Cấu trúc dữ liệu và thuật toán

PGS. TS. Phạm Tuấn Minh
Trường Công nghệ Thông tin, Đại học Phenikaa
minh.phamtuan@phenikaa-uni.edu.vn
https://sites.google.com/site/phamtuanminh/

Chương 1: Các kiến thức cơ bản

- 1.1 Giới thiệu môn học
- **1.2 Thuật toán và độ phức tạp**
- 1.3 Áp dụng đánh giá độ phức tạp của thuật toán
- 1.4 Đánh giá độ phức tạp của thuật toán chia để trị

Khái niệm thuật toán

- Thuật toán giải bài toán đặt ra là một thủ tục xác định bao gồm một dãy hữu hạn các bước cần thực hiện để thu được đầu ra cho một đầu vào cho trước cùa bài toán.
- Ví dụ: Tìm giá trị phần tử lớn nhất trong dãy số a[1..n]

```
int max = a[0];
for (i = 0; i < n; i++)
     if a[i] > max
         max = a[i];
```

1-3

Độ phức tạp của thuật toán

- Đánh giá độ phức tạp tính toán của thuật toán là đánh giá lượng tài nguyên các loại mà thuật toán đòi hỏi sử dụng.
- Hai loại tài nguyên quan trọng: Thời gian và bộ nhớ.
- Trong phạm vi môn học, tập trung vào đánh giá thời gian tính của thuật toán bởi một hàm của kích thước dữ liệu đầu vào

1-4

Các loại thời gian tính

- **Thời gian tính tốt nhất (Best case)** của thuật toán với đầu vào kích thước n: **Thời gian tối thiểu** cần thiết để thực hiện thuật toán với mọi bộ dữ liệu đầu vào kích thước n.
- **Thời gian tính tồi nhất (Worst case)** của thuật toán với đầu vào kích thước n: **Thời gian nhiều nhất** cần thiết để thực hiện thuật toán với mọi bộ dữ liệu đầu vào kích thước n.
- **Thời gian tính trung bình (Average case)** của thuật toán với đầu vào kích thước n: **Thời gian trung bình** cần thiết để thực hiện thuật toán trên tập hữu hạn các đầu vào kích thước n.

1-5

Ký hiệu tiệm cận

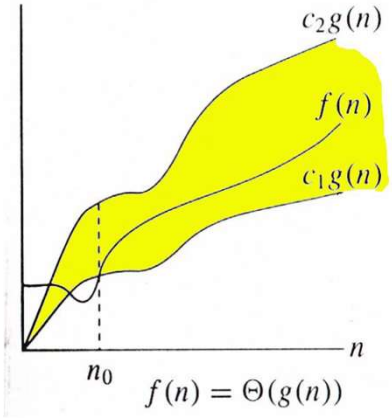
- Ký hiệu tiệm cận Θ, Ω, O
  - Dùng để mô tả thời gian tính của thuật toán
  - Thay vì nói chính xác thời gian tính, ta nói $\Theta(n^2)$
  - Được xác định đối với các hàm nhận giá trị nguyên không âm

1-6

Ký hiệu Θ

- Đối với hàm g(n), ta ký hiệu Θ(g(n)) là tập các hàm

  Θ(g(n)) = {f(n): tồn tại các hằng số c1, c2 và n0 sao cho 0 <= c1g(n) <= f(n) <= c2g(n), với mọi n >= n0 }
- g(n) là đánh giá **tiệm cận đúng** cho f(n)



1-7

Ký hiệu Θ

- $10n^2$ - 3n = $\Theta(n^2)$?
- Với giá trị nào của các hằng số n0, c1, và c2 thì bất đẳng thức trong định nghĩa là đúng?
- Lấy c1 bé hơn hệ số của số hạng với số mũ cao nhất, còn c2 lấy lớn hơn, ta có

