ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

THIẾT KẾ VÀ ĐÁNH GIÁ THUẬT TOÁN

Bài 3
Đệ quy (*Recursive*)

Nguyễn Thị Hồng Minh
minhnth@gmail.com

Nội dung

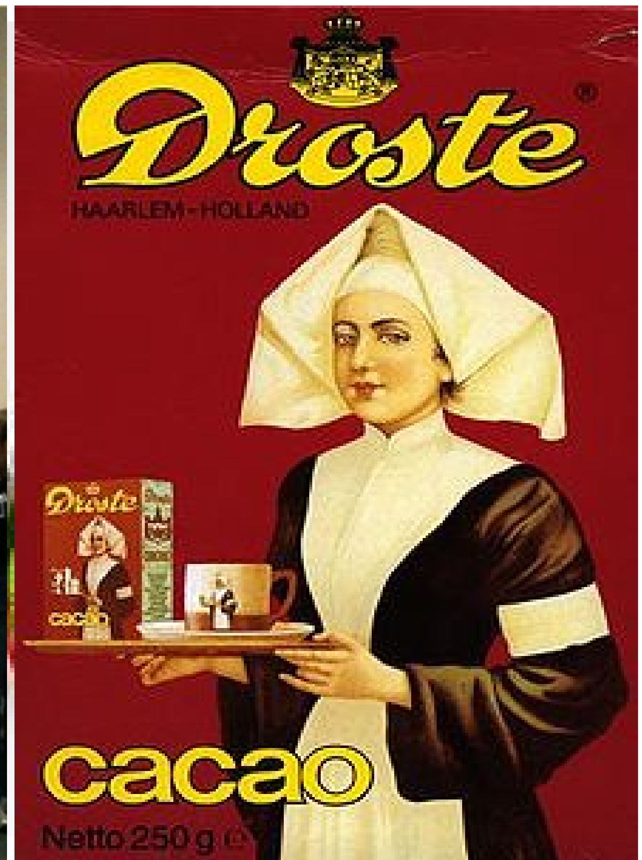
1. **Khái niệm về đệ quy**
2. **Lược đồ giải thuật đệ quy**
3. **Đánh giá độ phức tạp giải thuật đệ quy**
4. **Khử đệ quy**

Nội dung

1. **Khái niệm về đệ quy**
2. **Lược đồ giải thuật đệ quy**
3. **Đánh giá độ phức tạp giải thuật đệ quy**
4. **Khử đệ quy**

Khái niệm về giải thuật đệ quy

- **Hình ảnh đệ quy**

Khái niệm về giải thuật đệ quy

- **Giải thuật đệ quy**:
  - Nếu một bài toán T được thực hiện bằng giải thuật của một bài toán T' có dạng giống như T, thì đó là một giải thuật đệ quy.

- **Yêu cầu để giải thuật đệ quy thoả mãn tính dừng:**
  - T' phải "đơn giản hơn" T ;
  - T' giải được (không cần đệ quy) trong một số trường hợp nào đó (trường hợp suy biến.

- **Ví dụ:**
  - Tính giá trị S(n) = n!

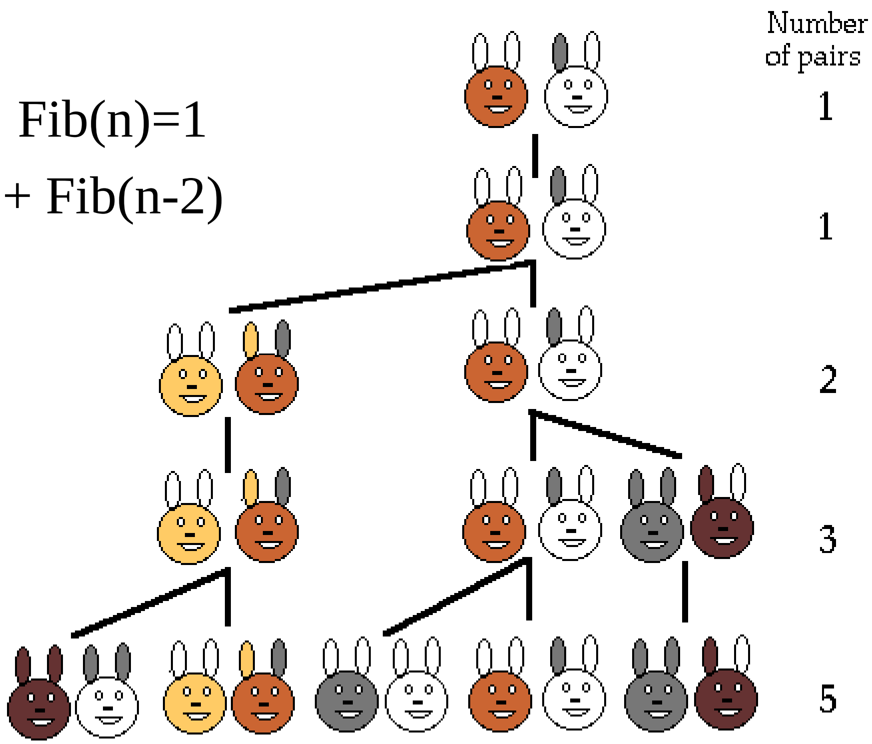
    Ví dụ: S(n)=n! ← S(n-1)=(n-1)! S ← (2-1)=(n-2)! ← …. ← S(1)=1! =1

