Quy nạp và Hồi quy

Nội dung

- Phép quy nạp toán học
- Định nghĩa đệ quy
- Các thuật toán quy nạp

Quy nạp toán học (Induction)

- Được sử dụng để chứng minh các mệnh đề dạng

$$\forall x\, P(x), x \in Z^+$$

- Gồm 2 bước:
  - **Cơ bản**: Chứng minh P(1) đúng
  - **Bước quy nạp**: Chứng minh P(n) -> P(n+1) đúng với mọi số dương n.
  - Kết luận ∀n P(n)

Quy nạp toán học

- Đặc trưng có thứ tự (Well-ordering property)
  - Mọi tập không rỗng của các số nguyên không âm có thành phần nhỏ nhất.
- Tính đúng đắn của quy nạp toán học
  - Giả sử tồn tại ít nhất một số n để P(n) sai.
    - Gọi S là một tập các số nguyên không âm sao cho P(n) sai. Do đó, $S \neq \emptyset$.
  - **Dựa vào đặc trưng có thứ tự**, S có thành phần nhỏ nhất, giả sử là k ($k > 1$),
    - $k - 1 > 0$ và P(k-1) đúng (vì k là số nguyên nhỏ nhất mà P(k) sai)
  - **Mà:** P(k-1) -> P(k) đúng
    - Do đó, P(k) đúng (phản chứng)
  - **Vì vậy, ∀n P(n).**

Quy nạp toán học

- **Ví dụ**: Chứng minh tổng của n số nguyên lẻ đầu tiên là n^2, với mọi số nguyên dương. P(n): $1 + 3 + 5 + 7 + \ldots + (2n-1) = n^2$

- **Chứng minh**:
  - **Bước cơ bản**: P(1) đúng, vì $1 = 1^2$
  - **Bước quy nạp**:
    - Giả sử P(n) đúng, $1 + 3 + 5 + 7 + \ldots + (2n-1) = n^2$
    - Khi đó P(n+1) đúng, vì

$$\underbrace{1 + 3 + 5 + 7 + \ldots + (2n - 1)}_{n^2} + (2n + 1) = n^2 + (2n+1) = (n+1)^2$$

Quy nạp toán học

- **Ví dụ:** Chứng minh $n < 2^n$ với mọi số nguyên dương n. P(n): $n < 2^n$
- **Chứng minh**
  - **Bước cơ bản:** $1 < 2^1$ (luôn đúng)
  - **Bước quy nạp:** Nếu P(n) đúng thì P(n+1) cũng đúng với mọi n.
    - Giả sử P(n): $n < 2^n$ đúng
    - P(n+1): $n+1 < 2^{n+1}$ đúng, vì

$$n+1 < 2^n + 1 < 2^n + 2^n = 2^n(1+1) = 2^n(2) = 2^{n+1}$$

Quy nạp toán học

- **Ví dụ**: Chứng minh $n^3 - n$ chia hết cho 3 với mọi số nguyên dương. P(n): n^3-n chia hết cho 3
- **Chứng minh**
  - **Bước cơ bản**: P(1): $1^3 - 1 = 0$ chia hết cho 3 (luôn đúng)
  - **Bước quy nạp**: Nếu P(n) đúng thì P(n+1) đúng với mọi số nguyên dương.
    - Giả sử P(n): n^3-n chia hết cho 3
    - Chứng minh P(n+1): $(n+1)^3 - (n+1)$ chia hết cho 3

$$
\begin{aligned}
(n+1)^3 - (n+1) &= n^3 + 3n^2 + 3n + 1 - n - 1 \\
&= (n^3 - n) + 3n^2 + 3n \\
&= (n^3 - n) + 3(n^2 + n)
\end{aligned}
$$

divisible by 3 divisible by 3

Quy nạp toán học

- **Ví dụ**: (Bài toán ném bánh) Một số lẻ người đứng trong sân sao cho khoảng cách giữa 2 người bất kỳ nào đều không giống nhau. Cùng một lúc, mỗi người sẽ ném 1 chiếc bánh về phía người gần nhất với mình. Chứng minh rằng có ít nhất một người không bị ai ném cả.

