Chương 2: Thuật toán đệ quy

Trịnh Anh Phúc, Nguyễn Đức Nghĩa [1]

[1]Bộ môn Khoa Học Máy Tính, Viện CNTT & TT,
Trường Đại Học Bách Khoa Hà Nội.

Ngày 14 tháng 7 năm 2015

Giới thiệu

1. Khái niệm đệ quy
   - Hàm đệ qui
   - Tập hợp được xác định đệ qui
2. Thuật toán đệ qui
3. Một số ví dụ minh họa
4. Phân tích thuật toán đệ qui
5. Chứng minh tính đúng đắn của thuật toán đệ qui
6. Thuật toán quay lui
   - Bài toán xếp hậu
   - Bài toán mã tuần

Thuật toán đệ qui

Khái niệm đệ qui

Trong thực tế chúng ta thường gặp những đối tượng đệ quy bao gồm chính nó hoặc được định nghĩa bởi chính nó. Ta nói các đối tượng đó được xác định một cách đệ qui

- Điểm quân số
- Các hàm được định nghĩa đệ qui
- Tập hợp được định nghĩa đệ qui
- Định nghĩa đệ qui về cây
- Fractal ....

Hàm đệ qui (resursive functions)

Định nghĩa

Các hàm đệ qui được xác định bởi số nguyên không âm n theo sơ đồ

- **Bước cơ sở** (Basic step) : Xác định giá trị hàm tại thời điểm $n = 0$ hay $f(0)$
- **Bước đệ qui** (Recursive step) : Cho giá trị của hàm $f(k)$ tại $k \leq n$ đưa ra qui tắc tính giá trị của $f(n+1)$.

Hàm đệ qui (resursive functions)

- VD1 :

$$f(0) = 3 \quad n = 0$$
$$f(n+1) = 2f(n) + 3 \quad n > 0$$

- VD2 :

$$f(0) = 1$$
$$f(n+1) = f(n) \times (n+1)$$

- VD3 : Định nghĩa đệ qui tổng $s_n = \sum_{i=1}^{n} a_i$

$$s_1 = a_1$$
$$s_n = s_{n-1} + a_n$$

Tập hợp được xác định đệ qui

Định nghĩa

Tập hợp cũng có thể được xác định đệ qui theo sơ đồ tương tự như hàm đệ qui

- **Bước cơ sở** : Định nghĩa tập cơ sở.
- **Bước đệ qui** : Xác định các qui tắc để sản sinh tập mới từ các tập đã có.

Tập hợp được xác định đệ qui (tiếp)

- **VD1** : Xét tập S đc định nghĩa đệ qui như sau
  - **Bước cơ sở** : 3 là phần tử của tập S.
  - **Bước đệ qui** : Nếu x thuộc S và y thuộc S thì $x + y$ thuộc S.

  Vậy tập S có phân tử đc tạo một cách đệ qui 3, 3+3 = 6, 3+6 = 9, 6+6 = 12, $\cdots$
- **VD2** : Định nghĩa đệ qui của xâu như sau : $\sum$ = Bảng chữ cái (alphabet) thì tập $\sum^*$ các xâu (string) trên bảng chữ cái A đc định nghĩa đệ qui như sau :
  - **Bước cơ sở** : Xâu rỗng các phần tử của $\sum^*$
  - **Bước đệ qui** : Nếu w thuộc $\sum^*$ và x thuộc $A \to wx$ thuộc $\sum^*$.

  Chẳng hạn tập các xâu nhị phân **B** thu được khi $\sum = \{0, 1\}$ bắt đầu từ xâu rỗng, 0 , 1, 00, 01, 10, 11

Tập hợp được xác định đệ qui (tiếp)

- **VD3** : Công thức toán học
  Một công thức hợp lệ của các biến, các số và các phép toán từ tập {+,-,*,/} có thể định nghĩa đệ qui như sau
  - **Bước cơ sở** : x là công thức hợp lệ nếu x là biến hoặc số.
  - **Bước đệ qui** : Nếu f, g là công thức hợp lệ thì

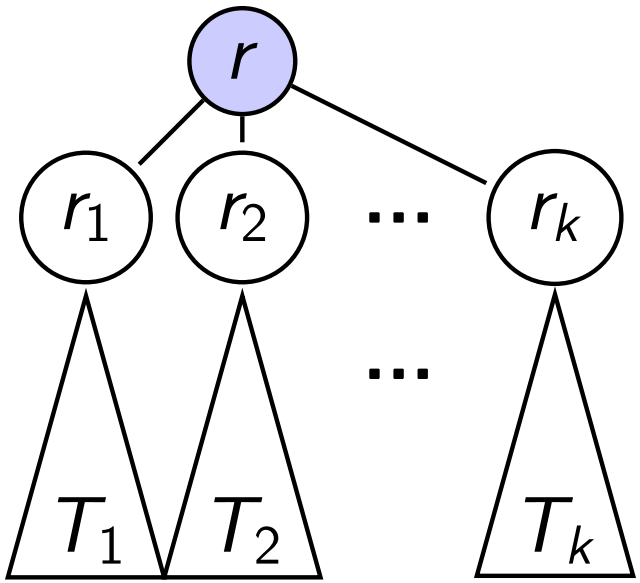
$$(f+g), (f-g), (f*g), (f/g)$$

    là công thức hợp lệ.

  Ta có các công thức hợp lệ sau
  (x-y)
  ((z/3)-y),((z/3) - (6+5))
  ((z/3)-(6+5))
  ((z/(2*3)) - (6+5))

Tập hợp được xác định đệ qui (tiếp)

- **VD4** : Cây có gốc r được định nghĩa đệ qui như sau
  - **Bước cơ sở** : Một nút là một cây có gốc r.
  - **Bước đệ qui** : Giả sử $T_1, T_2, \cdots, T_k$ là các cây với gốc là $r_1, \cdots, r_k$. Vậy nếu ta nối gốc mới tạo r với mỗi một trong số các gốc $r_1, \cdots, r_k$ bởi một cạnh tương ứng, ta lại thu được một cây mới có gốc vẫn là r.



1 Khái niệm đệ quy
- Hàm đệ qui
- Tập hợp được xác định đệ qui

2 Thuật toán đệ qui

3 Một số ví dụ minh họa

4 Phân tích thuật toán đệ qui

5 Chứng minh tính đúng đắn của thuật toán đệ qui

6 Thuật toán quay lui
- Bài toán xếp hậu
- Bài toán mã tuần

Thuật toán đệ qui

Thuật toán đệ qui

Định nghĩa : *Thuật toán đệ qui là thuật toán tự gọi đến chính mình với đầu vào kích thước nhỏ hơn.*

- Tất nhiên việc sử dụng thủ tục đệ qui thích hợp với xử lý dữ liệu, tập hợp, hàm, cây được định nghĩa cũng một cách đệ qui như các ví dụ vừa nêu.
- Các thuật toán được phát triển dựa trên phương pháp chia-để-trị (Divide and Conquer) thông thường được mô tả dưới dạng đệ qui.
- Hầu hết các ngôn ngữ lập trình đều cho phép gọi đệ qui của hàm - lệnh gọi đến chính nó trong thân chương trình.