Khái niệm về giải thuật đệ quy

- **Đặc trưng của các bài toán có thể giải bằng đệ quy**
  - Các bài toán phụ thuộc tham số;
  - Ứng với các giá trị đặc biệt nào đó của tham số thì bài toán có giải thuật để giải (trường hợp suy biến)
  - Trong trường hợp tổng quát bài toán có thể quy về dạng tương tự với một bộ giá trị mới của tham số và sau một số hữu hạn lần thì có thể dẫn tới trường hợp suy biến.

Khái niệm về giải thuật đệ quy

- **Một số ví dụ:**
  - Xác định số Fibonacci thứ n của dãy số Fibonacci 1, 1, 2, 3, 5, 8, 13 . . .
    - o Giải thuật tính Fib(n)
    - o TH suy biến n=1 hoặc n=2 thì Fib(n)=1
    - o TH tổng quát Fib(n) = Fib(n-1) + Fib(n-2)



Xem thêm về số Fibonacci:
http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html

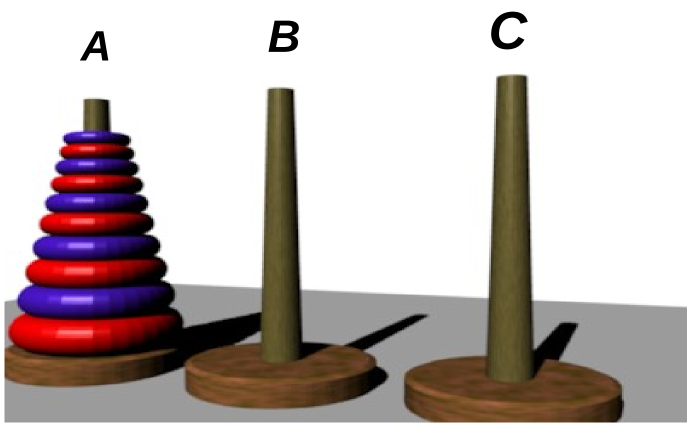
Khái niệm về giải thuật đệ quy

- **Một số ví dụ:**
  - Bài toán tháp Hà nội: Chuyển N tầng tháp từ A sang B lấy C làm trung gian.
    - Giải thuật chuyển tháp CHUYEN(N,A,B,C)
    - TH suy biến N=1 thì Chuyen1Tang(A,B)
    - TH tổng quát (N>1)

      CHUYEN(N-1,A,C,B);

      Chuyen1Tang(A,B);

