Kỹ thuật đệ qui

Kỹ thuật lập trình

ThS. Đặng Bình Phương (dbphuong@fit.hcmus.edu.vn)

Nội dung

- Giới thiệu về lập trình đệ qui
- Phân loại các dạng đệ qui
- Một số ứng dụng của giải pháp đệ qui
- Những ví dụ về giải pháp thay thế cho đệ qui
- Đồ án lập trình
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh

Giới thiệu về lập trình đệ qui

Ví dụ

- Cho S(n) = 1 + 2 + 3 + … + n
- Tính S(10) và S(11)

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

$$S(11) = \underbrace{1 + 2 + \dots + 10}_{S(10)} + 11 = 66$$

$$= S(10) + 11$$

$$= 55 + 11 = 66$$

Khái niệm đệ qui

- Khái niệm
  - Vấn đề đệ quy là vấn đề được định nghĩa bằng chính nó.
- 2 điều kiện quan trọng
  - Tồn tại bước đệ qui
  - Điều kiện dừng
- Ví dụ trong bài toán trước thì:
  - Bước đệ qui: S(n) = S(n – 1) + n
  - Điều kiện dừng: S(1) = 1

Phân loại các dạng đệ qui

Phân loại

- ***Đệ qui tuyến tính (đệ qui thông thường và đệ qui đuôi)***: Trong thân hàm có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.
- ***Đệ qui nhị phân***: Trong thân hàm có hai lời gọi hàm gọi lại chính nó một cách tường minh.
- ***Đệ qui hỗ tương***: Trong thân hàm này có lời gọi hàm tới hàm kia và bên trong thân hàm kia có lời gọi hàm tới hàm này.
- ***Đệ qui phi tuyến***: Trong thân hàm có lời gọi hàm lại chính nó nằm bên trong thân vòng lặp.

Ví dụ đệ qui tuyến tính

- Tính S(n) = 1 + 2 + … + n
  - S(n) = S(n – 1) + n
  - S(0) = 0

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

Ví dụ đệ qui đuôi

- Lời gọi đệ qui là thao tác cuối cùng.
- Tính S(n) = 1 + 2 + … + n
  - S(n) = S(n – 1) + n
  - S(0) = 0

```
long Tong(int n, int ret)      // Gọi hàm ret = 0
{
      if (n == 0)
            return ret;
      return Tong(n – 1, ret + n);
}
```

Ví dụ đệ qui nhị phân

- Tính số hạng thứ n của dãy Fibonacy
  - F(0) = F(1) = 1
  - F(n) = F(n – 1) + F(n – 2) n > 1

```
long Fibo(int n)
{
    if (n == 0 || n == 1)
        return 1;
    return Fibo(n–1) + Fibo(n – 2);
}
```

Ví dụ đệ qui hỗ tương

- Tính số hạng thứ n của dãy sau:
  - x(0) = 1, y(0) = 0
  - x(n) = x(n – 1) + y(n – 1)
  - y(n) = 3 * x(n – 1) + 2 * y(n – 1)

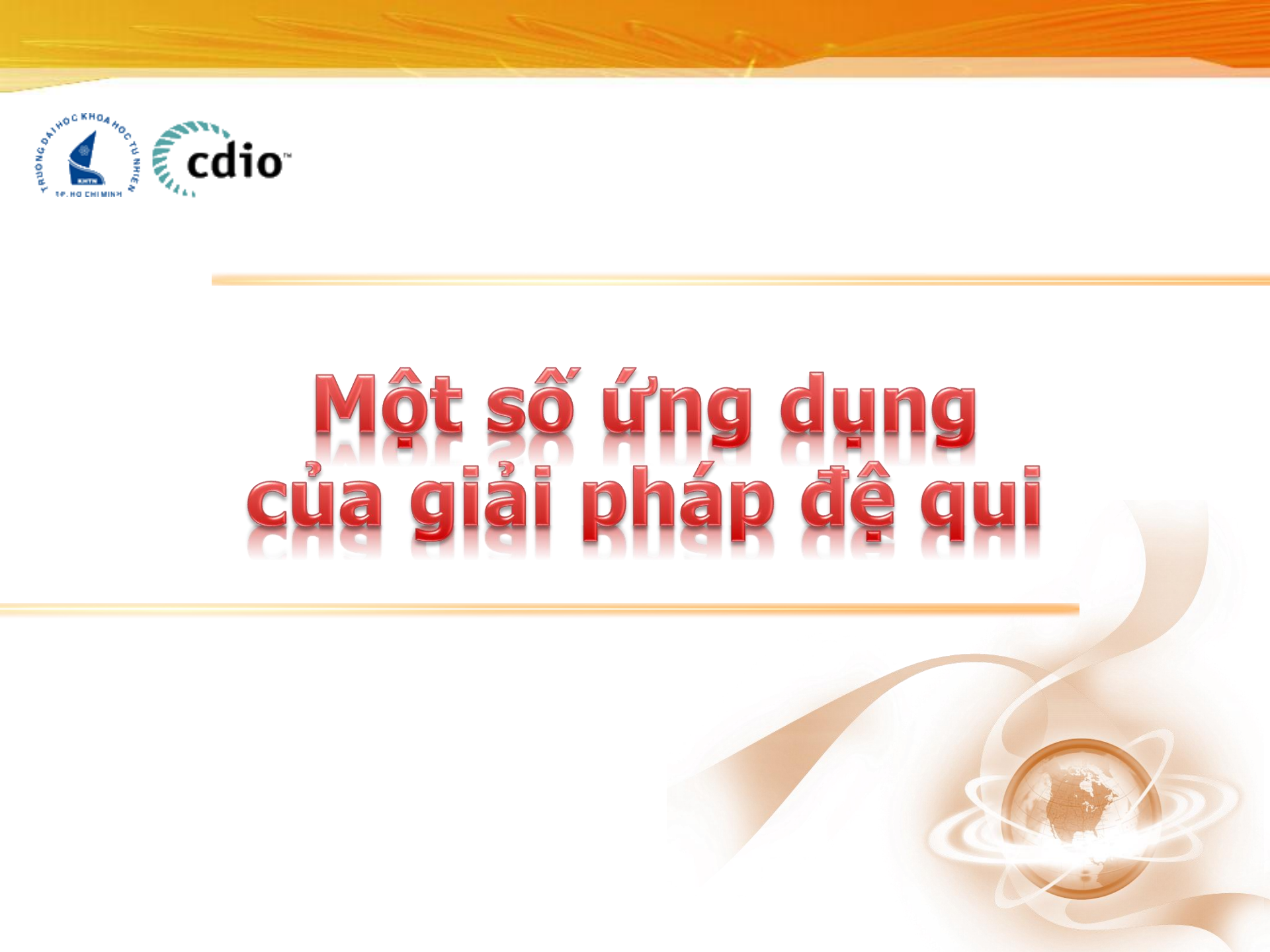
```
long xn(int n) {
    if (n == 0) return 1;
    return xn(n–1) + yn(n – 1);
}
long yn(int n) {
    if (n == 0) return 0;
    return 3 * xn(n–1) + 2 * yn(n – 1);
}
```