Quy nạp toán học

Lỗi mắc phải khi chứng minh bằng quy nạp

- "**Định lý:**" Với mọi số nguyên dương n, nếu x và y là 2 số nguyên dương thỏa mãn max(x, y) = n, thì x = y
- Chứng minh
  - **Bước cơ bản**: Với n = 1. Nếu max(x, y) = 1 và x, y đều nguyên dương, ta có x = 1 và y = 1. P(1) đúng.
  - **Bước quy nạp**: Với k là một số nguyên dương, giả sử rằng bất kể khi nào max(x, y) = k và x, y đều nguyên dương thì x=y. Nếu để max(x, y) = k + 1, thì max(x − 1, y − 1) = k, do đó theo giả thuyết quy nạp ta có x − 1 = y − 1.
    - Có nghĩa là x = y [đpcm]

Quy nạp mạnh (Strong induction)

- Quy nạp thông thường:
  - Dùng bước cơ bản: P(1)
  - Bước quy nạp: P(n-1) -> P(n)

- Quy nạp mạnh:
  - Dùng bước cơ bản: P(1)
  - Bước quy nạp: P(1) và P(2) … P(n-1) -> P(n)

Quy nạp mạnh

- **Ví dụ**: Chứng minh một số nguyên dương lớn hơn 1 có thể được biểu diễn dưới dạng tích của các số nguyên tố.
- **Chứng minh**:
  - **Bước cơ bản**: P(2) đúng
  - **Bước quy nạp**: Giả sử P(2), P(3),…P(n) đúng. Chứng minh P(n+1) cũng đúng. Có 2 trường hợp:
    - Nếu n+1 là số nguyên tố thì P(n+1) đương nhiên đúng
    - Nếu n+1 là hợp số thì nó có thể được biểu diễn là tích của 2 số nguyên (n+1) = a*b sao cho 1<a,b<n+1
      - Từ giả sử P(a) và P(b) đúng, ta có P(n+1) đúng.

Quy nạp mạnh

- **Ví dụ**: Một trò chơi trong đó 02 người chơi lần lượt lấy ra một số bất kỳ các que diêm (mà họ muốn) từ một trong 2 cột diêm. Người chơi nào lấy được que diêm cuối cùng sẽ thắng.
  - Chứng minh rằng nếu ban đầu 2 cột diêm chứa số que diêm bằng nhau thì người chơi thứ 2 luôn thắng

Định nghĩa đệ quy (Recursive definition)

- Một đối tượng (hàm, chuỗi, thuật toán, cấu trúc) trong một số trường hợp có thể được định nghĩa bằng chính nó. Quy trình này được gọi là **đệ quy (recursion)**.

Ví dụ:

- Định nghĩa đệ quy của một cấp số cộng:

  $a_n = a + nd$

  $a_n = a_{n-1} + d,\ a_0 = a$

- Định nghĩa đệ quy của một cấp số nhân:

  $x_n = ar^n$

  $x_n = rx_{n-1},\ x_0 = a$

Định nghĩa đệ quy

- Trong một số trường hợp, các đối tượng sẽ được định nghĩa một cách dễ dàng và dễ hiểu hơn khi sử dụng **định nghĩa đệ quy**.

Ví dụ:

- Thuật toán tính ước chung lớn nhất gcd:
  - gcd(79,35) = gcd(35,9)
  - Tổng quát hơn: gcd(a,b) = gcd(b, a mod b)
- Hàm giai thừa
  - n! = n(n-1)! và 0! = 1
- Các bộ sinh số giả ngẫu nhiên:
  - $x_{n+1}=(ax_n+c) \bmod m$

Định nghĩa đệ quy các hàm

- Để định nghĩa đệ quy một hàm trên tập các số nguyên không âm
  - Xác định giá trị của hàm tại 0
  - Đưa ra một quy tắc để tìm giá trị của hàm tại n+1 với các số nguyên i <= n.