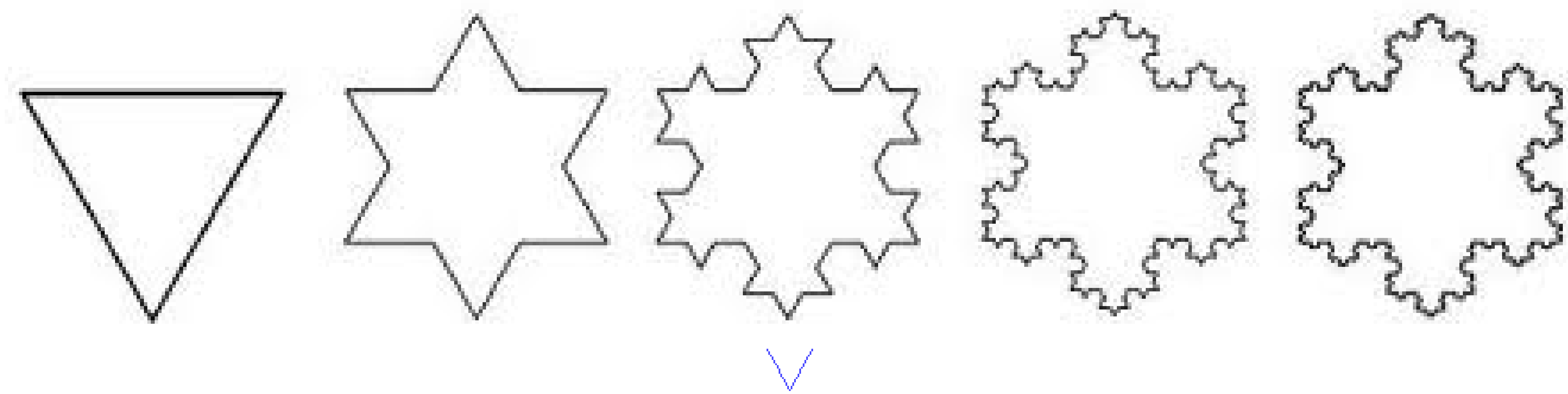
      CHUYEN(N-1,C,B,A);



