

# Kỹ Thuật lập trình

## Programming Technique

*Ts. Nguyễn Đức Thuận*  
*BM Hệ thống Thông Tin*



# Giới thiệu môn học

## Nội dung môn học

- **Chương 1:** Kỹ thuật tổ chức chương trình
- **Chương 2:** Lập trình có cấu trúc – Hàm nâng cao
- **Chương 3:** Đệ qui
- **Chương 4:** Thử sai-quaylui, Nhánh cận
- **Chương 5:**

# ĐỆ QUI

(Tham khảo bài giảng Th. Đặng Bình Phương)

# 1. DẪN NHẬP

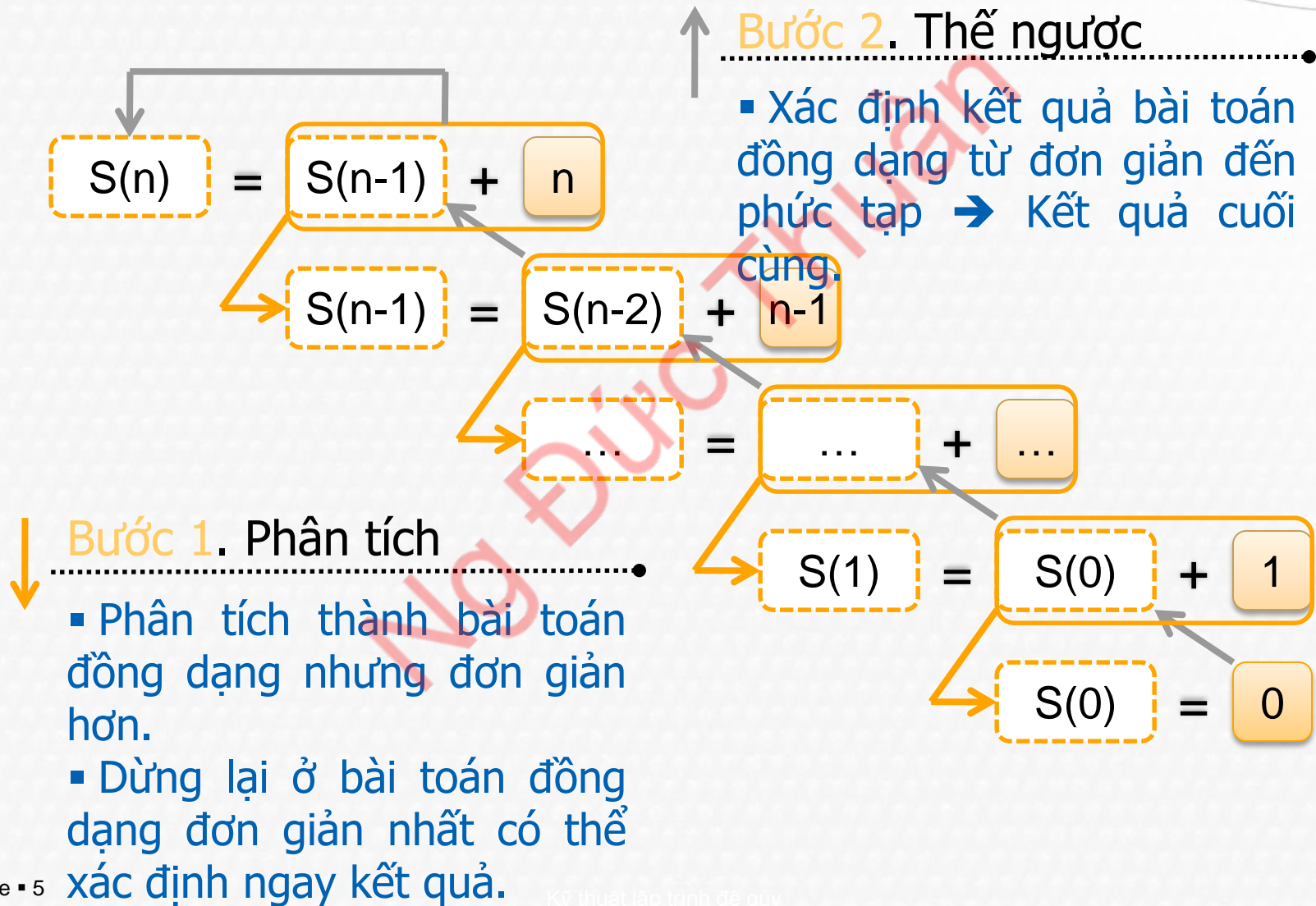
■ Cho  $S(n) = 1 + 2 + 3 + \dots + n$

$\Rightarrow S(10)? S(11)?$

$$S(10) = 1 + 2 + \dots + 10 = 55$$

$$\begin{aligned} S(11) &= 1 + 2 + \dots + 10 + 11 = 66 \\ &= S(10) + 11 \\ &= 55 + 11 = 66 \end{aligned}$$

# 1. DẪN NHẬP





## 2. ĐỆ QUI

### Khái niệm

*Một khái niệm, định nghĩa được gọi là đệ quy nếu trong khái niệm/định nghĩa có chứa lại chính nó.*

### Ví dụ:

**Người giàu:**

*Người có nhiều tiền*

*Người có cha mẹ là **người giàu***



## 2. ĐỆ QUI

- Ví dụ

$$n! = \begin{cases} 1, & n = 0 \\ (n-1)!n, & n > 0 \end{cases}$$

$$a^n = \begin{cases} 1, & n = 0 \\ a^{n-1} \cdot a, & n > 0 \end{cases}$$

- Cần phân biệt đệ qui và các khái niệm lặp, đệ từ ...

## 2. ĐỀ QUI

- Một khái niệm đề qui gồm 2 thành phần:
  - Thành phần neo (dừng) (anchor)
  - Thành phần đề qui

