độ phức tạp của giải thuật không phụ thuộc vào *giá trị* của dữ liệu đầu vào, mà phụ thuộc vào *kích thước* của dữ liệu đầu vào

→ độ phức tạp của giải thuật thường được định nghĩa là một hàm có tham số là kích thước của dữ liệu đầu vào

- VD:
  - Độ phức tạp của giải thuật tính n! là f(n)
  - Độ phức tạp của giải thuật sắp xếp mảng m phần tử là f(m)

Độ phức tạp của giải thuật (3)

- Người ta thường chỉ quan tâm đến độ phức tạp của giải thuật với **giả định số phần tử cần xử lý rất lớn** (n → ∞)
  - Như vậy, ta có thể bỏ qua các thành phần "rất bé" trong biểu thức tính độ phức tạp
  - VD. $f(n) = \mathbf{n^2} + 100n + \log_{10}n + 1000$
- Việc xác định độ phức tạp chính xác cho một giải thuật rất khó khăn, thậm chí nhiều khi không thể

→ ta có thể bỏ qua các thành phần phụ (ảnh hưởng không đáng kể)

VD.
```
for (i=0; i<n; i++) {
        a = a + b;
        if (c==0) a = 0;
    }  // độ phức tạp: f(n) = n
```

Độ phức tạp của giải thuật (4)

n	f(n)	n^2		100n		$\log_{10}n$		1000	
	Value	Value	%	Value	%	Value	%	Value	%
1	1,101	1	0.1	100	9.1	0	0.0	1000	90.82
10	2,101	100	4.76	1,000	47.6	1	0.05	1000	47.62
100	21002	10,000	47.6	10,000	47.6	2	0.991	1000	4.76
1,000	1,101,003	1,000,000	90.8	100,000	9.1	3	0.0003	1000	0.09
10,000	101,001,004	100,000,000	99.0	1,000,000	0.99	4	0.0	1000	0.001
100,000	10,010,001,005	10,000,000,000	99.9	10,000,000	0.099	5	0.0	1000	0.00

Mức tăng của các thành phần trong

$f(n) = \mathbf{n^2} + 100n + \log_{10}n + 1000$

Độ phức tạp của giải thuật (5)

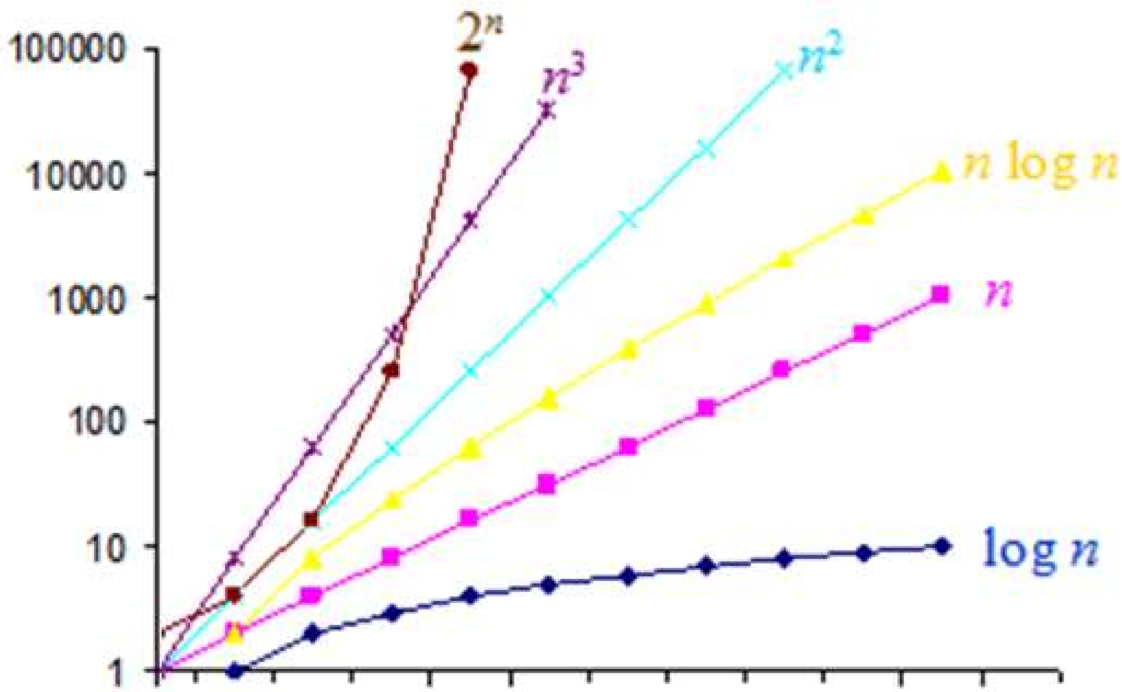
- Trường hợp tốt nhất (Best case)
  - Không phản ánh được thực tế
- Trường hợp trung bình (Average case)
  - Rất khó xác định, vì lệ thuộc nhiều yếu tố khách quan
- Trường hợp xấu nhất (Worst case)
  - Cho chúng ta một sự "bảo đảm tuyệt đối"
  - VD. Độ phức tạp của giải thuật sẽ không nhiều hơn n^2