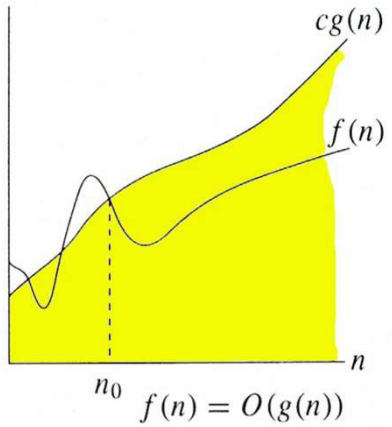
  $9n^2 \leq 10n^2 - 3n \leq 11n^2$, với mọi n ≥ 3.
- Đối với hàm đa thức: Để so sánh tốc độ tăng cần nhìn vào số hạng với số mũ cao nhất

1-8

Ký hiệu O

- Đối với hàm g(n), ta ký hiệu O(g(n)) là tập các hàm

  O(g(n)) = {f(n): tồn tại các hằng số dương c và n0 sao cho f(n) <= cg(n), với mọi n >= n0 }
- g(n) là **cận trên tiệm cận** của f(n)

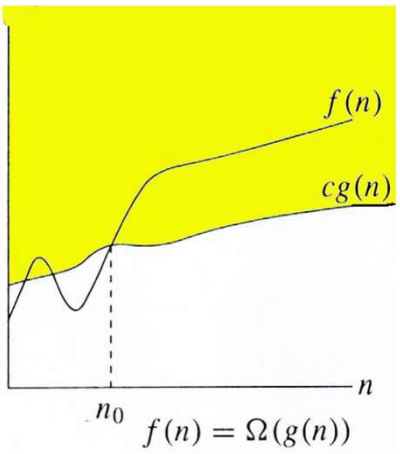


1-9

Ký hiệu Ω

- Đối với hàm g(n), ta ký hiệu Ω(g(n)) là tập các hàm

  Ω(g(n)) = {f(n): tồn tại các hằng số dương c và n0 sao cho cg(n) <= f(n), với mọi n >= n0 }
- g(n) là **cận dưới tiệm cận** của f(n)



1-10

Ví dụ

- g(n) = ?

$10n^2/2$ - n/2 = $\Theta(g(n))$

$5n^2 = \Omega$(g(n))

$100n^2$ + 100n = O(g(n))

Cận trên và cận dưới cho thời gian tính của bài toán

- Cho bài toán P, ta nói **cận trên cho thời gian tính** của P là O(g(n)) nếu để giải P tồn tại thuật toán giải với thời gian tính là O(g(n)).
- Cho bài toán P, ta nói **cận dưới cho thời gian tính** của P là Ω(g(n)) nếu mọi thuật toán giải P đều có thời gian tính là Ω(g(n)).
- Cho bài toán P, ta nói **thời gian tính** của P là Θ(g(n)) nếu P có cận trên là O(g(n)) và cận dưới là Ω(g(n)).

Một số lớp thuật toán

- O(1): hằng số (constant)
- O(log n): logarithmic
- O(n): tuyến tính (linear)
- O(n log n): trên tuyến tính (superlinear)
- O(n^2): bình phương (quadratic)
- O(n^3): bậc ba (cubic)
- O(n^k): đa thức (polynomial) (k ≥ 1)
- O(a^n): hàm mũ (exponential) (a > 1)

Chương 1: Các kiến thức cơ bản

- 1.1 Giới thiệu môn học
- 1.2 Thuật toán và độ phức tạp
- **1.3 Áp dụng đánh giá độ phức tạp của thuật toán**
- 1.4 Đánh giá độ phức tạp của thuật toán chia để trị

Cấu trúc tuần tự

- Giả sử P và Q là hai đoạn của thuật toán, có thể là một câu lệnh nhưng cũng có thể là một thuật toán con. Gọi Time(P), Time(Q) là thời gian tính của P và Q tương ứng. Khi đó ta có:
- Quy tắc tuần tự: Thời gian tính đòi hỏi bởi "P; Q", nghĩa là P thực hiện trước, tiếp đến là Q, sẽ là:

  Time(P; Q) = Time(P) + Time(Q),

  Time(P; Q) = Θ(max(Time(P), Time(Q)))