Thuật toán đệ qui

Cấu trúc của thuật toán đệ qui

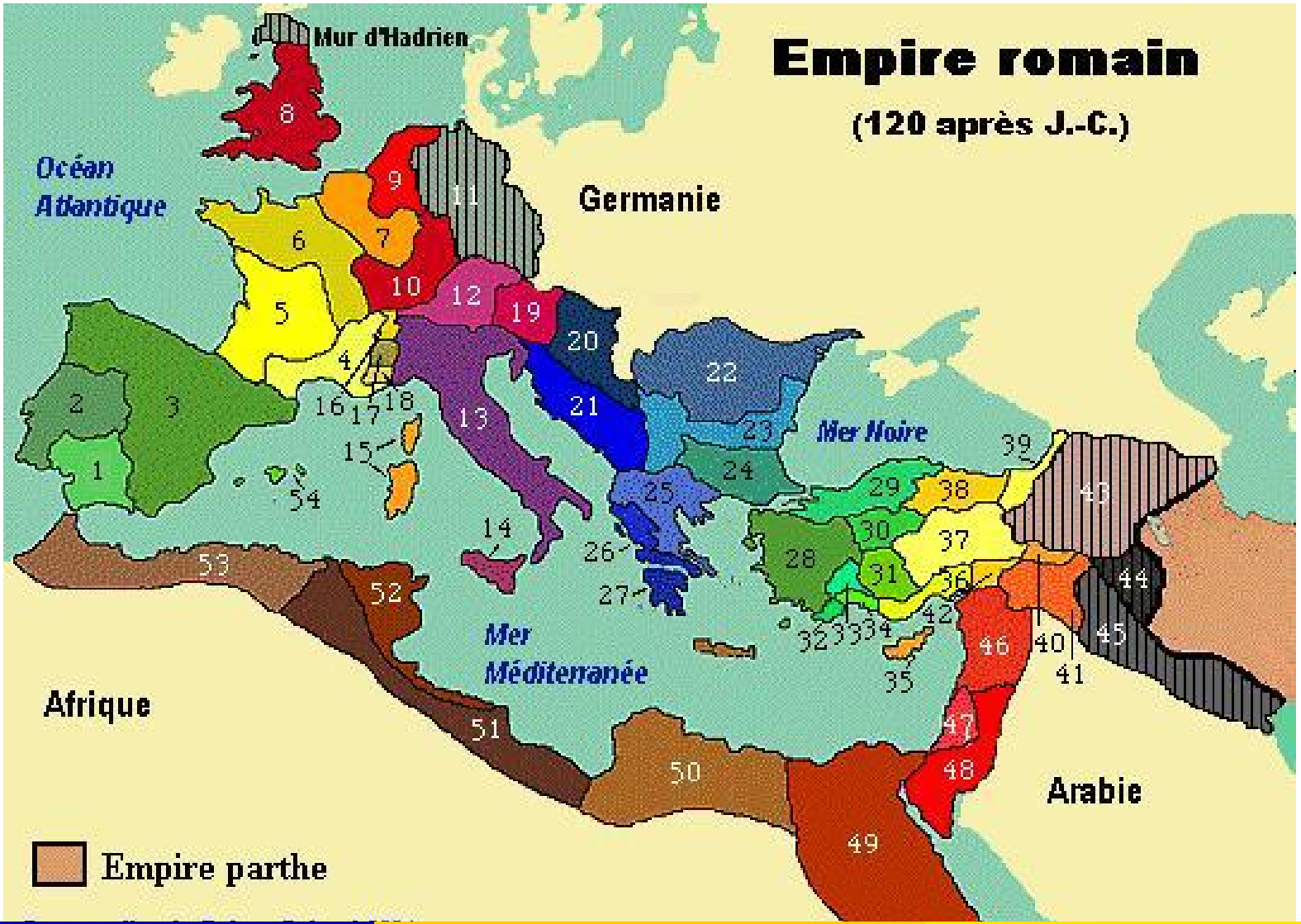
```
Function RecAlg(input)
begin
    if (kích thước đầu vào là nhỏ nhất) then
        thực hiện bước cơ sở /* giải bài toán với kích thước cơ sở*/
    else
        RegAlg(input với đầu vào nhỏ hơn);/* các bước đệ qui*/
        Tổ hợp lời giải của các bài toán con để thu được lời-giải;
        return(lời-giải)
    endif
end;
```

Thuật toán đệ qui

Ví dụ phương pháp chia-để-trị : cắt bánh ngọt

Thuật toán đệ qui

Ví dụ phương pháp chia-để-trị : quản lý hành chính



Thuật toán đệ qui

Phương pháp chia-để-trị

Phương pháp chia-để-trị (divide et impera) là một trong những phương pháp chính dùng để thiết kế thuật toán có tính đệ qui, nó bao gồm 3 thao tác chính như sau

- Chia (Divide) bài toán cần giải thành các bài toán con
  - Bài toán con có kích thước nhỏ hơn và có cũng dạng với bài toán cần giải.
- Trị (Conquer) các bài toán con
  - Giải các bài toán con một cách đệ qui
  - Bài toán con có kích thước đủ nhỏ sẽ được giải thực tiếp
- Tổ hợp (Combine) lời giải của các bài toán con
  - Thu đc lời giải của bài toán xuất phát.

Thuật toán đệ qui

Phương pháp chia-để-trị đc trình bày trong thủ tục đệ qui sau đây

Procedure D-and-C(n)
    if $n \leq n_0$ **then**
        Giải bài toán một cách trực tiếp
    else
        Chia bài toán thành a bài toán con kích thước n/b;
        for (mỗi bài toán trong a bài toán con) **do** D-and-C(n/b) **endfor**
        Tổng hợp lời giải của a bài toán con để thu được lời giải của bài toán gốc;
    endif
End

Thuật toán đệ qui

Phương pháp chia để trị đc trình bày trong thủ tục đệ qui (tiếp)

Các thông số quan trọng của thuật toán

- n_0 kích thước nhỏ nhất của bài toán con (còn gọi là neo đệ qui). Bài toán con với kích thước n_0 sẽ được giải trực tiếp.
- a - số lượng bài toán con cần giải.
- b - liên quan đến kích thước của bài toán con được chia.

