BÀI 2

CÁC KIẾN THỨC CƠ BẢN

Vũ Thương Huyền

huyenvt@tlu.edu.vn

NỘI DUNG

- **Hàm**
- **Độ tăng của hàm**
- **Thuật toán**
- **Độ phức tạp của thuật toán**

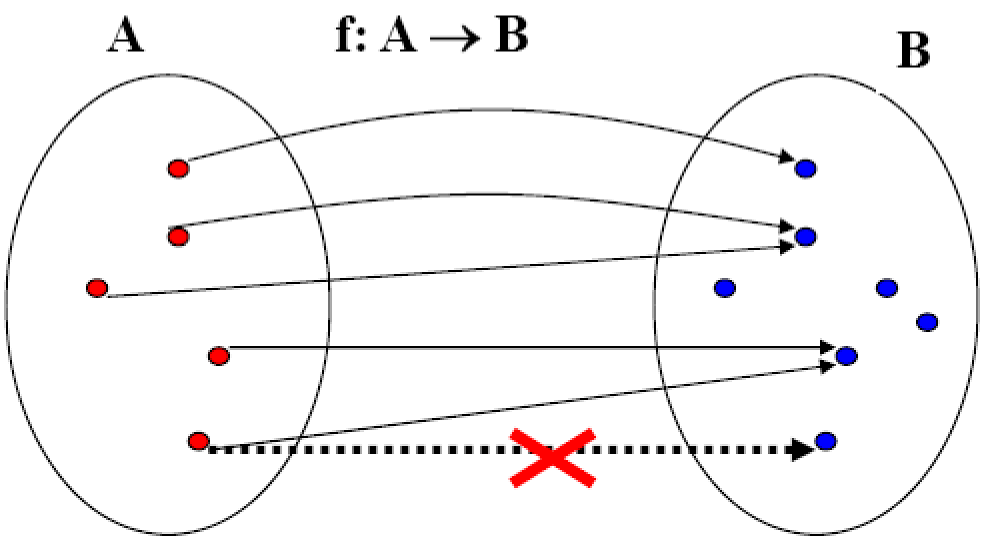
2.1 HÀM

1.8 HÀM

- Dùng để định nghĩa các cấu trúc rời rạc như dãy, xâu
- Dùng để biểu diễn thời gian một máy tính phải mất để giải một bài toán

Định nghĩa 1:

Cho A và B là hai tập hợp. Một ***hàm f*** từ A đến B là sự gán chính xác một phần tử của B cho mỗi phần tử của A. Ta viết $\boldsymbol{f(a) = b}$ nếu b là phần tử duy nhất của B được gán bởi hàm f cho phần tử a của A. Nếu f là hàm từ A đến B ta viết: $\boldsymbol{f: A \rightarrow B}$.



1.8 HÀM

Định nghĩa 2:

Nếu *f* là một hàm từ A đến B.

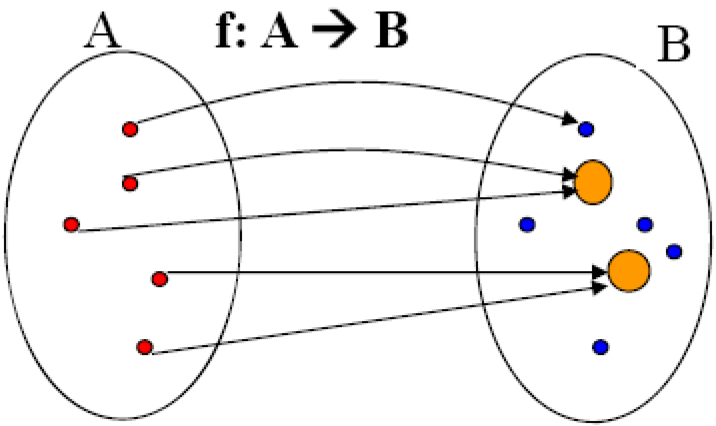
- A được gọi là **miền xác định** của *f* và B là **miền giá trị** của *f*.
- Nếu *f(a)* = *b*, *b* gọi là **ảnh** của *a* và *a* là **một nghịch ảnh** của *b*.
- Tập ánh xạ qua hàm *f* là tập các ảnh của các phần tử thuộc A
- *f* **ánh xạ** A đến B

