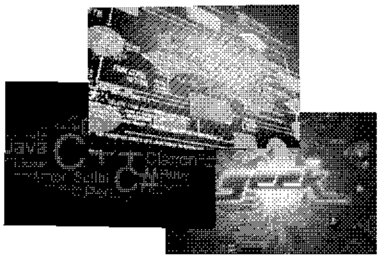
ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

NHẬP MÔN LẬP TRÌNH

CHƯƠNG II

GIỚI THIỆU VỀ THUẬT TOÁN



Nguyễn Trọng Chỉnh
chinhnt@uit.edu.vn

1

GIỚI THIỆU VỀ THUẬT TOÁN

- ❖THUẬT TOÁN
- ❖BIỂU DIỄN THUẬT TOÁN
- ❖ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

2

THUẬT TOÁN

❖KHÁI NIỆM THUẬT TOÁN
❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

3

THUẬT TOÁN

❖KHÁI NIỆM THUẬT TOÁN (Algorithm)

- Thuật toán (thuật giải) là tập hợp các thao tác có thứ tự sao cho khi thực hiện một số tao tác hữu hạn đó thì đạt được mục tiêu.
- Để xây dựng thuật toán, cần phải phân tích các mối liên hệ logic của bài toán và trình tự xác định giá trị của các mối quan hệ đó.

4

THUẬT TOÁN

❖KHÁI NIỆM THUẬT TOÁN (Algorithm)

Ví dụ 1: tìm giá trị lớn nhất của 3 số thực a, b, c.

- Phân tích:

+ Các số a, b, c. Giá trị lớn nhất m có quan hệ với a, b và c như sau: $m \geq a$, $m \geq b$, $m \geq c$.

+ Để tìm giá trị của m, lần lượt so sánh giá trị của nó với a, b, c, nếu giá trị của m nhỏ hơn thì xác định giá trị của nó là tham số đang xét.

+ Ban đầu giá trị của m chưa xác định nên m sẽ nhận giá trị của một trong 3 tham số a, b, c.

5

THUẬT TOÁN

❖KHÁI NIỆM THUẬT TOÁN (Algorithm)

Ví dụ 1: tìm giá trị lớn nhất của 3 số thực a, b, c.

- Thuật toán:

* Đầu vào: các số thực a, b, c.

* Đầu ra: số thực m là giá trị lớn nhất của a, b, c.

* Các bước thực hiện:

  + Bước 1: gán số thực m = a.
  + Bước 2: nếu m < b thì gán m = b.
  + Bước 3: nếu m < c thì gán m = c.
  + Bước 4: kết thúc.

6

THUẬT TOÁN

❖KHÁI NIỆM THUẬT TOÁN (Algorithm)

Ví dụ 2: tìm nghiệm của phương trình bậc hai

$$ax^2 + bx + c = 0.$$

- Phân tích:

+ phương trình bậc hai có các đặc trưng là số thực: hệ số a, b, c; Δ; có nhiều nhất 2 nghiệm x_1, x_2.

+ quan hệ giữa nghiệm x_1, x_2 và Δ nếu $\Delta \geq 0$ (công thức 1 và 2) là

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a} \qquad x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

7

THUẬT TOÁN

❖KHÁI NIỆM THUẬT TOÁN (Algorithm)

- Phân tích:

+ Δ nếu $\Delta < 0$, không tồn tại x_1, x_2

+ quan hệ giữa Δ và a, b, c nếu $a \neq 0$ (công thức 3) là

$$\Delta = b^2 - 4ac$$

+ nếu a=0, phương trình trở thành phương trình bậc nhất: bx + c = 0.

8

THUẬT TOÁN

❖KHÁI NIỆM THUẬT TOÁN (Algorithm)

- Thuật toán:

* Đầu vào: các số thực a, b, c là các hệ số của phương trình.
* Đầu ra: các số thực x_1, x_2 là nghiệm (nếu có).
* Các bước thực hiện:
  + Bước 1: nếu $a \neq 0$, tính Δ ; ngược lại, đến bước 3.
  + Bước 2: nếu $\Delta \geq 0$, tính x_1, x_2; ngược lại, đến bước 4.
  + Bước 3: phương trình không hợp lệ, đến bước 5.
  + Bước 4: phương trình vô nghiệm, đến bước 5.
  + Bước 5: kết thúc.