Thuật toán đệ qui

Phương pháp chia để trị trong thủ tục đệ qui (tiếp)

Ví dụ về **sắp xếp trộn** bài toán : sắp xếp mảng không thứ tự A[1..n]

- Chia (Divide)
  - Chia một dãy gồm n phần tử cần sắp xếp ra hai dãy, mỗi dãy gồm $n/2$ phần tử
- Trị (Conquer)
  - Sắp xếp mỗi dãy con một cách đệ qui sử dụng **sắp xếp trộn**
  - Khi dãy chỉ còn một phần tử thì trả lại phần tử này
- Tổ hợp (Combine)
  - Trộn hai dãy con được sắp xếp để thu được dãy được sắp xếp gồm tất cả các phần tử của cả hai dãy con

Thuật toán đệ qui

Phương pháp chia để trị trong thủ tục đệ qui (tiếp)

Thuật toán đệ qui được mô tả như sau

```
MERGE-SORT(A,p,r)
if (p < r) /* Kiểm tra điều kiện neo đệ qui */
then
    q ← ⌊(p + r)/2⌋ /* Chia (Divide)*/
    MERGE-SORT(A,p,q) /*Trị (Conquer)*/
    MERGE-SORT(A,q+1,r) /*Trị (Conquer)*/
    MERGE(A,p,q,r) /* Tổ hợp (Combine)*/
endif
End
```

Lệnh gọi thuật toán là : MERGE-SORT(A,1,n)

Thuật toán đệ qui

Hàm trộn MERGE(A,p,q,r)

Đầu vào : Mảng A và các chỉ số p, q, r sao cho $p \leq q < r$ trong đó các mảng con $A[p \cdots q]$ và $A[q+1 \cdots r]$
Đầu ra : Mảng con được sắp xếp $A[p \cdots r]$

Ý tưởng của thuật toán trộn :

- Có hai dãy con đã được sắp xếp
  - Chọn phần tử nhỏ hơn ở hai đầu dãy
  - Loại nó khỏi dãy con tương ứng và đưa vào dãy kết quả
- Lặp lại khi một trong hai dãy trở thành dãy rỗng
- Các phần tử còn lại của dãy con kia sẽ được đưa nốt vào đuôi của dãy kết quả

Thuật toán đệ qui

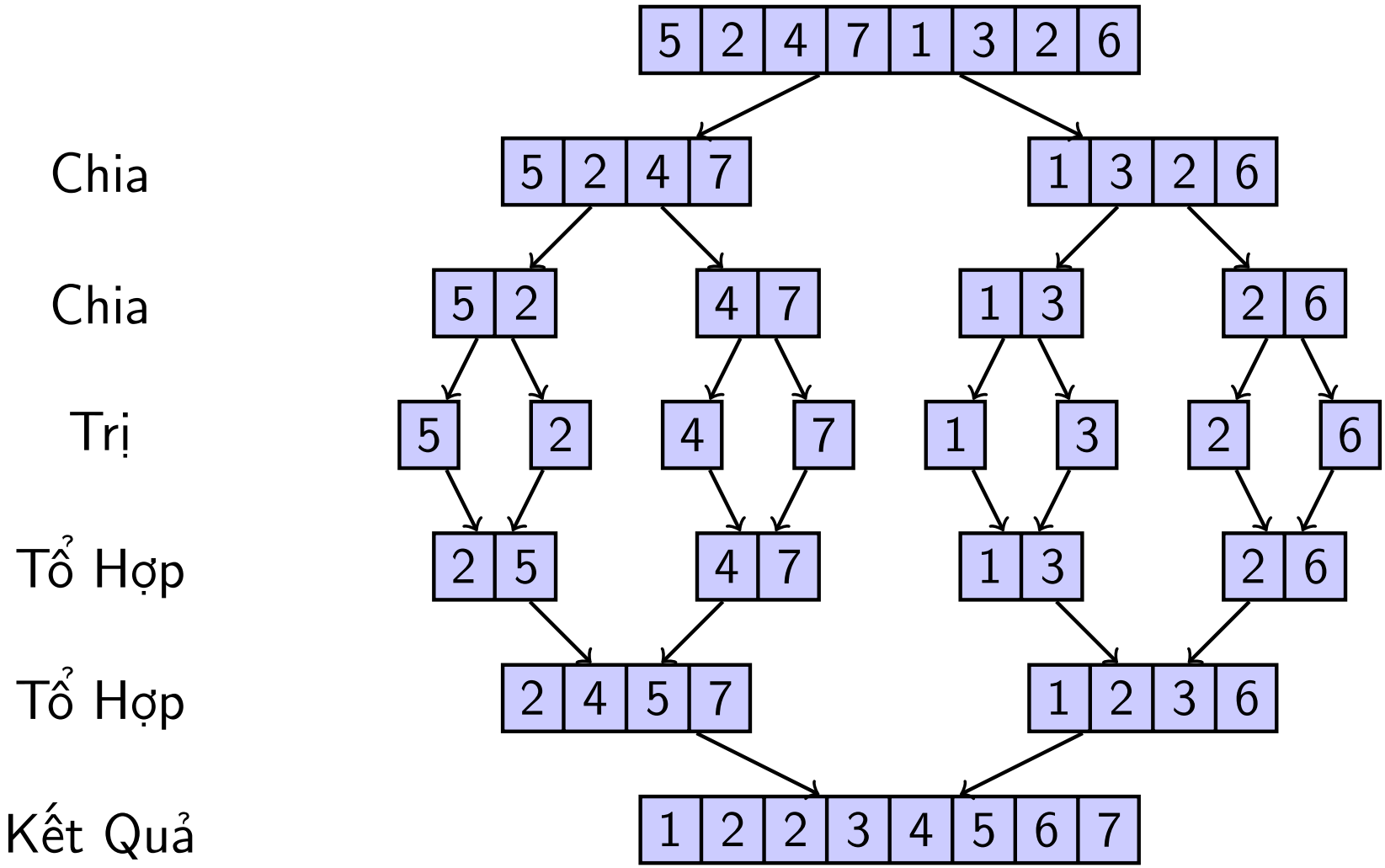
Sắp xếp trộn (tiếp)

MERGE(A,p,q,r)

1. Tính i $\leftarrow$ p và j $\leftarrow$ q+1 sao cho i là phần tử đầu tiên trỏ vào mảng con trái L[1..last1] và j là phần tử đầu tiên trỏ vào mảng con bên phải R[1..last2] còn last1 $\leftarrow$ q và last2 $\leftarrow$ r.
2. i $\leftarrow$ 1; j $\leftarrow$ 1;
3. **for** k$\leftarrow$ p **to** r **do**
4. **if** (L[i] $\leq$ R[j]) **then**
5. A[k] $\leftarrow$ L[i]
6. i $\leftarrow$ i+ 1
7. **else** A[k] $\leftarrow$ R[j]
8. j$\leftarrow$ j+1
9. **endif**

Thuật toán đệ qui

Minh họa sắp xếp trộn của dãy {5,2,4,7,1,3,2,6}.



1 Khái niệm đệ quy
- Hàm đệ qui
- Tập hợp được xác định đệ qui

2 Thuật toán đệ qui

3 Một số ví dụ minh họa

4 Phân tích thuật toán đệ qui

5 Chứng minh tính đúng đắn của thuật toán đệ qui

6 Thuật toán quay lui
- Bài toán xếp hậu
- Bài toán mã tuần

Thuật toán đệ qui

Ví dụ 1 : Tính n!