Ví dụ: Cho A= {1, 2, 3}, B ={a, b, c}

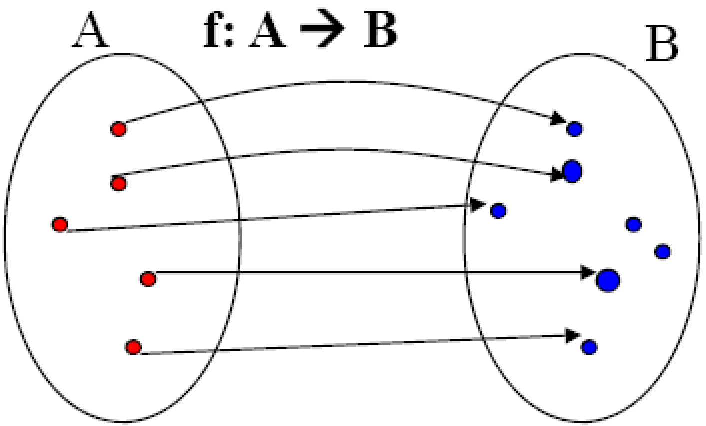
- Hàm *f* được định nghĩa: $1 \rightarrow c, 2 \rightarrow a, 3 \rightarrow c$
- $1 \rightarrow c$, c là ***ảnh*** của 1
- $2 \rightarrow a$, 2 là ***nghịch ảnh*** của a
- ***Miền xác định*** của f {1, 2, 3}, ***miền giá trị*** của f {a, b, c}
- ***Tập ánh xạ*** f {a, c}

Định nghĩa 5:

Một hàm f được gọi là ***đơn ánh*** hay ánh xạ ***một-một*** nếu và chỉ nếu $f(x) = f(y)$ kéo theo $x = y$ với mọi x và y trong miền xác định của f.



Không đơn ánh



Đơn ánh

Các hàm sau có là hàm đơn ánh không?

Ví dụ 1:

- Cho A = {1, 2, 3} và B = {a, b, c}, hàm *f* được cho như sau:
- $1 \rightarrow c, 2 \rightarrow a, 3 \rightarrow c$

Ví dụ 2:

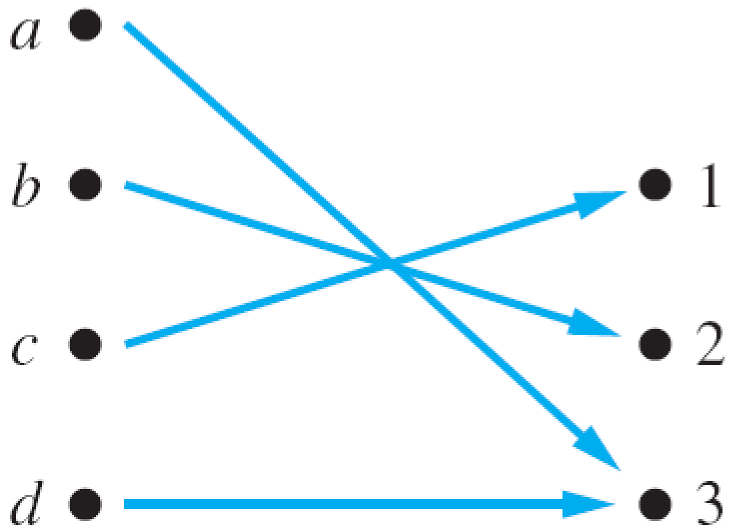
- Cho g: $Z \rightarrow Z$, với $g(x) = 2x - 1$

Ví dụ 3:

- Hàm $f(x) = x^2$, x thuộc tập các số nguyên, miền giá trị của *f* cũng là tập các số nguyên.

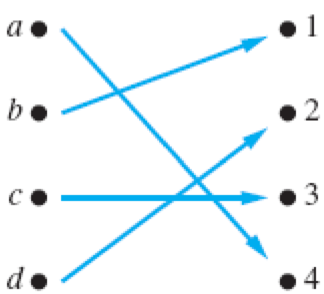
Định nghĩa 7:

Một hàm f từ A đến B được gọi là ***toàn ánh*** nếu và chỉ nếu với mọi phần tử $b \in B$ tồn tại một phần tử $a \in A$, với $f(a) = b$.

