Đệ quy (4)

Khoa Khoa học máy tính

Đệ quy

- Thuật toán được gọi là đệ quy khi nó được xây dựng dựa trên chính nó

- Đơn đệ quy (simple recursion)
  - Chẳng hạn, định nghĩa hàm tính x^n
    - Hàm được định nghĩa đệ quy

$$x^n = \begin{cases} 1 \; khi n = 0 \\ x \times x^{n-1} \; khi n \geq 1 \end{cases}$$

    - Thuật toán đệ quy

```
function ham_mu (x, n)
begin
        if (n=0) then ham_mu = 1
        else ham_mu = x * ham_mu(x, n-1)
        endif
end
```

Đệ quy

- Đa đệ quy (multiple recursion)
  - Một định nghĩa đệ quy có thể có nhiều hơn một lời gọi đệ quy
  - Ví dụ:
    - Định nghĩa dãy số Fibonaccci

      $F_0 = 1, F_1 = 1$

      $F_n = F_{n-1} + F_{n-2}$
    - Thuật toán

```
function fib (n)
begin
        if ((n=0) or (n=1)) then fib = 1
        else fib = fib(n-1) + fib(n-2)
        endif
end
```

Đệ quy

- Đệ quy chéo (mutual recursion)
  - Các định nghĩa được gọi là đệ quy chéo nếu chúng phụ thuộc lẫn nhau
  - Ví dụ
    - Định nghĩa số chẵn/lẽ

$$even(n) = \begin{cases} true\ if\ n = 0 \\ odd(n-1)\ else \end{cases} \qquad odd(n) = \begin{cases} false\ if\ n = 0 \\ even(n-1)\ else \end{cases}$$

    - Thuật toán

```
function even (n)
begin
        if (n=0) then even = true
        else even = odd(n-1)
        endif
end
```

```
function odd (n)
begin
        if (n=0) then odd = false
        else odd = even(n-1)
        endif
end
```

Đệ quy

- Đệ quy chồng (implicated recursion)
  - Các định nghĩa được gọi đệ quy lồng nhau
  - Ví dụ
    - Định nghĩa hàm Ackermann

$$A(m,n)=\begin{cases} n+1 \ if \ m=0 \\ A(m-1,1) \ if \ m>0, n=0 \\ A(m-1, A(m,n-1)) \ else \end{cases}$$

    - Thuật toán

```
function Ackermann (m, n)
begin
        if (m=0) then Ackermann = n+1
        else if (n=0) Ackermann = Ackermann(m-1,1)
             else Ackermann = Ackermann(m-1, Ackermann(m, n-1))
             endif
        endif
end
```

Đệ quy

- Nguyên tắc
  - Chúng ta cần có
    - một số trường hợp mà giải pháp xác định – « trường hợp đơn giản »: các trường hợp dừng của đệ quy
    - một cách để chuyển từ một « trường hợp phức tạp » thành « trường hợp đơn giản »
- Khó khăn
  - Cần bảo đảm rằng, đệ quy sẽ dừng khi gặp giải pháp đã biết
    - Hàm phải được định nghĩa trên toàn miền dữ liệu
- Giải pháp
  - Dãy các giá trị liền nhau của các tham số được gọi phải thay đổi đơn điệu và đạt đến một giá trị mà giải pháp tương ứng đã được xác định

Đệ quy

- Ví dụ 1
  - Thuật toán sau kiểm tra a có là ước số của b

```
function divisor (a, b) // giả sử a>0, b>0
begin
        if (a≥b) then
                if (a=b) divisor = true
                else divisor = false
                endif
        else divisor=divisor(a, b-a)
        endif
end
```

- Dãy các giá trị b, b-a, b-2a … liên tuc giảm cho đến khi a≥b thì sẽ dừng, trường hợp đã được xác định

Đệ quy

- Ví dụ 2
  - Thuật toán

```
function syracuse (n)
begin
        if (n=0 or n=1) then syracuse = 1
        else
                if (a mod 2 = 0) syracuse = syracuse(n/2)
                else syracuse = syracuse(3*n+1)
                endif
        endif
end
```

- Thuật toán được định nghĩa rỏ ràng
- Thuật toán có dừng ?