Hàm f(n) = n! được định nghĩa đệ qui như sau

- Bước cơ sở : f(0) = 0! = 1
- Bước đệ qui : f(n) = n f(n-1), với n>0

Hàm đệ qui viết bằng ngôn ngữ C

```
int Fact(int n){
    if(n==0) return 1;
    else return n*Fact(n-1);
}
```

Thuật toán đệ qui

Ví dụ 2 : Tính số Fibonacci

Dãy số Fibonacci đc định nghĩa như sau :

- Bước cơ sở : $F(0) = 1$, $F(1) = 1$;
- Bước đệ qui : $F(n) = F(n\text{-}1) + F(n\text{-}2)$ với $n \geq 2$

Hàm đệ qui viết bằng ngôn ngữ C

```
int FibRec(int n){
    if(n<=1) return 1;
    else return FibRec(n-1) + FibRec(n-2);
}
```

Thuật toán đệ qui

Ví dụ 3 : Tính hệ số nhị thức C_n^k

Hệ số $C(n,k)$ đc định nghĩa như sau :

- Bước cơ sở : $C(n,0) = 1, C(n,n) = 1 \ \ \forall n \geq 0$;
- Bước đệ qui : $C(n,k) = C(n-1,k-1) + C(n-1,k) \ \ 0 < k < n$

Hàm đệ qui viết bằng ngôn ngữ C

```
int C(int n,int k){
    if((k==0) || (k==n)) return 1;
    else return C(n-1,k-1) + C(n-1,k);
}
```

Thuật toán đệ qui

Ví dụ 4 : Tìm kiếm nhị phân

Cho mảng số $x = [1..n]$ được sắp xếp theo thứ tự không giảm và số y. Cần tìm chỉ số $(1 \leq i \leq n)$ sao cho $x[i] = y$.
Để đơn giản, ta giả thiết chỉ số i tồn tại. Thuật toán để giải bài toán dựa trên lập luận sau : với số y cho trước

- hoặc là bằng phần tử nằm giữa mảng x
- hoặc là nằm nửa bên trái mảng x
- hoặc là nằm nửa bên phải mảng x

Thuật toán đệ qui

Ví dụ 4 : Tìm kiếm nhị phân (tiếp)

```
function Bsearch(x[1..n], start, finish){
middle := (start + finish)/2;
if (y = x[middle]) return middle;
else {
    if (y < x[middle]) return Bsearch(x, start, middle-1);
    else /* y > x[middle] */
        return Bsearch(x, middle+1, finish);
    }
}
```

Thuật toán đệ qui

Ví dụ 4 : Tìm kiếm nhị phân (tiếp)

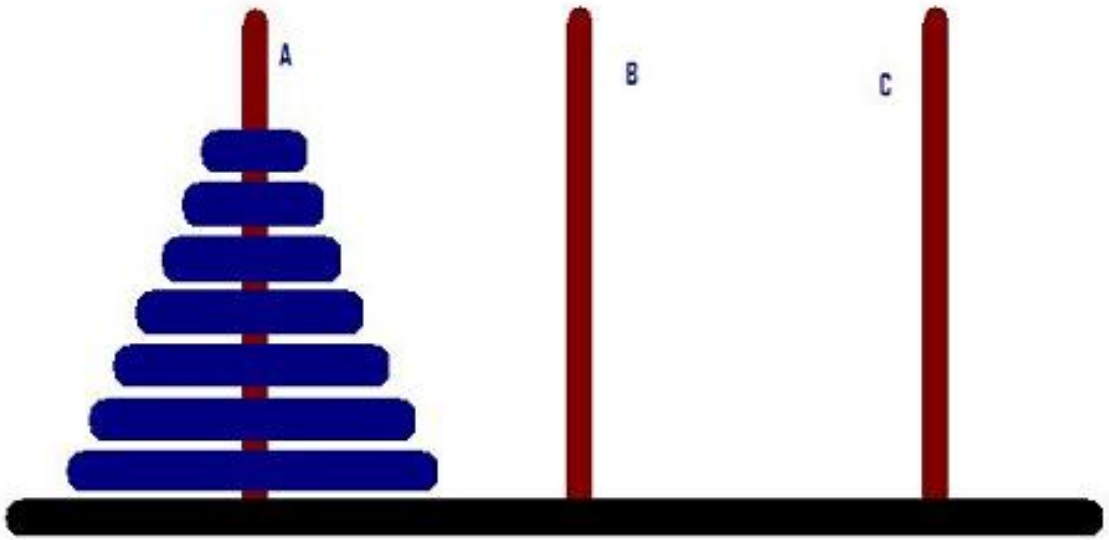
Hàm C trả lại giá trị chỉ số i nếu tìm thấy y, không thì trả lại -1.

```
int Bsearch(int *x, int start, int finish,int y){
    int middle;
    middle = (start+finish)/2;
    if(start==finish){/* Neo de qui */
       if(y==x[middle]) return middle; else return -1; }
    if(y==x[middle]) return middle;/* Tim thay */
    else{
    if(x[middle]<y) /* Tim y nam mang ben phai */
       return Bsearch(x,middle+1,finish,y);
    /* Tim y nam mang ben trai */
    else return Bsearch(x,start,middle-1,y);
    }
}
```

Thuật toán đệ qui

Ví dụ 5 : Bài toán tháp Hà Nội

Trò chơi tháp Hà Nội được trình bày như sau : Có 3 cọc A, B, C. Trên cọc A có một chồng gồm n cái đĩa đường kính giảm dần (xem hình vẽ). Cần phải chuyển chồng đĩa từ cọc A sang cọc C tuân theo qui tắc, mỗi lần chuyển một đĩa và chỉ được xếp đĩa có đường kính nhỏ lên trên đĩa có đường kính lớn hơn đồng thời được dùng cọc B làm cọc trung gian.



Thuật toán đệ qui

Ví dụ 5 : Bài toán tháp Hà Nội (tiếp)

Bài toán đặt ra là tìm cách chơi đòi hỏi số lần di chuyển đĩa ít nhất. Các lập luận sau đây được sử dụng để xây dựng thuật toán giải quyết bài toán đặt ra

- Nếu $n = 1$ thì ta chỉ việc chuyển đĩa cọc A sang cọc C
- Trong trường hợp $n \geq 2$ việc di chuyển đĩa gồm các bước đệ qui như sau
  1. chuyển $n - 1$ đĩa từ cọc A đến cọc B sử dụng cọc C làm trung gian. Bước này cũng phải thực hiện với số lần di chuyển nhỏ nhất, nghĩa là ta phải giải bài toán tháp Hà Nội với $n - 1$ đĩa.
  2. chuyển 1 đĩa đường kính lớn nhất từ cọc A đến cọc C.
  3. chuyển $n - 1$ đĩa từ cọc B đến cọc C - sử dụng cọc A làm trung gian. Bước này cũng phải thực hiện với số lần di chuyển nhỏ nhất, nghĩa là ta lại phải giải bài toán tháp Hà Nội với $n - 1$ đĩa.