9

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

+ **Đầu vào (Input)**: Giá trị đầu vào được chỉ định rõ ràng.
+ **Đầu ra (output)**: Giá trị đầu ra là lời giải của thuật toán đối với bài toán đã cho.

Ví dụ: đầu và là các số thực như trong 2 ví dụ trên.

10

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

+ **Tính dừng**: Thuật toán phải dừng sau một số hữu hạn bước thực hiện.

Ví dụ: thuật toán tính n!

* Đầu vào: giá trị nguyên dương n
* Đầu ra: giá trị nguyên r
* Các bước thực hiện:
  + Bước 1: r = 1.
  + Bước 2: r = r * n.
  + Bước 3: n = n – 1, đến bước 2.
  + Bước 4: kết thúc.

11

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

Các bước tính n! như trên không có điểm dừng. Vì vậy, các bước tính toán trên không phải là một thuật toán.

12

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

+ **<u>Tính xác định</u>**: các thao tác tại mỗi bước phải được mô tả rõ ràng, chính xác.

Ví dụ: tìm số lớn nhất trong 3 số a, b, c

* Đầu vào: các số thực a, b, c.
* Đầu ra: số thực m là giá trị lớn nhất của a, b, c
* Các bước thực hiện:
  + Bước 1: chọn đại m.
  + Bước 2: so sánh m với một trong hai số còn lại, lấy số lớn.
  + Bước 3: so sánh m với số còn lại, lấy số lớn.
  + Bước 4: kết thúc.

13

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

Các bước thực hiện không được mô tả rõ ràng, chính xác. Vì vậy, các bước thực hiện này không phải là một thuật toán.

14

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

+ **Tính đúng đắn**: lời giải của thuật toán phải đáp thỏa yêu cầu của bài toán.

Ví dụ: tìm giá trị nhỏ nhất của 3 số thực a, b, c.

* Đầu vào: các số thực a, b, c.
* Đầu ra: số thực m là giá trị nhỏ nhất của a, b, c.
* Các bước thực hiện:
  + Bước 1: gán số thực m = a.
  + Bước 2: nếu m < b thì gán m = b.
  + Bước 3: nếu m < c thì gán m = c.
  + Bước 4: kết thúc.

15

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

Các bước thực hiện trên trả về là giá trị lớn nhất của 3 số thực a, b, c. Vì vậy, các bước thực hiện này không phải là thuật toán cho bài toán tìm giá trị nhỏ nhất của 3 số thực a, b, c.

16

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

+ **Tính phổ dụng**: có thể áp dụng để giải các bài toán trong lớp bài toán đó (lớp bài toán là các bài toán cùng dạng nhưng khác tham số).

Ví dụ tìm nghiệm của phương trình bậc hai

$$ax^2 + bx + c = 0.$$

17

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

* Đầu vào: các số thực a, b, c là các hệ số của phương trình.
* Đầu ra: các số thực x_1, x_2 là nghiệm (nếu có).
* Các bước thực hiện:
  + Bước 1: nếu $a + b + c \neq 0$, đến bước 2; ngược lại, tính
  $$x_1 = 1, x_2 = c/a.$$
  + Bước 2: nếu $a - b + c \neq 0$, đến bước 3; ngược lại, tính
  $$x_1 = -1, x_2 = -c/a.$$
  + Bước 3: phương trình vô nghiệm.
  + Bước 4: kết thúc

18

THUẬT TOÁN

❖CÁC ĐẶC TRƯNG CỦA THUẬT TOÁN

Các bước thực hiện trên chỉ giải được phương trình bậc 2 trong trường hợp $a \neq 0$ có $a + b + c = 0$ hoặc $a - b + c = 0$ là hai trường hợp đặc biệt. Vì vậy, các bước thực hiện trên không phải là thuật toán cho bài toán tìm nghiệm phương trình bậc 2.

