Chương 2
Đệ quy và giải thuật đệ quy

Ths. Phạm Thanh An
Bộ môn Khoa học máy tính- Khoa CNTT
Trường Đại học Ngân hàng TP.HCM

LOGO

Nội dung

- Khái niệm đệ quy
- Giải thuật và chương trình đệ quy
- Thiết kế giải thuật đệ quy
- Ưu nhược điểm của đệ quy
- Một số dạng giải thuật đệ quy thường gặp
- Giải thuật đệ qui quay lui (backtracking)
- Một số bài toán giải bằng giải thuật đệ quy điển hình
- Đệ quy và quy nạp toán học

Mục tiêu

- Trang bị cho sinh viên các khái niệm và cách thiết kế giải thuật đệ qui, giải thuật đệ qui quay lui.
- Giới thiệu một số bài toán điển hình được giải bằng giải thuật đệ qui.
- Phân tích ưu và nhược điểm khi sử dụng giải thuật đệ qui

Khái niệm về đệ qui

- Đệ quy: Đưa ra 1 định nghĩa có sử dụng chính khái niệm đang cần định nghĩa(quay về).
- Ví dụ
  - **Người** = con của hai ***người*** khác.
  - Trong toán học:
    - Số tự nhiên: 0 là số tự nhiên, n là số tự nhiên nếu n- 1 là số tự nhiên
    - Hàm n!
- Khái niệm về đệ

Giải thuật và hàm đệ quy

- Giải thuật đệ quy
  - Nếu bài toán T được thực hiện bằng lời giải của bài toán T ' có dạng giống T là lời giải đệ quy
  - Giải thuật tương ứng với lời giải như vậy gọi là giải thuật đệ quy.
- Hàm đệ quy

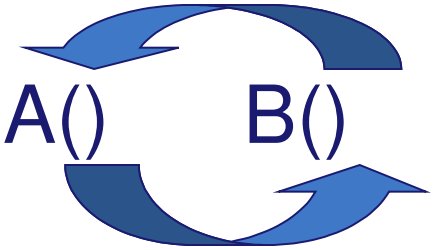
Giải thuật đệ quy

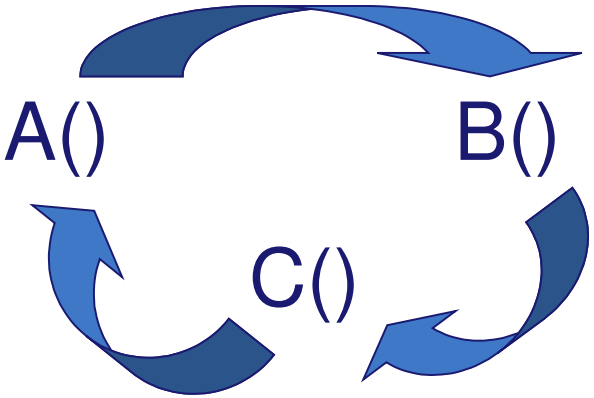
- Ví dụ: Xét bài toán tìm một từ trong quyển từ điển:

```
If (từ điển là một trang)
   tìm từ trong trang này
else {
     Mở từ điển vào trang "giữa"
     Xác định xem nửa nào của từ điển chứa từ cần
   tìm;
     if (từ đó nằm ở nửa trước)
              tìm từ đó ở nửa trước
     else     tìm từ đó ở nửa sau.
}
```

Phân loại giải thuật đệ qui

- Đệ quy phân thành 2 loại :
  - Đệ quy trực tiếp:
  - Đệ quy gián tiếp (Tương hỗ):





Cài đặt hàm đệ quy

- Hàm đệ quy về cơ bản gồm hai phần:
  - Phần cơ sở (Phần neo):
  - Phần đệ quy:

Cài đặt hàm đệ quy (tt)

- Cấu trúc hàm đệ qui như sau

```
If (suy biến)
    <Giải quyết trường hợp suy biến>;
Else
    { <tiền xử lý đệ qui>;
     <Gọi đệ qui> ;
     <Xử lý hậu đệ qui>;
     }
```