Thuật toán đệ qui

Ví dụ 5 : Bài toán tháp Hà Nội (tiếp)

Trong trường hợp $n \geq 2$, hai bước nhỏ 1 và 3 cần số lần di chuyển ít nhất khi thực hiện hai bước này là $2 \times h_{n-1}$, do đó nếu gọi số lần di chuyển đĩa ít nhất là h_n, ta có công thức đệ qui sau

$$h_1 = 1,$$
$$h_2 = 2h_1 + 1, ...$$
$$h_n = 2h_{n-1} + 1, n \geq 2$$

sử dụng phương pháp thế từng bước, ta có

$$h_n = 2^n - 1$$

như vậy độ phức tạp của thuật toán là hàm số mũ

Thuật toán đệ qui

Ví dụ 5 : Bài toán tháp Hà Nội (tiếp)

Mã giả của thuật toán đệ qui giải bài toán tháp Hà Nội như sau

```
Procedure HanoiTower(n,a,b,c)
    if (n=1) then
        <chuyển đĩa từ cọc A sang cọc C>
    else
        HanoiTower(n-1,A,C,B)
        HanoiTower(1,A,B,C)
        HanoiTower(n-1,B,A,C)
    endif
End
```

Thuật toán đệ qui

Ví dụ 5 : Bài toán tháp Hà Nội (tiếp)

Mã nguồn C của thuật toán đệ qui giải bài toán tháp Hà Nội như sau

```
# include <stdio.h>
# include <conio.h>
int i=0;
int main(){
    int disk;
    printf(" Nhập số đĩa : ");
    scanf("%d",&disk);
    move(disk,'A','C','B');
    printf(" Tổng số lần di chuyển %d",i);
    getch();
    return 0;
}
```

Thuật toán đệ qui

Ví dụ 5 : Bài toán tháp Hà Nội (tiếp)

Mã nguồn C của thuật toán đệ qui giải bài toán tháp Hà Nội như sau

```
void move(int n, char start, char finish, char spare){
    if(n==1){
        printf("chuyen dia tu coc %c sang coc %c ",start,finish);
        i++;
        return;
    }else{
        move(n-1,start,spare,finish);
        move(1,start,finish,spare);
        move(n-1,spare,finish,start);
    }
}
```

1. Khái niệm đệ quy
   - Hàm đệ qui
   - Tập hợp được xác định đệ qui
2. Thuật toán đệ qui
3. Một số ví dụ minh họa
4. Phân tích thuật toán đệ qui
5. Chứng minh tính đúng đắn của thuật toán đệ qui
6. Thuật toán quay lui
   - Bài toán xếp hậu
   - Bài toán mã tuần

Phân tích thuật toán đệ qui

Các bước tiến hành phân tích thuật toán đệ qui

- Gọi T(n) là thời gian tính của thuật toán
- Xây dựng công thức đệ qui cho T(n)
- Giải công thức đệ qui thu được để đưa ra đánh giá cho T(n)

Vì ta chỉ cần một đánh giá sát cho tốc độ của T(n) nên việc giải công thức đệ qui đối với T(n) được hiểu là việc đưa ra đánh giá tốc độ tăng của T(n) trong ký hiệu tiệm cận.

Phân tích thuật toán đệ qui (tiếp)

Ví dụ 1 : Thuật toán FibRec

```
int FibRec(int n){
    if(n<=1) return n;
    else return FibRec(n-1) + FibRec(n-2);
}
```

Gọi T(n) là số phép toán cộng phải thực hiện trong lệnh gọi FibRec(n), ta có

- $T(0) = 0$, $T(1) = 0$
- $T(n) = T(n\text{-}1) + T(n\text{-}2) + 1$, $n>1$

Chú ý : Phép toán cộng trong trường hợp (n>1), bằng qui nạp ta có : $T(n) = \Theta(F_n)$, tương đương thời gian tính tăng tốc độ cỡ $(1.6)^n$

Phân tích thuật toán đệ qui (tiếp)

Ví dụ 2 : Thủ tục chia để trị

Procedure D-and-C(n)
begin
    if $n \leq n_0$ **then**
        Giải bài toán một cách trực tiếp
    else
        Chia bài toán thành a bài toán con kích thước n/b;
        for (mỗi bài toán trong a bài toán con) **do** D-and-C(n/b);
        Tổ hợp lời giải của a bài toán con để thu được lời giải của bài toán gốc;
    endif
end

Phân tích thuật toán đệ qui (tiếp)

Ví dụ 2 : Thủ tục chia để trị (tiếp)

Gọi T(n) là thời gian giải bài toán kích thước n. Thời gian của giải thuật chia-để-trị đc đánh giá thời gian thực hiện 3 thao tác của thuật toán

- **Chia** bài toán thành a bài toán con, mỗi bài toán kích thước n/b, đòi hỏi thời gian $D(n)$
- **Trị** các bài toán con : a T(n/b)
- **Tổ hợp** các lời giải : $C(n)$

Vậy ta có công thức đệ qui sau đây để tính T(n) :

$$T(n) = \begin{cases} \Theta(1), & n \leq n_0 \\ aT(n/b) + D(n) + C(n), & n > n_0 \end{cases}$$

Phân tích thuật toán đệ qui (tiếp)

Định lý thợ - Master theorem

Xét T(n) là công thức theo giải thuật Chia-để-trị đệ quy trên

$$T(n) = aT(n/b) + f(n)$$

trong đó

1. n là kích thước bài toán
2. $a \geq 1$ là số bài toán con
3. n/b là kích thước của các bài toán con với $b > 1$ giả thiết chúng đều bằng nhau
4. $f(n)$ là chi phí gồm chia bài toán thành a bài toán con D(n) và tổ hợp a bài toán con thành lời giải C(n)

Phân tích thuật toán đệ qui (tiếp)

Định lý thợ - Master theorem (tiếp)

Xét T(n) là công thức thời gian tính theo giải thuật đệ quy Chia-để-trị

$$T(n) = aT(n/b) + f(n)$$

xảy ra ba tình huống

1. nếu $f(n) = O(n^{log_b a - \epsilon})$ với hằng số $\epsilon > 0$ thì $T(n) = \Theta(n^{\log_b a})$
2. nếu $f(n) = \Theta(n^{\log_b a})$ thì $T(n) = \Theta(n^{\log_b a} \log n)$
3. nếu $f(n) = \Omega(n^{\log_b a + \epsilon})$ với hằng số $\epsilon > 0$ và nếu $af(n/b) \leq cf(n)$ cho các hằng số $c < 1$ và kích thước n đủ lớn thì $T(n) = \Theta(f(n))$

Định lý thợ - Master theorem (tiếp)

Xét T(n) là công thức thời gian tính theo giải thuật đệ quy Chia-để-trị

$$T(n) = aT(n/b) + f(n)$$

xảy ra ba tình huống

1. nếu $f(n) = O(n^{\log_b a - \epsilon})$ với hằng số $\epsilon > 0$ thì $T(n) = \Theta(n^{\log_b a})$
2. nếu $f(n) = \Theta(n^{\log_b a})$ thì $T(n) = \Theta(n^{\log_b a} \log n)$
3. nếu $f(n) = \Omega(n^{\log_b a + \epsilon})$ với hằng số $\epsilon > 0$ và nếu $af(n/b) \leq cf(n)$ cho các hằng số $c < 1$ và kích thước n đủ lớn thì $T(n) = \Theta(f(n))$

Trước khi áp dụng định lý thợ, ta cần hiểu ý nghĩa của định lý. Với mỗi của ba tình huống, ta so sánh hàm $f(n)$ với hàm $n^{\log_b a}$ du di khoảng $\epsilon > 0$ đủ nhỏ. Với việc so sánh độ lớn của hai hàm giúp ta xác định được thời gian tính $T(n)$ của giải thuật.