19

BIỂU DIỄN THUẬT TOÁN

❖SƠ ĐỒ KHỐI

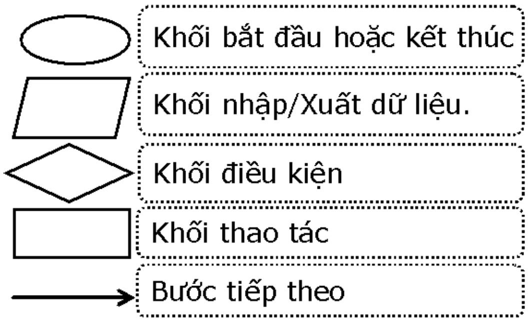
❖MÃ GIẢ

20

BIỂU DIỄN THUẬT TOÁN

❖SƠ ĐỒ KHỐI

- Sử dụng để trình bày thuật giải theo một hình thức thống nhất.
- Sơ đồ khối gồm có các thành phần sau:

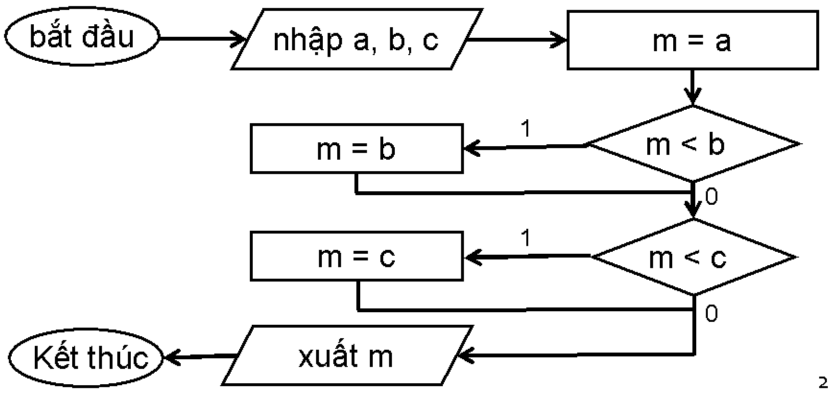


21

BIỂU DIỄN THUẬT TOÁN

❖SƠ ĐỒ KHỐI

Ví dụ 3: thuật giải tìm số lớn nhất trong 3 số a, b, c.



22

BIỂU DIỄN THUẬT TOÁN

❖MÃ GIẢ

- Mã giả là sử dụng một ngôn ngữ vay mượn ngôn ngữ lập trình nào đó để biểu diễn thuật toán.

Ví dụ 4: thuật giải tìm số lớn nhất trong 3 số a, b, c

* Đầu vào: 3 số a, b, c

* Đầu ra: m là số lớn nhất của a, b, c

* Các bước thực hiện

```
m = a
if (m < b) m = b
if (m < c) m = c
```

23

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖KHÁI NIỆM

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

24

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖KHÁI NIỆM

- Độ phức tạp của thuật toán là tốc độ tăng chi phí cho việc thực hiện thuật toán dựa trên tốc độ tăng số lượng giá trị đầu vào. Chi phi cho việc thực hiện thuật toán gồm 2 dạng: chi phí về thời gian và chi phí về không gian.

  + Chi phí về thời gian: là số lượng các phép tính mà thuật toán sẽ thực hiện (độc lập với phần cứng).

  + Chi phí về không gian: kích thước bộ nhớ để lưu trữ các dữ liệu sử dụng khi thực hiện thuật toán.

25

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖KHÁI NIỆM

<u>Ví dụ 5</u>: giả sử một thuật toán có độ phức tạp về thời gian theo kích thước đầu vào n được ước lượng là n^2, điều đó có nghĩa là số lượng phép toán có tốc độ tăng theo hàm số n^2 nếu kích thước đầu vào tăng lên n.

26

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖KHÁI NIỆM

- Độ phức tạp tính toán được dùng để so sánh giữa hai thuật toán của một bài toán nhằm chọn lựa thuật toán hiệu quả hơn.
- Chi phí về không gian phụ thuộc vào cấu trúc dữ liệu đặc trưng để thực hiện thuật toán nên trong phạm vi môn học này chỉ xét đến chi phí về thời gian.
- Giữa chi phí về thời gian và chi phí không gian có mối quan hệ tỉ lệ nghịch. Một thuật toán nếu giảm được chi phí về thời gian thì phải tăng chi phí về không gian, và ngược lại.

27

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

Cho n là kích thước dữ liệu đầu vào, số lượng phép tính của một thuật toán được tính theo các quy tắc sau:

- Số phép tính cho một công việc đơn giản A (gán, so sánh, các phép toán số học) là

$$f_A(n) = 1.$$

- Số phép tính cho một công việc C gồm 2 công việc liên tiếp A và B là:

$$f_C(n) = f_A(n) + f_B(n)$$

28

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

- Số phép tính cho một công việc C chứa hai công việc được lựa chọn A và B là:

$$f_A(n) = \max(f_B(n), f_C(n));$$

Ví dụ 6:

A = { a = 100; }, $f_A(n) = 1$.