***Phần đề qui thể hiện tính “quy nạp”***

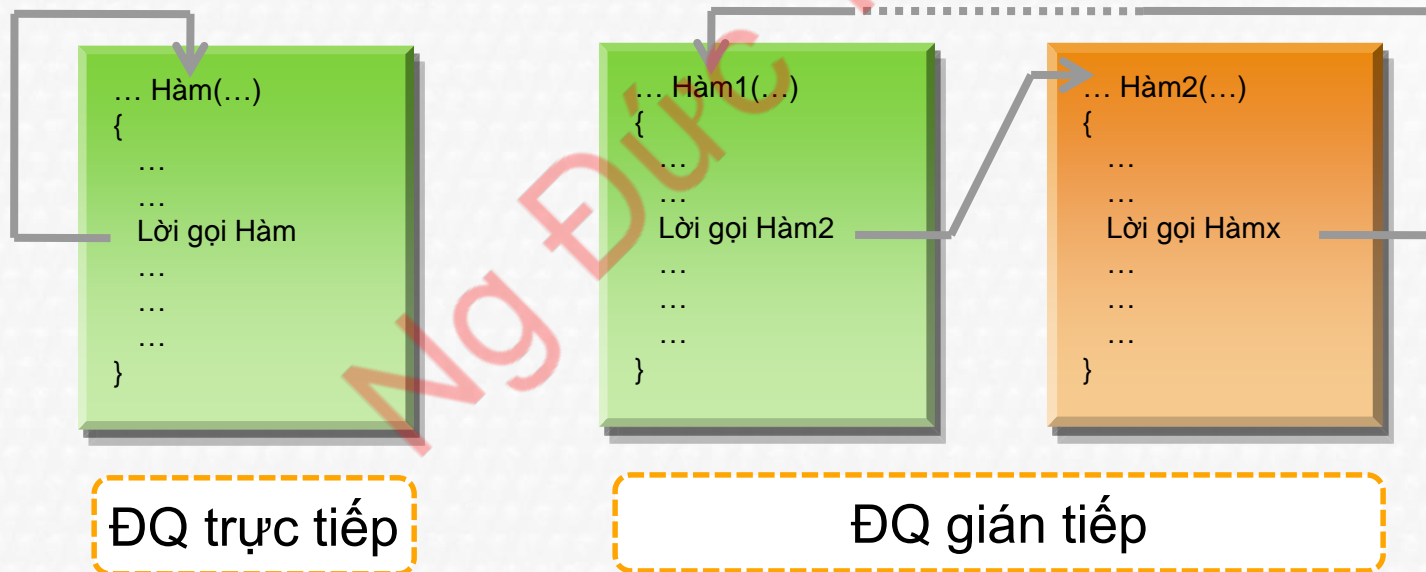
***Phần neo đảm bảo cho tính dừng.***



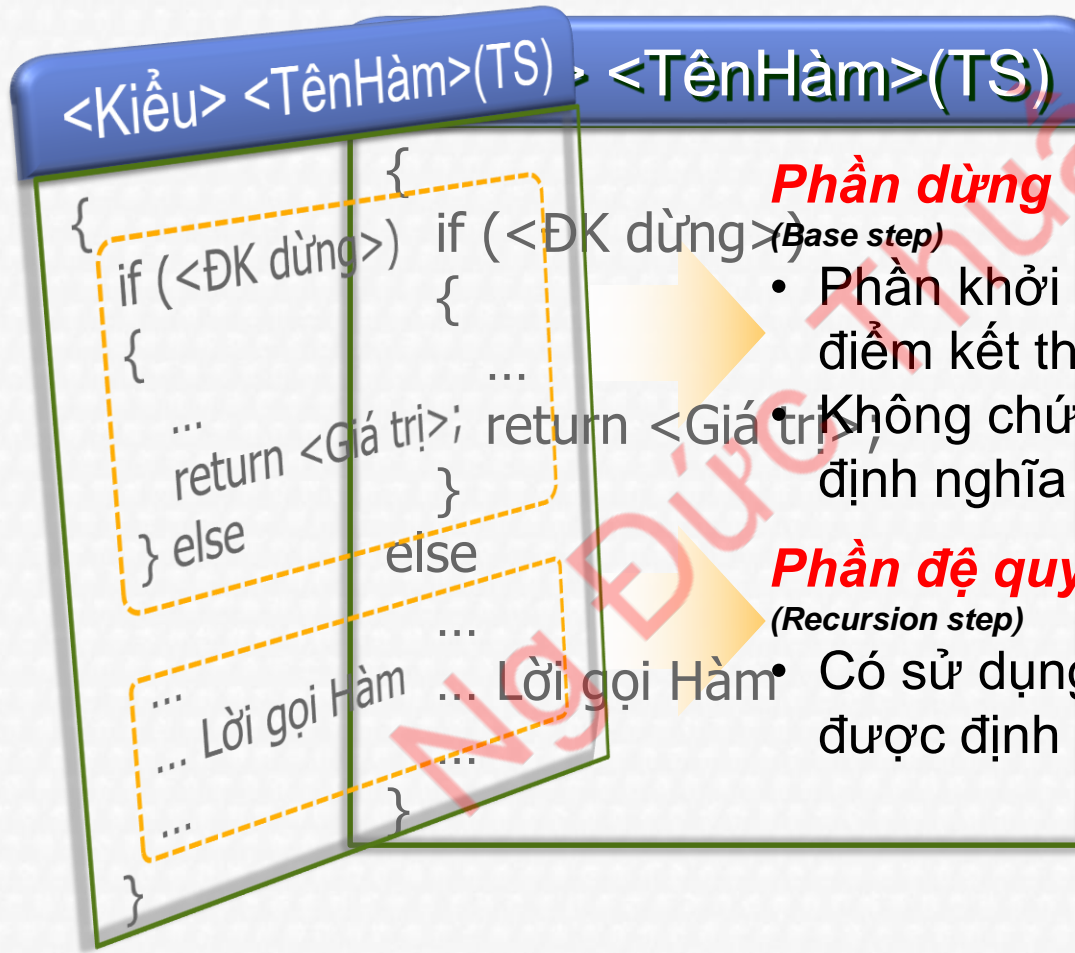
### 3. LẬP TRÌNH ĐỆ QUY

#### ▪ Hàm đệ quy

- Một hàm được gọi là đệ quy nếu bên trong thân của hàm đó có lời gọi hàm lại chính nó một cách trực tiếp hay gián tiếp.



# Cấu trúc hàm đệ quy



## **Phần dừng**

(Base step)

- Phần khởi tính toán hoặc điểm kết thúc của thuật toán
- Không chứa phần đang được định nghĩa

## **Phần đệ quy**

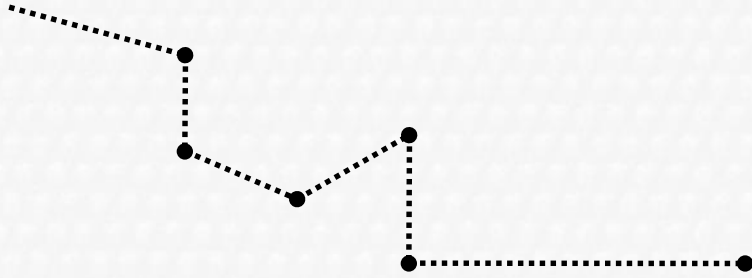
(Recursion step)

- Có sử dụng thuật toán đang được định nghĩa.

# Phân loại



# Đệ quy tuyến tính



## Cấu trúc chương trình

```
<Kiểu> TênHàm (<TS>) {  
    if (<ĐK dừng>) {  
        ...  
        return <Giá Trị>;  
    } else  
        ... TênHàm(<TS>); ...  
}
```

## Ví dụ

Tính  $S(n) = 1 + 2 + \dots + n$

→  $S(n) = S(n - 1) + n$

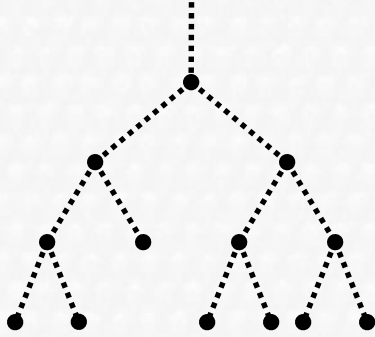
ĐK dừng:  $S(0) = 0$

∴ Chương trình ∴

```
long Tong (int n)  
{  
    if (n == 0)  
        return 0; else  
        return Tong(n-1) + n;  
}
```



# Đệ quy nhị phân



## Cấu trúc chương trình

```
<Kiểu> TênHàm (<TS>) {  
    if (<ĐK dừng>) {  
        ...  
        return <Giá Trị>;  
    } else  
        ... TênHàm(<TS>);  
    ...  
    TênHàm(<TS>);  
    ...  
}
```

## Ví dụ

Tính số hạng thứ  $n$  của dãy Fibonacci:

$f(0) = f(1) = 1$

$f(n) = f(n-1) + f(n-2), n > 1$

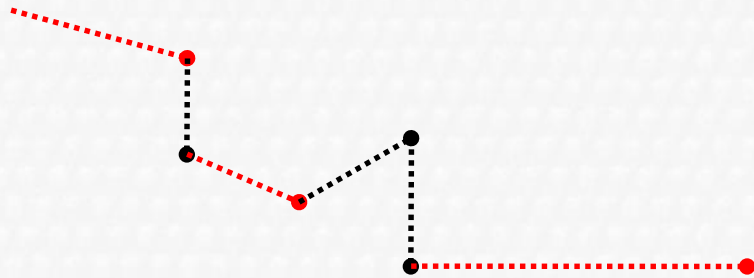
ĐK dừng:  $f(0)=1$  và  $f(1)=1$

∴ Chương trình ∴

```
long Fibo (int n)  
{  
    if (n == 0 || n == 1)  
        return 1; else  
        return Fibo(n-1)+Fibo(n-1);  
}
```



# Đệ quy hỗ tương



## Cấu trúc chương trình

```
<Kiểu> TênHàm1(<TS>) {  
    if (<ĐK dừng>)  
        return <Giá trị>;  
    ... TênHàm2(<TS>); ...  
}  
  
<Kiểu> TênHàm2(<TS>) {  
    if (<ĐK dừng>)  
        return <Giá trị>;  
    ... TênHàm1(<TS>); ...  
}
```

## Ví dụ

Tính số hạng thứ  $n$  của dãy:

$$x(0) = 1, y(0) = 0$$

$$x(n) = x(n-1) + y(n-1)$$

$$y(n) = 3*x(n-1) + 2*y(n-1)$$

ĐK dừng  $x(0) = 1, y(0) = 0$

∴ Chương trình ∴

```
long yn (int n);
```

```
long xn (int n) {
```

```
    if (n == 0) return 1; else
```

```
    return xn (n-1)+yn (n-1);
```

```
}
```

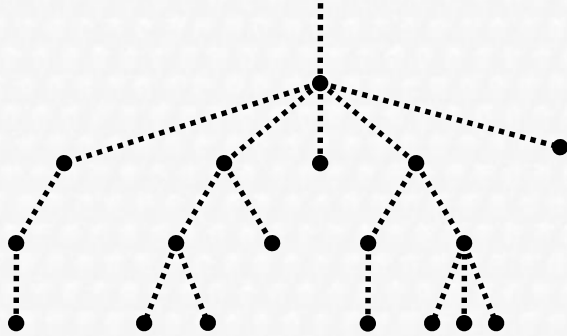
```
long yn (int n) {
```

```
    if (n == 0) return 0; else
```

```
    return 3*xn (n-1)+2*yn (n-1);
```

```
}
```

# Đệ quy phi tuyến



## Cấu trúc chương trình

```
<Kiểu> TênHàm(<TS>) {  
    if (<ĐK dừng>) {  
        ...  
        return <Giá Trị>;  
    }  
    ... Vòng lặp {  
        ... TênHàm(<TS>); ...  
    }  
    ...  
}
```

## Ví dụ

Tính số hạng thứ  $n$  của dãy:

$$x(0) = 1$$

$$x(n) = n^2x(0) + (n-1)^2x(1) + \dots + 2^2x(n-2) + 1^2x(n-1)$$

ĐK dừng:  $x(0) = 1$

∴ Chương trình ∴

```
long xn(int n)  
{  
    if (n == 0) return 1; else {  
        long s = 0;  
        for (int i=1; i<=n; i++)  
            s = s + i*i*xn(n-i);  
        return s;}  
}
```

# Các bước xây dựng hàm đệ quy

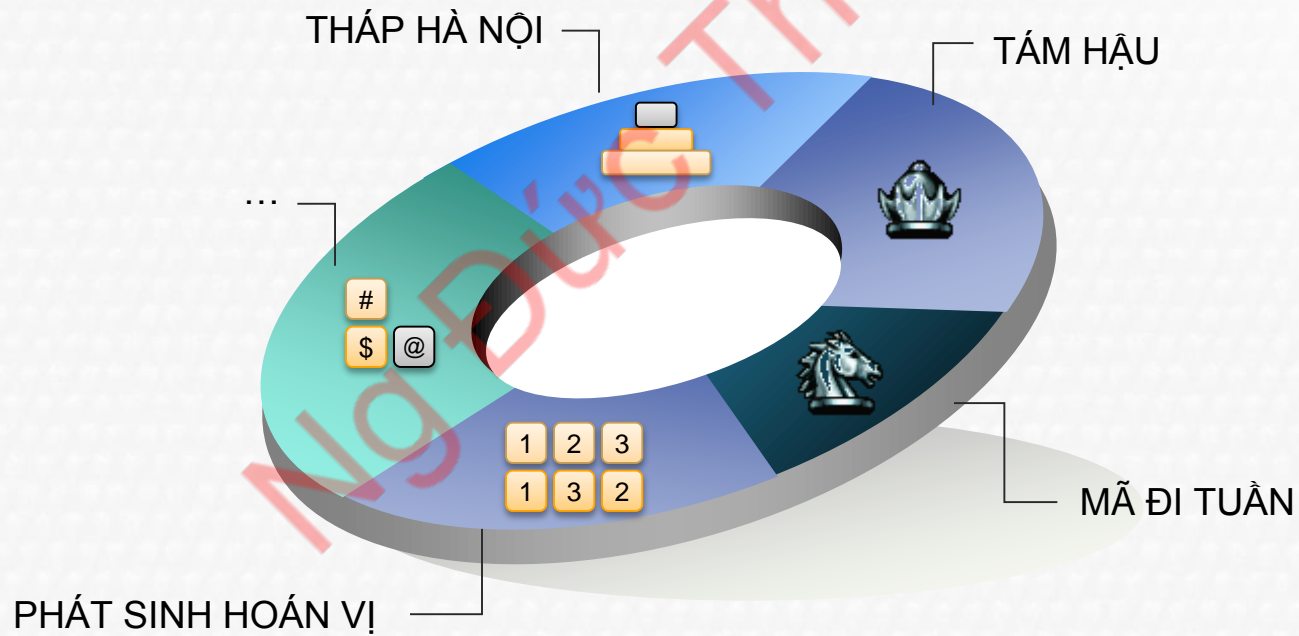


- Tổng quát hóa bài toán cụ thể thành bài toán tổng quát.
- Thông số hóa cho bài toán tổng quát
- VD:  $n$  trong hàm tính tổng  $S(n)$ , ...

- Chia bài toán tổng quát ra thành:
  - Phần không đệ quy.
  - Phần như bài toán trên nhưng kích thước nhỏ hơn.
- VD:  $S(n) = S(n - 1) + n$ , ...

- Các trường hợp suy biến của bài toán.
- Kích thước bài toán trong trường hợp này là nhỏ nhất.
- VD:  $S(0) = 0$

# Một số bài toán kinh điển





# Tháp Hà Nội

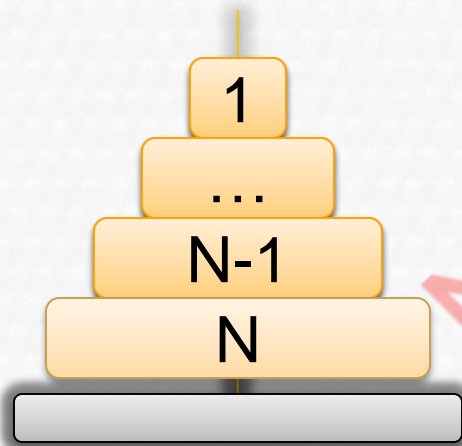
## ■ Mô tả bài toán

- Có 3 cột A, B và C và cột A hiện có N đĩa.
- Tìm cách chuyển N đĩa từ cột A sang cột C sao cho:
  - Một lần chuyển 1 đĩa
  - Đĩa lớn hơn phải nằm dưới.
  - Có thể sử dụng các cột B làm cột trung gian.

