Thuật toán (Algorithm)

Thuật toán: *Định nghĩa*

"*Một **thuật toán** là một chuỗi các bước (chỉ dẫn) chính xác để thực hiện một phép tính toán hoặc giải quyết một bài toán.*"

- ***Ví dụ***: *Thuật toán tìm phần tử lớn nhất trong một chuỗi hữu hạn*

procedure max($a_1, a_2, \ldots, a_n$: *các số nguyên*)

$max := a_1$

for $i := 2$ **to** n

if $max < a_i$ **then** $max := a_i$

return max {max là phần tử lớn nhất}

Thuật toán: *Các đặc trưng*

- **Đầu vào (Input)**: Thuật toán có các giá trị đầu vào thuộc một tập xác định.
- **Đầu ra (Output)**: từ mỗi tập giá trị đầu vào, thuật toán tạo ra các giá trị đầu ra thuộc một tập xác định. Các giá trị này có thể là lời giải cho bài toán mà thuật toán đang giải quyết.
- **Tính xác định (Definiteness)**: Các bước của thuật toán được xác định một cách chính xác.
- **Tính chính xác (Correctness)**: Thuật toán cần tạo ra các kết quả chính xác với mỗi tập đầu vào.
- **Tính hữu hạn (Finiteness)**: Thuật toán cần tạo ra các giá trị đầu ra sau một số hữu hạn (có thể lớn) các bước đối với mọi tập đầu vào.
- **Tính hiệu quả (Effectiveness)**: Các bước của thuật toán cần được thực hiện một cách chính xác trong một khoảng thời gian hữu hạn.
- **Tính tổng quát (Generality)**: Thuật toán cần khả dụng cho mọi bài toán thuộc cùng một dạng mong muốn, chứ không chỉ áp dụng cho một tập đặc biệt các giá trị đầu vào.

Thuật toán: *Ví dụ*

- *Thuật toán tìm kiếm tuyến tính*

procedure linearSearch$(x, a_1, a_2, \dots, a_n)$

$// \ x\text{: số nguyên}, a_1, a_2, \dots, a_n\text{: các số nguyên phân biệt})$

$i := 1$

while $(i \leq n$ **and** $x \neq a_i)$

$i := i + 1$

if $i \leq n$ **then** $location := i$

else $location := 0$

return $location$ {$location$ là chỉ số của số hạng bằng x, hoặc là 0 nếu không tìm thấy x}

Thuật toán: Ví dụ

- *Thuật toán tìm kiếm nhị phân*

procedure binarySearch($x, a_1, a_2, \dots, a_n$)

//x: *số nguyên*, $a_1, a_2, \dots, a_n$: *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 $i < j$

$m := \lfloor (i + j)/2 \rfloor$

if $x > a_m$ **then** $i := m + 1$

else $j := m$

if $x = a_i$ **then** $location := i$

else $location := 0$

return $location$ {$location$ là chỉ số dưới của số hạng bằng x, hoặc là 0 nếu không tìm thấy x}

Thuật toán: Ví dụ

- *Thuật toán sắp xếp nổi bọt*

procedure bubbleSort($a_1, \ldots, a_n$: *các số thực với* $n \geq 2$)