B = { a = a + 12; }, $f_B(n) = 1$.

C = { a = 100; a = a + 12; }, $f_C(n) = f_A(n) + f_B(n) = 2$.

29

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

Ví dụ 7:

```
D =   {   if a = 10
              a = 100;
          else {
              a = 100;
              a = a + 12;
          }
      }
```

$f_D(n) = \max(f_A(n), f_C(n)) + 1 = 3$.

30

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

Ví dụ 8: tính số lượng phép tính của đoạn chương trình A sau:

```
int s = 0;
for (i =1; i <= n; i++) {
   s = s + i;
}
```

31

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

Các phép tính thực hiện gồm:

- 1 phép gán s = 0

→ số lượng phép tính: 1

- liên tiếp n lần phép tính s = s + I

→ số lượng phép tính: 1+..+1 = n

→ Số lượng phép tính của đoạn chương trình: n+1.

32

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

Ví dụ 9: tính số lượng phép tính của đoạn chương trình A sau:

```
for (i =1; i < n; i++)
   for (j=i+1; j<=n; j++)
       if (a[i] > a[j]) {
           int tmp = a[i];
           a[i] = a[j];     a[j] = a[i];
       }
```

33

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

- Đặt đoạn chương trình C là

```
       if (a[i] > a[j]) {
           int tmp = a[i];
           a[i] = a[j];     a[j] = a[i];
       }
```

- Đặt đoạn chương trình B là

```
   for (j=i+1; j<=n; j++)
       C
```

34

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖TÍNH SỐ LƯỢNG PHÉP TÍNH

- Số lượng phép tính của C là $f_C(n) = 4$.
- Số lượng phép tính của B là $f_B(n) = (n-i).f_C(n) = 4(n-i)$.
- Số lượng phép tính của A là

$$
\begin{aligned}
f_A(n) &= 4(n-1) + 4(n-2) + \ldots + 4.1 \\
&= 4[(n-1) + (n-2) + \ldots + 1] = 4.n.(n-1)/2 \\
&= 2.(n^2+n)
\end{aligned}
$$

35

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

Cho một thuật toán A thực hiện trên dữ liệu đầu vào có kích thước n. Chi phí về thời gian để thực hiện giải thuật này là f(n). Khi đó, độ phức tạp tính toán của thuật toán có thể được ước lượng theo các bậc như sau:

* **Bậc Big-O (Ô lớn)**

- Độ phức tạp của thuật toán A là O(g(n)) nếu tồn tại các hằng số $c>0$, $n_0>0$ sao cho

$$f(n) \le c.g(n) \ \forall\ n \ge n_0$$

36

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

Ví dụ 10: với đoạn chương trình A trong ví dụ 9, ta có:

$$f_A(n) = 2(n^2 - n)$$

Xét hàm số $g(n) = n^2$

Với c = 2, n_0=1, ta có:

$$2 \cdot g(n) = 2n^2 \geq f_A(n)$$

Vậy, thuật toán A có độ phức tạp tính toán là $O(n^2)$

37

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

- Bậc Big-O thường được sử dụng trong đánh giá độ phức tạp tính toán của thuật toán để ước lượng độ phức tạp không thể cao hơn của thuật toán. Hàm g(n) được gọi là giới hạn trên của f(n).
- Lưu ý: nếu độ phức tạp tính toán của thuật toán A là O(g(n)), thì độ phức tạp của thuật toán A cũng là O(h(n)) với h(n) là giới hạn trên của g(n)

Ví dụ 11: giả sử thuật toán A có số phép tính là $f(n)=n^2+2n+1$, khi đó, độ phức tạp tính toán của A có thể là $O(n^2)$, $O(n^3)$, $O(2^n)$, $O(n^n)$,...

38

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

* **Bậc Big-Ω (Ômêga lớn)**

 - Độ phức tạp của thuật toán A là Ω(g(n)) nếu tồn tại các hằng số c>0, n_0>0 sao cho

$$f(n) \geq c.g(n) \ \forall \ n \geq n_0$$

39

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

Ví dụ 12: với đoạn chương trình A trong ví dụ 9, ta có:

$$f_A(n) = 2(n^2 - n)$$

Xét hàm số $g(n) = n^2$

Với c=1, n_0=2, ta có:

$$g(n) = 2\left(n^2 - \frac{1}{2}n^2\right) \leq f_A(n)$$

Vậy, thuật toán A có độ phức tạp tính toán là Ω(n^2)

40

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

- Bậc Big-Ω dùng để ước lượng độ phức tạp không thể thấp hơn của thuật toán. Hàm g(n) được gọi là giới hạn dưới của f(n).
- Lưu ý: nếu độ phức tạp tính toán của thuật toán A là $\Omega(g(n))$, thì độ phức tạp của thuật toán A cũng là $\Omega(h(n))$ với h(n) là giới hạn dưới của g(n).