Ví dụ đệ qui phi tuyến

- Tính số hạng thứ n của dãy sau:
  - $x(0) = 1$
  - $x(n) = n^2x(0) + (n-1)^2x(1) + \ldots + 2^2x(n-2) + 1^2x(n-1)$

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

Một số ứng dụng của giải pháp đệ qui

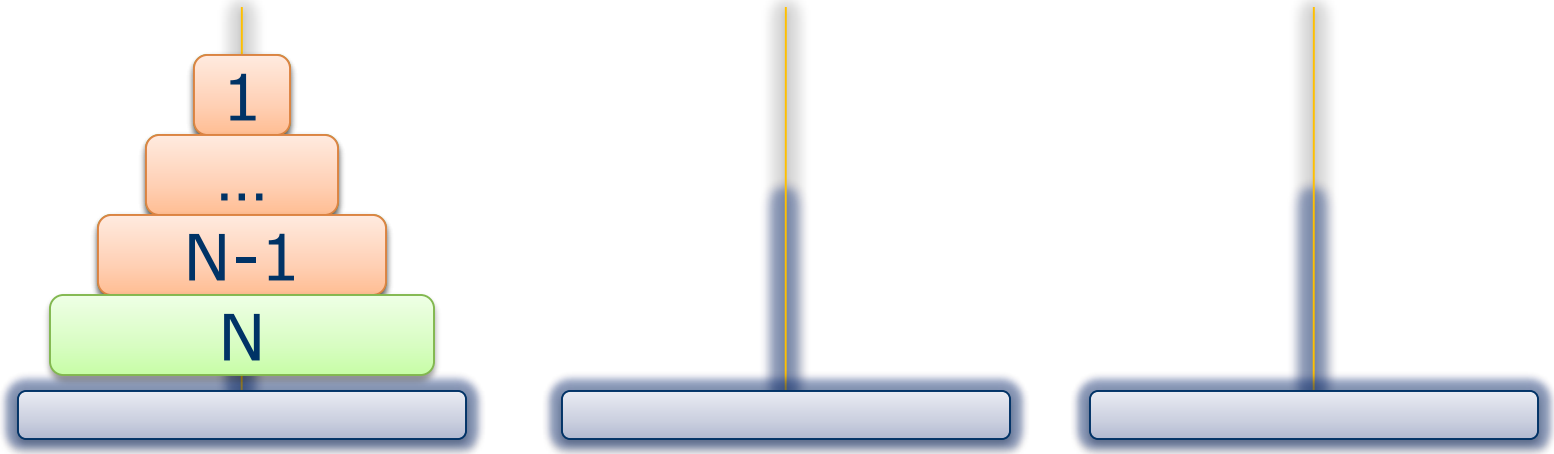


Bài toán tháp Hà Nội

- 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 A, B, C làm cột trung gian.