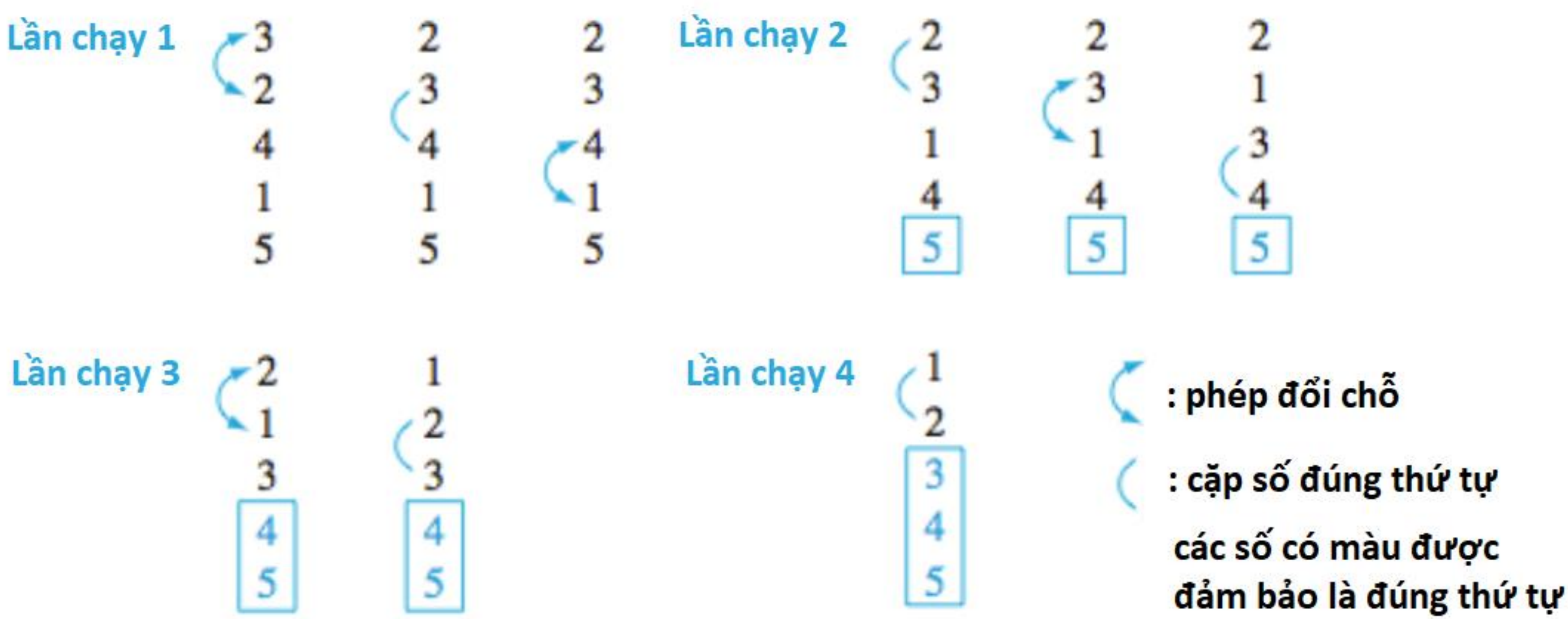
    for $i := 1$ **to** $n - 1$

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

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

    {$a_1, \ldots, a_n$ được sắp xếp giá trị tăng dần}

Thuật toán: Ví dụ



Hình 1: Các bước của thuật toán sắp xếp nổi bọt

Thuật toán: Ví dụ

- *Thuật toán sắp xếp chèn*

procedure insertionSort($a_1, a_2 \ldots, a_n$: *các số thực với* $n \geq 2$)
for j $:= 2$ **to** n
    $i := 1$
    while $a_j > a_i$
        $i := i + 1$
    $m := a_j$
    for $k := 0$ **to** $j - i - 1$
        $a_{j-k} := a_{j-k-1}$
    $a_i := m$
$\{a_1, \ldots, a_n$ được sắp xếp giá trị tăng dần$\}$

Thuật toán tham lam (Greedy algorithm)

- Giải quyết các vấn đề tối ưu hóa (Optimization problems)
  - Bài toán đổi tiền n xu
    - Dùng đồng 25 xu, 10 xu, 5 xu, 1 xu
    - Dùng ít đồng tiền nhất có thể
- Chọn giải pháp tốt nhất tại mỗi bước thay vì xem xét toàn bộ chuỗi các bước có thể cho lời giải tối ưu nhất

Thuật toán tham lam: Ví dụ

- *Thuật toán đổi tiền tham lam*

procedure change $(c_1, c_2 \dots, c_r, n)$

// $c_1, c_2 \dots, c_r$: $các\ mệnh\ giá\ của\ xu, với\ c_1 > c_2 > \cdots > c_r;\ n$: $số\ nguyên\ dương$

for $i := 1$ **to** r

$d_i := 0$ {d_i đếm số lượng xu mệnh giá c_i được sử dụng}

while $n \geq c_i$

$d_i := d_i + 1$ {thêm một xu mệnh giá c_i}

$n := n - c_i$

{d_i là số lượng xu thuộc mệnh giá c_i trong tiền đổi với $i = 1, 2, \dots, r$}

Thuật toán tham lam: Ví dụ

- Thảo luận trên lớp
  - Bổ đề 1 – Trang 199

Độ tăng của hàm

- **Khái niệm O-lớn (Big-O):** Cho f và g là các hàm từ tập số nguyên hoặc tập số thực đến tập số thực. Ta nói $f(x)$ là $O(g(x))$ nếu tồn tại hằng số C và k sao cho:

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

với mọi $x > k$. [Đọc là "$f(x)$ là O-lớn (big-O) theo $g(x)$."]

- **Chú ý:** Dễ thấy, $f(x)$ tăng chậm hơn một bội số cố định của $g(x)$ khi x tăng tới vô hạn.
  - C và k được gọi là "chứng cứ (*witnesses*)"

Độ tăng của hàm

- **Chú ý:** $f(x)$ là $O(g(x))$ có thể được viết là $f(x) = O(g(x))$.
  - Dấu bằng ở đây không thể hiện phép bằng thực sự.
  - Có thể viết là $f(x) \in O(g(x))$ bởi vì $O(g(x))$ biểu diễn một tập các hàm là $O(g(x))$.

- Khi $f(x)$ là $O(g(x))$, và $h(x)$ là một hàm có giá trị tuyệt đối lớn hơn $g(x)$ với giá trị x đủ lớn,
  - Suy ra $f(x)$ là $O(h(x))$.
  - Nói cách khác, hàm $g(x)$ trong mối quan hệ $f(x)$ là $O(g(x))$ có thể được thay thế bằng một hàm với giá trị tuyệt đối lớn hơn.

Độ tăng của hàm: Ví dụ

- **Chứng minh:**

$$f(x) = x^2 + 2x + 1 \text{ là } O(x^2)$$

- **Lời giải:**

- Ta thấy rằng khi $x > 1$ ta có:

  $$0 \leq x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$$

  Do đó, chúng ta có thể chọn $C = 4$ và $k = 1$ để thấy rằng $f(x)$ là $O(x^2)$. Nghĩa là, $f(x) = x^2 + 2x + 1 < 4x^2$ với mọi $x > 1$.

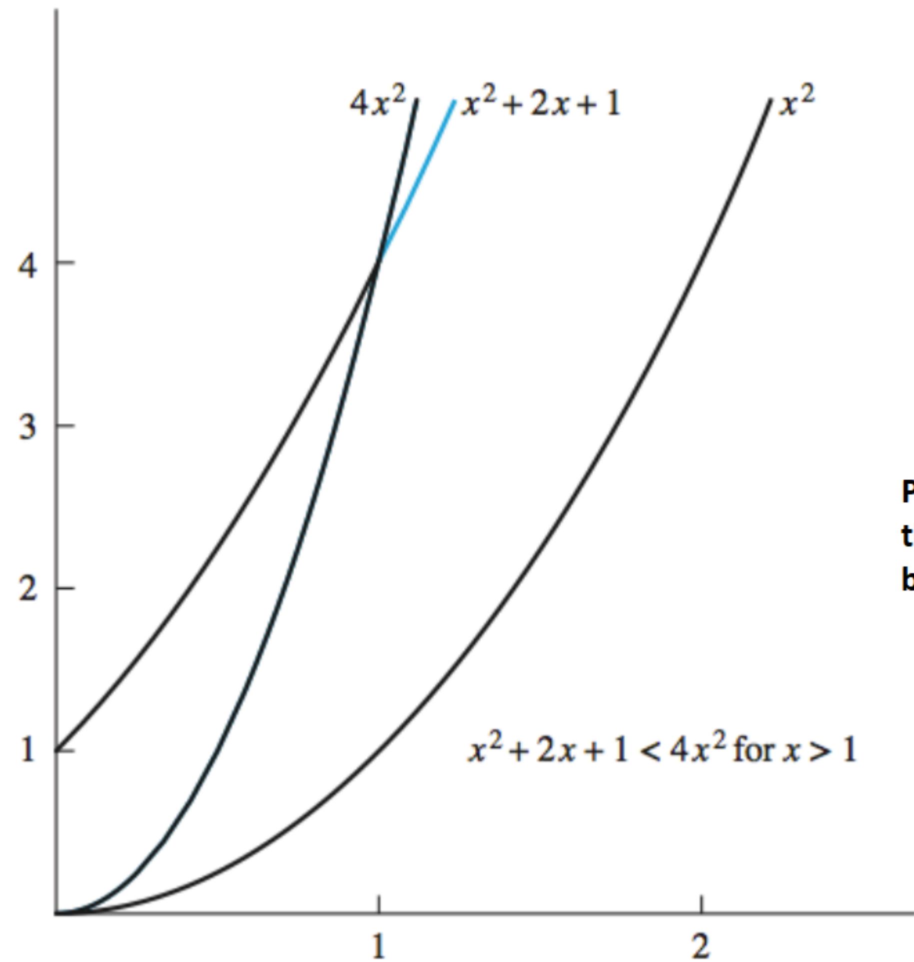
  (Lưu ý rằng không cần sử dụng giá trị tuyệt đối ở đây vì tất cả các hàm trong đẳng thức đều dương khi x dương.)