  → Ta thường dùng độ đo "xấu nhất"

Độ phức tạp của giải thuật (6)

- Độ phức tạp thường gặp đối với các giải thuật thông thường:
  - Độ phức tạp hằng số: O(1). Số phép tính không phụ thuộc vào độ lớn đầu vào
  - Độ phức tạp tuyến tính: O(n). Số phép tính có xu hướng tỉ lệ thuận với độ lớn đầu vào
  - Độ phức tạp logarit: O(log n)
  - Độ phức tạp đa thức: O(P(n)). Với P(n) là đa thức bậc 2 trở lên. Vd. $O(n^2)$, $O(n^3)$
  - Độ phức tạp hàm mũ: $O(2^n)$

So sánh các hàm số



$\log n$	n	$n \log n$	n^2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65,536
5	32	160	1,024	32,768	4,294,967,296

Bài tập

- Tính độ phức tạp của giải thuật Bubble sort:
  - Trường hợp tốt nhất ?
  - Trường hợp xấu nhất ?

Nội dung

1. Chi phí của giải thuật
2. Độ phức tạp của giải thuật
3. Big-O, Big-Ω, Big-Θ

Big-O (1)

- Lịch sử:
  - Ký hiệu Big-O được giới thiệu năm 1894 bởi **Paul Bachmann** (Đức) trong cuốn sách *Analytische Zahlentheorie* ("Analytic Number Theory") (tái bản lần 2)
  - Ký hiệu này (sau đó) được phổ biến rộng rãi bởi nhà toán học **Edmund Landau**, nên còn gọi là ký hiệu Landau (Landau notation), hay Bachmann-Landau notation
  - **Donald Knuth** là người đưa ký hiệu này vào ngành Khoa học máy tính (Computer Science) năm 1976 – "*Big Omicron and big Omega and big Theta*" - ACM SIGACT News, Volume 8, Issue 2

Big-O (2)

- Định nghĩa:
  - Cho f(n) và g(n) là hai hàm số
  - Ta nói: $f(n) = O(g(n))$ khi n→∞, nếu tồn tại các số dương *c* và *K* sao cho:

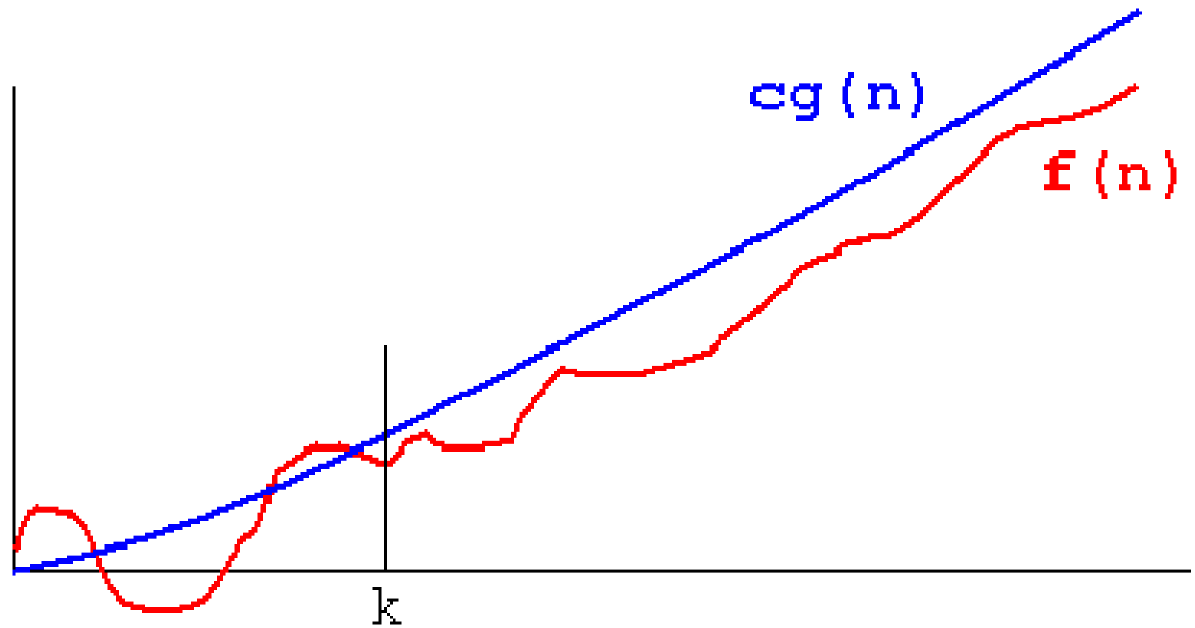
$$|f(n)| <= c*|g(n)| \quad \forall n>=K$$