- Tình huống 1, $f(n) = O(n^{log_b a - \epsilon})$, nghĩa là hàm $n^{\log_b a}$ lớn hơn thì $T(n) = \Theta(n^{\log_b a})$
- Tình huống 2, $f(n) = \Theta(n^{\log_b a})$, nghĩa là hai hàm có cùng kích thước thì t chỉ cần nhân thời gian tính với hệ số $\log n$ để trở thành $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(f(n) \log n)$
- Tình huống 3, $f(n) = \Omega(n^{\log_b a + \epsilon})$, nghĩa là hàm $f(n)$ lớn hơn thì $T(n) = \Theta(f(n))$

Như vậy hàm nào lớn hơn sẽ đại diện trong ký hiệu tiệm cận $\Theta(g(n))$. Để cho đỡ phức tạp ta dùng định lý thợ rút gọn có công thức $f(n)$ tường minh theo slide tiếp theo

Phân tích thuật toán đệ qui (tiếp)

Định lý thợ phát biểu tương đương (tiếp)

Giả sử $a \geq 1$ và $b > 1, c \geq 0$ là các hằng số. Xét T(n) là công thức đệ qui

$$T(n) = aT(n/b) + cn^k$$

xác định với $n \geq 0$ có ba tình huống

1. nếu $a > b^k$ thì $T(n) = \Theta(n^{\log_b a})$
2. nếu $a = b^k$ thì $T(n) = \Theta(n^k \log n)$
3. nếu $a < b^k$ thì $T(n) = \Theta(n^k)$

Phân tích thuật toán đệ qui (tiếp)

- **VD1** : $T(n) = 3T(n/4) + cn^2$ trong ví dụ này : a=3, b=4, k=2 do $3 < 4^2$ ta áp dụng tình huống 3 nên $T(n) = \Theta(n^2)$
- **VD2** : $T(n) = 2T(n/2) + \sqrt{n}$ trong ví dụ này : a=2, b=2, k=1/2 do $2 > \sqrt{2}$ ta áp dụng tình huống 1 nên $T(n) = \Theta(n^{\log_b a}) = \Theta(n)$.
- **VD3** : $T(n) = 16T(n/4) + n$ trong ví dụ này : a=16, b=4, k=1 do $16 > 4$ ta áp dụng tình huống 1 nên $T(n) = \Theta(n^2)$
- **VD4** : $T(n) = T(3n/7) + 1$ trong ví dụ này : a=1, b=7/3, k=0 do $a = b^k$ ta áp dụng tình huống 2 nên $T(n) = \Theta(n^k \log n) = \Theta(\log n)$
- **VD5** : Phân tích Bsearch

$$T(1) = c$$
$$T(n) = T(n/2) + d$$

Từ đó theo định lý thợ $T(n) = \Theta(\log n)$.

Thuật toán đệ qui

Định nghĩa đệ qui và qui nap toán học

Định nghĩa đệ qui và qui nạp toán học có những nét tương đồng và bổ sung cho nhau. Chứng minh bằng qui nạp toán học thường dùng làm cơ sở để xây dựng giải thuật đệ qui để giải quyết bài toán. Chứng minh bằng qui nạp thường gồm hai phần

- **Bước cơ sở qui nạp** : tương đương *bước cơ sở* trong định nghĩa đệ qui
- **Bước chuyển qui nạp** : tương đương *bước đệ qui* trong định nghĩa đệ qui

Chứng minh tính đúng đắn của thuật toán đệ qui

Vậy để chừng minh tính đúng đắn của thuật toán đệ qui thông thường ta sử dụng qui nạp toán học. Ngược lại, cách chứng minh bằng đệ qui cũng thường là cơ sở để xây dựng nhiều thuật toán đệ qui.

- **VD1** Chứng minh Fact(n) = n! vậy ta sẽ chứng minh bằng qui nạp toán học
  - **Bước cơ sở qui nạp** : Ta có Fact(0) = 1 = 0!
  - **Bước chuyển qui nạp** : Giả sử Fact(n-1) cho giá trị của (n-1)! ta phải chứng minh Fact(n) cho giá trị n!. Thật vậy, do giá trị trả lại của Fact(n)

$$n * Fact(n-1) \underbrace{\Rightarrow}_{\text{theo giả thiết qui nạp}} n * (n-1)! = n!$$

Chứng minh tính đúng đắn của thuật toán đệ qui (tiếp)

- **VD2** : Cho mặt phẳng trên đó vẽ *n* đường thẳng. Chứng minh mệnh đề sau bằng qui nạp : P(n) luôn có thể tô các phần được chia bởi n đường thẳng bởi chỉ hai mầu : xanh và đỏ (Xem chứng minh trong sách).

  Mã giả giải thuật tô hai mầu mặt phẳng như sau

  Procedure PaintColor(n,A,B)
  <xét đường thẳng n>
  <phân chia các phần mặt phẳng thành hai tập A,B>
  if (n=1) **then**
  A $\leftarrow$ Xanh; B $\leftarrow$ Đỏ;
  else
  PaintColor(n-1,A,B)
  <đảo mầu các phần mặt phẳng thuộc A>
  endif
  End

1 Khái niệm đệ quy
- Hàm đệ qui
- Tập hợp được xác định đệ qui

2 Thuật toán đệ qui

3 Một số ví dụ minh họa

4 Phân tích thuật toán đệ qui

5 Chứng minh tính đúng đắn của thuật toán đệ qui

6 Thuật toán quay lui
- Bài toán xếp hậu
- Bài toán mã tuần

Thuật toán quay lui

Định nghĩa về bài toán liệt kê

Thuật toán quay lui (backtracking algorithm) là thuật toán đệ qui cơ bản dùng để giải quyết nhiều bài toán, đặc biệt là bài toán dạng liệt kê được phát biểu như sau :
Cho $A_1, A_2, \cdots A_n$ là các tập hữu hạn. Ký hiệu

$$X = A_1 \times A_2 \times \cdots \times A_n = \{(x_1, x_2, \cdots, x_n) : x_i \in A_i, i = 1, 2, \cdots, n\}$$

Giả sử P là tính chất cho trên tập X, vấn đề đặt ra là liệt kê tất cả các phần tử của X thỏa mãn P

$$D = \{(x_1, x_2, \cdots, x_n) \in X \text{ thỏa mãn tính chất } P\}$$

tập D được gọi tập các lời giải chấp nhận được.

