Cấu trúc dữ liệu và giải thuật

Trần Hồng Diệp

E-mail: diepthd@gmail.com

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

CHƯƠNG 2: Thiết kế và phân tích giải thuật

I. Thiết kế giải thuật.

II. Phân tích giải thuật.

I. Thiết kế giải thuật.

Mô đun hoá để giải quyết bài toán:

- Vấn đề thực tế đặt ra:
  - ✓ Các bài toán mong muốn được giải quyết trên máy tính ngày càng đa dạng, phức tạp $\Rightarrow$ có thể có giải thuật và giải thuật đơn giản để giải quyết được bằng máy tính?
  - ✓ Các bài toán lớn và rất lớn, để có thể giải quyết được và giải quyết hiệu quả hơn $\Rightarrow$ có thể chia cho nhiều đơn vị cùng giải quyết không?

I. Thiết kế giải thuật.

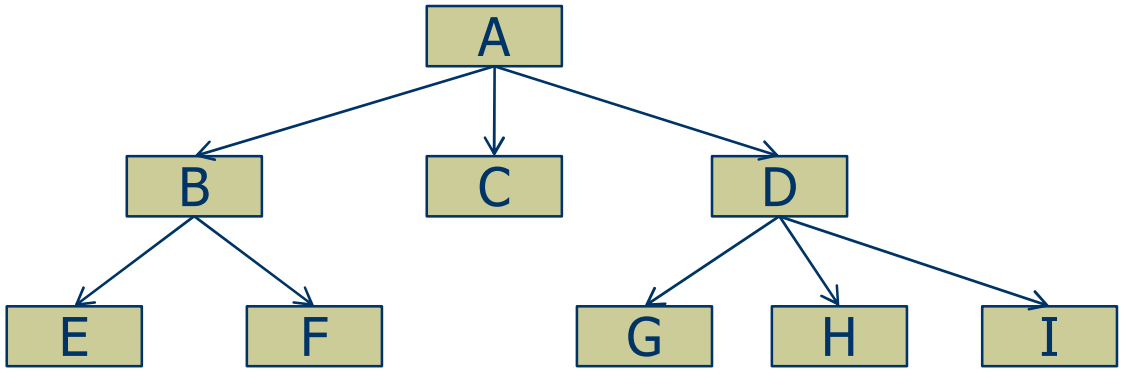
Mô đun hoá để giải quyết bài toán:

- Ý tưởng:
  - ✓ Phân rã bài toán phức tạp thành các bài toán con đơn giản hơn và có khả năng giải quyết trên máy tính.
  - ✓ Phân rã bài toán lớn thành nhiều bài toán nhỏ có thể chia cho các nhóm, các cá nhân phát triển song song, độc lập và cuối cùng ghép nối lại.

I. Thiết kế giải thuật.

Mô đun hoá để giải quyết bài toán:

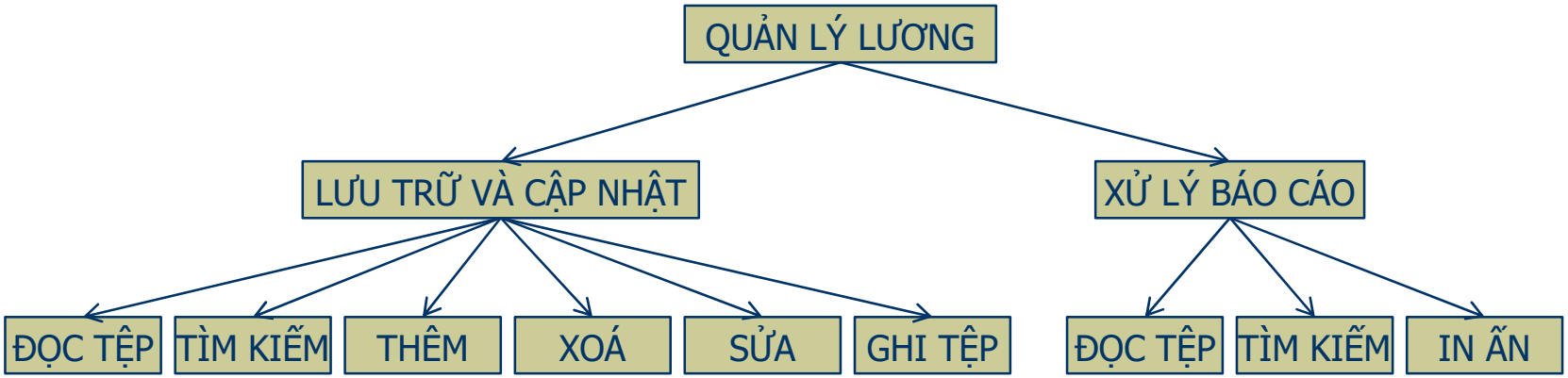
- Giải quyết: Phương pháp phân tích TOP-DOWN:
  - ✓ Coi bài toán cần giải quyết là mô-đun chính và chia nó thành các mô-đun con, tương tự như vậy: mỗi mô-đun con lại chia thành các mô-đun con của nó…