- Ngoài ra, với $x > 2$, chúng ta có $2x \leq x^2$ và $1 \leq x^2$, do đó, với $x > 2$ ta có:

  $$0 \leq x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2$$

  Do đó, $C = 3$ và $k = 2$ cũng là "chứng cứ/witnesses" cho mối quan hệ $f(x)$ là $O(x^2)$.

Độ tăng của hàm: Ví dụ



Phần đồ thị của f(x) = $x^2 + 2x + 1$ thỏa mãn f(x) < $4x^2$ được thể hiện bằng màu xanh.

Hình 1: Hàm $x^2 + 2x + 1$ là $O(x^2)$.

Độ tăng của hàm: Ví dụ

- **Chứng minh:**
  - Đặt $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, với $a_0, a_1, \ldots, a_{n-1}, a_n$ là các số thực. Khi đó, $f(x)$ là $O(x^n)$.
- **Lời giải:**
  - Sử dụng bất đẳng thức tam giác (xem Bài tập 7, mục 1.8) với $x > 1$, ta có:

$$\begin{aligned} |f(x)| &= |a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0| \\ &\leq |a_n|x^n + |a_{n-1}|x^{n-1} + \cdots + |a_1|x + |a_0| \\ &= x^n\left(|a_n| + \frac{|a_{n-1}|}{x} + \cdots + \frac{|a_1|}{x^{n-1}} + \frac{|a_0|}{x^n}\right) \\ &\leq x^n(|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|) \end{aligned}$$

  - Qua đó,

$$|f(x)| \leq Cx^n,$$

với $C = |a_n| + |a_{n-1}| + \cdots + |a_0|$ với mọi $x > 1$. Do đó, $C = |a_n| + |a_{n-1}| + \cdots + |a_0|$ và $k = 1$ cho thấy $f(x)$ là $O(x^n)$.

Độ tăng của hàm

- **Định lý:**
  - Giả sử $f_1(x)$ là $O\big(g_1(x)\big)$ và $f_2(x)$ là $O\big(g_2(x)\big)$
    - Thì $(f_1 + f_2)(x)$ là $O(\max(|g_1(x)|, |g_2(x)|))$.
  - Giả sử $f_1(x)$ và $f_2(x)$ đều là $O\big(g(x)\big)$
    - Thì $(f_1 + f_2)(x)$ là $O\big(g(x)\big)$.
  - Giả sử $f_1(x)$ là $O\big(g_1(x)\big)$ và $f_2(x)$ là $O\big(g_2(x)\big)$
    - Thì $(f_1 f_2)(x)$ là $O\big(g_1(x) g_2(x)\big)$.