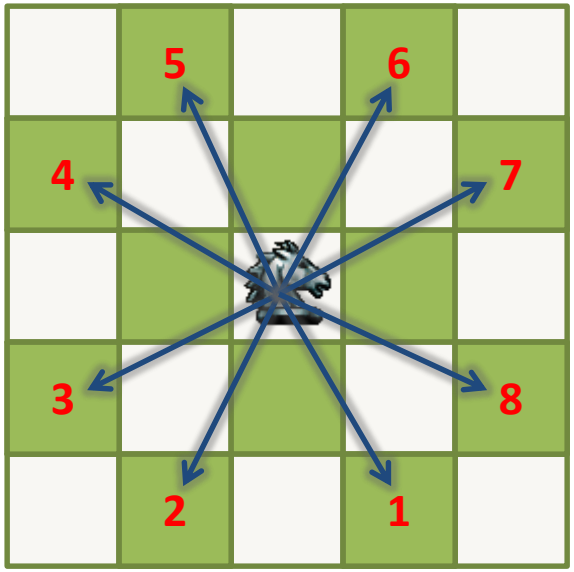
Bài toán tháp Hà Nội

N đĩa A → C = ? đĩa A → B + Đĩa N A → C + N-1 đĩa B → C



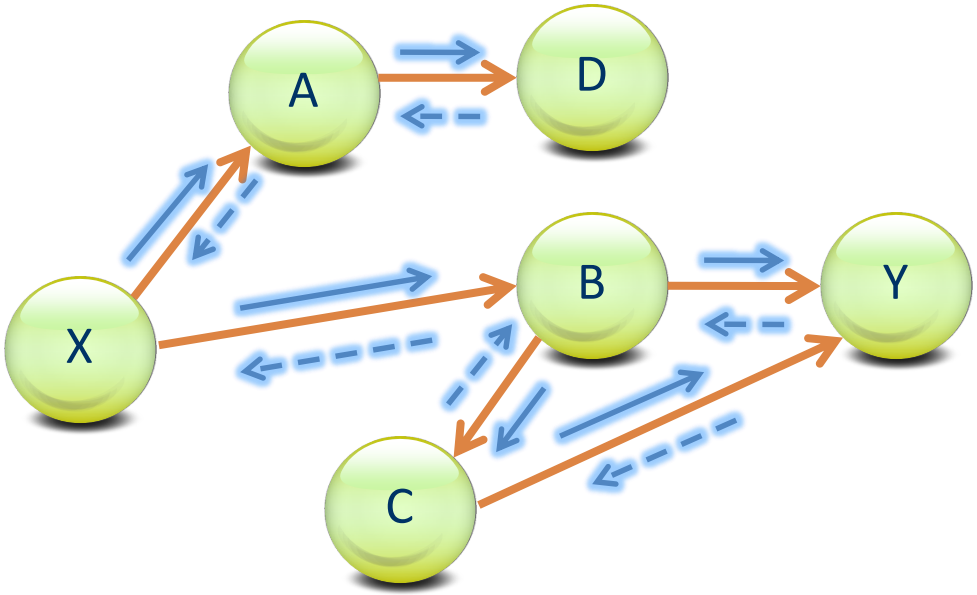
Bài toán mã đi tuần

- 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:



Bài toán tìm đường đi

- Ví dụ tìm đường đi từ X đến Y (sử dụng đệ qui lần ngược - backtracking)



Một số ứng dụng khác

- Tính các công thức truy hồi
- Phát sinh hoán vị, tổ hợp, chỉnh hợp
- Bài toán tìm cực trị hay tính toán lặp
- Bài toán sắp xếp trên mảng
- Bài toán trên lưới ô vuông
- Bài toán phân tích biểu thức và tính giá trị

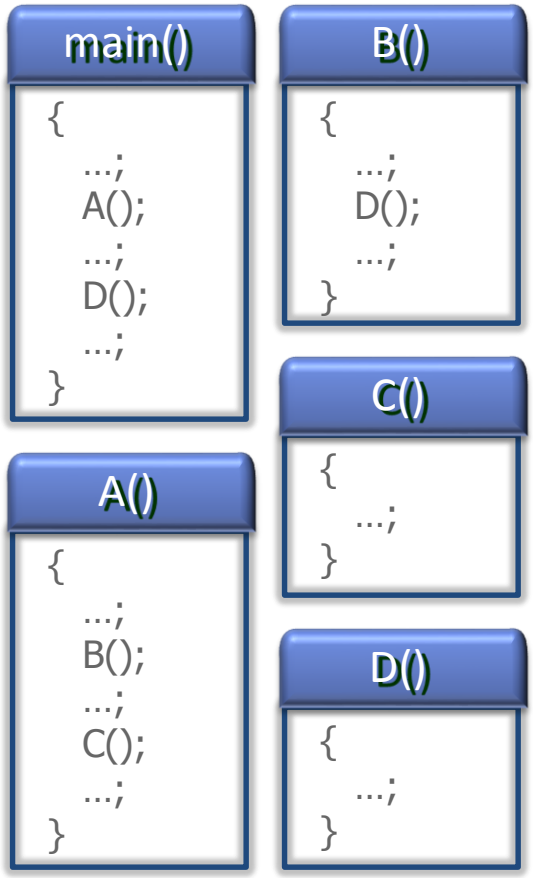
Những ví dụ về giải pháp thay thế cho đệ qui