Ví dụ: định nghĩa hàm giai thừa

- 0! = 1
- n! = n(n-1)!

Định nghĩa đệ quy các hàm

Ví dụ:

- Giả sử một hàm đệ quy trên tập các số nguyên dương:
  - f(0) = 3
  - f(n+1) = 2f(n) + 3
- Giá trị của f(0) là gì? 3
  - f(1) = 2f(0) + 3 = 2(3) + 3 = 6 + 3 = 9
  - f(2) = f(1+1) = 2f(1) + 3 = 2(9) + 3 = 18 + 3 = 21
  - f(3) = f(2+1) = 2f(2) + 3 = 2(21) + 3 = 42 + 3 = 45
  - f(4) = f(3+1) = 2f(3) + 3 = 2(45) + 3 = 90 + 3 = 93

Định nghĩa đệ quy

Ví dụ:

- Hàm f(n) = 2n + 1 với n = 0,1,2,… được định nghĩa đệ quy như sau:
  - f(0) = 1
  - f(n+1) = f(n) +2
- Chuỗi: $a_n = n^2$ với n = 1,2,3,… được định nghĩa đệ quy như sau:
  - $a_1 = 1$
  - $a_{n+1} = a_n^2 + (2n+1)$, n >= 1

Các định nghĩa đệ quy

Ví dụ:

- Định nghĩa đệ quy của tổng n số nguyên dương đầu tiên F(n) = $\sum_{i=1}^{n} i$
  - F(1) = 1
  - F(n+1) = F(n) + (n+1), n >= 1

Định nghĩa đệ quy: xâu ký tự

- Cho trước tập các chữ cái $\sum$
  - Ví dụ: $\sum = \{a,b,c,d\}$
- $\sum^*$
  - Một tập gồm tất cả các xâu có chứa những kí tự trong $\sum$
  - Ví dụ: $\sum^* = \{\text{“”},a,aa,aaa,aaa\ldots,ab,\ldots b,bb,bbb,\ldots\}$
- Định nghĩa đệ quy của $\sum^*$
  - Xâu rỗng $\boldsymbol{\lambda} \in \sum^*$
  - Nếu $w \in \sum^*$ và $x \in \sum$ thì $wx \in \sum^*$

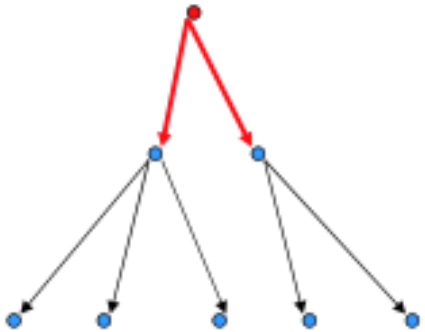
Định nghĩa đệ quy: xâu ký tự (String)

Ví dụ:

- Tìm một định nghĩa đệ quy của l(w), là độ dài của xâu w.
- **Giải pháp:**
  - l(“”) = 0; (độ dài của xâu rỗng)
  - l(wx) = l(w) + 1 nếu $w \in \sum^*$ và $x \in \sum$.

Định nghĩa đệ quy

- Các cấu trúc dữ liệu
  - Ví dụ: Cây có gốc (rooted tree)



- Bước cơ bản:
  - Một nút đơn (đỉnh) là một cây có gốc
- Bước đệ quy:
  - Giả sử T1, T2, … Tk là các cây có gốc, thì đồ thị với một gốc r liên kết với T1, T2,…Tk là một cây có gốc.