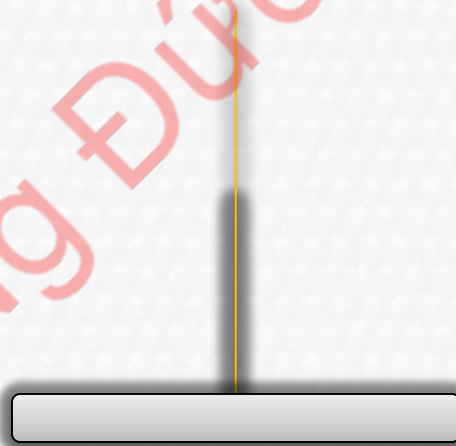


# Tháp Hà Nội

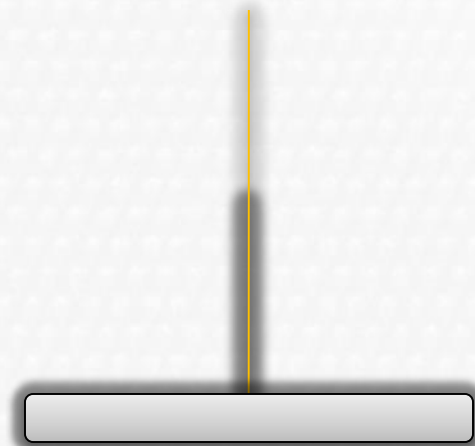
$$N \text{ đĩa } A \rightarrow C = ? \text{ đĩa } A \rightarrow B + \text{Đĩa } N A \rightarrow C + N-1 \text{ đĩa } B \rightarrow C$$



Cột nguồn A



Cột trung gian B



Cột đích C

*Chuyển n đĩa từ a → b, c là cọc trung gian*

```
void HaNoi(int n, char a, char c, char b)
```

```
{
```

```
    if (n == 1)
```

```
        printf("\n %c -> %c", a, c);
```

```
    else
```

```
    {
```

```
        HaNoi(n-1, a, b, c);
```

```
        printf("\n %c -> %c", a, c);
```

```
        HaNoi(n-1, b, c, a);
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    HaNoi(5, 'A', 'C', 'B');
```

```
}
```

*Điều kiện dừng*

*Chuyển n-1 đĩa từ a → b*

*Chuyển đĩa lớn từ a → c*

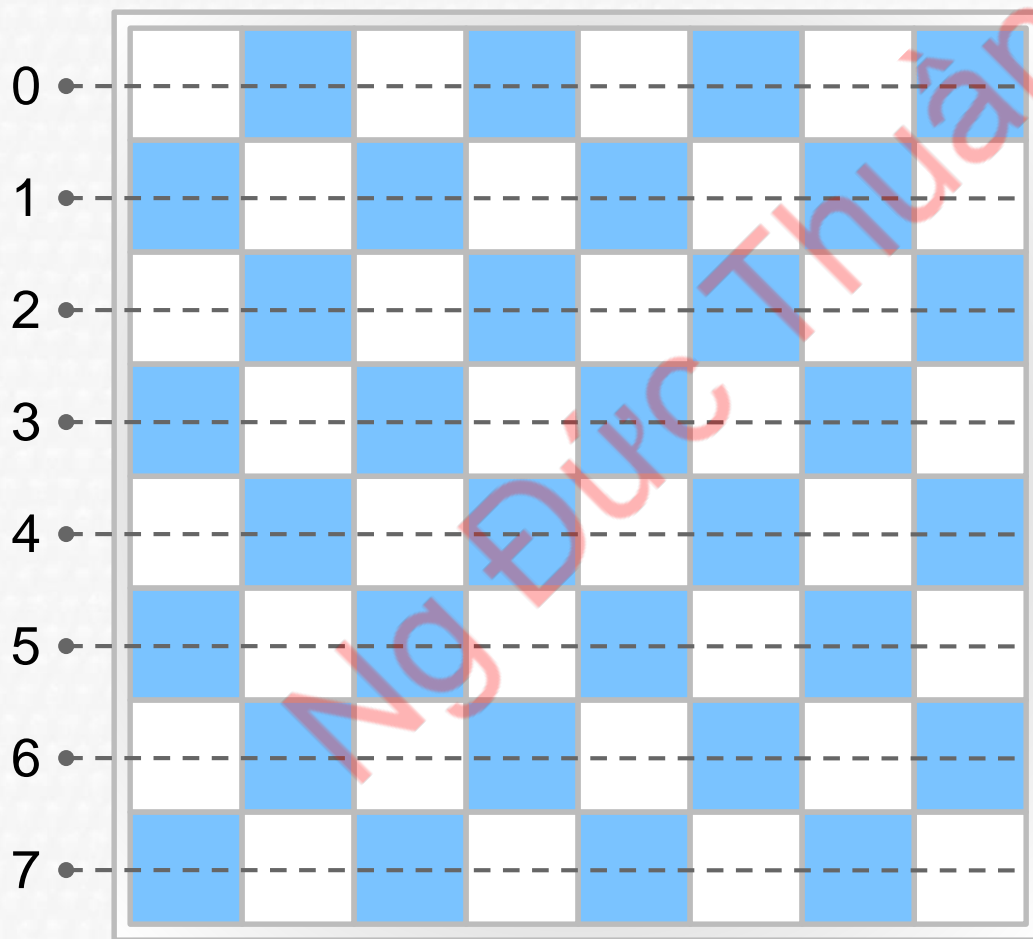
*Chuyển n-1 đĩa từ b → c*

# Tám hậu

## ■ Mô tả bài toán

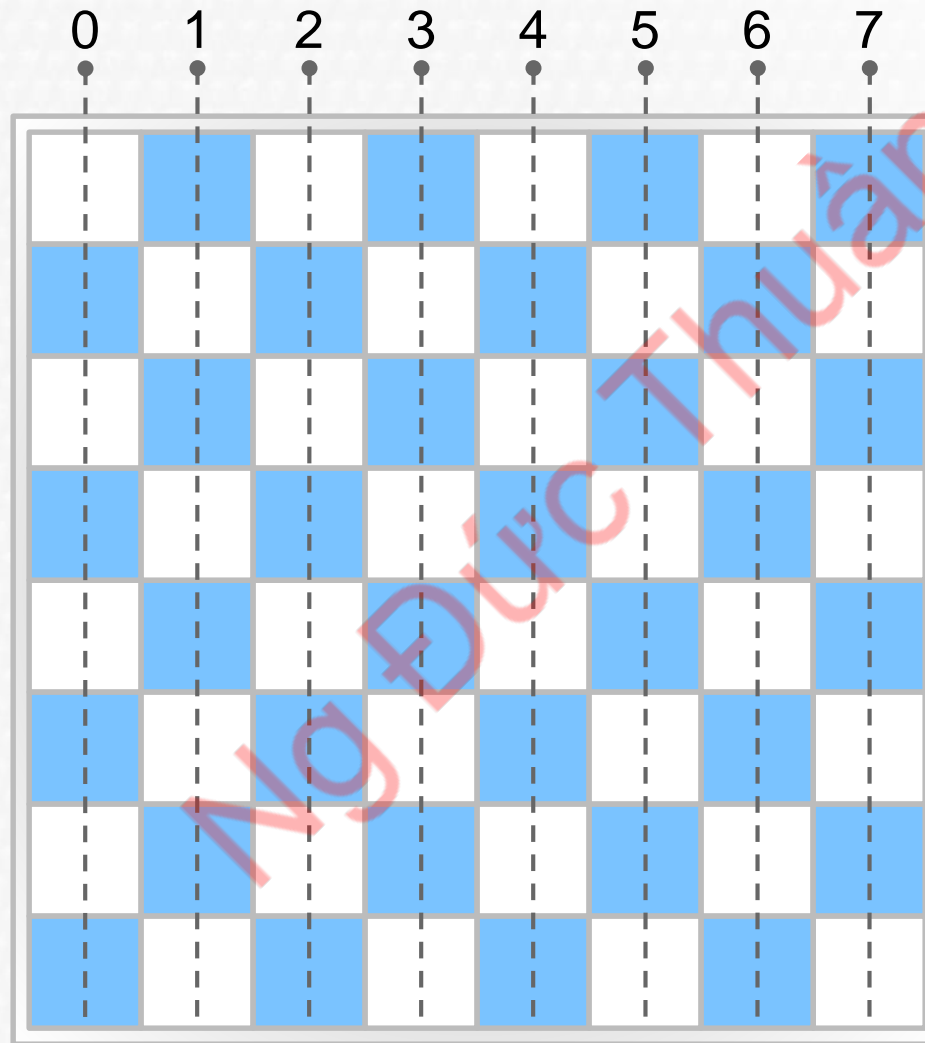
- Cho bàn cờ vua kích thước  $8 \times 8$
- Hãy đặt 8 hoàng hậu lên bàn cờ này sao cho không có hoàng hậu nào “ăn” nhau:
  - Không nằm trên cùng dòng, cùng cột
  - Không nằm trên cùng đường chéo xuôi, ngược.