Phân tích vòng lặp for

- for i =1 to m do P(i);
- Giả sử thời gian thực hiện P(i) là t(i). Khi đó thời gian thực hiện vòng lặp for sẽ là

$$\sum_{i=1}^{n} t(i)$$

Câu lệnh đặc trưng

- Câu lệnh đặc trưng là câu lệnh được thực hiện thường xuyên, ít nhất là cũng như bất kỳ câu lệnh nào trong thuật toán.
- Nếu giả thiết thời gian thực hiện mỗi câu lệnh là bị chặn bởi hằng số thì thời gian tính của thuật toán sẽ cùng cỡ với số lần thực hiện câu lệnh đặc trưng
- => Để đánh giá thời gian tính có thể đếm số lần thực hiện câu lệnh đặc trưng

1-17

Ví dụ 1

```
function Fibiter(n)
begin
   i:=0; j:=1;
   for k:=1 to n do
   begin
     j:= j+i;
     i:= j-i;
   end;
   Fibiter:= j;
end;
```

- Câu lệnh đặc trưng?
- Số lần thực hiện câu lệnh đặc trưng?
- Thời gian tính của thuật toán?

1-18

Ví dụ 2

```
int maxSum =0;
for (int i=0; i<n; i++) {
   for (int j=i; j<n; j++) {
      int sum = 0;
      for (int k=i; k<=j; k++)
         sum += a[k];
      if sum > maxSum
         maxSum = sum;
   }
}
```

- Câu lệnh đặc trưng?
- Số lần thực hiện câu lệnh đặc trưng?
- Thời gian tính của thuật toán?

Chương 1: Các kiến thức cơ bản

- 1.1 Giới thiệu môn học
- 1.2 Thuật toán và độ phức tạp
- 1.3 Áp dụng đánh giá độ phức tạp của thuật toán
- **1.4 Đánh giá độ phức tạp của thuật toán chia để trị**

Thuật toán chia để trị

```
procedure D-and-C (n);
begin
    if n < no then
        Giải bài toán một cách trực tiếp
    else
    begin
        Chia bài toán thành a bài toán con kích thước n/b;
        for (mỗi bài toán trong a bài toán con) do D-and-C(n/b);
        Tổng hợp lời giải của a bài toán con để thu được lời giải của bài toán gốc;
    end
end
```

- Gọi **T(n)** - thời gian giải bài toán kích thước n. Thời gian của chia để trị được đánh giá dựa trên đánh giá thời gian thực hiện ba thao tác của thuật toán:
  - Chia bài toán ra thành a bài toán con mỗi bải toán kích thước n/b: đòi hỏi thời gian: **D{n)**.
  - Trị (giải) các bài toán con: **aT(n/b)**.
  - Tổ hợp các lời giải: **C(n).**

Thuật toán chia để trị

- Công thức đệ quy để tính T(n):

$$T(n) = \begin{cases} \Theta(1), & n \le n_0 \\ aT(n/b) + D(n) + C(n), & n > n_0 \end{cases}$$

Định lý thợ rút gọn

- Định lý thợ rút gọn cung cấp công cụ để đánh giá số hạng tổng quát của dãy số thỏa mãn công thức đệ quy dạng:

$$T(n) = aT\left(\frac{n}{b}\right) + cn^k$$

trong đó: a >= 1 và b > 1, c > 0 là các hằng số

1. Nếu a > b^k, thì T(n) = Θ ($n^{log_b a}$).
2. Nếu a = b^k, thì T(n) = Θ ($n^k lgn$).
3. Nếu a < b^k, thì T(n) = Θ (n^k).

1-23

Ví dụ

$$T(n) = 3T\left(\frac{n}{4}\right) + cn^2$$

trong đó: a = 3 và b = 4, k=2

Vì 3 < 4^2, tương ứng trường hợp 3 nên T(n) = Θ (n^2).

1-24

Cấu trúc dữ liệu và thuật toán

- Nội dung bài giảng được biên soạn bởi PGS. TS. Phạm Tuấn Minh.