Đệ quy

- Không thể xác định tính dừng (1)
  - Vấn đề
    - Có thể xây dựng chương trình tự động kiểm tra một chương trình P có thể dừng khi thực thi trên bộ dữ liệu D ?
    - Vào
      - chương trình P
      - bộ dữ liệu D
    - Ra
      - đúng, nếu chương trình P dừng trên bộ dữ liệu D
      - Sai, nếu ngược lại

Đệ quy

- Không thể xác định tính dừng (2)
  - Giả sử tồn tại chương trình *terminate* kiểm tra tự động tính dừng của một chương trình
  - Từ chương trình *terminate* chúng ta xây dựng chương trình sau

```
program Q
begin
        result = terminate(Q)
        while (result = true)
                wait(1 minute)
        endwhile
end
```

⇨ Vấn đề dừng là không thể xác định !

Đệ quy

- Thứ tự của lời gọi đệ quy
  - Hãy cho biết kết quả của hai thuật toán sau

```
T(n) // n≥0
begin
   if (n=0) then do nothing
   else
           T(n-1)
           print(n) //in n
  endif
end
```

```
G(n) // n≥0
begin
   if (n=0) then do nothing
   else
           print(n) //in n
           G(n-1)
  endif
end
```

  - Thuật toán đệ quy in dãy nhị phân tương ứng của một số nguyên

Đệ quy

- Ví dụ thuật toán đệ quy (1)
  - Tháp Hà Nội: có 3 cọc A, B và C, mỗi cọc có thể chồng các đĩa có kích thước khác nhau, nguyên tắc chồng đĩa to dưới đĩa nhỏ trên; yêu cầu chuyển n đĩa trên cọc A sang cọc C với các điều kiện:
    - Mỗi lần chỉ được chuyển một đĩa
    - Không khi nào có tình huống đĩa to chồng trên đĩa nhỏ
    - Được sử dụng cọc B làm cọc trung gian khi chuyển đĩa

Đệ quy

- Ví dụ thuật toán đệ quy (2)
  - Giả thiết
    - chúng ta giải quyết được bài toán với n-1 đĩa
  - Nguyên tắc
    - Để chuyển n đĩa từ cọc A sang cọc C, thực hiện
      - chuyển n-1 đĩa nhỏ hơn từ cọc A sang cọc B
      - chuyển đĩa lớn nhất từ cọc A sang cọc C
      - chuyển n-1 đĩa nhỏ hơn từ cọc B sang cọc C
  - Thuật toán

```
Hanoi(n, A, B, C)
begin
   if (n=1) then chuyển đĩa lớn từ cọc A sang cọc C
   else   Hanoi(n-1, A, C, B)
          chuyển đĩa lớn từ cọc A sang cọc C
          Hanoi(n-1, B, A, C)
  endif
end
```

Đệ quy

- Ví dụ thuật toán đệ quy (3)
  - Tính độ phức tạp
    - Tính số lần chuyển đĩa

$$C(n) = \begin{cases} 1 & if\ n = 1 \\ C(n-1) + 1 + C(n-1) & else \end{cases}$$

$$C(n) = \begin{cases} 1 & if\ n = 1 \\ 2C(n-1) + 1 & else \end{cases}$$

$C(n) = 2^n - 1$

Đệ quy

- Chi phí đệ quy (1)
  - Chi phí lời gọi hàm

```
…
f(x, y, z)
…
```

```
f(a, b, c)
…
end
```

1. Lưu lại ngữ cảnh
2. Tạo ngăn xếp lưu các tham số, địa chỉ quay về, …
3. Chuyển đến hàm
4. Lấy từ ngăn xếp
5. Thực thi
6. Tạo ngăn xếp chứa kết quả trả về
7. Quay về vị trí gọi hàm
8. Lấy kết quả từ ngăn xếp
9. Phục hồi lại ngữ cảnh

Đệ quy

- Chi phí đệ quy (2)
  - Ví dụ

```
factorial(n)
begin
  factorial = 1
  for i = 2 to n
      factorial = factorial * i
  endfor
end
```

n-1	phép nhân
n	phép gán
n	phép gán (vòng lặp)
n-1	phép tăng 1 (vòng lặp)
n	phép so sánh

```
factorial(n)
begin
  if (n = 1) then
     factorial = 1
  else
     factorial = n * factorial(n-1)
  endif
end
```

n-1	phép nhân
n	phép gán
n-1	phép trừ (tính n-1)
n	phép so sánh
n	**lời gọi hàm**

