Cấu trúc dữ liệu và Giải thuật

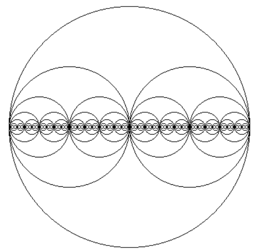
Chương II
Giải thuật đệ qui

Giải thuật đệ qui

Nội dung

- Các khái niệm cơ bản
- Một số ví dụ
- Phân tích giải thuật đệ qui

Một số đối tượng đệ qui



Một số đối tượng đệ qui

- Hàm đệ qui:
  - Là hàm được xác định phụ thuộc vào một biến nguyên không âm n theo sơ đồ:
    - Bước cơ sở : xác định giá trị hàm tại một giá trị n giá trị nhỏ nhất có thể của biến
    - Bước đệ qui: Cho giá trị f(k) , đưa ra qui tắc để tính f(k+1)

Một số đối tượng đệ qui

- Tập hợp đệ qui
  - Là tập được xác định như sau
    - Bước cơ sở: Định nghĩa tập cơ sở
    - Bước đệ qui: Xác định qui tắc để sản sinh tập mới từ tập đã có

Một số đối tượng đệ qui

- Định nghĩa đệ qui của xâu ký tự
  - A = bảng chữ cái, tập các xâu S trên bảng chữ cái A được xác định
    - Xâu rỗng là xâu trong S
    - Nếu w thuộc S và x là một ký tự trong A thì wx là xâu trong S

Một số đối tượng đệ qui

- Cây
  - Định nghĩa đệ qui của cây
    - Một nút tạo thành 1 cây
    - Nếu có n cây T_1, T_2, …, T_n với nút gốc là r_1, r_2, … , r_n; r là một nút có quan hệ cha-con r_1, r_2, … , r_n thì tồn tại một cây mới T nhận r làm gốc

Giải thuật đệ qui

- Định nghĩa: Giải thuật đệ qui là giải thuật được định nghĩa sử dụng chính giải thuật có dạng giống nó
- Cấu trúc của một thuật toán đệ qui bao gồm 2 bước
  - Bước cơ sở
    - Với những giá trị đầu vào đủ nhỏ, bài toán có thể giải quyết trực tiếp
  - Bước đệ qui
    - Lời gọi đến giải thuật đang định nghĩa
    - Lời gọi đệ qui phải được định nghĩa để nó tiến gần hơn đến bước cơ sở

Các dạng giải thuật đệ qui

- Đệ qui trực tiếp : A→ A
- Đệ qui gián tiếp: A→B →…→A
- Đệ qui đuôi
  - Lời gọi đệ qui luôn luôn nằm cuối cùng trong giải thuật

Giải thuật đệ qui

- Ví dụ: Hàm tính n!

$$Fact(n) = \begin{cases} 1 \; if \;\; n = 0 \\ n * Fact(n-1) \; if \;\; n > 0 \end{cases}$$

```
Function recursiveFactorial(n)
Begin
{Tính giá trị n! }
1.   if n = 0 then return 1
              else  return n*FACT(n-1);
2. End.
```

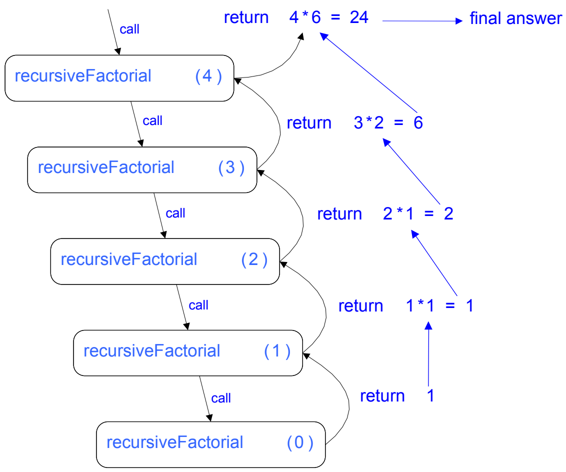
Trường hợp cơ sở

Lời gọi đệ qui

Tổ hợp kết quả

Giải thuật đệ qui

– Hình dung việc thực hiện giải thuật tính n!