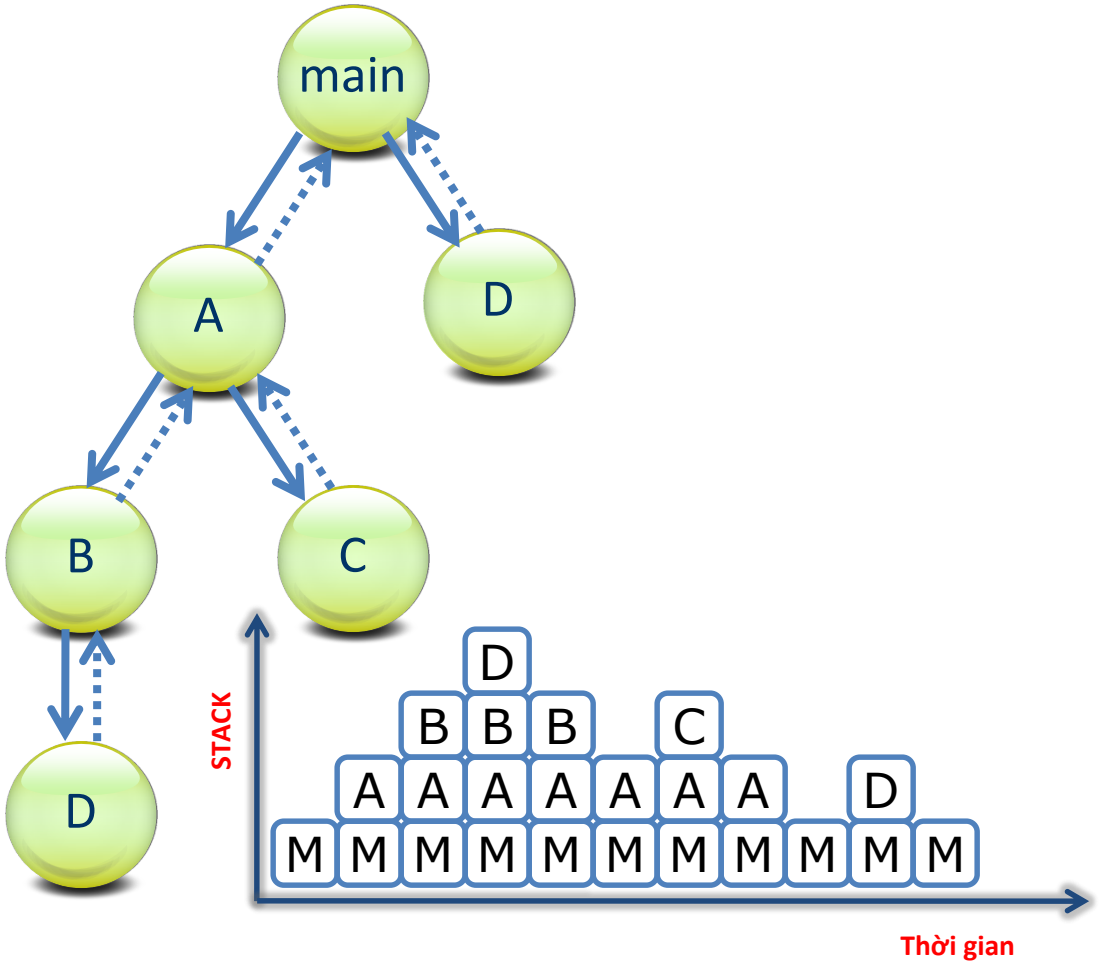
Ví dụ

- Sử dụng vòng lặp để:
  - Tính ước số chung lớn nhất
  - Phần tử thứ n của dãy Fibonacy
  - …
- Sử dụng ngăn xếp để:
  - Sắp xếp mảng sử dụng theo thuật toán sắp xếp nhanh (quick sort)
  - …

Kỹ thuật theo dõi hoạt động của chương trình đệ qui

Cơ chế gọi hàm và STACK





Cơ chế gọi hàm và STACK

- Nhận xét
  - Lưu thông tin trạng thái còn dở dang mỗi khi gọi đệ quy.
  - Thực hiện xong một lần gọi cần khôi phục thông tin trạng thái trước khi gọi.
  - Lệnh gọi cuối cùng sẽ hoàn tất đầu tiên.

Đồ án lập trình

Tính tích 2 chuỗi số cực lớn

- Yêu cầu: Tính tích 2 chuỗi số cực lớn X, Y
- Gợi ý:
  - $X = X_{2n-1}\dots X_n X_{n-1}\dots X_0$, $Y = Y_{2n-1}\dots Y_n Y_{n-1}\dots Y_0$
  - Đặt $X_L = X_{2n-1}\dots X_n$, $X_N = X_{n-1}\dots X_0 \Rightarrow X = 10^n X_L + X_N$
  - Đặt $Y_L = Y_{2n-1}\dots Y_n$, $Y_N = Y_{n-1}\dots Y_0 \Rightarrow Y = 10^n Y_L + Y_N$