Trong đó: Chuyen1Tang(X,Y) là thao tác chuyển một tầng tháp từ X →Y

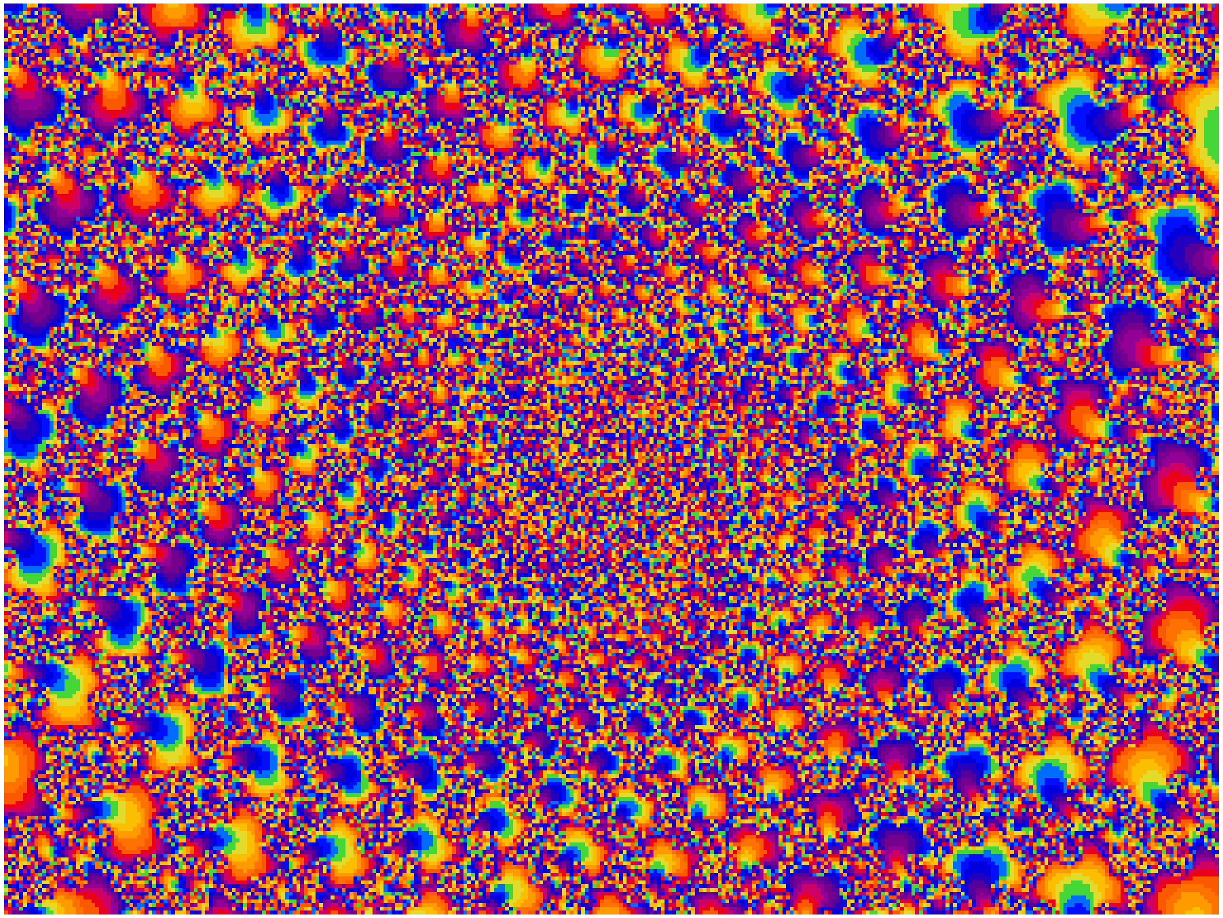
Khái niệm về giải thuật đệ quy

- **Đệ quy và hình học Fractal**
  - Fractal (Phân dạng): hình được dựng từ các hình đơn giản, lặp đi lặp lại với các tỷ lệ (vị trí, màu sắc…) khác nhau.



Khái niệm về giải thuật đệ quy

- **Đệ quy và hình học Fractal**



Nội dung

1. **Khái niệm về đệ quy**
2. **Lược đồ giải thuật đệ quy**
3. **Đánh giá độ phức tạp giải thuật đệ quy**
4. **Khử đệ quy**

Lược đồ giải thuật đệ quy

- **Lược đồ**

```
Recursive_Algorithm(pₙ) ≡
   if (pₙ = p₀)   //TH suy biến
       <Thực hiện giải thuật trường hợp suy biến>;
   else   //TH tổng quát
       <Lệnh;>
       Recursive_Algorithm(pₙ₋₁);
       <Lệnh;>
   endif
End.
```

$$p_n \leftarrow p_{n-1} \leftarrow p_{n-2} \leftarrow \dots \leftarrow p_o$$

Lược đồ giải thuật đệ quy

- **Ví dụ**
  - **Tính n!**

```
S(n):int ≡      //Hàm đệ quy tính giá trị n giai thừa
      if (n==1)
          S = 1;
      else
          S = n * S(n-1)
 End.
```

Lược đồ giải thuật đệ quy

- **Ví dụ**
  - **Tính Fib(n)**

```
Fib(n):int ≡ //Hàm đệ quy tính giá trị số Fib thứ n
     if (n==1||n==2)
         Fib = 1;
     else
         Fib = Fib(n-1)+Fib(n-2);
     End.
```

Lược đồ giải thuật đệ quy

- **Ví dụ**
  - **Tháp Hà Nội**

```
ChuyenThap(n,A,B,C) ≡ //Thủ tục đệ quy chuyển tháp n tầng từ A sang B
    if (n==1)
        Chuyen1Tang(A,B);
    else{
        ChuyenThap(n-1,A,C,B);
        Chuyen1Tang(A,B);
        ChuyenThap(n-1,C,B,A);
    }
  End.

  Chuyen1Tang(A,B) ≡ //Thủ tục chuyển 1 tầng tháp từ A sang B
      print(A,"→",B)
  End.
```

Nội dung

1. **Khái niệm về đệ quy**
2. **Lược đồ giải thuật đệ quy**
3. **Đánh giá độ phức tạp giải thuật đệ quy**
4. **Khử đệ quy**

Độ phức tạp giải thuật đệ quy

- Xác định quan hệ truy hồi trong phép đệ quy
  - Gọi $T(n)$ là độ phức tạp của giải thuật đệ quy với kích thước bài toán n
  - c = const là độ phức tạp thuật toán trong trường hợp suy biến khi $n=n_0$

    $T(n_0) = c$
  - Với $f(n)$ là hàm biến đổi tham số n, nếu giải thuật được thực hiện a lần bài toán con với tham số $f(n)$ thì

    $T(n) = a.T(f(n)) + g(n)$ với $g(n)$ là độ phức tạp các thao tác ngoài lời gọi đệ quy
- Công thức truy hồi xác định độ phức tạp giải thuật đệ quy

$$T(n) = \begin{cases} c & \text{khi } n = n_0 \\ a.T(f(n)) + g(n) & \text{trường hợp ngược lại} \end{cases}$$

Độ phức tạp giải thuật đệ quy

- Ví dụ tìm công thức truy hồi
  - Thuật toán S(n)

$$T_S(n) = \begin{cases} 1 & \text{khi } n = 1 \\ T_S(n-1) + 1 & \text{khi n} > 1 \end{cases}$$

  - Thuật toán Fib(n)

$$T_{Fib}(n) = \begin{cases} 1 & \text{khi } n = 1,2 \\ T_{Fib}(n-1) + T_{Fib}(n-2) + 1 & \text{khi n} > 1 \end{cases}$$