Giải thuật đệ qui

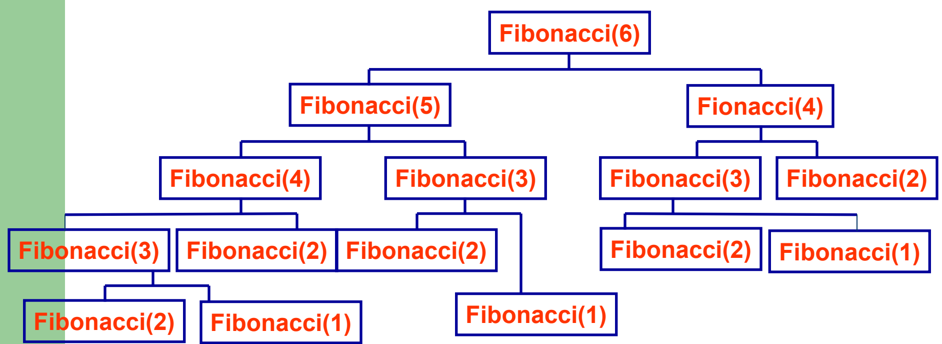
– Dãy Fibonacci

$$Fibonacci(n) = \begin{cases} 0 & if \;\; n = 0 \\ 1 & if \;\; n = 1 \\ Fibonacci(n-1) + Fibonacci(n-2) & otherwise \end{cases}$$

```
Function Fibonacci(n)
Begin
{Tính giá trị n! }
1.   if n <= 1 then return n
            else  return (Fibonacci(n-1)+Fibonacci(n-2));
2. End.
```

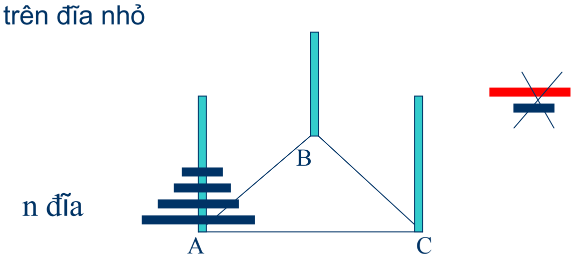
Giải thuật đệ qui

- Thực hiện tính Fibonacci(6)



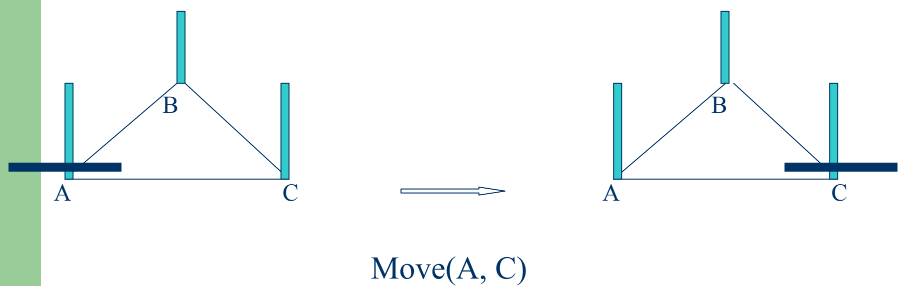
Giải thuật đệ qui

- Bài toán Tháp Hà nội
  - Có 3 cọc A, B, C và n đĩa có kích thước khác nhau
  - Ban đầu, các đĩa được xếp có thứ tự đĩa to ở trên, đĩa nhỏ ở dưới tại cọc A
  - Mục tiêu là chuyển n đĩa này sang cọc C với điều kiện mỗi lần được chuyển 1 đĩa, không được đặt đĩa to ở trên đĩa nhỏ