I. Thiết kế giải thuật.

Mô đun hoá để giải quyết bài toán:

- Ví dụ: Bài toán hỗ trợ quản lý lương cán bộ:



II. Phân tích giải thuật

- Vấn đề:
  - ✓ Một bài toán có thể có nhiều giải thuật giải quyết.
  - ✓ Căn cứ nào để nói giải thuật này "tốt" hay "tốt hơn" giải thuật kia.
  - ⇒ Khái niệm về "Độ phức tạp tính toán của giải thuật"

II. Phân tích giải thuật

- Độ phức tạp tính toán của giải thuật:
  - ✓ Độ phức tạp tính toán của giải thuật là sự đánh giá lượng tài nguyên các loại mà giải thuật đòi hỏi sử dụng.
  - ✓ Có hai loại tài nguyên quan trọng nhất: Thời gian và Bộ nhớ.
  - ✓ Trong đó thời gian tính được quan tâm chủ yếu

II. Phân tích giải thuật

- Đo thời gian tính của giải thuật:
  - ✓ Thời gian tính của giải thuật phụ thuộc nhiều yếu tố, nhưng chủ yếu là kích thước dữ liệu vào.
  - ✓ Độ phức tạp thời gian tính của giải thuật được đánh giá bằng một hàm của dữ liệu vào.
  - ✓ Xác định thời gian tính của giải thuật bằng số các phép tính căn bản (phép toán tích cực) mà giải thuật đòi hỏi thực hiện.

II. Phân tích giải thuật

❖ Hàm:

- Khái niệm
  - ✓ Quan hệ vào-ra từ một tập đến tập khác
  - ✓ Chỉ ra quy tắc: Một phần tử tập vào → nhiều nhất một phần tử tập ra
  - ✓ Ký hiệu: $f : X \rightarrow Y$

II. Phân tích giải thuật

❖ Hàm:

◆ Khái niệm

- ✓ Khi $x \in X$ tương đương với $y \in Y$

  "hàm f xác định tại x và có giá trị là y"
- ✓ Khi $x \in X$ không có phần tử tương đương

  "hàm f không xác định tại x"
- ✓ Ký hiệu $f(x) = y$
- ✓ x là biến của hàm f

II. Phân tích giải thuật

❖ Hàm:

- Khái niệm
  - ✓ Khi $X = X_1 \text{ x } X_2 \text{ x } \ldots \text{ x } X_k$
    gọi là hàm ***k ngôi***, hay hàm ***k biến***"
  - ✓ Tập D gọi là ***Miền xác định*** của f nếu:
    $D = \{\forall x \mid x \in X \text{ và } f(x) \text{ xác định } \}$;
    Tập R các giá trị y tương ứng là ***Miền giá trị*** của f
    $R = \{ y \mid y \in Y, \ \exists x \in X : f(x) = y \}$;

II. Phân tích giải thuật

❖ Hàm:

◆ Khái niệm

- ✓ f là hàm hoàn toàn xác định nếu:

  f xác định $\forall x \in X$ tức là $D=X$

- ✓ f là hàm lên nếu: $R=Y$
- ✓ Hàm nói chung: là *bộ phận* và *đơn trị*

II. Phân tích giải thuật

❖ Biên độ tiệm cận của hàm:

Khi n là đủ lớn và

- *O-lớn*:
  - ✓ Cho hai hàm $f(n)$ và $g(n)$
  - ✓ Nếu tồn tại hai số tự nhiên c và n_o thỏa mãn:

    $f(n) \leq c.g(n) \quad \forall n \geq n_o$
  - ➢ ***f(n)* là *O-lớn* của *g(n)***

II. Phân tích giải thuật

❖ Biên độ tiệm cận của hàm:

Khi n là đủ lớn và

- *o-bé*:
  - ✓ Cho hai hàm $f(n)$ và $g(n)$
  - ✓ Với mỗi số thực $c > 0$ nếu tồn tại số tự nhiên n_c :

    $f(n) < c.g(n) \quad \forall n \geq n_c$
  - ➢ ***f(n)* là *o-bé* của *g(n)***

II. Phân tích giải thuật

❖ Biên độ tiệm cận của hàm:

Khi n là đủ lớn và

- Cận trên tiệm cận:
  - ✓ Khi $f(n)$ là O-lớn của $g(n)$ ta viết:

    $f(n) = O[g(n)]$
  - ➢ **$g(n)$ là *cận trên tiệm cận* (cận trên) của $f(n)$**

II. Phân tích giải thuật

❖ Biên độ tiệm cận của hàm:

Khi n là đủ lớn và

- Cận dưới tiệm cận:
  - ✓ Khi $f(n) = O[g(n)]$ ta có thể viết:

    $f(n) = \Omega[g(n)]$
  - ➢ ***g(n)* là *cận dưới tiệm cận* (cận dưới) của *f(n)***

II. Phân tích giải thuật

❖ Biên độ tiệm cận của hàm:

Khi n là đủ lớn và

- Cận sát tiệm cận:
  - ✓ Khi $f(n) = O[g(n)]$ và $f(n) = \Omega[g(n)]$ ta có thể viết: $f(n) = \theta[g(n)]$
  - ➢ ***g(n)*** **là *cận sát tiệm cận* (cận chặt tiệm cận) của *f(n)***

II. Phân tích giải thuật

❖ Biên độ tiệm cận của hàm:

- Cận trên tiệm cận

Ví dụ: $f(n) = 5n^3 + 3n^2 + 18n + 7$

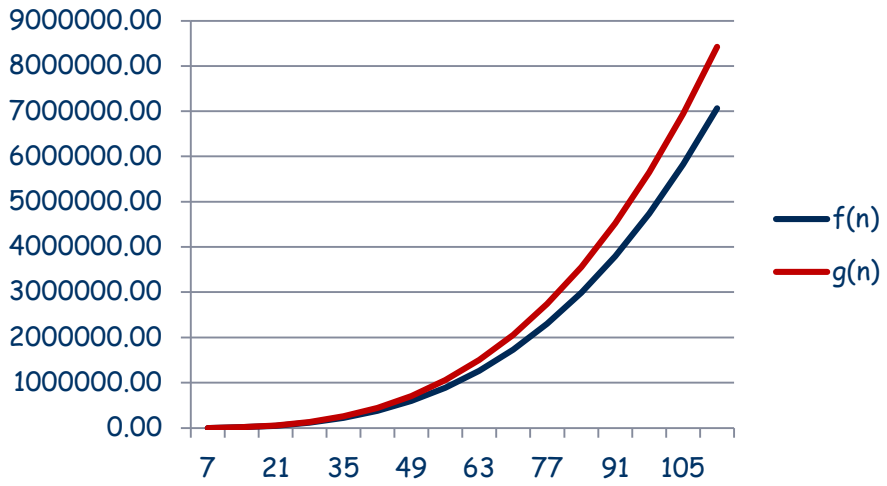
Chọn: $g(n) = n^3$

II. Phân tích giải thuật

- Chọn: $c = 6$ và $n_o = 7$
- Khi đó:

$$5n^3 + 3n^2 + 18n + 7 \leq 6n^3$$

- $f(n) = O[n^3]$



- ***g(n)* là *cận trên tiệm cận* (cận trên) của *f(n)***

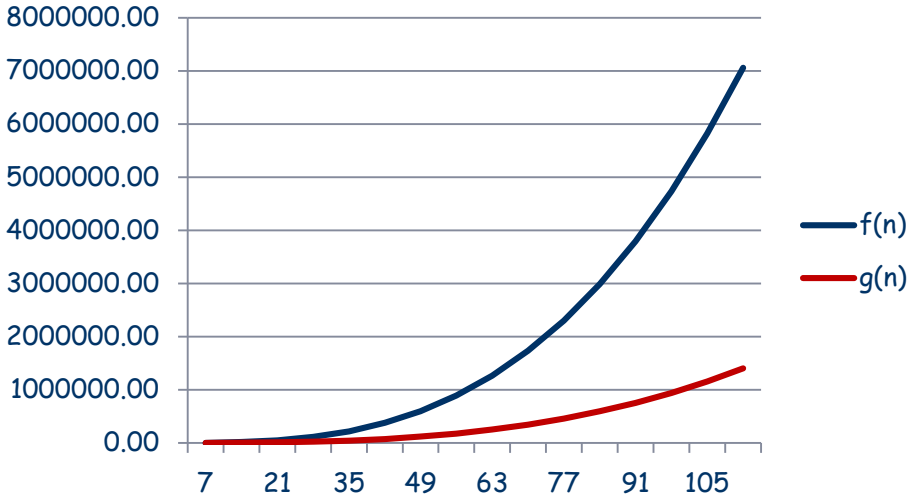
Ta cũng có thể chứng minh được $f(n) = O[n^4]$

II. Phân tích giải thuật

- Chọn: $c = 1$ và $n_o = 7$
- Khi đó:

$$5n^3 + 3n^2 + 18n + 7 \geq n^3$$

- $f(n) = \Omega[n^3]$



- ***g(n)* là *cận dưới tiệm cận* của *f(n)***
- ***g(n)* cũng là *cận sát tiệm cận* của *f(n)***

$$f(n) = \theta[n^3]$$

II. Phân tích giải thuật

- Các loại thời gian tính của giải thuật:
  - Thời gian tối thiểu để thực hiện giải thuật với mọi bộ dữ liệu vào kích thước n, còn gọi là thời gian tốt nhất, kí hiệu $\Omega(f(n))$
  - Thời gian tối đa để thực hiện giải thuật với mọi bộ dữ liệu vào kích thước n, hay thời gian tồi nhất, kí hiệu O(f(n))
  - Thời gian trung bình để thực hiện giải thuật với mọi bộ dữ liệu vào hữu hạn kích thước n, kí hiệu $\theta(f(n))$

II. Phân tích giải thuật

Qui tắc tổng: cho 2 đoạn chương trình:

- ✓ P_1 có độ phức tạp tính toán là $T_1=O(f(n))$
- ✓ P_2 có độ phức tạp tính toán là $T_2=O(g(n))$

Nếu P_1 và P_2 được thực hiện liên tiếp thì độ phức tạp tính toán chung là:

$$T=T1+T2=O(max(f(n);g(n)))$$

(*Tương tự đối với các kí hiệu Ω và θ*)

II. Phân tích giải thuật

Qui tắc nhân: cho 2 đoạn chương trình:

- ✓ P_1 có độ phức tạp tính toán là $T_1=O(f(n))$
- ✓ P_2 có độ phức tạp tính toán là $T_2=O(g(n))$

Nếu P_1 và P_2 được thực hiện lồng nhau thì độ phức tạp tính toán chung là:

$$T=T1.T2=O(f(n).g(n))$$

(*Tương tự đối với các kí hiệu Ω và θ*)

Chú ý: khi độ phức tạp O(f(n)) mà f(n) là hằng số thì đều được coi là O(1)

II. Phân tích giải thuật

- Ví dụ: Cho giải thuật sau:

```
Tb:=0; d:=0;
For i:=1 to n do
    If a[i]>0 then begin Tb:=Tb+a[i]; d:=d+1 end;
Tb:=Tb/d;
```

Ta thấy câu lệnh cơ bản ở đây là Tb:=Tb+a[i] có độ phức tạp hằng số O(1); câu lệnh này lồng trong câu lệnh For có độ phức tạp O(n) $\Rightarrow$ độ phức tạp của đoạn chương trình trên là O(1.n)=O(n)

II. Phân tích giải thuật

- Một số mức độ phức tạp thường gặp:
  - ✓ Độ phức tạp mức đa thức: logn, n, nlogn, n^2, n^3…
  - ✓ Độ phức tạp mức hàm mũ: 2^n, n!, n^n…
  - ⇒ Nói chung độ phức tạp thuộc mức đa thức thường là chấp nhận được trong thực tế.

<HẾT CHƯƠNG 2>