  - Thuật toán ChuyenThap(n,A,B,C)

$$T_{ChuyenThap}(n) = \begin{cases} 1 & \text{khi } n = 1 \\ 2T_{ChuyenThap}(n-1) + 1 & \text{khi n} > 1 \end{cases}$$

Độ phức tạp giải thuật đệ quy

- Giải công thức truy hồi
  - Sử dụng phép thế liên tiếp

$$T_S(n) = \begin{cases} 1 \quad \text{khi } n = 1 \\ T_S(n-1) + 1 \text{ khi n} > 1 \end{cases}$$

$$\begin{aligned} T_S(n) &= T_S(n-1) + 1 = T_S(n-2) + 1 + 1 \\ &= T_S(n-3) + 1 + 1 + 1 \\ &= \ldots \\ &= T_S(n-(n-1)) + \underbrace{1 + \ldots + 1}_{n-1 \text{ lân}} \\ &= 1 + 1 + \ldots + 1 = n \end{aligned}$$

$$\Rightarrow T_S(n) = O(n)$$

Độ phức tạp giải thuật đệ quy

- Giải công thức truy hồi
  - Định lí chính (General Theorem): Nếu n là lũy thừa của b thì công thức truy hồi:

$$T(n) = \begin{cases} 1 & \text{khi } n \le 1 \\ aT(n/b) + n^d & \text{khi } n > 1,\ a \ge 1, b > 1, d \ge 0 \end{cases}$$

Có lời giải là:

$$T(n) = \begin{cases} O(n^d) & \text{khi } a < b^d \\ O(n^d \log n) & \text{khi } a = b^d \\ O(n^{\log_b a}) & \text{khi } a > b^d \end{cases}$$

Chứng minh: Sử dụng phép thế liên tiếp.

Tham khảo Ian ParBerry's book, chương 4

Độ phức tạp giải thuật đệ quy

- Chứng minh định lí chính

$$T(n) = a^{\log_b n} T\left(n / b^{\log_b n}\right) + \ldots + a^2 \left(n / b^2\right)^d + a\left(n / b\right)^d + n^d$$

$$= n^{\log_b a} + n^d \sum_{j=0}^{\log_b n - 1} \left(a / b^d\right)^j$$

- Trường hợp 1: $a < b^d$

$$T(n) < n^{\log_b a} + n^d \frac{1}{1 - \left(a / b^d\right)} = O(n^d)$$

- Trường hợp 1: $a = b^d$

$$T(n) = n^{\log_b a} + n^d \log_b n = O(n^d \log_b n)$$

- Trường hợp 1: $a > b^d$

$$T(n) < n^{\log_b a} + n^d \left(a / b^d\right)^{\log_b n} \frac{1}{(a / b^d) - 1} = n^{\log_b a}\left(1 + \frac{1}{(a / b^d) - 1}\right) = O(n^{\log_b a})$$

Độ phức tạp giải thuật đệ quy

- Áp công thức truy hồi
  - $T(n) = 2T(n/2) + n^2$

    $a = 2, b = 2, d = 2, a < b^d \rightarrow T(n) = O(n^d) = O(n^2)$
  - $T(n) = 2T(n/2) + n$

    $a = 2, b = 2, d = 1, a = b^d \rightarrow T(n) = O(n^d \log_b n) = O(n \log n)$
  - $T(n) = 2T(n/2) + 1$

    $a = 2, b = 2, d = 0, a > b^d \rightarrow T(n) = O(n^{\log_b a}) = O(n)$

Độ phức tạp giải thuật đệ quy

- Áp công thức truy hồi
  - $T(n) = 4T(n/2) + n^3$

    $a = 4, b = 2, d = 3, a < b^d \rightarrow T(n) = O(n^d) = O(n^3)$

  - $T(n) = 4T(n/2) + n^2$

    $a = 4, b = 2, d = 1, a = b^d \rightarrow T(n) = O(n^d \log n) = O(n^2 \log n)$

  - $T(n) = 4T(n/2) + n$

    $a = 4, b = 2, d = 1, a > b^d \rightarrow T(n) = O(n^{\log_b a}) = O(n^2)$

Nội dung

1. **Khái niệm về đệ quy**
2. **Lược đồ giải thuật đệ quy**
3. **Đánh giá độ phức tạp giải thuật đệ quy**
4. **Khử đệ quy**