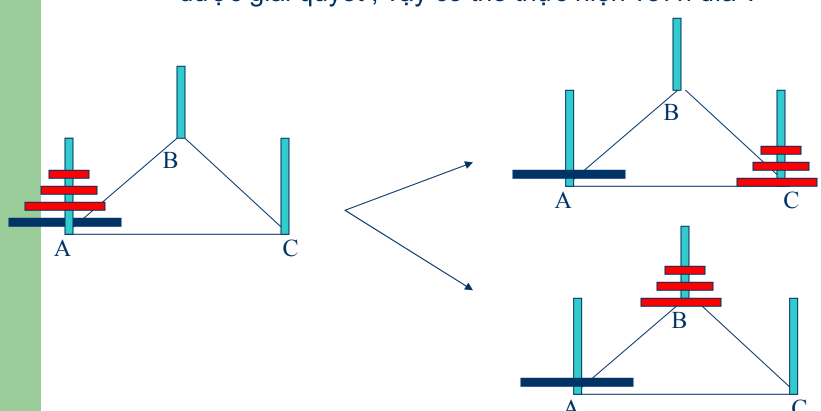
Giải thuật đệ qui

- Bước cơ sở : n <= 1, giải quyết trực tiếp

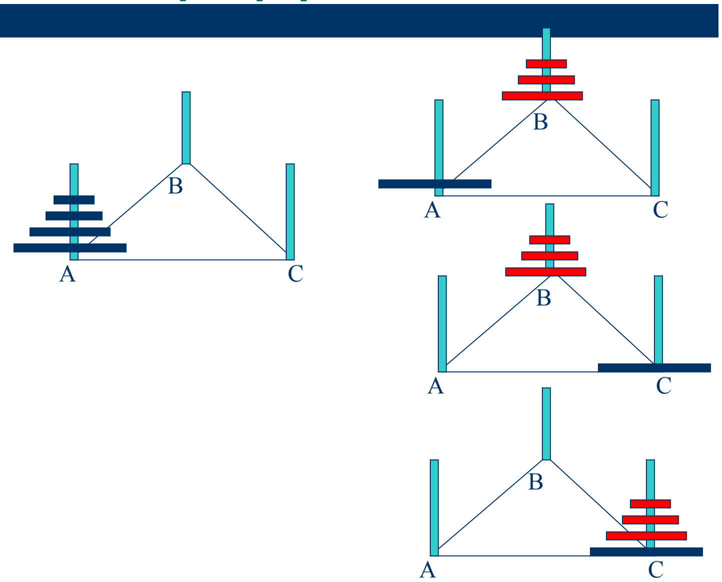


Giải thuật đệ qui

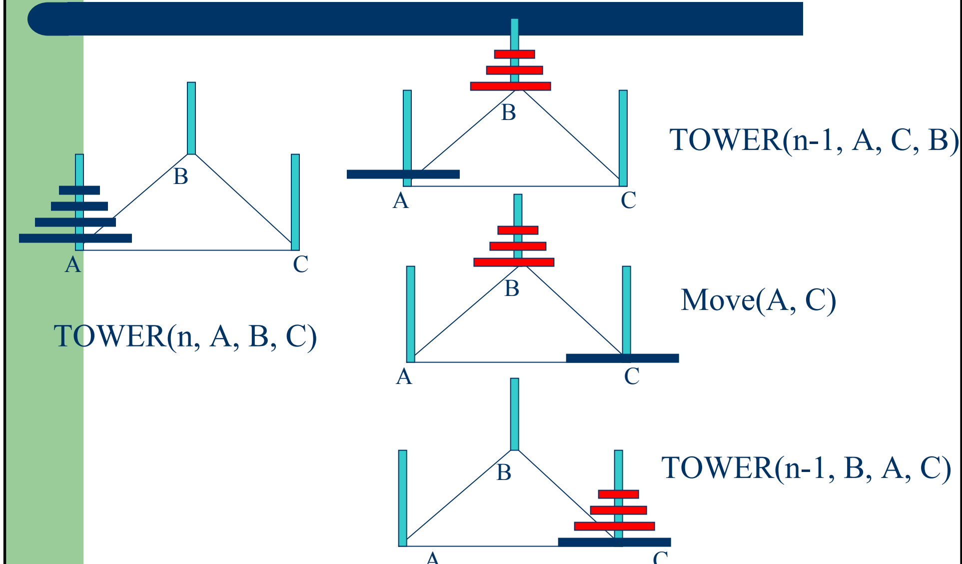
- Bước đệ qui: Giả sử rằng bài toán chuyển n-1 đĩa đã được giải quyết , vậy có thể thực hiện với n đĩa ?