<u>Ví dụ 11</u>: giả sử thuật toán A có số phép tính là

$f(n)=n^n + n^2 + 1$, khi đó, độ phức tạp tính toán của A có thể là $\Omega(n^n)$, $\Omega(2^n)$, $\Omega(n^3)$, $\Omega(n^2)$.

41

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

* **<u>Bậc Big-Θ (Têta lớn)</u>**

- Độ phức tạp của thuật toán A là $\Theta(g(n))$ nếu tồn tại các hằng số $c_0>0$, $c_1>0$, $n_0>0$ sao cho

$$C_0.g(n) \le f(n) \le c_1.g(n) \ \forall \ n \ge n_0$$

42

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

Ví dụ 13: với đoạn chương trình A trong ví dụ 9, ta có:

$$f_A(n) = 2(n^2 - n)$$

Xét hàm số $g(n) = n^2$

Với $c_0 = 1$, $c_1 = 2$, $n_0 = 2$, ta có:

$$g(n) \leq f_A(n) \leq 2 \cdot g(n)$$

Vậy, thuật toán A có độ phức tạp tính toán là $\Theta(n^2)$

43

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

- Bậc Big-Θ dùng để ước lượng độ phức tạp tương đương của thuật toán. Hàm g(n) được gọi là giới hạn chặt của f(n).

44

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

* **Một số tính chất chung**

- Nếu thuật toán A có số phép tính dựa trên kích thước đầu vào n là một đa thức P(n) bậc k, khi đó, độ phức tạp tính toán của A là có thể là $O(n^k)$, $\Omega(n^k)$, $\Theta(n^k)$.
- Nếu thuật toán A có số phép tính dựa trên kích thước đầu vào n là $\log_a f(n)$, do $\log_a f(n)=\log_a b.\log_b f(n)$ nên độ phức tạp tính toán của A có thể ghi là O(log f(n)) mà không cần ghi cơ số. Điều này cũng đúng với bậc Ω và Θ.

45

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

* **Nhận xét:**

- Trong các ký pháp để đánh giá thuật toán, Bậc Big-Θ thể hiện độ phức tạp tính toán tốt nhất
- Trong nhiều trường hợp không thể xác định được giới hạn chặt của hàm số biểu diễn số lượng phép tính của thuật toán, người ta thường dùng bậc Big-O để thể hiện độ phức tạp tính toán.
- Khi đánh giá thuật toán, người ta sử dụng hàm giới hạn g(n) đơn giản và sát với f(n) nhất có thể

46

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖CÁC BẬC ĐÁNH GIÁ THUẬT TOÁN

Các độ phức tạp thường dùng được xếp theo thứ tự tốc độ tiến đến vô cùng nhanh dần

Độ phức tạp	Tên gọi độ phức tạp tương ứng
O(1)	Độ phức tạp hằng số
O(log(n))	Độ phức tạp logarit
O(n)	Độ phức tạp tuyến tính
$O(n^k)$	Độ phức tạp đa thức
$O(k^n)$	Độ phức tạp hàm mũ
O(n!)	Độ phức tạp giai thừa

47

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

❖BÀI TẬP

Thuật toán A1 và A2 dùng để xác định giá trị x có tìm được trong dãy a[n] hay không, trong đó a[n] là dãy số được sắp theo thứ tự tăng dần. A1 và A2 được cho bởi hai đoạn chương trình sau. Cho biết thuật toán nào tốt hơn.

Thuật toán A1

```
for (int i = 0; i < n; i++) {
    if (x == a[i]) printf("tim thay");
}
```

48

Thuật toán A2

```
r = n;
   while (l < r) {
           mid = (l+r)/2;
           if (x == a[mid]) {
               printf("tim thay");    l = r;
           } else {
               if (x > a[mid])
                   l = mid + 1;
               else
                   r = mid - 1;
           }
   }
```

49