Khử đệ quy

- **Khái niệm**

  Ưu điểm của các giải thuật đệ quy: Ngắn gọn, trong sáng

  Nhược điểm: Lời gọi thực hiện phức tạp hơn, đôi khi khó hình dung

  Lí do: Mỗi lần gọi đệ quy, một tập các giá trị cục bộ được tạo ra để lưu các giá trị trung gian phục vụ cho các lượt tính ngược lại

  Thay đổi: Chủ động quản lí các giá trị trung gian bằng ngăn xếp, thay lời gọi đệ quy bằng vòng lặp $\Rightarrow$ thuật toán khử đệ quy

Khử đệ quy một thuật toán đệ quy là thay thế thuật toán đệ quy bằng một thuật toán ***tương đương*** nhưng bỏ đi các lời gọi đệ quy và thay vào đó bằng việc sử dụng các vòng lặp cùng với sự hỗ trợ của ngăn xếp dữ liệu (*stack*)

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[AP]***

```
P(x) ≡
  if (E(x))
      D(x);
  else
      A(x);
      P(f(x));
  endif
End.
```

Triển khai chi tiết

```
P(x) ≡
     if E(x)
        D(x);
     else
        A(x);
        x = f(x);
        if (E(x))
           D(x);
        else
           A(x);
           x = f(x);
           ...
        endif
     endif
End.
```