Một số dạng giải thuật đệ quy đơn giản thường gặp

- **Đệ quy tuyến tính.** Hàm đệ qui tuyến tính dạng:

```
P (<tham số>)
{
        if (điều kiện dừng)
        {
          <Xử lý trường hợp neo>
        }
        Else
        {
        <Thực hiện một số công việc (nếu có)>
          P(<tham số>);
         <Thực hiện một số công việc (nếu có)>
        }
 }
```

Một số dạng giải thuật đệ quy đơn giản thường gặp (tt)

- Ví dụ 1 : Hàm Fact(n) tính số hạng n của dãy n!, định nghĩa như sau:
  - $fact_0 = 1$;
  - $f_n = n*fact_{n-1;}$ (n>=1)

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

Một số dạng giải thuật đệ quy đơn giản thường gặp (tt)

- **Đệ quy nhị phân.**

```
P (<tham số>)
{
        if (điều kiện dừng)
        {
          <Xử lý trường hợp neo>
        }
        Else
        {
         <Thực hiện một số công việc (nếu có)>
          P(<tham số>);
         <Thực hiện một số công việc (nếu có)>
          P(<tham số>);
          <Thực hiện một số công việc (nếu có)>
        }
 }
```

Một số dạng giải thuật đệ quy đơn giản thường gặp (tt)

- Ví dụ 1: Tính số hạng thứ n của dãy Fibonaci được định nghĩa như sau:
  - $f_1 = f_0 = 1$;
  - $f_n = f_{n-1} + f_{n-2}$; (n>1)

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

Một số dạng giải thuật đệ quy đơn giản thường gặp (tt)

- **Đệ quy phi tuyến.**

```
P (<danh sách tham số>) {
  for (int i = 1; i<=n; i++)
      {
       <Thực hiện một số công việc (nếu có)>
      if (điều kiện dừng)
            {
            <Xử lý trường hợp neo>
            }
      else
            {
                  <Thực hiện một số công việc (nếu có)>
                  P (<danh sách tham số>);
            }
      }
}
```

Một số dạng giải thuật đệ quy đơn giản thường gặp (tt)

- Ví dụ : Cho dãy $\{X_n\}$ xác định theo công thức truy hồi :
  - $X_0 = 1$; $X_n = n^2X_O + (n-1)^2X_1 + \ldots + 2^2X_{n-2} + 1^2X_{n-1}$

```
int X(int n ) ;
{ if ( n == 0 ) return 1 ;
else
{ int tg = 0 ;
for (int i = 0 ; i<n ; i++ ) tg = tg + sqr(n-i)*X(i);
return ( tg ) ;
}
```

Một số dạng giải thuật đệ quy đơn giản thường gặp (tt)

- **Đệ qui tương hỗ:**

```
P2(<danh sách tham số>);// khai báo nguyên mẫu
P1(<danh sách tham số>)
{
    <Thực hiện một số công việc (nếu có)>
    …P2 (<danh sách tham số>);
    <Thực hiện một số công việc (nếu có)>
}

P2 (<danh sách tham số>)
    {
     <Thực hiện một số công việc (nếu có)>
    P1 (<danh sách tham số>);
    <Thực hiện một số công việc (nếu có)>
}
```

Một số dạng giải thuật đệ quy đơn giản thường gặp (tt)

- Ví dụ: Tính số hạng thứ n của hai dãy $\{X_n\}$, $\{Y_n\}$ được định nghĩa như sau:
  - $X_0 = Y_0 = 1$; $X_n = X_{n-1} + Y_{n-1}$; (n>0) ; $Y_n = n_2 X_{n-1} + Y_{n-1}$; (n>0)

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

```
long TinhYn (int n)
    {
     if(n==0)
       return 1;
       return n*n*TinhXn(n-
1) +          TinhYn(n-1);
     }
```

Thiết kế giải thuật đệ qui

- Để xây dựng giải thuật đệ quy, ta cần thực hiện tuần tự 3 nội dung sau :
  - Thông số hóa bài toán .
  - Tìm các trường hợp neo cùng giải thuật giải tương ứng .
  - Tìm giải thuật giải trong trường hợp tổng quát bằng phân rã bài toán theo kiểu đệ quy.