# Tám hậu – Các dòng



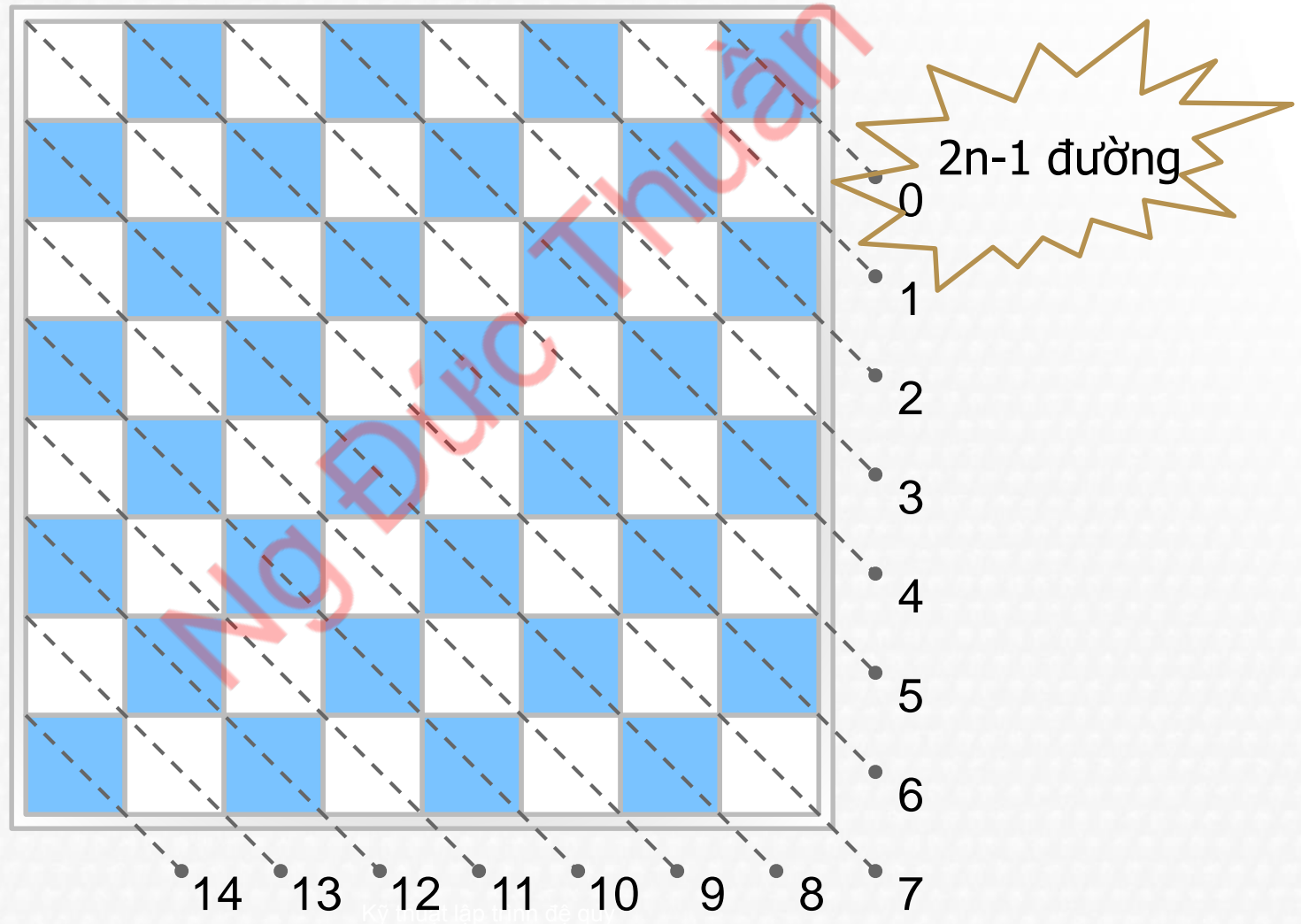
n đường

# Tám hậu – Các cột

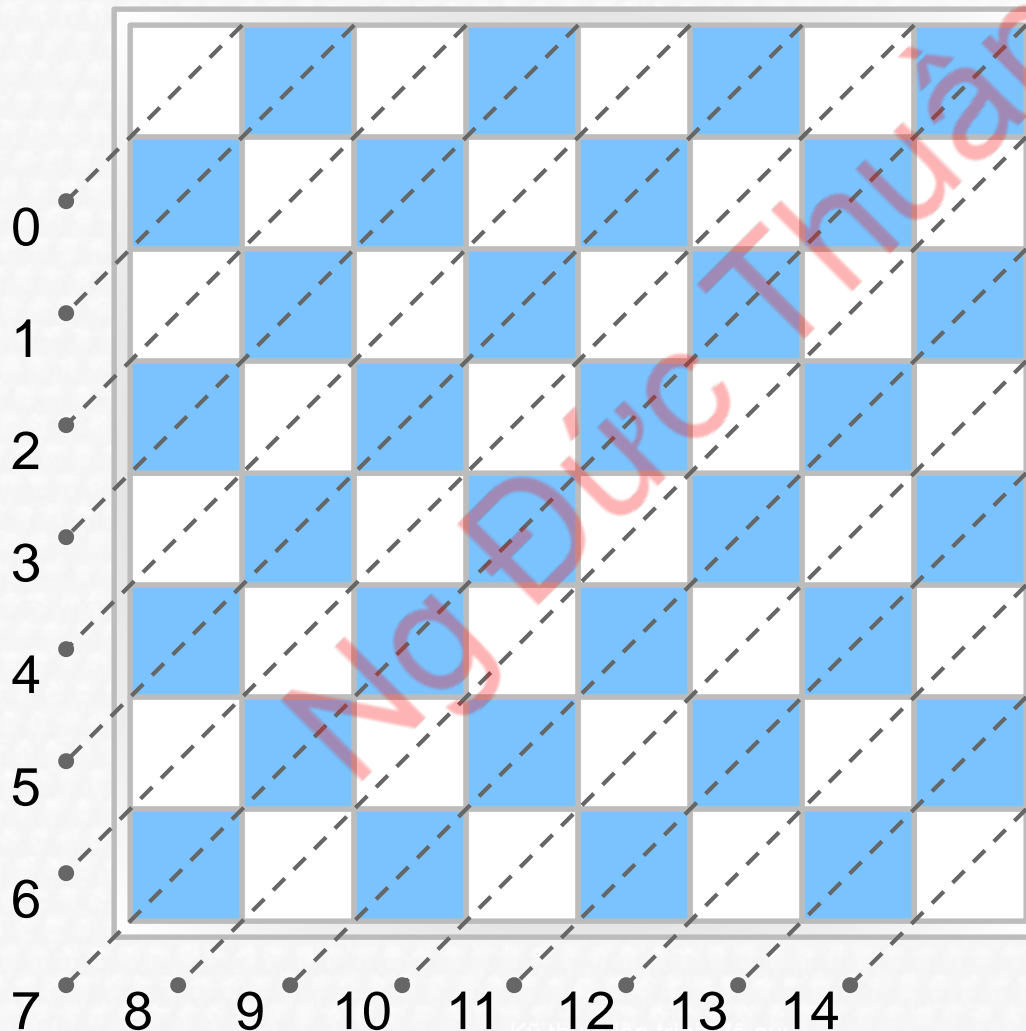




# Tám hậu – Các đường chéo xuôi

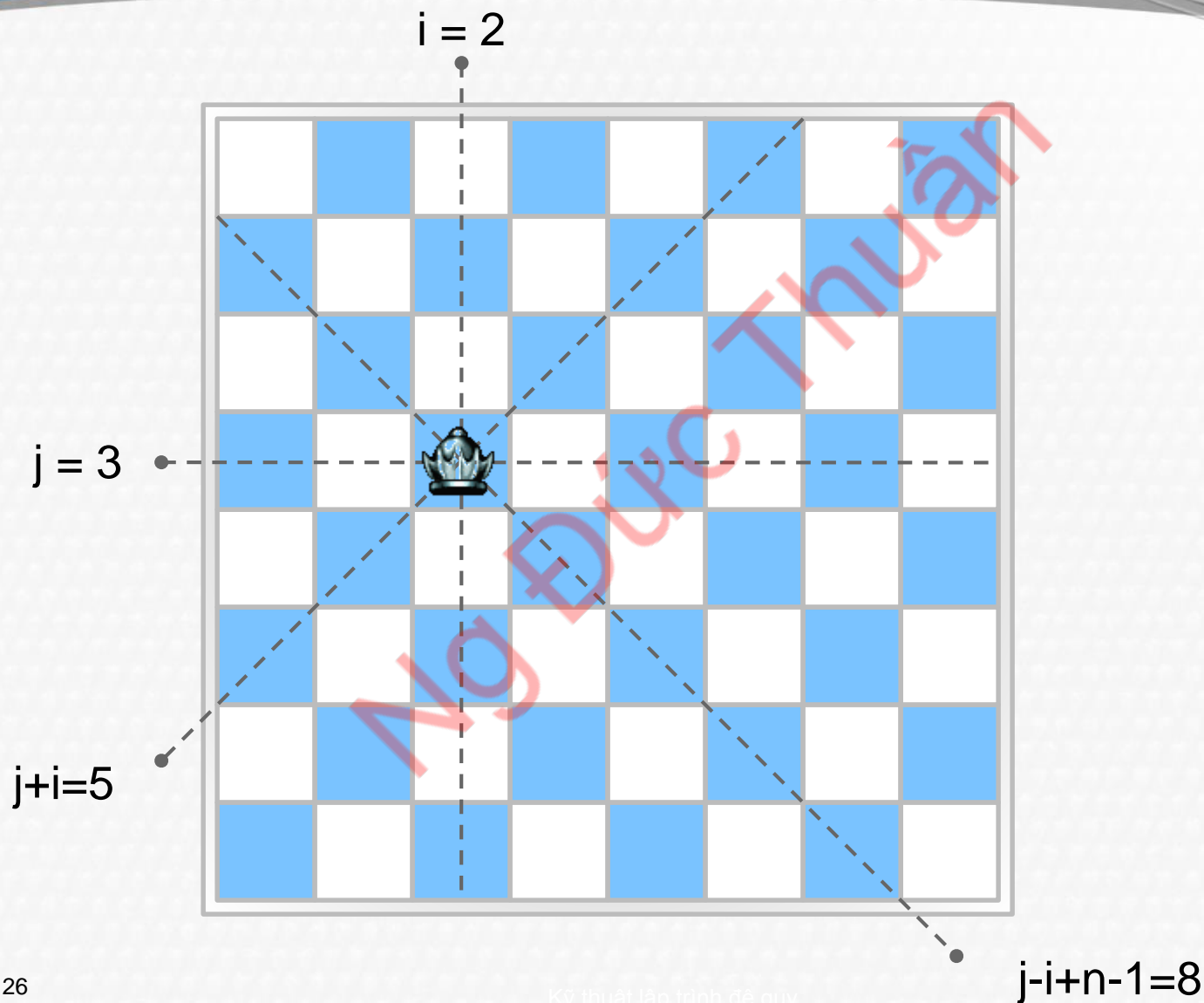


# Tám hậu – Các đường chéo ngược



$2n-1$  đường

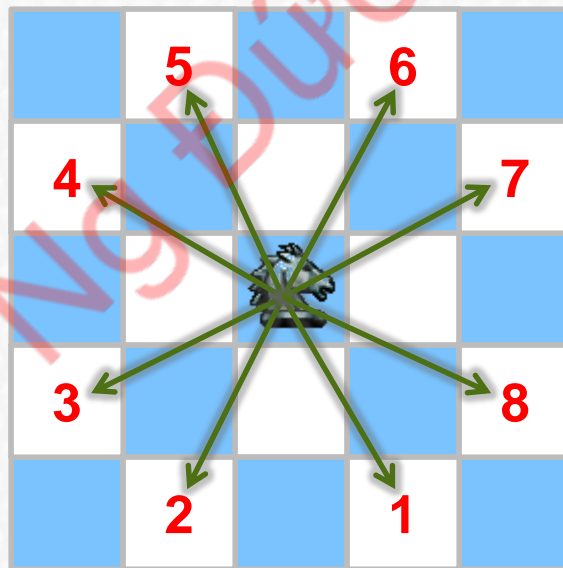
# Tám hậu – Các dòng



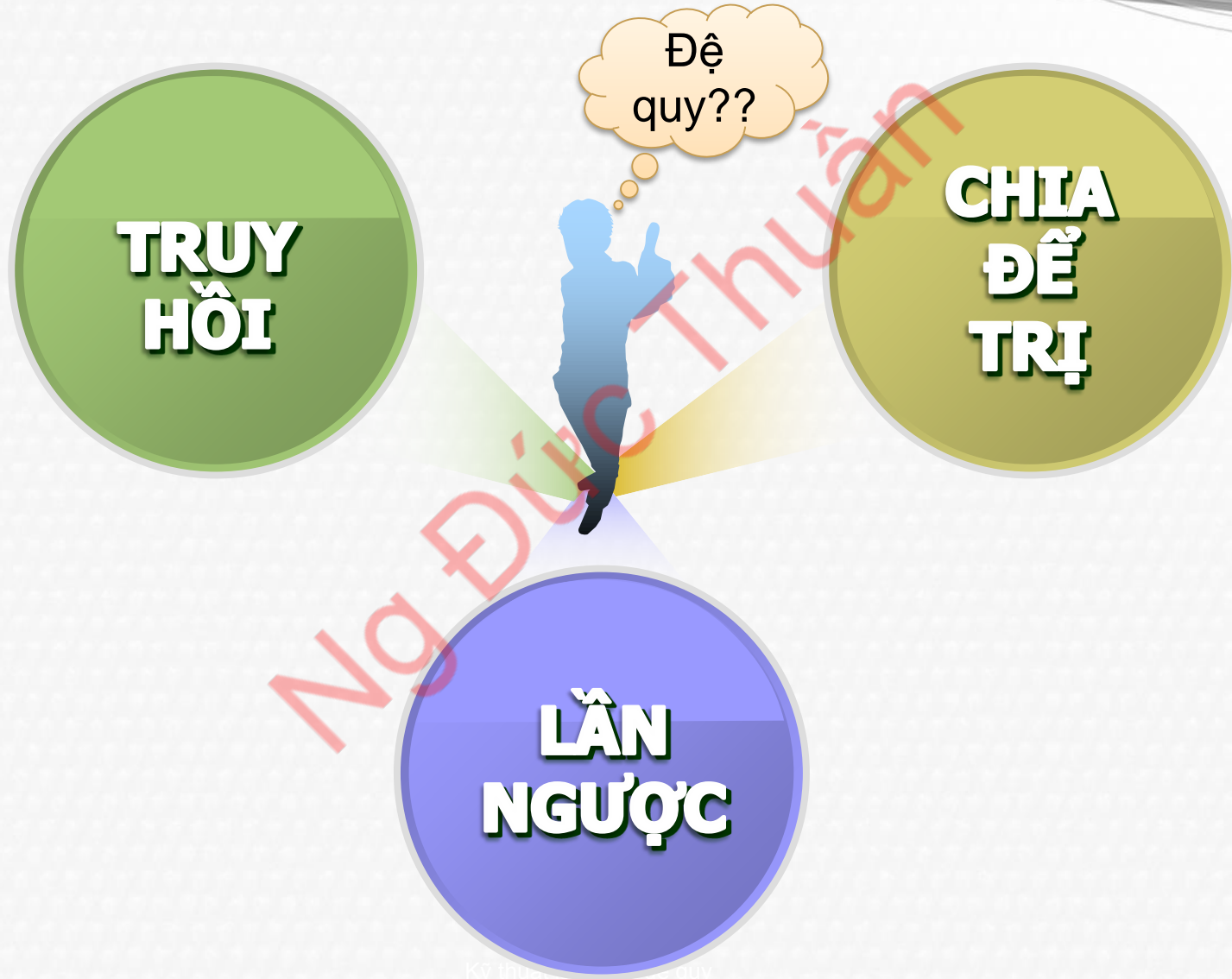
# Mã đi tuần

## ■ Mô tả bài toán

- Cho bàn cờ vua kích thước 8x8 (64 ô)
- Hãy đi con mã 64 nước sao cho mỗi ô chỉ đi qua 1 lần (xuất phát từ ô bất kỳ) theo luật:



# Các vấn đề đệ quy thông dụng

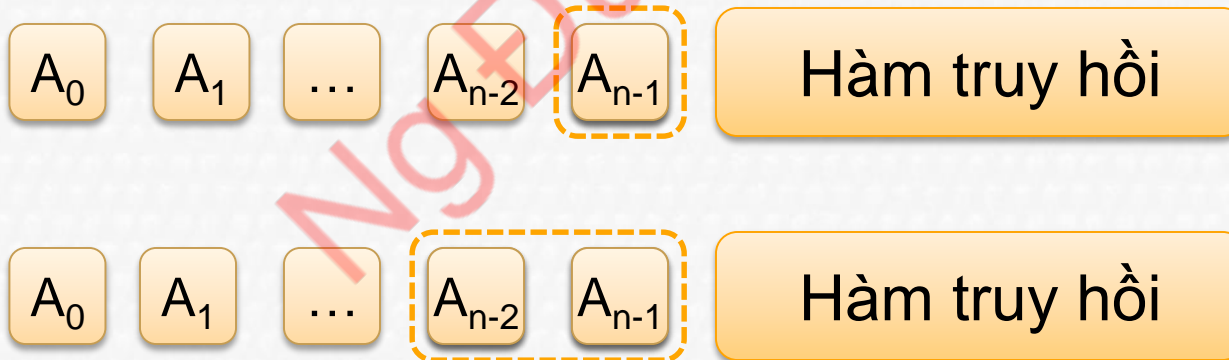




## a. Hệ thức truy hồi

### ■ Khái niệm

- Hệ thức truy hồi của 1 dãy  $A_n$  là công thức biểu diễn phần tử  $A_n$  thông qua 1 hoặc nhiều số hạng trước của dãy.



## a. Hệ thức truy hồi

### ■ Ví dụ 1

- Vi trùng cứ 1 giờ lại nhân đôi. Vậy sau 5 giờ sẽ có mấy con vi trùng nếu ban đầu có 2 con?

### ■ Giải pháp

- Gọi  $V_h$  là số vi trùng tại thời điểm  $h$ .

- Ta có:

- $V_h = 2V_{h-1}$

- $V_0 = 2$

➔ Độ quy tuyến tính với  $V(h) = 2 * V(h-1)$  và điều kiện dừng  $V(0) = 2$

## a. Hệ thức truy hồi

### ▪ Ví dụ 2

- Gửi ngân hàng 1000 USD, lãi suất 12%/năm. Số tiền có được sau 30 năm là bao nhiêu?

### ▪ Giải pháp

- Gọi  $T_n$  là số tiền có được sau  $n$  năm.

- Ta có:

- $T_n = T_{n-1} + 0.12T_{n-1} = 1.12T_{n-1}$

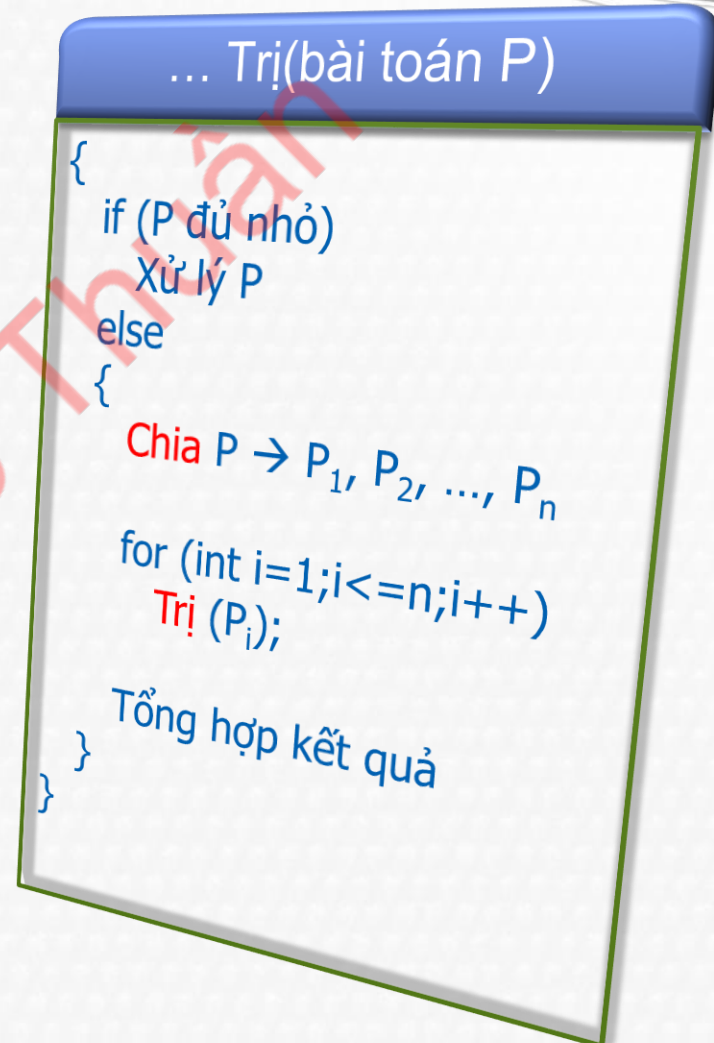
- $V(0) = 1000$

→ Đệ quy tuyến tính với  $T(n) = 1.12 \cdot T(n-1)$  và điều kiện dừng  $V(0) = 1000$