Thuật toán quay lui (tiếp)

Các ví dụ về bài toán liệt kê

- **VD1** : Liệt kê xâu nhị phân độ dài n dẫn về liệt kê các các phần tử của tập

$$B^n = \{(x_1, x_2, \cdots, x_n) : x_i \in \{0, 1\}, i = 1, 2, \cdots, n\}$$

- **VD2** : Liệt kê các tập con m phần tử của tập $N = \{1, 2, \cdots, n\}$ dẫn về liệt kê tập con có thứ tự

$$S(m, n) = \{(x_1, x_2, \cdots, x_m) \in N^m : 1 \leq x_1 < x_2 < \cdots < x_m \leq n\}$$

- **VD3** : Tập hoán vị các số tự nhiên $N = \{1, 2, \cdots, n\}$ là tập

$$\Pi_n = \{(x_1, x_2, \cdots, x_n) \in N^n : x_i \neq x_j, i \neq j\}$$

Thuật toán quay lui (tiếp)

Lời giải bộ phân

Ta gọi lời giải cấp bộ phận cấp k với $0 \leq k \leq n$ là bộ có thứ tự gồm k thành phần $(a_1, a_2, \cdots, a_k)$ trong đó $a_i \in A_i$ với $i = 1, 2, \cdots, k$.

- Với k=0, ta có lời giải bộ phận cấp 0 hay lời giải rỗng ()
- Với k=n, ta có một lời giải chấp nhận được của bài toán

Thuật toán quay lui (tiếp)

Các bước chung của thuật toán quay lui

1. thuật toán bắt đầu với lời giải rỗng ()
2. dựa trên tính chất P, ta xác định được phần tử $a_1 \in A_1$ vào vị trí thứ nhất của lời giải bộ phận cấp 1 (a_1), gọi là Ứng Cử Viên (viết tắt UCV)
3. tại bước tổng quát : giả sử ta đang có lời giải bộ phận cấp k-1 là $(a_1, a_2, \cdots, a_{k-1})$, ta sẽ gọi những UCV vào vị trí k vào vị trí thứ k thuộc tập S_k. Có hai tình huống xảy ra ...

Thuật toán quay lui (tiếp)

Các bước chung của thuật toán quay lui (tiếp)

- tại bước tổng quát : ... Có hai tình huống xảy ra
  - **tình huống 1** : $S_k \neq \emptyset$ khi đó lấy $a_k \in S_k$, bổ sung vào lời giải bộ phận cấp $k-1$ đang có thu được lời giải bộ phận cấp k là $(a_1, a_2, \cdots, a_k)$
    - nếu $k = n$, ta thu được một lời giải chấp nhận được
    - nếu $k < n$, ta tiếp tục xây dựng lời giải bộ phận cấp $k+1$
  - **tình huống 2** : $S_k = \emptyset$ là tình huống ngõ cụt. Do không thể tìm phát triển được thành lời giải đầy đủ, ta sẽ phải *quay lui* để tìm UCV mới vào vị trí $k-1$ của lời giải.
    - Nếu tìm thấy UCV thì bổ sung vào vị trí $k-1$ rồi tiếp tục xây dựng thành phần k
    - Nếu không tìm thấy ta sẽ phải *quay lui* để tìm UCV mới vào vị trí $k-2, \cdots$ Nếu quay lại tận lời giải rỗng mà vẫn không tìm đc UCV vào vị trí 1 thì thuật toán kết thúc (bài toán vô nghiệm).

Thuật toán quay lui (tiếp)

Thủ tục đệ qui của thuật toán quay lui

Procedure Backtrack(k)

1. Xây dựng S_k
2. **for** $y \in S_k$ **do** /* với mỗi UCV y từ S_k*/
3. $a_k \leftarrow y$
4. **if** (k=n) **then** <Ghi nhận lời giải $(a_1, a_2, \cdots, a_k)$>
5. **else** Backtrack(k+1)
6. **endif**
7. **endfor**

End

Lệnh gọi ban đầu của giải thuật Backtrack(1)

Thuật toán quay lui (tiếp)

Hai vấn đề mấu chốt của thuật toán quay lui

Để cài đặt thuật toán quay lui giải các bài toán cụ thể, ta cần giải quyết hai vấn đề cơ bản sau

- Tìm thuật toán xây dựng tập UCV tại mỗi bước k là S_k
- Tìm cách mô tả các tập này để có thể cài đặt thao tác liệt kê các phần tử của vòng lặp **for** ở bước 2

hiệu quả của thuật toán liệt kê phụ thuộc vào việc ta có xác định được chính xác các tập UCV hay không

Thuật toán quay lui (tiếp)

Các lưu ý

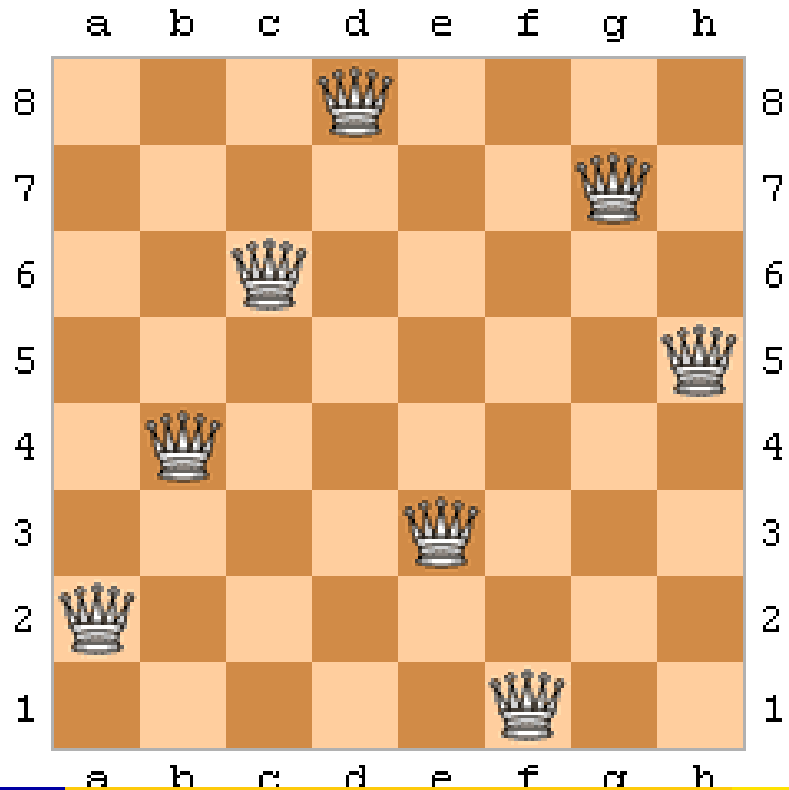
- Nếu chỉ cần tìm một lời giải (chấp nhận được) thì cần tìm cách chấm dứt các thủ tục gọi đệ qui lồng nhau sinh ra bởi lệnh gọi Backtrack(1) sau khi ghi nhận lời giải đầu tiên.
- Nếu kết thúc thuật toán mà không thu được lời giải nào thì có nghĩa bài toán không có lời giải.
- Thuật toán dễ dàng mở rộng cho bài toán liệt kê với chiều dài hữu hạn không nhất thiết cùng độ dài n. Lúc đó câu lệnh ở bước 4 đc sửa thành
  if <$(a_1, a_2, \cdots, a_k)$ là lời giải> **then** <Ghi nhận lời giải $(a_1, a_2, \cdots, a_k)$>