Thuật toán đệ quy

- Một thuật toán được gọi là "**đệ quy**" nếu nó giải quyết một vấn đề bằng cách giải quyết vấn đề tương tự nhưng với kích thước đầu vào nhỏ hơn.
  - Dễ cài đặt
- **Ví dụ**:

ALGORITHM 1 A Recursive Algorithm for Computing $n!$.

procedure *factorial*(n: nonnegative integer)
if $n = 0$ **then return** 1
else return $n \cdot factorial(n - 1)$
{output is $n!$}

Thuật toán đệ quy

- Thuật toán đệ quy tính a^n.

ALGORITHM 2 A Recursive Algorithm for Computing a^n.

procedure *power*(a: nonzero real number, n: nonnegative integer)
if $n = 0$ **then return** 1
else return $a \cdot power(a, n - 1)$
{output is a^n}

Thuật toán đệ quy

- Thuật toán đệ quy tính ước chung lớn nhất

ALGORITHM 3 A Recursive Algorithm for Computing gcd(a, b).

procedure $gcd(a, b$: nonnegative integers with $a < b)$
if $a = 0$ **then return** b
else return $gcd(b \bmod a, a)$
{output is gcd(a, b)}

Thuật toán đệ quy

- Thuật toán đệ quy tính lũy thừa modulo $b^n \bmod m$

ALGORITHM 4 Recursive Modular Exponentiation.

procedure *mpower*$(b, n, m$: integers with $b > 0$ and $m \geq 2, n \geq 0)$
if $n = 0$ **then**
    return 1
else if n is even **then**
    return $mpower(b, n/2, m)^2$ **mod** m
else
    return $(mpower(b, \lfloor n/2 \rfloor, m)^2$ **mod** $m \cdot b$ **mod** $m)$ **mod** m
{output is b^n **mod** m}

Thuật toán đệ quy

❑ Thuật toán đệ quy tìm kiếm tuyến tính

ALGORITHM 5 A Recursive Linear Search Algorithm.

procedure *search*(i, j, x: i, j, x integers, $1 \leq i \leq j \leq n$)
if $a_i = x$ **then**
    return i
else if $i = j$ **then**
    return 0
else
    return *search*($i + 1, j, x$)
{output is the location of x in $a_1, a_2, \ldots, a_n$ if it appears; otherwise it is 0}