Biểu diễn số nguyên

Số nguyên và Phép chia

- **Lý thuyết số** là một nhánh của toán học, nghiên cứu về các số nguyên và các tính chất của chúng.
- **Số nguyên (Integer):**
  - Tập số nguyên $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
  - Tập số nguyên dương $\mathbb{Z}^+ = \{1, 2, \dots\}$
- Lý thuyết số có nhiều ứng dụng trong khoa học máy tính:
  - Đánh chỉ mục (Indexing) phục vụ lưu trữ và tổ chức dữ liệu (data)
  - Mã hóa
  - Mã sửa lỗi
  - Bộ tạo số ngẫu nhiên

Phép chia (Division)

- **Định nghĩa:** Giả sử a và b là 2 số nguyên, trong đó a $\neq$ 0. Ta nói **b chia hết cho a** nếu có một số nguyên c sao cho b = ac. Nếu b chia hết cho a ta nói **a là một ước số của b** và **b là bội số của a**.
  - b chia hết cho a được kí hiệu là a|b
- **Ví dụ 1:**
  - 4|24 Đúng hay sai? **Đúng**
    - 4 là ước của 24
    - 24 là bội của 4
  - 3|7 Đúng hay sai? **Sai**

Chia hết (Divisibility)

- **Tính chất:**

  Gọi a, b, c là các số nguyên, khi đó:

  1. Nếu $a|b$ và $a|c$ thì $a|(b+c)$
  2. Nếu $a|b$ thì $a|bc$ với mọi c
  3. Nếu $a|b$ và $b|c$ thì $a|c$

- **Chứng minh 1:** Nếu $a|b$ và $a|c$ thì $a|(b+c)$

Từ định nghĩa của sự Chia hết ta có: $b = au$ và $c = av$ với u,v là 2 số nguyên. Thì $(b+c) = au + av = a(u+v)$

Do đó, b+c chia hết cho a.

Chia hết (Divisibility)

- **Tính chất:**

  Gọi a, b, c là các số nguyên. Có những tính chất sau:

  1. Nếu $a|b$ và $a|c$ thì $a|(b+c)$
  2. Nếu $a|b$ thì $a|bc$ với mọi c
  3. Nếu $a|b$ và $b|c$ thì $a|c$

- **Chứng minh 2:** Nếu $a|b$ thì $a|bc$ với mọi c

Nếu $a|b$ thì sẽ tồn tại số nguyên u nào đó sao cho $b = au$

Nhân 2 vế với c ta có $bc = auc$, vì vậy theo định nghĩa, $a|bc$

Do đó, bc chia hết cho a.

Phép chia

❑ Định lý:

Đặt a là một số nguyên và d là một số nguyên dương. Tồn tại các số nguyên duy nhất, q và r, với $0 \leq r < d$, sao cho:

$$a = dq + r$$

❑ Các định nghĩa

- a là số bị chia
- d là số chia
- q là thương số
- r là số dư của phép chia

❑ Các ký hiệu

- $q = a \text{ div } d$
- $r = a \text{ mod } d$

❑ Ví dụ

$$a = 14, d = 3$$

- $14 = 3 \times 4 + 2$
- $\frac{14}{3} = 3.666$
- $14 \text{ div } 3 = 4$
- $14\ mod\ 3 = 2$

Biểu diễn số nguyên

- Thường ngày, chúng ta dùng hệ thập phân, hay hệ cơ số 10, để biểu diễn số nguyên. Ví dụ ta viết 965, nghĩa là $9\times10^2 + 6\times10^1 + 5\times10^0$.
- Cơ số b=10 có thể thay bằng bất kỳ số nguyên dương lớn hơn 1 nào.
  - b = 2 (hệ nhị phân)
  - b = 8 (hệ bát phân)
  - b=16 (hệ thập lục) rất quan trọng trong tính toán.
  - Người Maya cổ dùng hệ cơ số 20 và người Babylon dùng hệ cơ số 60.

Biểu diễn hệ cơ số b

- **Định lý 1:** Cho b là một số nguyên dương lớn hơn 1. Nếu n là một số nguyên dương, nó có thể được biểu diễn duy nhất bởi:

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0$$

với k là số nguyên không âm, $a_0, a_1, \dots, a_k$ là các số nguyên không âm nhỏ hơn b, và $a_k \neq 0$. $a_j, j = 0, \dots, k$ được gọi là các chữ số hệ-b của biểu diễn.

  - Biểu diễn của n trong **Định lý 1** được gọi là **Hệ cơ số $\boldsymbol{b}$ của $\boldsymbol{n}$** và được kí hiệu là $(a_k a_{k-1} \dots a_1 a_0)_b$.
  - Ta thường bỏ số 10 trong biểu diễn thập phân.