E(x): điều kiện suy biến

D(x): lời giải trong trường hợp suy biến

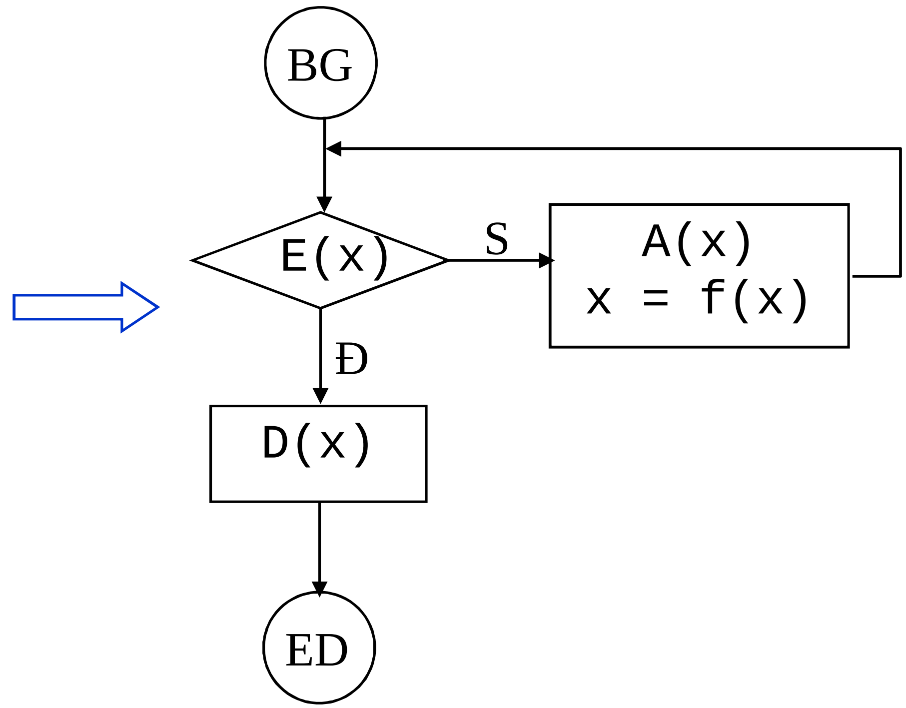
A(x): thao tác trước lời gọi đệ quy

f(x): hàm biến đổi tham số lời gọi đệ quy

Khử đệ quy

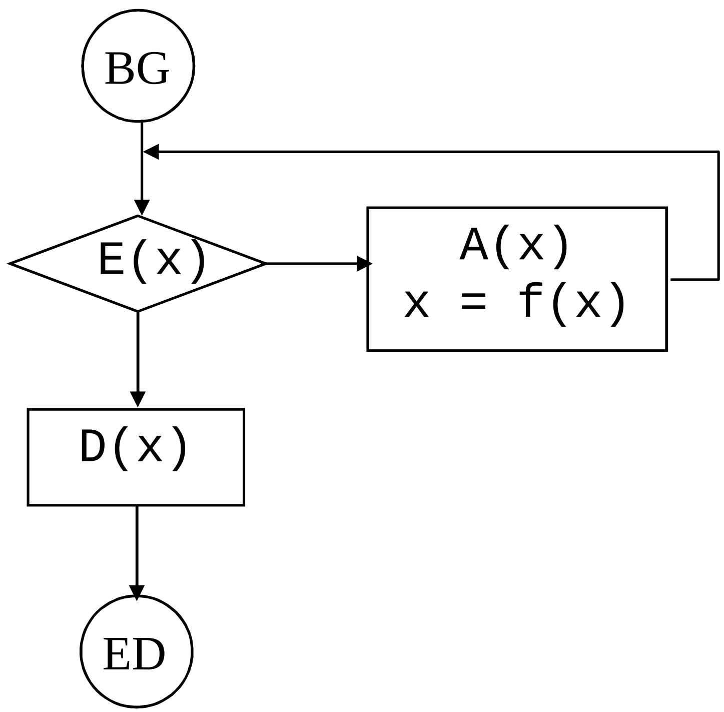
- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[AP]*** - sơ đồ thực hiện

```
P(x) ≡
     if E(x)
        D(x)
     else
        A(x)
        x = f(x)
        if (E(x))
           D(x)
        else
           A(x)
           x = f(x)
           ...
       endif
     endif
End.
```



Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[AP]*** - khử đệ quy




```
P(x) ≡
      while not E(x) do
          A(x);
          x := f(x);
      endwhile
      D(x);
End.
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[AP]***

Ví dụ: phân tích số nguyên thành các thừa số nguyên tố, in theo thứ tự tăng dần.

```
PhanTich(n) ≡
        d = 2;
        if (n=1)
           exit
        else
           while (n mod d <> 0)do
              d := d+1
           write(d);
           PhanTich(n div d);
       endif
End.
```

Các thành phần của giải thuật đệ quy dạng ED[AP] của thuật toán PhanTich

```
Tham số (x) ≡(n)
E(x)≡  n=1
D(x)≡  exit
A(x)≡  d = 2;
       while (n mod d<>0) do
          d := d+1
       write(d);
f(x)≡  f(n) = n div d
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[AP]***

```
P(x) ≡
      while not E(x) do
          A(x);
          x := f(x);
      endwhile
      D(x);
End.
```

```
PhanTich(n)≡
      d = 2;
      while (n>1) do
         while (n mod d <> 0) do
              d := d+1;
         write(d);
         n = n div d;
      endwhile;
      exit;
End.
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APB]***

```
P(x) ≡
   if (E(x))
         D(x);
  else
         A(x);
         P(f(x));
         B(x);
  endif
End.
```

E(x): điều kiện suy biến

D(x): lời giải trong trường hợp suy biến

A(x), B(x): thao tác ngoài lời gọi đệ quy

f(x): hàm biến đổi tham số lời gọi đệ quy

⟹ *Triển khai chi tiết*

```
P(x) ≡
     if E(x)
        D(x);
     else
        A(x);
        x = f(x);     //f1
        if (E(x))
           D(x);
        else
           A(x);
           x = f(x); //f2
           ...
       endif
       B(x) //x=f2(x)
     endif
     B(x) //x=f1(x)
End.
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APB]***

```
P(x) ≡
     if E(x)
        D(x);
     else
        A(x);
        x = f(x);    //f1
        if (E(x))
           D(x);
        else
           A(x);
           x = f(x); //f2
           ...
       endif
       B(x) //x=f2(x)
     endif
     B(x) //x=f1(x)
End.
```

P kết thúc sau k+1 lần gọi:

- $E(f^k(x))$ = true với

  $f^k(x) = f(f\ldots(f(x))\ldots)$

  $f^o(x) = x$

- Sau khi thực hiện $D(f^k(x))$ sẽ phải thực hiện $B(f^k(x))$

- Tiếp đó sẽ thực hiện $B(f^{k-1}(x))$… và cuối cùng là B(x).