    $\Rightarrow X*Y = 10^{2n} X_L Y_L + 10^n (X_L Y_L + X_N Y_N) + X_N Y_N$

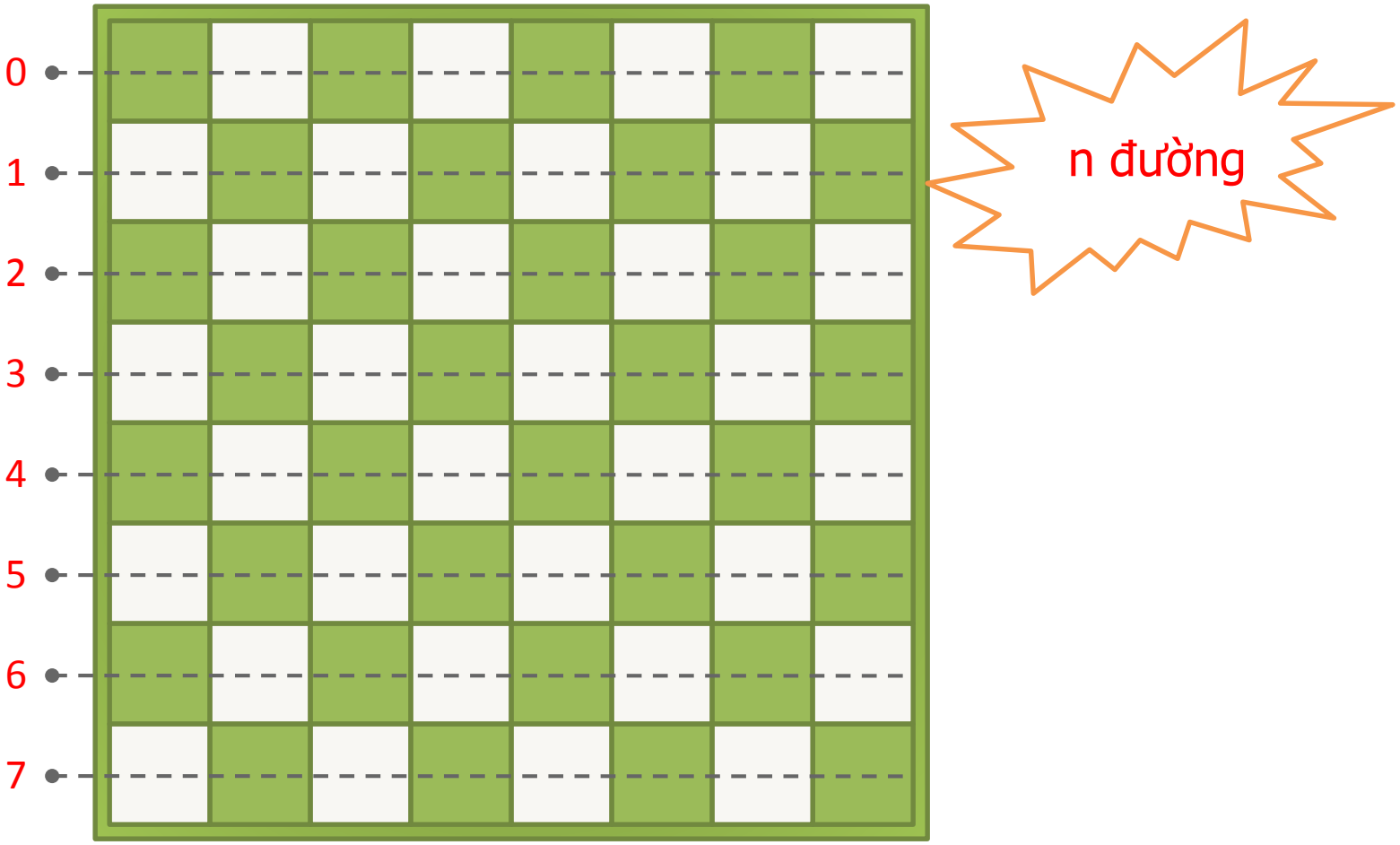
    và $X_L Y_L + X_N Y_N = (X_L - X_N)(Y_N - Y_L) + X_L Y_L + X_N Y_N$

$\Rightarrow$ Nhân 3 số nhỏ hơn (độ dài ½) đến khi có thể nhân được ngay.