Hệ cơ số 2 (Hệ nhị phân)

- Máy vi tính biểu diễn số nguyên và thực hiện tính toán với dạng nhị phân của số nguyên.
  - Các chữ số duy nhất được dung là 0 và 1.
- **Ví dụ:** Biểu diễn thập phân của số nguyên có dạng nhị phân là $(1\ 0101\ 1111)_2$?
  - **Đáp án:**
    $(1\ 0101\ 1111)_2 = 1\times2^8 + 0\times2^7 + 1\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 = 351$
- **Ví dụ:** Biểu diễn thập phân của số nguyên có dạng nhị phân là $(11011)_2$?
  - **Đáp án:**
    $(11011)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 27$

Hệ cơ số 8

- Hệ cơ số 8 sử dụng các chữ số {0, 1, 2, 3, 4, 5, 6, 7}
- **Ví dụ:** Biểu diễn hệ cơ số 10 của số $(7016)_8$?
  - **Đáp án:**
    $7\times8^3 + 0\times8^2 + 1\times8^1 + 6\times8^0 = 3598$
- **Ví dụ:** Biểu diễn hệ cơ số 10 của số $(111)_8$?
  - **Đáp án:**
    $1\times8^2 + 1\times8^1 + 1\times8^0 = 64 + 8 + 1 = 73$

Hệ cơ số 16

- Hệ cơ số 16 sử dụng 16 kí tự: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
  - Các chữ cái từ A đến F biểu diễn các số thập phân 10 đến 15
- **Ví dụ:** Biểu diễn thập phân của số $(2AE0B)_{16}$?
  - **Đáp án:** $2\times16^4 + 10\times16^3 + 14\times16^2 + 0\times16^1 + 11\times16^0 = 175627$
- **Ví dụ:** Biểu diễn thập phân của số $(E5)_{16}$?
  - **Đáp án:** $14\times16^1 + 5\times16^0 = 224 + 5 = 229$

Chuyển đổi Hệ cơ số

- Các bước tìm biểu diễn hệ cơ số b của số nguyên n:
  - Chia n cho b để thu được thương số và số dư:

$$n = bq_0 + a_0 \; (0 \leq a_0 \leq b)$$

  - Số dư, a_0 là chữ số bên phải nhất trong biểu diễn hệ cơ số b của n. Tiếp theo, chia q_0 cho b.

$$q_0 = bq_1 + a_1 \; (0 \leq a_1 \leq b)$$

  - Số dư a_1 là số thứ 2 từ phải sang trong biểu diễn hệ cơ số b của n.
  - Tiếp tục bằng cách chia các thương số cho b, lấy các chữ số tiếp theo từ phần dư. Quá trình kết thúc khi thương bằng 0.

Chuyển đổi Hệ cơ số

- **Ví dụ:** Tìm biểu diễn hệ cơ số 8 của $(12345)_{10}$
  - **Đáp án:**
    Chia liên tục cho 8:
    $12345 = 8 \times 1543 + 1$
    $1543 = 8 \times 192 + 7$
    $192 = 8 \times 24 + 0$
    $24 = 8 \times 3 + 0$
    $3 = 8 \times 0 + 3$
  - Biểu diễn cần tìm là các phần dư được viết từ phải sang trái: $(30071)_8$

Các phép toán trong hệ nhị phân

- *Thuật toán Cộng các số nguyên*

procedure add(a, b: *các số nguyên dương*)
{biểu diễn nhị phân của a và b lần lượt là $(a_{n-1}a_{n-2}\ldots a_1a_0)_2$ và $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$}
$c := 0$
for $j := 0$ **to** $n-1$
    $d := \lfloor (a_j + b_j + c)/2 \rfloor$
    $s_j := a_j + b_j + c - 2d$
    $c := d$
$s_n := c$
return $(s_0, s_1, \ldots, s_n)$ {biểu diễn nhị phân của tổng là $(s_n s_{n-1} \ldots s_0)_2$}