  - *<u>Giải thích:</u>* *f* là big-O của *g* nếu tồn tại số dương *c* sao cho *f* không thể lớn hơn *c*g* khi *n* đủ lớn

- Cách đọc: f(n) là big-O của g(n)

- Ý nghĩa:
  - g(n) là ***giới hạn trên (upper bound)*** của f(n); hay
  - Khi n lớn, f(n) tăng tương đương bằng g(n)

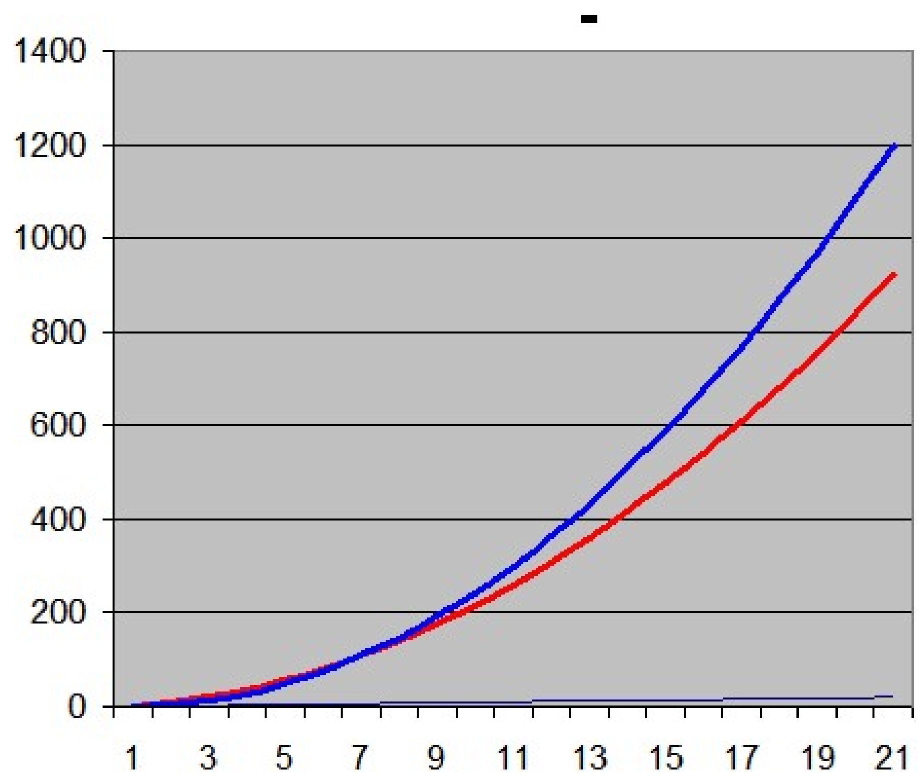
Big-O (3)



Khi n đủ lớn (n>=K), thì g(n) là giới hạn trên của f(n)

Big-O (4)

- VD. $f(n) = 2n^2 + 6n + 1 = O(n^2)$, $g(n) = n^2$
  - Thật vậy, ta chọn được $c = 3$ và $K = 7$
  - $\forall n >= 7 \rightarrow f(n) < 3 * g(n)$



Big-O (5)

- Khi áp dụng big-O vào việc ước lượng độ phức tạp của giải thuật, ta nên chọn g(n):
  - càng đơn giản càng tốt,
  - bỏ qua các hằng số và các thành phần có lũy thừa thấp
- Nhờ vậy, ta có thể ước lượng độ phức tạp của giải thuật một cách đơn giản hơn
  - Thay vì phát biểu *"độ phức tạp của giải thuật là $2n^2 + 6n + 1$"*, ta sẽ nói *"giới hạn (chặn) trên của độ phức tạp của giải thuật là n^2"*

Big-O (6)

- Trắc nghiệm: xác định O(g(n)) của các hàm sau đây
  - $f(n) = 10$
  - $f(n) = 5n + 3$
  - $f(n) = (n+1)^2$