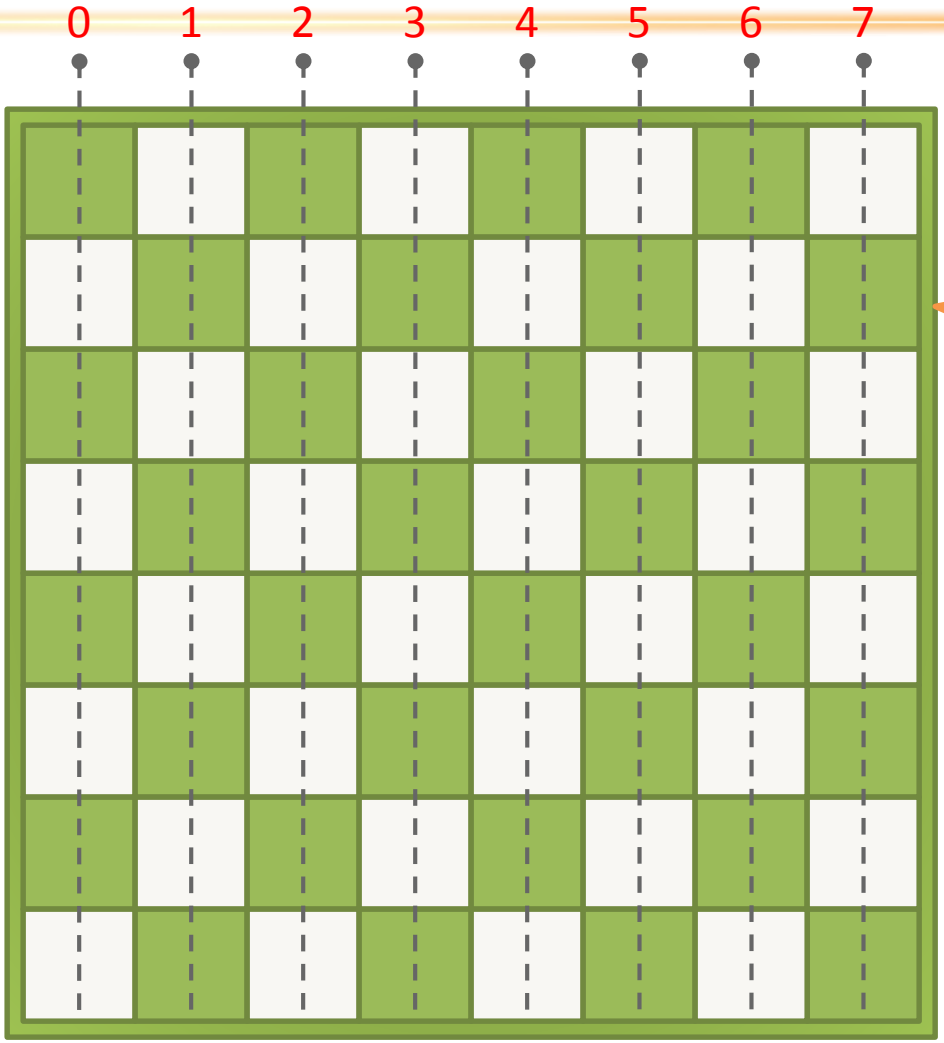
Bài toán tám hậu

- Mô tả bài toán
  - Cho bàn cờ vua kích thước 8x8
  - 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.
- Gợi ý: sử dụng đệ qui quay lui và sử dụng các mảng đánh dấu cột, đường chéo xuôi, đường chéo ngược.