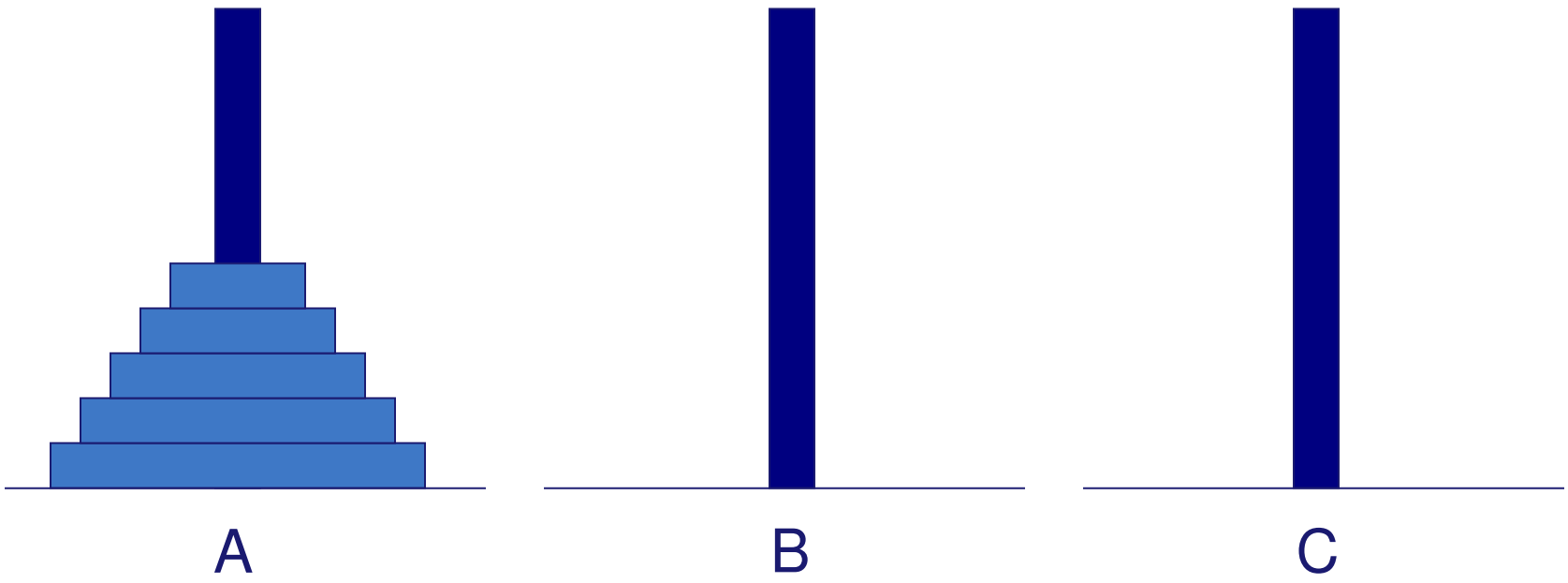
Ưu và nhược điểm của đệ qui

- Ưu điểm của đệ quy
  - Sáng sủa, dễ hiểu, nêu rõ bản chất vấn đề
  - Tiết kiệm thời gian hiện thực mã nguồn
- Nhược điểm của đệ quy
  - Tốn nhiều bộ nhớ, thời gian thực thi lâu
  - Một số bài toán không có lời giải đệ quy