→ Chi phí đệ quy rất lớn do lời gọi hàm

Khử đệ quy

- Chuyển thuật toán đệ quy thành thuật toán tương đương không chứa lời gọi đệ quy
  - Sử dụng vòng lặp

- Hai trường hợp đệ quy
  - Đệ quy kết thúc (terminal recursion)
    - Thuật toán được gọi là đệ quy kết thúc nếu nó không chứa bất kỳ xử lý nào sau lời gọi đệ quy
  - Đệ quy không kết thúc (non terminal recursion)
    - Thuật toán được gọi là đệ quy không kết thúc nếu nó chứa các xử lý nào sau lời gọi đệ quy

Khử đệ quy

- Đệ quy kết thúc (tail-recursion)
  - Sơ đồ tổng quát của thuật toán đệ quy kết thúc

```
P(U)
begin
    if C then
            D
            P(α(U))
    else
            T
    endif
end
```

```
P'(U)
begin
    while C do
            D
            U = α(U)
    endwhile
    T
end
```

U	:	danh sách các tham số
C	:	điều kiện phụ thuộc U
D	:	xử lý cơ bản của thuật toán
$\alpha(U)$	:	biểu diễn sự chuyển đổi tham số
T	:	xử lý dừng

Khử đệ quy

- Đệ quy kết thúc (2)
  - Ví dụ: khử đệ quy của thuật toán sau

```
bsearch(X, A, l, r)
begin
   if (l≤r) then
            m = (l+r)/2
            if (X = A[m]) then bsearch = m
            else if (X < A[m]) then bsearch = bsearch(X, A, l, m-1)
                  else bsearch = bsearch(X, A, m+1, r)
                  endif
            endif
  else
            bsearch = 0
  endif
end
```

Khử đệ quy

- Đệ quy kết thúc (3)
  - Ví dụ: thuật toán lặp tương đương

```
bsearch'(X, A)
begin
   l = 1
   r = n
   while (l≤r) do
            m = (l+r)/2
            if (X = A[m]) then bsearch' = m; return
            else if (X < A[m]) then r = m-1
                  else l = m+1
                  endif
            endif
  endwhile
  bsearch' = 0
end
```

Khử đệ quy

- Đệ quy không kết thúc (non tail-recursion)
  - Cần ghi nhớ lại *ngữ cảnh* của lời gọi đệ quy
    - điển hình là các *tham số* của lời gọi đệ quy
  - Sử dụng cấu trúc *ngăn xếp* (stack) để ghi nhớ ngữ cảnh
    - Các thao tác với ngăn xếp
      - create
      - isempty
      - push
      - pop
      - top
  - Hai cách khử đệ quy không kết thúc

Khử đệ quy

- Đệ quy không kết thúc (2)
  - Cách 1

```
Q(U)
begin
   if C(U) then
           B(U)
           Q(α(U))
           E(U)
   else
           T(U)
  endif
end
```

```
Q'(U)
begin
   create(S)
   while C(U) do
           B(U)
           push(S, U)
           U = α(U)
   endwhile
   T(U)
   while not isempty(S) do
           U = top(S)
           E(U)
           pop(S)
   endwhile
end
```

Khử đệ quy

- Đệ quy không kết thúc (3)
  - Cách 1
    - Minh hoạ

Gọi $Q(U_0)$
  $C(U_0)$ đúng
  $B(U_0)$
  Gọi $Q(\alpha(U_0))$
    $C(\alpha(U_0))$ đúng
    $B(\alpha(U_0))$
    Gọi $Q(\alpha(\alpha(U_0)))$
      $C(\alpha(\alpha(U_0)))$ sai
      $T(\alpha(\alpha(U_0)))$
    $E(\alpha(U_0))$
  $E(\alpha(\alpha(U_0))$

$\Rightarrow$

Gọi $Q'(U_0)$?