Tám hậu – Dòng

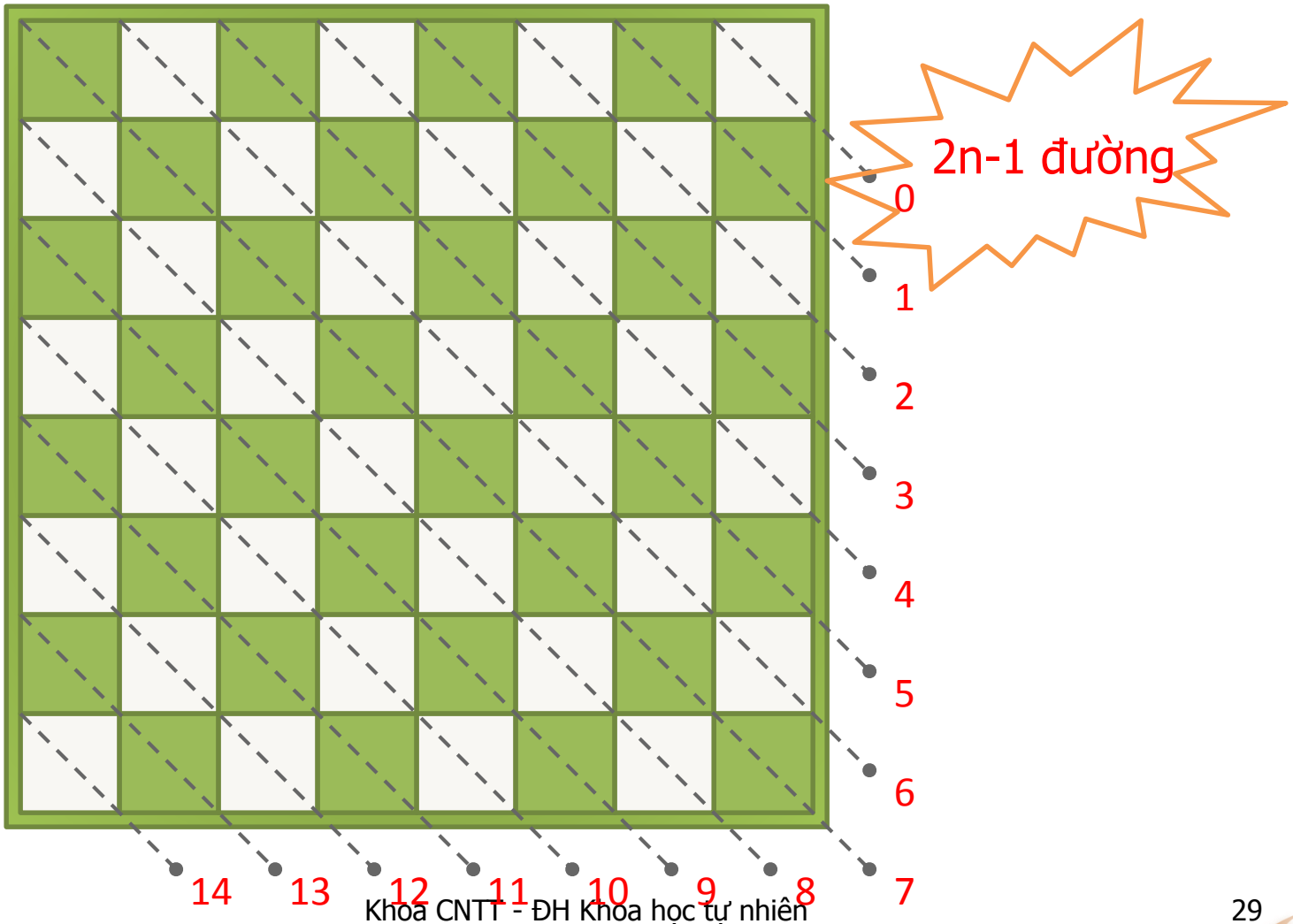


Tám hậu – Cột

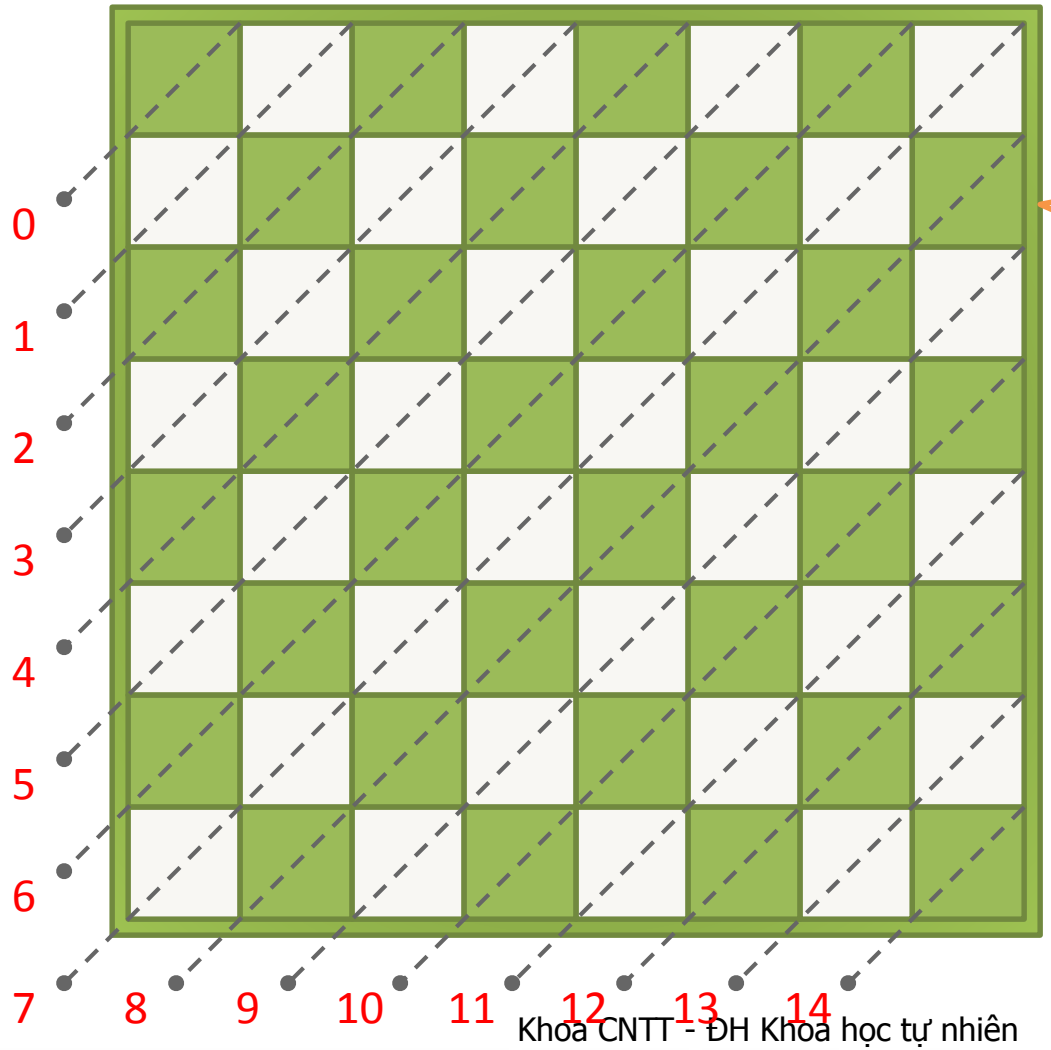


n đường

Tám hậu – Đường chéo xuôi

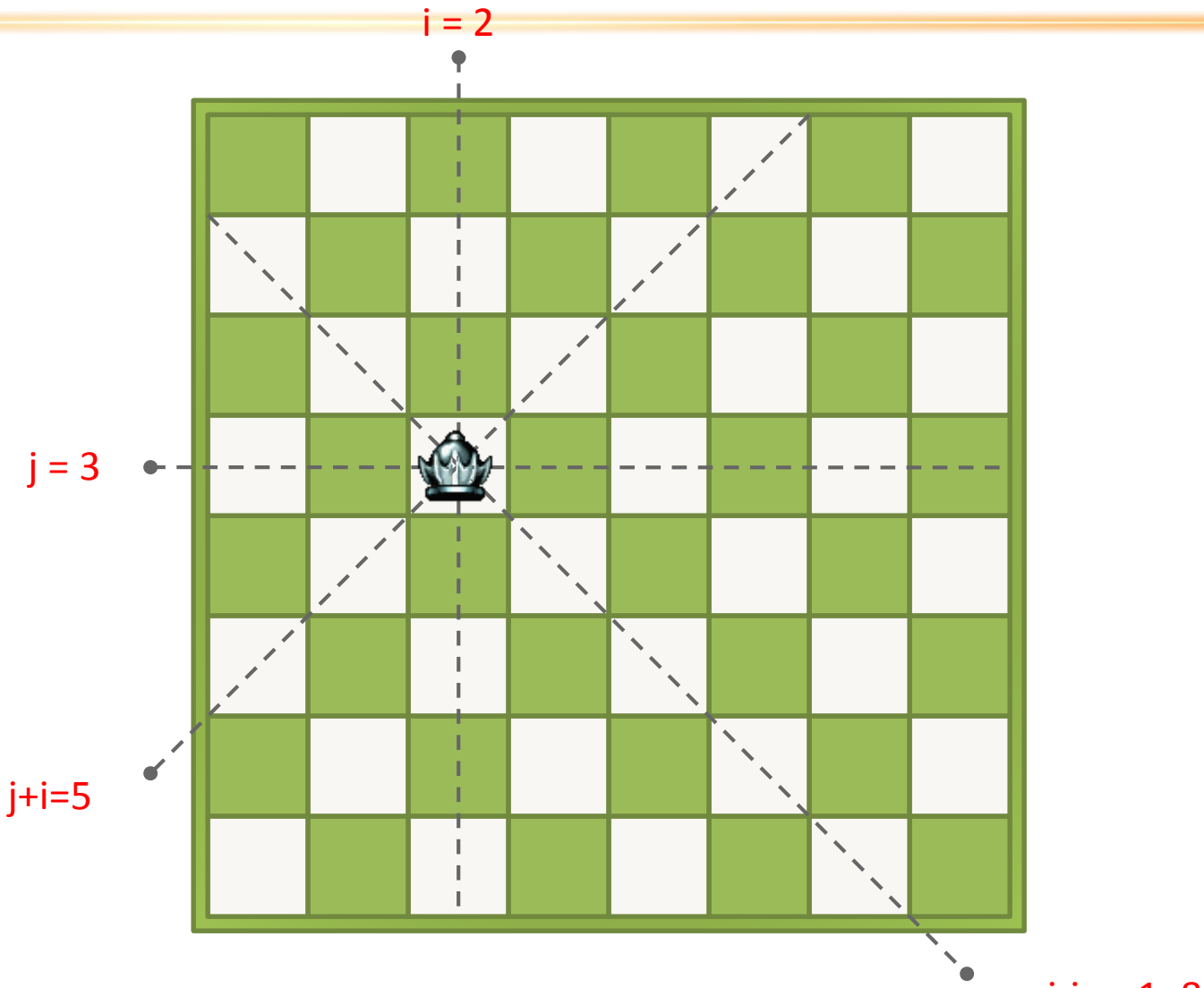


Tám hậu – Đường chéo ngược



2n-1 đường

Tám hậu – Tính toán



Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp

Độ phức tạp

- Một công cụ thường dùng khi phân tích độ phức tạp của giải thuật đệ qui là sử dụng cây đệ qui.
  - Chiếm dụng bộ nhớ: tỉ lệ với cấp (chiều sâu) của cây đệ qui (không phụ thuộc số nút có trên cây)
  - Thời gian chạy: đếm số tác vụ cần thực hiện trên cây đệ qui.

Độ phức tạp

- Ví dụ bài toán tháp Hà Nội:
  - Chiếm dụng bộ nhớ: tỉ lệ với chiều sâu của cây (bậc O(n) với n là số đĩa cần chuyển)
  - Thời gian chạy: tỉ lệ với số lần thực hiện tác vụ (số nút có trên cây) = $1 + 2 + 2^2 + 2^3 + \ldots + 2^{n-1} = 2^n - 1$ (bậc $O(n^2)$)

Độ phức tạp

- Cách khác để đánh giá độ phức tạp là thành lập phương trình đệ qui và giải phương trình này.
- Ví dụ bài toán tháp Hà Nội:
  - Phương trình đệ qui:
    - $T(n) = 2T(n - 1) + 1, (T(1) = 1$
  - Sử dụng truy hồi tính được: $T(n) = 2^n - 1$
  - Vậy $T(n) = O(n^2)$

Vấn đề khử đệ qui