Thuật toán đệ quy

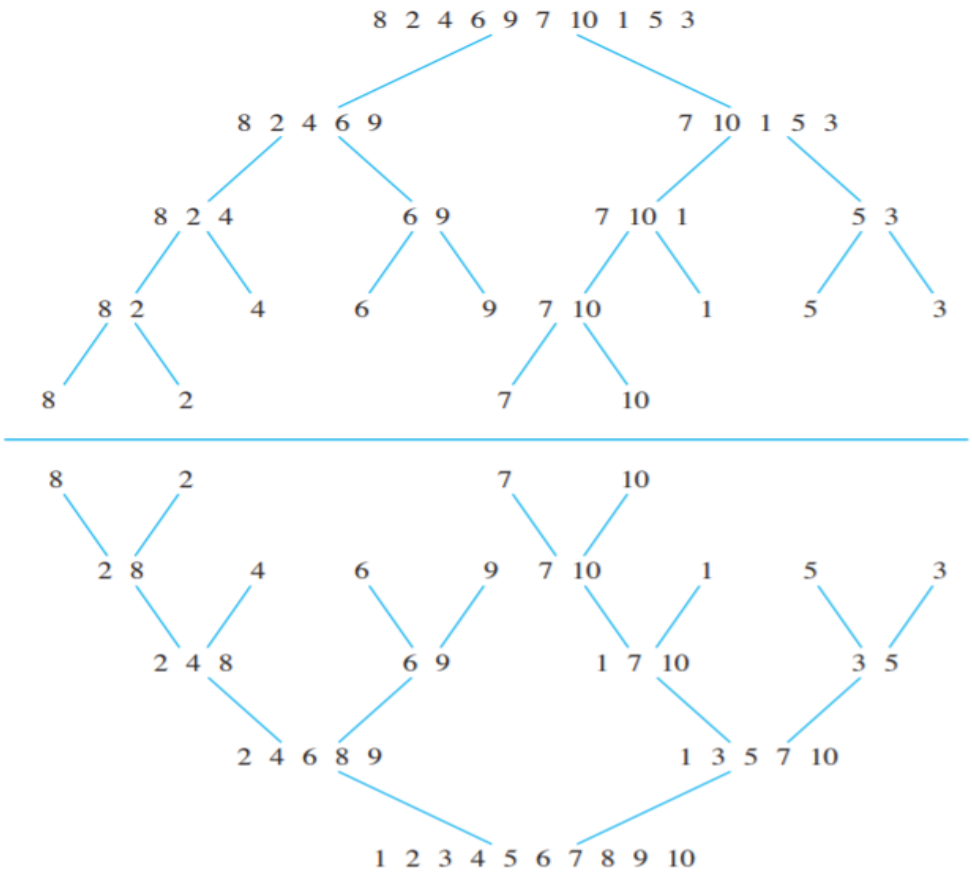
❑ Thuật toán đệ quy tìm kiếm nhị phân

ALGORITHM 6 A Recursive Binary Search Algorithm.

procedure *binary search*(i, j, x: i, j, x integers, $1 \leq i \leq j \leq n$)
$m := \lfloor (i + j)/2 \rfloor$
if $x = a_m$ **then**
    return m
else if ($x < a_m$ and $i < m$) **then**
    return *binary search*($i, m - 1, x$)
else if ($x > a_m$ and $j > m$) **then**
    return *binary search*($m + 1, j, x$)
else return 0
{output is location of x in $a_1, a_2, \ldots, a_n$ if it appears; otherwise it is 0}

Thuật toán đệ quy

- Thuật toán sắp xếp trộn (merge sort)
  - Ví dụ: 8, 2, 4, 6, 9, 7, 10, 1, 5, 3



Thuật toán đệ quy

❑ Thuật toán sắp xếp trộn (merge sort)

- Độ phức tạp *O(n log n)*

ALGORITHM 9 A Recursive Merge Sort.

procedure *mergesort*($L = a_1, \ldots, a_n$)
if $n > 1$ **then**
    $m := \lfloor n/2 \rfloor$
    $L_1 := a_1, a_2, \ldots, a_m$
    $L_2 := a_{m+1}, a_{m+2}, \ldots, a_n$
    $L := merge(mergesort(L_1), mergesort(L_2))$
{L is now sorted into elements in nondecreasing order}

- Hai danh sách đã được sắp xếp gồm m và n phần tử có thể được trộn vào thành 1 danh sách được sắp xếp bằng cách sử dụng không nhiều hơn m + n − 1 phép so sánh.

ALGORITHM 10 Merging Two Lists.

procedure *merge*(L_1, L_2: sorted lists)
$L :=$ empty list
while L_1 and L_2 are both nonempty
    remove smaller of first elements of L_1 and L_2 from its list; put it at the right end of L
    if this removal makes one list empty **then** remove all elements from the other list and append them to L
return L{L is the merged list with elements in increasing order}

Thuật toán đệ quy

- Tính đúng đắn của thuật toán đệ quy được chứng minh bằng phương pháp quy nạp [mạnh] toán học
- Ví dụ thuật toán a^n
  - **Bước cơ bản**: Nếu n = 0, bước đầu tiên của thuật toán cho chúng ta biết power(a, 0) = 1. Điều này đúng vì a^0 =1 với mọi $a \neq 0$.
  - **Bước đệ quy**: Giả sử thuật toán tính chính xác được a^k, cần phải chứng minh rằng nó cũng tính chính xác được a^{k+1}
    - power(a, k+1) = a * power(a, k) = a * a^k = a^{k+1}