Khử đệ quy

- Đệ quy không kết thúc (4)
  - Cách 1
    - Ví dụ

```
T(n) // n≥0
begin
   if (n=0) then do nothing
   else
            T(n-1)
            print(n) //in n
  endif
end
```

```
T'(n) // n≥0
begin
   create(S)
   if (n=0) then do nothing
   else
            while (n>0) do
               push(S, n)
               n = n-1
            endwhile
            while (not isempty(S)) do
              n = top(S)
              print(n) //in n
              pop(S)
            endwhile
  endif
end
```

Khử đệ quy

- Đệ quy không kết thúc (5)
  - Cách 2

```
Q(U)
begin
   if C(U) then
           B(U)
           Q(α(U))
           E(U)
   else
           T(U)
  endif
end
```

```
Q'(U)
begin
   create(S)
   push(S, (newcall, U))
   while not isempty(S) do
           (state, V) = top(S)
           pop(S)
           if (state = newcall) then
              U = V
              if C(U) then
                 B(U)
                 push(S, (end, U))
                 push(S, (newcall, α(U)))
              else T(U)
              endif
           endif
           if (state = end) then
              U = V
              E(U)
           endif
   endwhile
end
```

Khử đệ quy

- Đệ quy không kết thúc (6)
  - Cách 2
    - Minh hoạ

Gọi $Q(U_0)$
  $C(U_0)$ đúng
  $B(U_0)$
  Gọi $Q(\alpha(U_0))$
    $C(\alpha(U_0))$ đúng
    $B(\alpha(U_0))$
    Gọi $Q(\alpha(\alpha(U_0)))$
      $C(\alpha(\alpha(U_0)))$ sai
      $T(\alpha(\alpha(U_0)))$
    $E(\alpha(U_0))$
  $E(\alpha(\alpha(U_0)))$

⇒

Gọi $Q'(U_0)$?

Khử đệ quy

- Đệ quy không kết thúc (7)
  - Cách 2
    - Ví dụ

```
T(n) // n≥0
begin
   if (n=0) then do nothing
   else
           T(n-1)
           print(n) //in n
  endif
end
```

```
T'(n)
begin
   create(S)
   push(S, (newcall, n))
   while not isempty(S) do
           (state, k) = top(S)
           pop(S)
           if (state = newcall) then
              if (k>0) then
                 push(S, (end, k))
                 push(S, (newcall, k-1))
              else do nothing
              endif
           endif
           if (state = end) then
              print(k)
           endif
   endwhile
end
```

Khử đệ quy

- Thuật toán sử dụng vòng lặp thường hiệu quả hơn
- Thuật toán đệ quy thường dễ xây dựng hơn
- Phần lớn các trình biên dịch có thể tự động khử đệ quy kết thúc
- Luôn có thể khử đệ quy của một thuật toán

Khử đệ quy

- Bài tập (1)
  - Bài 1
    - Định nghĩa dãy số Fibonaccci

      $Fib_0 = 1, Fib_1 = 1$

      $Fib_n = Fib_{n-1} + Fib_{n-2}$
    - Hãy thực hiện
      1. Xây dựng thuật toán đệ quy tính Fib(n)
      2. Chứng minh rằng độ phức tạp (bởi số phép cộng) của thuật toán là $\Omega(2^{n/2})$
      3. Xây dựng thuật toán tính cặp (Fib(n), Fib(n-1)) với n > 0
      4. Sử dụng thuật toán trong câu 3 để xây dựng thuật toán mới tính Fib(n)
      5. Đánh giá độ phức tạp (bởi số phép cộng) của thuật toán trên

Khử đệ quy

- Bài tập (2)
  - Bài 2

    Ước số chung lớn nhất của hai số nguyên dương được định nghĩa như sau

    - nếu $x = y$ thì $usc(x, y) = x$
    - nếu $x > y$ thì $usc(x, y) = usc(x-y, y)$
    - nếu $x < y$ thì $usc(x, y) = usc(x, y-x)$

    1. Xây dựng thuật toán đệ quy tính ước số chung lớn nhất hai số nguyên dương
    2. Khử đệ quy của thuật toán

  - Bài 3
    1. Xây dựng thuật toán đệ quy in dãy nhị phân tương ứng của một số nguyên
    2. Khử đệ quy thuật toán trên