1. Khái niệm đệ quy
   - Hàm đệ qui
   - Tập hợp được xác định đệ qui
2. Thuật toán đệ qui
3. Một số ví dụ minh họa
4. Phân tích thuật toán đệ qui
5. Chứng minh tính đúng đắn của thuật toán đệ qui
6. Thuật toán quay lui
   - Bài toán xếp hậu
   - Bài toán mã tuần

Thuật toán quay lui (tiếp)

Phát biểu bài toán xếp hậu

Liệt kê tất cả các cách sắp xếp n quân hậu trên bàn cờ $n \times n$ sao cho chúng không ăn lẫn nhau - không có hai con nằm trên cùng dòng, cột hay đường chéo



Thuật toán quay lui (tiếp)

Biểu diễn bài toán xếp hậu

- Đánh số các cột và dòng của bàn cờ từ 1 đến n. Một cách xếp hậu có thể biểu diễn bởi bộ $(a_1, a_2, \cdots, a_n)$ trong đó a_i là tọa độ cột của con hậu ở dòng i
- Các điều kiện đặt ra với bộ $(a_1, a_2, \cdots, a_n)$
  - $a_i \neq a_j$ với mọi $i \neq j$ (hai hậu nằm trên hai dòng i và j không cùng một cột)
  - $|a_i - a_j| \neq |i - j|$ (không cùng nằm trên đường chéo)

Thuật toán quay lui (tiếp)

Biểu diễn bài toán xếp hậu (tiếp)

Như vậy bài toán được dẫn về bài toán liệt kê

$$D = \{(x_1, x_2, \cdots, x_n) \in N^n : a_i \neq a_j \text{ và } |a_i - a_j| \neq |i - j|, i \neq j\}$$

Thuật toán quay lui (tiếp)

Hàm nhận biết UCV

Mã nguồn ngôn ngữ C

```
int UCVh(int j, int k){
// UCVh nhận giá trị 1
// khi và chỉ khi j ∈ S_k
int i;
for(i=1;i<k;i++)
    if((j==a[i])||(fabs(j-a[i])==k-i)) // Vi phạm tính chất
        return 0;
return 1;
}
```

Thuật toán quay lui (tiếp)

Hàm xếp hậu sử dụng thuật toán quay lui

Mã nguồn ngôn ngữ C

```
int Hau(int i){
int j;
for(j=1;j<=n;j++)
    if(UCVh(j,i)){
        a[i] = j;
        if(i==n) Ghinhan();
        else Hau(i+1);
    }
}
```

Gọi từ thân chương trình chính **Hau(1)**

1. Khái niệm đệ quy
   - Hàm đệ qui
   - Tập hợp được xác định đệ qui
2. Thuật toán đệ qui
3. Một số ví dụ minh họa
4. Phân tích thuật toán đệ qui
5. Chứng minh tính đúng đắn của thuật toán đệ qui
6. Thuật toán quay lui
   - Bài toán xếp hậu
   - Bài toán mã tuần

Thuật toán quay lui (tiếp)

Phát biểu bài toán mã tuần

Liệt kê đường đi của một con mã từ vị trí xuất phát sao cho nó tuần qua mỗi ô của bàn cờ đúng một lần

35	40	47	44	61	08	15	12
46	43	36	41	14	11	62	09
39	34	45	48	07	60	13	16
50	55	42	37	22	17	10	63
33	38	49	54	59	06	23	18
56	51	28	31	26	21		03
29	32	53	58	05	02	19	24
52	57	30	27	20	25	04	01

Thuật toán quay lui (tiếp)

Biểu diễn bài toán mã tuần

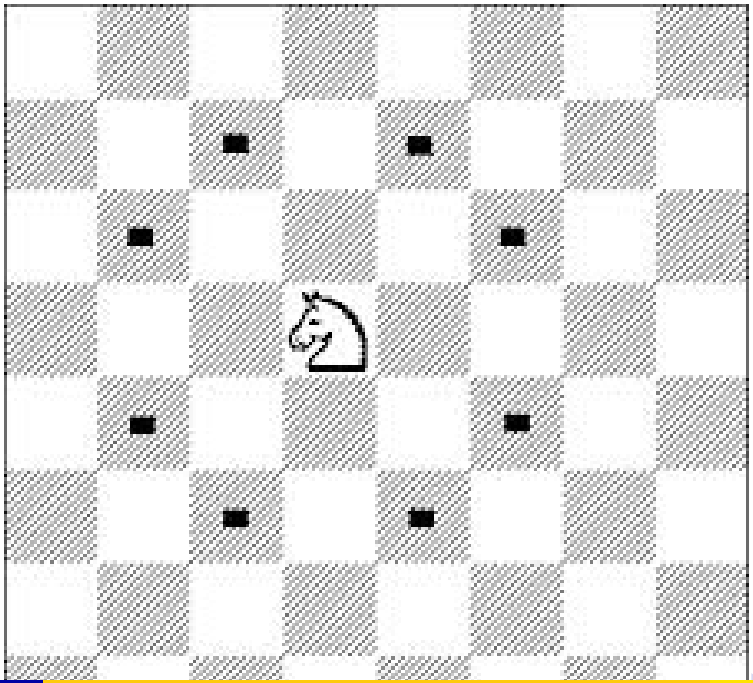
- Sử dụng một mảng số nguyên hai chiều bằng kích thước bàn cờ $sol[n, n]$ để chứa thứ tự con mã khi di chuyển trên bàn cờ
- Khởi tạo mảng sol với giá là -1 cho mọi vị trí
- Có tất cả $n^2 - 1$ phép dịch chuyển hợp lệ nếu quan mã đi qua các ô đúng một lần
- Khi ô ở vị trí x,y được xác định hợp lệ, ta gán $sol[x, y]$ thứ tự bước di chuyển,

Thuật toán quay lui (tiếp)

8 khả năng di chuyển của quân mã từ vị trí (x,y)

Ngoài việc xem xét khả năng di chuyển của quân mã, để di chuyển hợp lệ ta cần kiểm tra thêm

- Vị trí hàng, cột có vượt ra ngoài bàn cờ không
- Vị trí đã đi qua hay chưa $sol[x_{next}, y_{next}] \neq -1$



Thuật toán quay lui (tiếp)

Function KnightTour(x,y,k,sol)

1. **if** (k=n^2-1) **then** In ra lời giải sol; **return** true **endif**
2. **for** $(x_{next}, y_{next}) \in$ 8 khả năng dịch chuyển của quân mã **do**
3. **if** (x_{next}, y_{next}) hợp lệ **then**
4. sol[x,y] $\leftarrow$ k
5. **if** (KnightTour(x_{next}, y_{next},k+1,sol)=true) **then**
6. **return** true
7. **else**
8. sol[x,y] $\leftarrow$ -1 // Quay lui
9. **endif**
10. **endif**
11. **endfor**
12. **return** false

End

Gọi từ thân chương trình chính KnightTour(x_{start}, y_{start},0,sol)