Các phép toán trong hệ nhị phân

- $ab = a(b_0 2^0 + b_1 2^1 + \cdots + b_{n-1} 2^{n-1}) = a(b_0 2^0) + a(b_1 2^1) + \cdots + a(b_{n-1} 2^{n-1})$
- *Thuật toán Tích các số nguyên*

procedure multiply(a, b: *các số nguyên dương*)

{biểu diễn nhị phân của a và b lần lượt là $(a_{n-1}a_{n-2} \ldots a_1 a_0)_2$ và $(b_{n-1}b_{n-2} \ldots b_1 b_0)_2$}

for $j := 0$ **to** $n - 1$
    if $b_j = 1$ **then** $c_j := a$ chuyển j vị trí
    else $c_j := 0$
{$c_0, c_1, \ldots, c_{n-1}$ là các tích thành phần}
$p := 0$
for $j := 0$ **to** $n - 1$
    $p := p + c_j$
return p {p là giá trị của ab}

Các phép toán trong hệ nhị phân

- $b^n = b^{a_{k-1}2^{k-1}+\cdots+a_1 2^1+a_0} = b^{a_{k-1}2^{k-1}} \dots b^{a_1 2} b^{a_0}$
- *Thuật toán Lũy thừa mô đun*

procedure modular exponentiation(b: *số nguyên*, $n = (a_{k-1}a_{k-2} \dots a_1 a_0)_2$, m: *số nguyên dương*)

$x := 1$
$power := b$ **mod** m
for $i := 0$ **to** $k - 1$
    if $a_i = 1$ **then** $x := (x \cdot power)$ **mod** m
    $power := (power \cdot power)$ **mod** m
return x {x bằng b^n **mod** m}

Ví dụ về Lũy thừa Mô đun

- **Ví dụ:** Dùng Thuật toán 3 để tìm 3^{644} **mod** 645.
  - **Đáp án:** Thuật toán 3 khởi tạo $x = 1$ và $power = 3$ **mod** 645 = 3. Trong phép tính 3^{644} **mod** 645, thuật toán này xác định 3^{2^j} **mod** 645 với $j = 1, 2, \dots, 9$ bằng cách liên tục bình phương và giảm bằng modulo 645. Nếu $a_j = 1$ (với a_j là bit ở vị trí thứ j trong biểu diễn nhị phân của 644, hay chính là $(1010000100)_2$), nó nhân giá trị hiện tại của x với 3^{2^j} **mod** 645 và giảm kết quả bằng modulo 645. Sau đây là các bước:

i=0: Vì $a_0 = 0$, ta có $x = 1$ và $power = 3^2$ **mod** $645 = 9$ **mod** $645 = 9$;
i=1: Vì $a_1 = 0$, ta có $x = 1$ và $power = 9^2$ **mod** $645 = 81$ **mod** $645 = 81$;
i=2: Vì $a_2 = 1$, ta có $x = 1 \cdot 81$ **mod** $645 = 81$ và $power = 81^2$ **mod** $645 = 6561$ **mod** $645 = 111$;
i=3: Vì $a_3 = 0$, ta có $x = 81$ và $power = 111^2$ **mod** $645 = 12{,}321$ **mod** $645 = 66$;
i=4: Vì $a_4 = 0$, ta có $x = 81$ và $power = 66^2$ **mod** $645 = 4356$ **mod** $645 = 486$;
i=5: Vì $a_5 = 0$, ta có $x = 81$ và $power = 486^2$ **mod** $645 = 236{,}196$ **mod** $645 = 126$;
i=6: Vì $a_6 = 0$, ta có $x = 81$ và $power = 126^2$ **mod** $645 = 15{,}876$ **mod** $645 = 396$;
i=7: Vì $a_7 = 0$, ta thấy rằng $x = (81 \cdot 396)$ **mod** $645 = 471$ và $power = 396^2$ **mod** $645 = 156{,}816$ **mod** $645 = 81$;
i=8: Vì $a_8 = 0$, ta có $x = 471$ và $power = 81^2$ **mod** $645 = 6561$ **mod** $645 = 111$;
i=9: Vì $a_9 = 1$, ta thấy rằng $x = (471 \cdot 111)$ **mod** $645 = 36$.

Kết thúc nội dung bài giảng