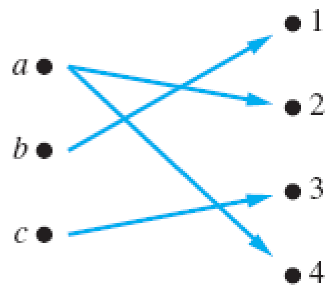


Định nghĩa 8:

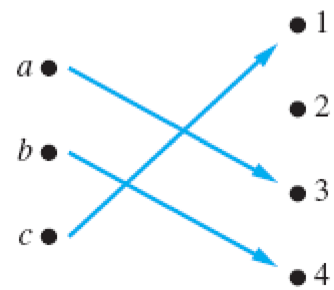
Một hàm f là một ***song ánh*** nếu nó vừa là *đơn ánh* vừa là *toàn ánh*.



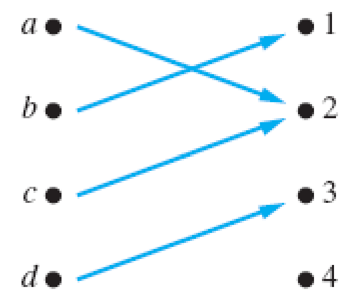
(1)?



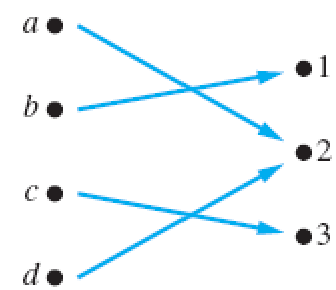
(2)?



(3)?



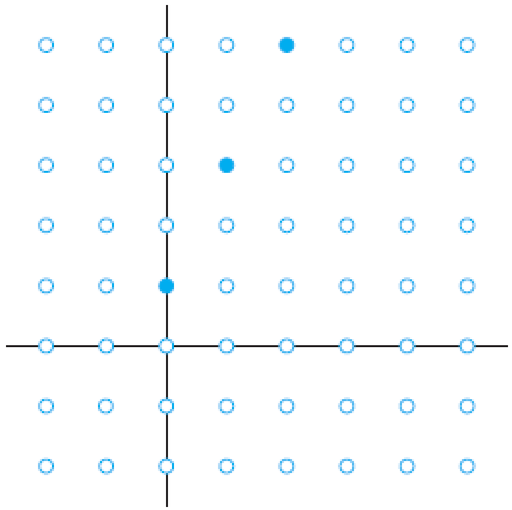
(4)?



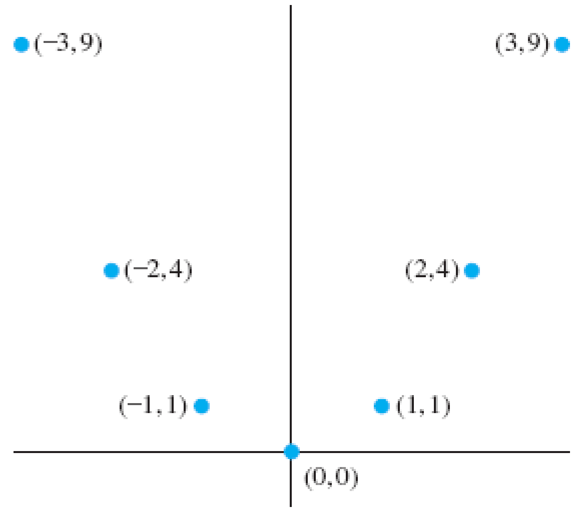
(5)?

Định nghĩa 11:

Cho f là hàm từ tập A đến tập B. ***Đồ thị*** của hàm f là tập các cặp sắp thứ tự $\{(\boldsymbol{a}, \boldsymbol{b}) |\ \boldsymbol{a} \in \boldsymbol{A}$ và $\boldsymbol{f}(\boldsymbol{a}) = \boldsymbol{b}\}$.



$f(x) = 2x + 1$



$f(x) = x^2$

Một số hàm quan trọng:

- Hàm sàn
- Hàm trần

Định nghĩa 12:

Hàm sàn gán cho số thực *x* số nguyên lớn nhất có giá trị nhỏ hơn hoặc bằng *x*. Giá trị của hàm sàn được kí hiệu $\lfloor x \rfloor$. ***Hàm trần*** gán cho số thực *x* số nguyên nhỏ nhất có giá trị lớn hơn hoặc bằng *x*. Giá trị của hàm trần được kí hiệu là $\lceil x \rceil$.