## b. Chia để trị (divide & conquer)

### ■ Khái niệm

- Chia bài toán thành nhiều bài toán con.
- Giải quyết từng bài toán con.
- Tổng hợp kết quả từng bài toán con để ra lời giải.



## b.Chia để trị (divide & conquer)

### ■ Ví dụ

- Cho dãy A đã sắp xếp thứ tự tăng. Tìm vị trí phần tử x trong dãy (nếu có)

### ■ Giải pháp

- $mid = (l + r) / 2;$
  - Nếu  $A[mid] = x \rightarrow$  trả về mid.
  - Ngược lại
    - Nếu  $x < A[mid] \rightarrow$  tìm trong đoạn  $[l, mid - 1]$
    - Ngược lại  $\rightarrow$  tìm trong đoạn  $[mid + 1, r]$
- $\rightarrow$  Sử dụng đệ quy nhị phân.



## 2.Chia để trị (divide & conquer)

### ▪ Một số bài toán khác

- Bài toán tháp Hà Nội
- Các giải thuật sắp xếp: QuickSort, MergeSort
- Các giải thuật tìm kiếm trên cây nhị phân tìm kiếm, cây nhị phân nhiều nhánh tìm kiếm.

### ▪ Lưu ý

- Khi bài toán lớn được chia thành các bài toán nhỏ hơn mà những bài toán nhỏ hơn này không đơn giản nhiều so với bài toán gốc thì **không nên** dùng kỹ thuật chia để trị.

## c. Lăn ngược (Backtracking)

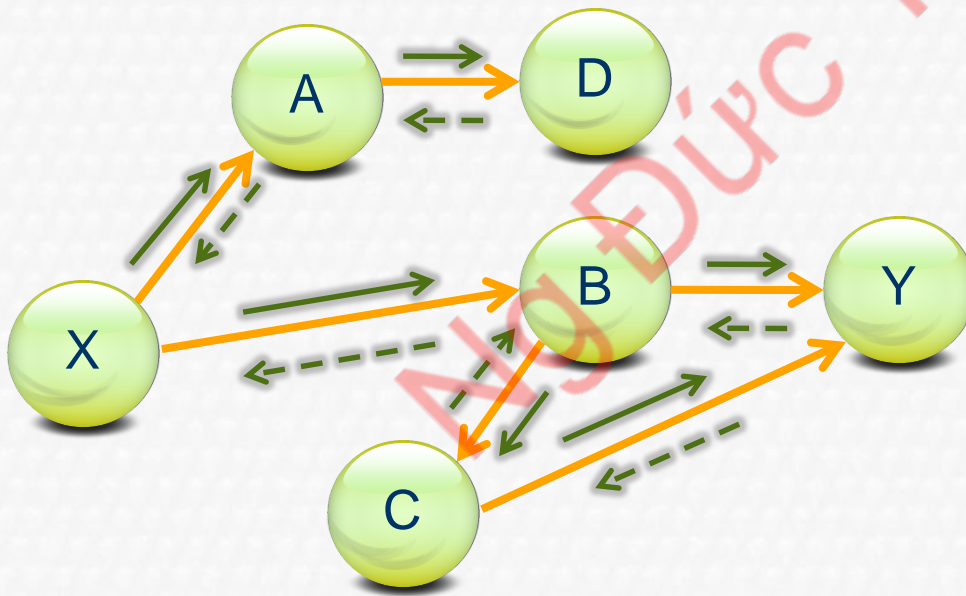
### ■ Khái niệm

- Tại bước có nhiều lựa chọn, ta chọn thử 1 bước để đi tiếp.
- Nếu không thành công thì “lăn ngược” chọn bước khác.
- Nếu đã thành công thì ghi nhận lời giải này đồng thời “lăn ngược” để truy tìm lời giải mới.
- Thích hợp giải các bài toán kinh điển như bài toán 8 hậu và bài toán mã đi tuần.

## c. Lăn ngược (Back tracking)

### ■ Ví dụ

- Tìm đường đi từ **X** đến **Y**.



Kết quả			
X	B	Y	
X	B	C	Y

# Thuật toán Thử sai-Quay lui

```
void try(int i)
```

```
{ Duyệt < không gian trạng thái ứng viên ở bước i >
```

```
  { ghi nhận trạng thái chấp nhận được;
```

```
  if (i < n) || (chưa tìm thấy nghiệm) try(i+1);
```

```
  else
```

```
    Kiểm tra nghiệm – xuất kết quả;
```

```
    xóa trạng thái đã ghi nhận;
```

```
  }
```

# Nhận xét

## ■ Ưu điểm

- Sáng sủa, dễ hiểu, nêu rõ bản chất vấn đề.
- Tiết kiệm thời gian thực hiện mã nguồn.
- Một số bài toán rất khó giải nếu không dùng đệ quy.

## ■ Khuyết điểm

- Tốn nhiều bộ nhớ, thời gian thực thi lâu.
- Một số tính toán có thể bị lặp lại nhiều lần.
- Một số bài toán không có lời giải đệ quy.



# BÀI TẬP

Ng Đức Thuận

# LẬP CHƯƠNG TRÌNH CHO CÁC BÀI TẬP SAU

- **Bài 1:** Các bài tập trên mảng sử dụng đệ quy.
- **Bài 2:** Viết hàm đệ quy xác định chiều dài chuỗi.
- **Bài 3:** Hiển thị n dòng của tam giác Pascal.

$$- a[i][0] = a[i][i] = 1$$

$$- a[i][k] = a[i-1][k-1] + a[i-1][k]$$

Dòng 0: 1

Dòng 1: 1 1

Dòng 2: 1 2 1

Dòng 3: 1 3 3 1

Dòng 4: 1 4 6 4 1

# LẬP CHƯƠNG TRÌNH CHO CÁC BÀI TẬP SAU

- **Bài 4:** Viết hàm đệ quy tính  $C(n, k)$  biết
  - $C(n, k) = 1$  nếu  $k = 0$  hoặc  $k = n$
  - $C(n, k) = 0$  nếu  $k > n$
  - $C(n, k) = C(n-1, k) + C(n-1, k-1)$  nếu  $0 < k < n$
- **Bài 5:** Đổi 1 số thập phân sang cơ số khác.
- **Bài 6:** Tính các tổng truy hồi.
- **Bài 7:** Bài toán In ra hoán vị  $n$  số tự nhiên đầu tiên”.
- **Bài 8:** Bài toán “8 hậu”.
- **Bài 9:** Bài toán “Mã đi tuần”.

# Cám ơn đã theo dõi