- Giải thuật giải bài toán đệ qui thường rất gọn gàng, dễ hiểu nên dễ chuyển thành chương trình tuy nhiên tốn không gian nhớ và thời gian xử lý.
- Một cách tổng quá người ta chỉ ra rằng mọi giải thuật đệ qui đều có thể thay thế bằng một giải thuật không đệ qui (bằng vòng lặp hoặc ngăn xếp)

Vấn đề khử đệ qui

- Thông thường các bước xây dựng chương trình cho một bài toán khó khi không tìm được giải thuật không đệ qui là:
  - Dùng quan niệm đệ qui để tìm giải thuật cho bài toán.
  - Mã hóa giải thuật đệ qui.
  - Khử đệ qui để có được một chương trình không đệ qui.

Thuật ngữ
và bài đọc thêm tiếng Anh

Thuật ngữ tiếng Anh

- ***backtracking***: quay lui, lần ngược.
- ***binary recursion***: đệ qui nhị phân.
- ***devide and conquer***: chia để trị.
- ***linear recursion***: đệ qui tuyến tính.
- ***mutual recursion***: đệ qui hỗ tương.
- ***nested recursion***: đệ qui phi tuyến.
- ***recursion***: đệ qui.
- ***recursion step***: bước đệ qui.
- ***stopping condition***: điều kiện dừng.
- ***tail recursion***: đệ qui đuôi.

Bài đọc thêm tiếng Anh

- **Theory and Problems of Fundamentals of Computing with C++**, John R.Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.