Ví dụ:

- $\lfloor 2,1 \rfloor = ?$
- $\lceil 2,1 \rceil = ?$
- $\lfloor -2,1 \rfloor = ?$
- $\lceil -2,1 \rceil = ?$

BÀI TẬP

- **Bài 1**: Hãy xác định xem hàm *f*: $Z \times Z \to Z$ có toàn ánh không?

  a) $f(m,n) = 2m - n$ b) $f(m,n) = |m| - |n|$

- **Bài 2**: Hãy xác định xem hàm *f*: $R \to R$ có song ánh không?

  a) $f(x) = -3x + 4$ b) $f(x) = \frac{(x+1)}{(x+2)}$

2.2 ĐỘ TĂNG CỦA HÀM

Đánh giá thuật toán như thế nào?

- Thời gian đòi hỏi để giải một bài toán phụ thuộc vào số phép toán được sử dụng
- Ước lượng thời gian bằng cách nhân thời gian đòi hỏi với một hằng số.
- Sử dụng khái niệm big-O: đánh giá số phép toán được dùng trong một thuật toán khi đầu vào của nó tăng

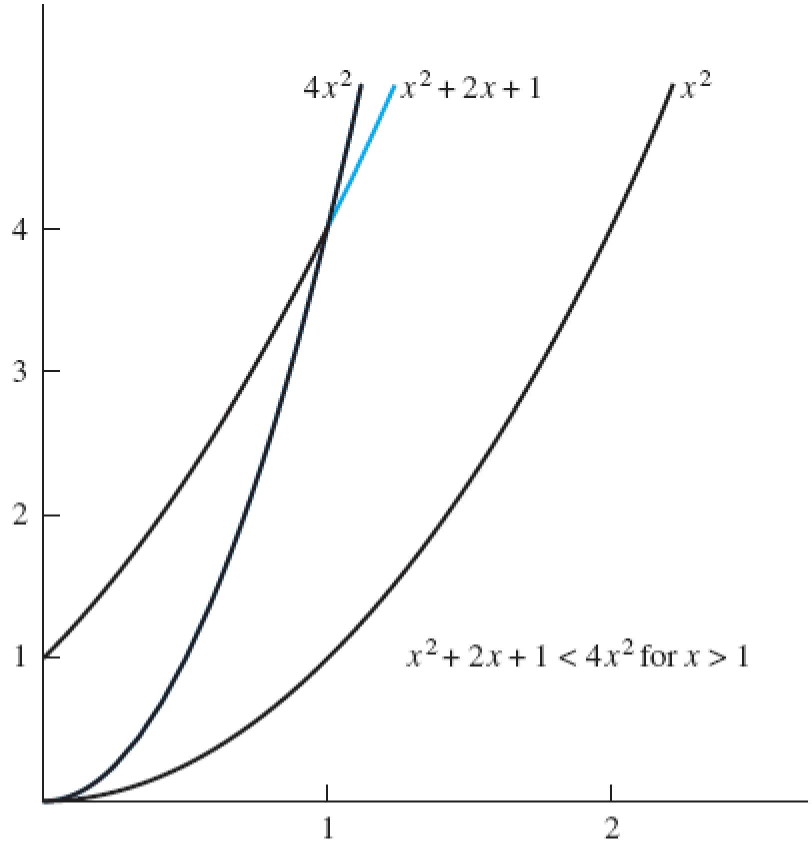
Định nghĩa 1:

Cho hàm *f* và *g* là hai hàm từ tập các số nguyên hoặc số thực đến tập các số thực. Ta nói *f(x)* là ***O*(*g*(*x*))** (đọc là *f(x)* là big-O của *g(x)*) nếu tồn tại hai hằng số *C* và *k* sao cho:

$$|f(x)| \leq C\,|g(x)|,$$

với mọi $x>k$

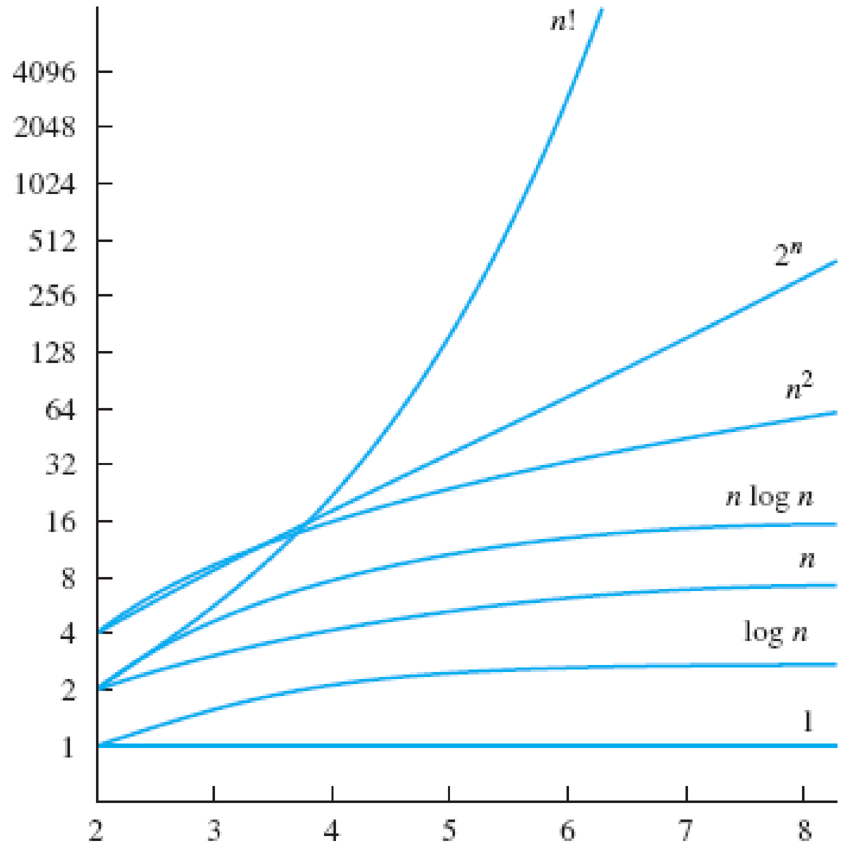
Ví dụ : Chứng minh rằng $f(x) = x^2 + 2x + 1$ là $O(x^2)$



Định lí 1:

Cho $f(x) = \mathbf{a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0}$, ở đây $a_0, a_1, \ldots, a_n$ là các số thực. Khi đó $f(x)$ là $\mathbf{O(x^n)}$.

- $\mathbf{1 + 2 + \ldots + n}$ là $O(n^2)$
- $\mathbf{n!}$ là $O(n^n)$
- $\mathbf{logn!}$ là $O(nlogn)$



2.2 ĐỘ TĂNG CỦA TỔ HỢP CÁC HÀM

Định lí 2:

Cho $f_1(x)$ là $O(g_1(x))$ *và* $f_2(x)$ *là* $O(g_2(x))$. Khi đó $\boldsymbol{(f_1 + f_2)(x)}$ *là* $\boldsymbol{O(max(|g_1(x)|, |g_2(x)|))}$.

Hệ quả 1:

Cho $f_1(x)$ và $f_2(x)$ đều là $O(g(x))$. Khi đó $\boldsymbol{(f_1 + f_2)(x)}$ *là* $\boldsymbol{O(g(x))}$.

Định lí 3:

Cho $f_1(x)$ là $O(g_1(x))$ *và* $f_2(x)$ *là* $O(g_2(x))$. Khi đó $\boldsymbol{(f_1 f_2)(x)}$ *là* $\boldsymbol{O(g_1(x)\, g_2(x))}$.