Big-Ω

- Định nghĩa:
  - Cho f(n) và g(n) là hai hàm số
  - Ta nói: *f(n)* = Ω*(g(n))* khi n→∞, nếu tồn tại các số dương *c* và *K* sao cho:

$$|f(n)| >= c*|g(n)| \quad \forall n>=K$$

  - *Giải thích:* *f* là big-Ω của *g* nếu tồn tại số dương *c* sao cho *f* lớn hơn *c*g* khi *n* đủ lớn

- Cách đọc: f(n) là big-Omega của g(n)

- Ý nghĩa:
  - g(n) là ***giới hạn dưới (chặn dưới - lower bound)*** của f(n)

Big-Θ

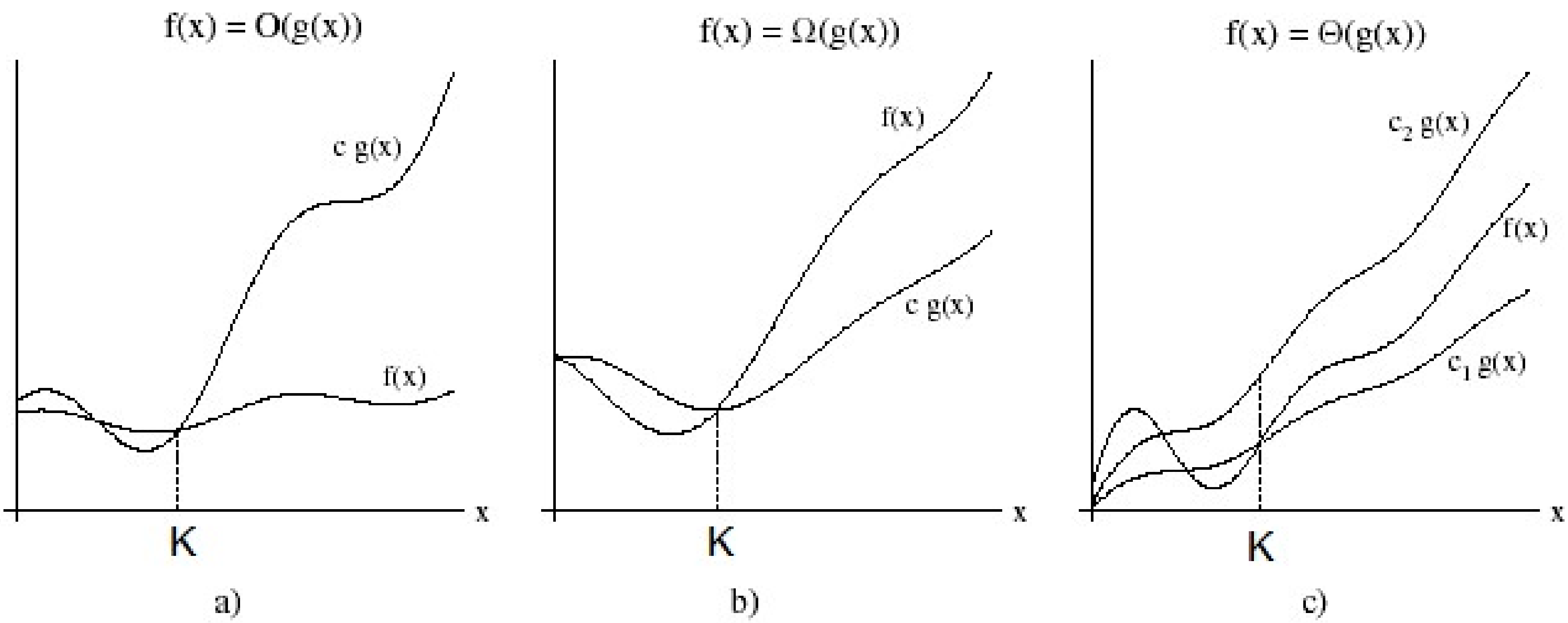
- Định nghĩa:
  - Cho f(n) và g(n) là hai hàm số
  - Ta nói: $f(n) = \Theta(g(n))$ khi n→∞, nếu tồn tại các số dương c_1, c_2 và K sao cho:

$$c_1*|g(n)| <= |f(n)| <= c_2*|g(n)| \quad \forall n>=K$$

- Cách đọc: f(n) là big-Theta của g(n)

- Ý nghĩa:
  - g(n) là ***giới hạn chặt (tight bound)*** của f(n)

Big-O, Big-Ω, Big-Θ



Minh họa big-O, big-Ω, big-Θ

Q & A