Tóm lại, giá trị của x mỗi lần biến đổi x=f(x) được cất giữ thành chồng (stack) rồi dỡ ra theo chiều từ trên xuống để thực hiện B(x)

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APB]***

Sử dụng ngăn xếp *S*, kiểu phần tử phù hợp với *x* và hai vòng lặp để xây dựng thuật toán khử đệ quy:

```
P(x)≡
      Start (S );          //Khởi tạo ngăn xếp S
      while not E(x) do // Vòng lặp khử lời gọi đệ quy
         A(x);
         Push(x,S);        //Cất x vào ngăn xếp
         x := f(x);
      endwhile;
      D(x)
      while not Is_Empty(S) do  //Vòng lặp gỡ ngăn xếp
         Pick(S,x);       //Lấy x vào ngăn xếp
         B(x)
      endwhile;
End.
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APB]***

Ví dụ: Đổi số thập phân sang nhị phân

```
Int2Bin(n)≡
   if (n>0)
       Int2Bin(n div 2);
       Write(n mod 2);
   endif;
End.
```

Các thành phần của giải thuật đệ quy dạng ED[APB] của thuật toán Int2Bin

```
Tham số (x) ≡(n)
E(x)≡ n=0
D(x)≡ ∅
A(x)≡  ∅
f(x)≡ f(n) = n div 2
B(x)≡ write(n mod 2)
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APB]***

Ví dụ: Đổi số thập phân sang nhị phân

```
Int2Bin(n)≡
   if (n>0)
       Int2Bin(n div 2);
       Write(n mod 2);
   endif;
End.
```

Sử dụng mảng làm ngăn xếp

```
int S[50];
int st;
```

```
IntToBin(n)  ≡
    st = 0; //Start(S)
    while (n>0) do
      st++;
      S[st]=n mod 2; //Push(S)
      n = n div 2;
    endwhile;
    while (st<>0) do
      write(s[st]); //Pick(S)
      st--;
    endwhile;
End.
```

Dùng xâu kí tự làm ngăn xếp sẽ cho chương trình ngắn gọn hơn

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APBP]***

```
P(x) ≡
  if (E(x))
       D(x);
  else
       A(x);
       P(f(x));
       B(x);
        P(g(x));
  endif
End.
```

E(x): điều kiện suy biến

D(x): lời giải trong trường hợp suy biến

A(x), B(x): thao tác ngoài lời gọi đệ quy

f(x), g(x): hàm biến đổi tham số của hai
lời gọi đệ quy

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APBP]***

```
P(x) ≡
   if (E(x))
         D(x);
   else
         A(x);
         P(f(x));
         B(x);
          P(g(x));
   endif
 End.
```

```
P(x)≡
   Start(S);
   Push((x,1),S);
   repeat
      while not E(x) do
        A(x);
        Push((x,2),S);
        x := f(x);
      endwhile;
      D(x);
      Pick (S,(x,m));
      if (m=2)
        B(x); x= g(x);
      endif;
    until (m=1);
End.
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - ***Dạng ED[APBP]***

Ví dụ: Bài toán tháp Hà Nội

```
ChuyenThap(n,A,B,C) ≡
    if (n==1)
        Chuyen1Tang(A,B);
    else
        ChuyenThap(n-1,A,C,B);
        Chuyen1Tang(A,B);
        ChuyenThap(n-1,C,B,A);
    endif
End.
```

```
P(x) ≡ ChuyenThap(n,A,B,C)
E(x) ≡   (n=1)
D(x) ≡ Chuyen1Tang(A,B)
A(x) ≡ ∅
f(x) ≡ f(n,A,B,C)=(n-1,A,C,B)
     do đó : n→n-1, A→A,
             B → C, C → B
B(x) ≡ Chuyen1Tang(A,B)
g(x) ≡ g(n,A,B,C)=(n-1,C,B,A)
     do đó : n → n-1, A → C,
             B → B, C → A
```

Khử đệ quy

- Khử đệ quy một số dạng thường gặp
  - *Dạng ED[APBP]*

Ví dụ: Bài toán tháp Hà Nội

```
P(x) ≡ ChuyenThap(n,A,B,C)
E(x) ≡   (n=1)
D(x) ≡ Chuyen1Tang(A,B)
A(x) ≡ ∅
f(x) ≡ f(n,A,B,C)=(n-1,A,C,B)
     do đó : n→n-1, A→A,
             B→C, C→B
B(x) ≡ Chuyen1Tang(A,B)
g(x) ≡ g(n,A,B,C)=(n-1,C,B,A)
     do đó : n→n-1, A→C,
             B→B, C→A
```

```
ChuyenThap(n,A,B,C) ≡
    Start(S);
    Push((n,A,B,C,1),S);
    repeat
      while (n>1) do
        Push((n,A,B,C,2),S);
        n--;
        DoiCho(B,C);
      endwhile;
      Chuyen1Tang(A,B);
      Pick(S,(n,A,B,C,m))
      if (m=2)
         Chuyen1Tang(A,B);
         n--;
         DoiCho(A,C);
      endif
    until (m=1);
End.
```