2.2 ĐỘ TĂNG CỦA TỔ HỢP CÁC HÀM

Ví dụ 1 : Cho một đánh giá big-O đối với hàm:

$$f(n) = 3n\log(n!) + (n^2 + 3)\log n$$

Ví dụ 2 : Cho một đánh giá big-O đối với hàm:

$$f(x) = (x+1)\log(x^2 + 1) + 3x^2$$

BÀI TẬP

- **Bài 3**: Với các hàm $g(x)$ sau đây x^3 có là $O(g(x))$ không:

  a) $g(x) = x^2$

  b) $g(x) = x^3$

  c) $g(x) = x^2 + x^3$

KHÁI NIỆM BIG-OMEGA VÀ BIG-THETA

Định nghĩa 2:

Cho f và g là hai hàm từ tập các số nguyên hoặc tập các số thực đến tập các số thực. Nói rằng $f(x)$ là $\mathbf{\Omega(g(x))}$ nếu và chỉ nếu tồn tại các hằng số C và k, sao cho:

$$|f(x)| \geq C|g(x)| \text{ với mọi } x > k$$

Ví dụ: Hàm $f(x) = 8x^3 + 5x^2 + 7$ là $\Omega(g(x))$, với $g(x) = x^3$.

KHÁI NIỆM BIG-OMEGA VÀ BIG-THETA

Định nghĩa 3:

Cho f và g là hai hàm từ tập các số nguyên hoặc tập các số thực đến tập các số thực. Nói rằng $f(x)$ là $\boldsymbol{\Theta(g(x))}$ nếu và chỉ nếu $f(x)$ là $O(g(x))$ và $f(x)$ là $\Omega(g(x))$. Khi $f(x)$ là $\boldsymbol{\Theta(g(x))}$ ta nói rằng $f(x)$ là big-Theta của $g(x)$ và $f(x)$ ***cùng bậc*** với $g(x)$.

Định lí 4:

Cho $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, trong đó $a_0, a_1, \ldots, a_n$ là các số thực với $a_n \neq 0$. Khi đó $f(x)$ cùng bậc với x^n.

BÀI TẬP

- **Bài 4**: Chứng minh rằng:

  a) $3x + 7$ là $\boldsymbol{\Theta(x)}$

  b) $2x^2 + x - 7$ là $\boldsymbol{\Theta(x^2)}$

  c) $log_{10}(x)$ là $\boldsymbol{\Theta(log_2\,(x))}$

2.3 THUẬT TOÁN

2.1 THUẬT TOÁN

Định nghĩa 1:

Thuật toán là tập hợp hữu hạn các lệnh chính xác để thực hiện tính toán hoặc giải một bài toán.

Tính chất của thuật toán

- Đầu vào
- Đầu ra
- Tính xác định
- Tính đúng đắn
- Tính hữu hạn
- Tính hiệu quả
- Tính tổng quát

Mô tả thuật toán

- Dùng ngôn ngữ tự nhiên
- Dùng giả mã
- Sử dụng lưu đồ
- Sử dụng ngôn ngữ lập trình

Ví dụ :

THUẬT TOÁN : Tìm phần tử lớn nhất trong dãy hữu hạn

Procedure $max(a_1, a_2, \ldots a_n$: số nguyên)
$max := a_1$
for $i := 2$ **to** n
  if $max < a_i$ **then** $max := a_i$
{ max là phần tử lớn nhất}

2.1 CÁC THUẬT TOÁN TÌM KIẾM

- **Tìm kiếm** là bài toán xác định vị trí của một phần tử trong bảng liệt kê
- Tổng quát: xác định vị trí x trong dãy $a_1, a_2, a_3, \ldots a_n$

- 2 loại thuật toán tìm kiếm:
  - **Tìm kiếm tuyến tính**
  - **Tìm kiếm nhị phân**

Tìm kiếm tuyến tính

- So sánh x với a_1, nếu $x = a_1$ thì vị trí tìm được là 1
- Khi $x \neq a_1$ so sánh x *với* a_2
- ……

THUẬT TOÁN : Thuật toán tìm kiếm tuyến tính

Procedure *linear search* (x: nguyên, a_1, a_2, ... a_n: các số nguyên phân biệt)
$i := 1$
while ($i \leq n$ và $x \neq a_i$)
        $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* **:= *0***
{ *location* là chỉ số của số hạng bằng x hoặc là 0 nếu không tìm được x}

Tìm kiếm nhị phân

- Sử dụng cho dãy đã sắp xếp tăng dần
- So sánh phần tử *x* với số hạng ở giữa của dãy, nếu bằng thì trả về vị trí cần tìm
- Nếu *x* nhỏ hơn tìm bên trái dãy
- Nếu *x* lớn hơn tìm bên phải dãy

Ví dụ :

- Tìm kiếm giá trị **15** trong dãy:

1 3 5 6 8 9 10 15 24 39 40

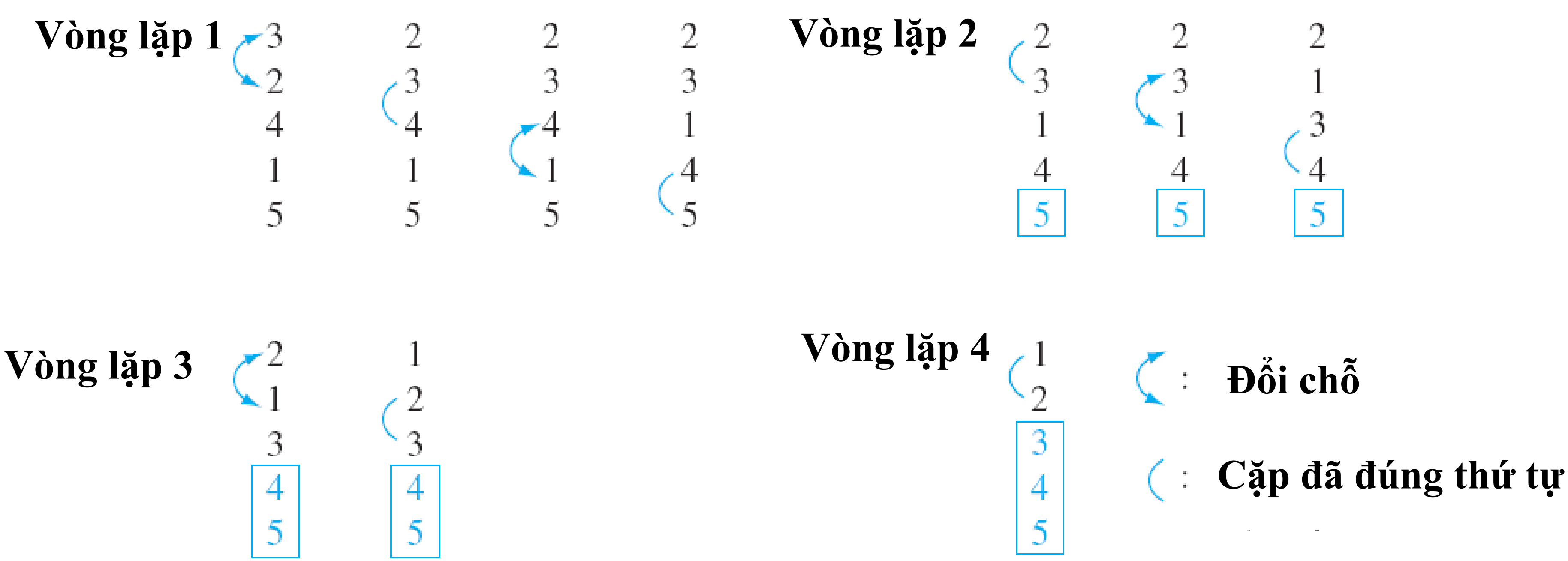
THUẬT TOÁN : **Thuật toán tìm kiếm nhị phân**

Procedure *binary search* (*x*: nguyên, a_1, a_2, ... a_n: các số nguyên tăng dần)
i := 1 {*i* là điểm mút trái của khoảng tìm kiếm}
j := n {*j* là điểm mút phải của khoảng tìm kiếm}
while $i < j$
begin
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
end
if $x = a$ **then** *location* :=*i*
else *location* :=*0*
{ *location* là chỉ số của số hạng bằng *x* hoặc là 0 nếu không tìm được *x*}

Sắp xếp kiểu nổi bọt

- So sánh liên tiếp các phần tử kề nhau
- Đổi chỗ cho nhau nếu chúng chưa có thứ tự đúng

Ví dụ : • Sắp xếp danh sách 3, 2, 4, 1, 5.



THUẬT TOÁN : Thuật toán sắp xếp nổi bọt

Procedure *bubble sort* $(a_1, a_2, \ldots a_n)$

for $i := 1$ **to** $n - 1$

    for $j := 1$ **to** $n - i$

        if $a_j > a_{j+1}$ **then** đổi chỗ a_j và a_{j+1}

$\{a_1, a_2, \ldots, a_n$ đã được sắp xếp$\}$

Sắp xếp kiểu chèn

- Bắt đầu với phần tử thứ 2
- So sánh phần tử thứ 2 với phần tử thứ nhất:
  - Chèn vào trước phần tử thứ nhất nếu nhỏ hơn hoặc bằng
  - Chèn vào sau phần tử thứ nhất nếu lớn hơn
- So sánh phần tử thứ 3 với phần tử thứ nhất và so sánh tiếp với phần tử thứ 2.