Giải thuật đệ qui



Giải thuật đệ qui



Giải thuật đệ qui

```
Procedure TOWER( n, A, B, C)
Begin
{n là số đĩa ban đầu trên cọc A, cọc đầu tiên được chỉ
định là cọc chứa các đĩa cần chuyển, cọc thứ 2 là cọc
trung chuyển, cọc thứ 3 là cọc cần chuyển đĩa đến }
if n < 1 then return
else begin
                call TOWER(n-1, A, C, B)
                call MOVE(A,C)
                call TOWER( n-1, B, A, C)
    end
End
```

Phân tích thuật toán đệ qui

- Hàm thời gian thực hiện giải thuật T(n) là hàm đệ qui với tham số n

$$T(n) = \begin{cases} a & \text{if } n = 1 \\ 2T(n/2) + bn + c & \text{if } n > 1 \end{cases}$$

Phân tích thuật toán đệ qui

- Ví dụ 1
  - T(0) = 1
  - T(n) = 2 + T(n-1)

```
Procedure f(n)
{n là số nguyên không âm}
Begin
        if (n > 0) then begin
                writeln(n) ;
                Call f(n-1);
        end
End
```

Phân tích giải thuật đệ qui

- Ví dụ 2
  - Trường hợp cơ sở
    T(1) = 2
  - Đệ qui
    T(n) = c + 2* T(n/2)

```
Function  g( n)
Begin
   if (n =1)  then
      return 2;
   else
      return 3 * g(n / 2) + g( n / 2) + 5;

End.
```

Phân tích thời gian thực hiện giải thuật

- Cách thức giải công thức đệ qui của thời gian thực hiện giải thuật đệ qui
  - Phương pháp lặp

Phân tích giải thuật đệ qui

- Phương pháp lặp
  - Giải công thức đệ qui của thời gian thành một tổng các toán hạng cụ thể
    - Lặp lại việc thay thế hàm cho đến khi bắt gặp trường hợp cơ sở
    - Tính tổng

Phân tích giải thuật đệ qui

– Ví dụ: $T(n) = c + T(n/2)$

$T(n) = c + T(n/2)$
$= c + c + T(n/4)$
$= c + c + c + T(n/8)$

Giả sử $n = 2^k$

$T(n) = c + c + \dots + c + T(1)$

$= c\log n + T(1)$

Vậy ta có $T(n) = O(\log n)$

Phân tích giải thuật đệ qui

– Ví dụ: $T(n) = n + 2T(n/2)$

$T(n) = n + 2T(n/2)$
$= n + 2(n/2 + 2T(n/4))$
$= n + n + 4T(n/4)$
$= n + n + 4(n/4 + 2T(n/8))$
$= n + n + n + 8T(n/8)$
$\dots = in + 2^iT(n/2^i)$

Giả sử $n = 2^k$ thì ta sẽ rút gọn được

$T(n) = kn + 2^kT(1)$
$= n\log n + nT(1)$

Vậy $T(n) = O(n\log n)$

Phân tích giải thuật đệ qui

- Phân tích giải thuật tính giai thừa

T(0) = c
T(n) = b + T(n - 1)
= b + b + T(n - 2)
= b +b +b + T(n - 3)
…
= kb + T(n - k)
Khi k = n, ta có:
T(n) = nb + T(n - n)
= bn + T(0)
= bn + c.
Vậy T(n) = O(n).

```
Function recursiveFactorial(n)
Begin
{Tính giá trị n! }
1.   if n = 0 then return 1
           else  return n*FACT(n-1);
2. End.
```

Phân tích giải thuật đệ qui

- Phân tích giải thuật Tháp Hà Nội

T(1) = a
T(n) = b+ 2T(n-1)

```
Procedure TOWER( n, A, B, C)
Begin
if n < 1 then return
else begin
    call TOWER(n-1, A, C, B);
    call MOVE(A,C);
    call TOWER( n-1, B, A, C);
    end
End
```