Ví dụ :

- Sắp xếp danh sách **3, 2, 4, 1, 5.**

THUẬT TOÁN : Thuật toán sắp xếp kiểu chèn

Procedure *insertion sort* (a_1, a_2, ... $a_{n:}$ các số thực với $n \geq 2$)
for $j := 2$ **to** n
begin
    $i := 1$
    while $a_i < a_j$
        $i := i + 1$
    $m := a_j$
    for $k := 0$ **to** $j - i - 1$
        $a_{j-k} := a_{j-k-1}$
    $a_i := m$
end {a_1, a_2, ..., a_n đã được sắp xếp}

BÀI TẬP

- **Bài 2**: Sắp xếp danh sách **6, 2, 3, 1, 5, 4** theo thứ tự ***tăng dần*** bằng phương pháp:

    a) Sắp xếp kiểu nổi bọt

    b) Sắp xếp kiểu chèn

    c) Sắp xếp kiểu lựa chọn (tham khảo trong sách)

    d) Sắp xếp kiểu chèn nhị phân (tham khảo trong sách)

2.4 ĐỘ PHỨC TẠP THUẬT TOÁN

2.3 ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Hiệu quả của một thuật toán:

- Thời gian mà máy tính sử dụng để giải bài toán
- Dung lượng bộ nhớ đòi hỏi khi thực hiện thuật toán

Độ phức tạp thời gian:

- Biểu diễn qua số các phép toán được dùng trong thuật toán
- Các phép toán để đo:
    - Phép so sánh
    - Phép cộng, trừ, nhân, chia

Ví dụ: Độ phức tạp thời gian của thuật toán tìm kiếm phần tử lớn nhất là $\Theta(n)$

2.3 ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Độ phức tạp trong trường hợp xấu nhất:

- Là trường hợp phải dùng tối đa các phép toán để giải bài toán theo thuật toán đang xét.

Độ phức tạp trong trường hợp trung bình:

- Tìm số bước trung bình các phép toán được dùng để giải toàn bộ các giá trị đầu vào
- Phức tạp hơn phân tích trong trường hợp xấu nhất

2.3 ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Ví dụ 1: Xác định độ phức tạp trong trường hợp xấu nhất của thuật toán **sắp xếp kiểu nổi bọt** qua số các phép so sánh

Ví dụ 2: Xác định độ phức tạp trong trường hợp xấu nhất của thuật toán **sắp xếp kiểu chèn** qua số các phép so sánh

Các thuật ngữ thường dùng cho độ phức tạp tính toán

Độ phức tạp	Thuật ngữ
$O(1)$	Độ phức tạp hằng số
$O(\log n)$	Độ phức tạp logarit
$O(n)$	Độ phức tạp tuyến tính
$O(n\log n)$	Độ phức tạp nlogn
$O(n^b)$	Độ phức tạp đa thức
$O(b^n)$	Độ phức tạp hàm mũ
$O(n!)$	Độ phức tạp giai thừa

Các thuật ngữ thường dùng cho độ phức tạp tính toán

Kích thước bài toán	**Các phép toán bit được sử dụng**					
n	***logn***	***n***	***nlogn***	n^2	2^n	***n!***
10	3.10^{-9}s	10^{-8}s	3.10^{-8}s	10^{-7}s	10^{-6}s	3.10^{-3}s
10^2	7.10^{-9}s	10^{-7}s	7.10^{-7}s	10^{-5}s	4.10^{13} năm	*
10^3	10^{-8}s	10^{-6}s	10^{-5}s	10^{-3}s	*	*
10^4	$1.3*10^{-9}$s	10^{-5}s	10^{-4}s	10^{-1}s	*	*
10^5	$1.7*10^{-8}$s	10^{-4}s	$2*10^{-3}$s	10s	*	*
10^6	$2*10^{-8}$s	10^{-3}s	$2*10^{-2}$s	17 phút	*	*