Phân tích giải thuật đệ qui

$T(n) = 2T(n-1) + b$
$= 2[2T(n-2) + b] + b \qquad = 2^2 T(n-2) + 2b + b$
$= 2^2 [2T(n-3) + b] + 2b + b \qquad = 2^3 T(n-3) + 2^2 b + 2b + b$
$= 2^3 [2T(n-4) + b] + 2^2 b + 2b + b \qquad = 2^4 T(n-4) + 2^3 b + 2^2 b + 2^1 b + 2^0 b$
$= \ldots\ldots$
$= 2^k T(n-k) + b[2^{k-1} + 2^{k-2} + \ldots 2^1 + 2^0]$

$$= 2^k T(n-k) + b\sum_{i=0}^{k-1} 2^i$$

$$= 2^k T(n-k) + b(2^k - 1)$$

Khi n = k-1 ta có $T(n) = 2^{n-1} T(1) + b(2^{n-1} - 1)$

$$= (a+b)\, 2^{n-1} - b$$

$$= \left(\frac{a+b}{2}\right) 2^n - b$$

Khử đệ qui

- Một hàm đệ qui có thể được giải quyết tương đương bằng việc sử dụng vòng lặp và stack

Khử đệ qui

Algorithm P (val n <integer>)

```
1  if (n = 0)
   1    print ("Stop")
2  else
   1    Q(n)
   2    P(n - 1)
   3    R(n)
```

End P

Khử đệ qui

Algorithm P (n)

```
1  if (n = 0)
   1    print ("Stop")
2  else
   1    Q(n)
   2    P(n - 1)
   3    R(n)
```

End P

Algorithm P (n)

```
1  createStack (s)
2  loop (n > 0)
        1  Q(n)
        2  push(s, n)
        3  n = n - 1
3  print ("Stop")
4  loop (not emptyStack (s))
        1  popStack(s, n)
        2  R(n)
```

End P

Khử đệ qui

```
Algorithm  P (n)
1   if (n = 0)
    1     print("Stop")
2   else
    1     Q(n)
    2     P(n - 1)
End  P
```

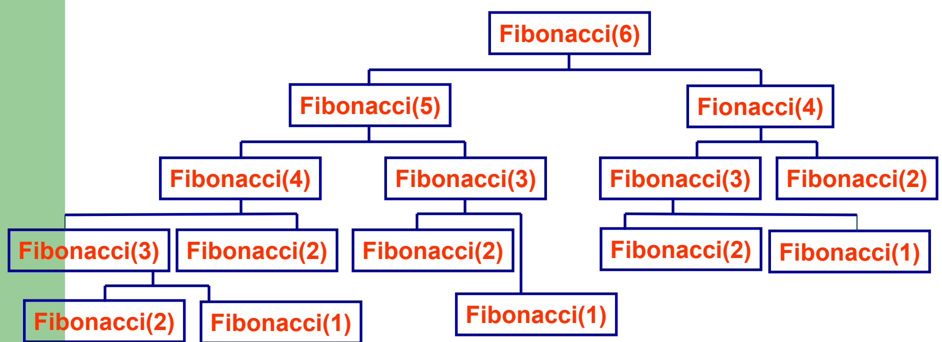
Khử đệ qui

```
Algorithm  P (n)
1   if (n = 0)
        1       print("Stop")
2       else
        1       Q(n)
        2       P(n - 1)
End  P
```

```
Algorithm  P (n)
1  loop (n > 0)
   1  Q(n)
   2  n = n - 1
2  print("Stop")
End  P
```

Đệ qui có nhớ

- Một kỹ thuật sử dụng khi trong các bài toán đệ qui có việc lặp đi lặp lại lời gọi một bài toán con nào đó
- Làm tăng tính hiệu quả của giải thuật đệ qui



Đệ qui có nhớ

– Ý tưởng khắc phục:
- Ghi lại lời giải của các bài toán con sử dụng một biến trong giải thuật
- Ví dụ: Bài toán tính hệ số nhị thức

$$C(n,0) = 1 \quad (n \geq 0)$$
$$C(n,n) = 1 \quad (n \geq 0)$$
$$C(n,k) = C(n-1,k-1) + C(n-1,k) \quad 0 < k < n$$

Đệ qui có nhớ

- Hàm đệ qui của bài toán

```
Function C(n,k)
Begin
    if ( k == 0) || (k ==n) then return 1;
    else return C(n-1,k-1) + C( n-1,k);
End
```

- Hàm đệ qui có nhớ

```
Function C(n,k)
Begin
    if  D[n,k] > 0 then return D[n,k];
    else D[n,k] = C(n-1,k-1) + C( n-1,k);
    return D[n,k];
End
```