Một số bài toán giải bằng giải thuật đệ qui điển hình

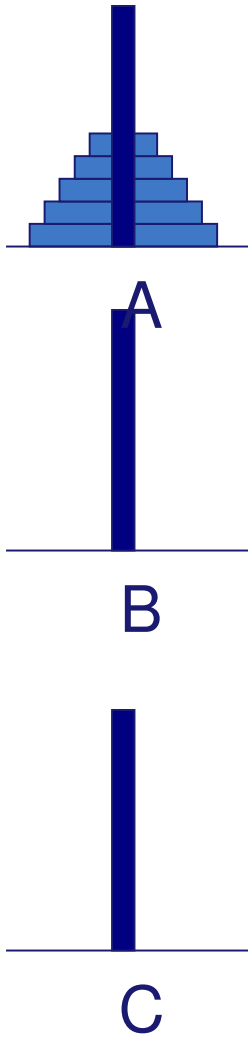
- Bài toán Tháp Hà Nội
- Bài toán chia thưởng

Bài toán tháp Hà Nội



Bài toán tháp Hà Nội

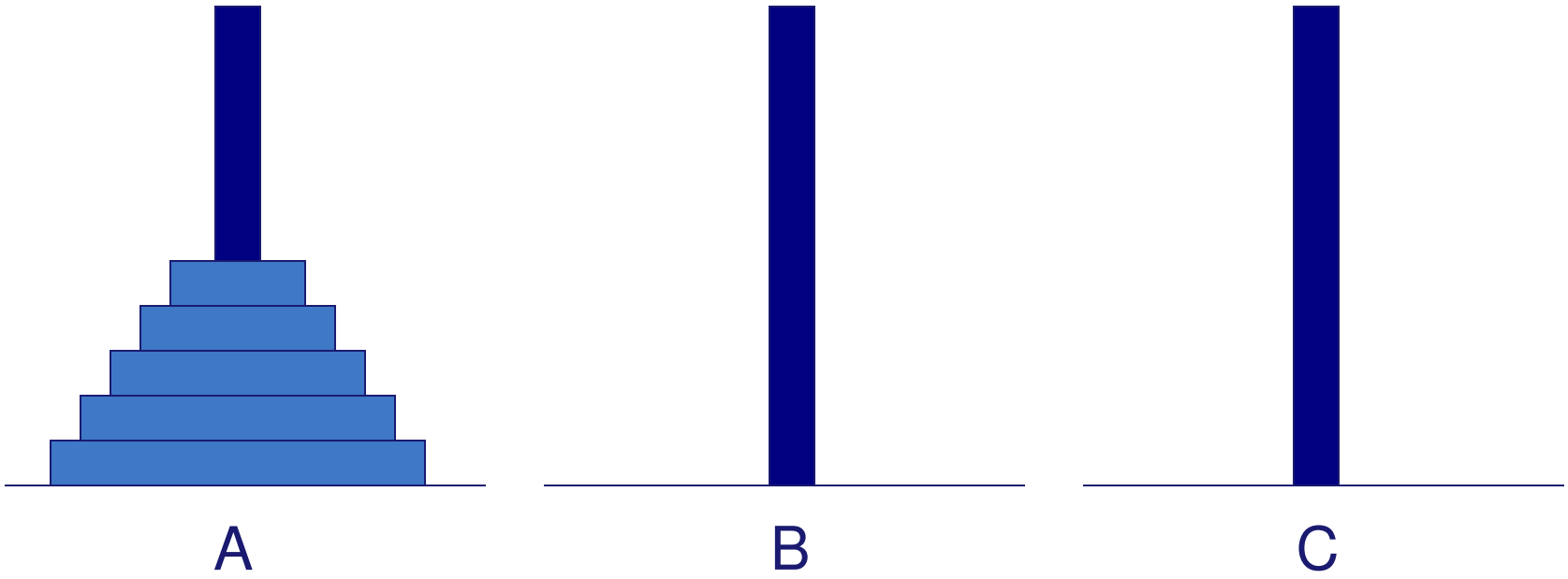
- Bài toán tháp Hà nội : n đĩa
  - Mỗi lần chỉ di chuyển một đĩa
  - Đĩa lớn luôn nằm dưới đĩa nhỏ
  - Được phép sử dụng một cọc trung gian
  - Ký hiệu
    - A: cọc nguồn
    - B: cọc trung gian
    - C: cọc đích



Bài toán tháp Hà Nội

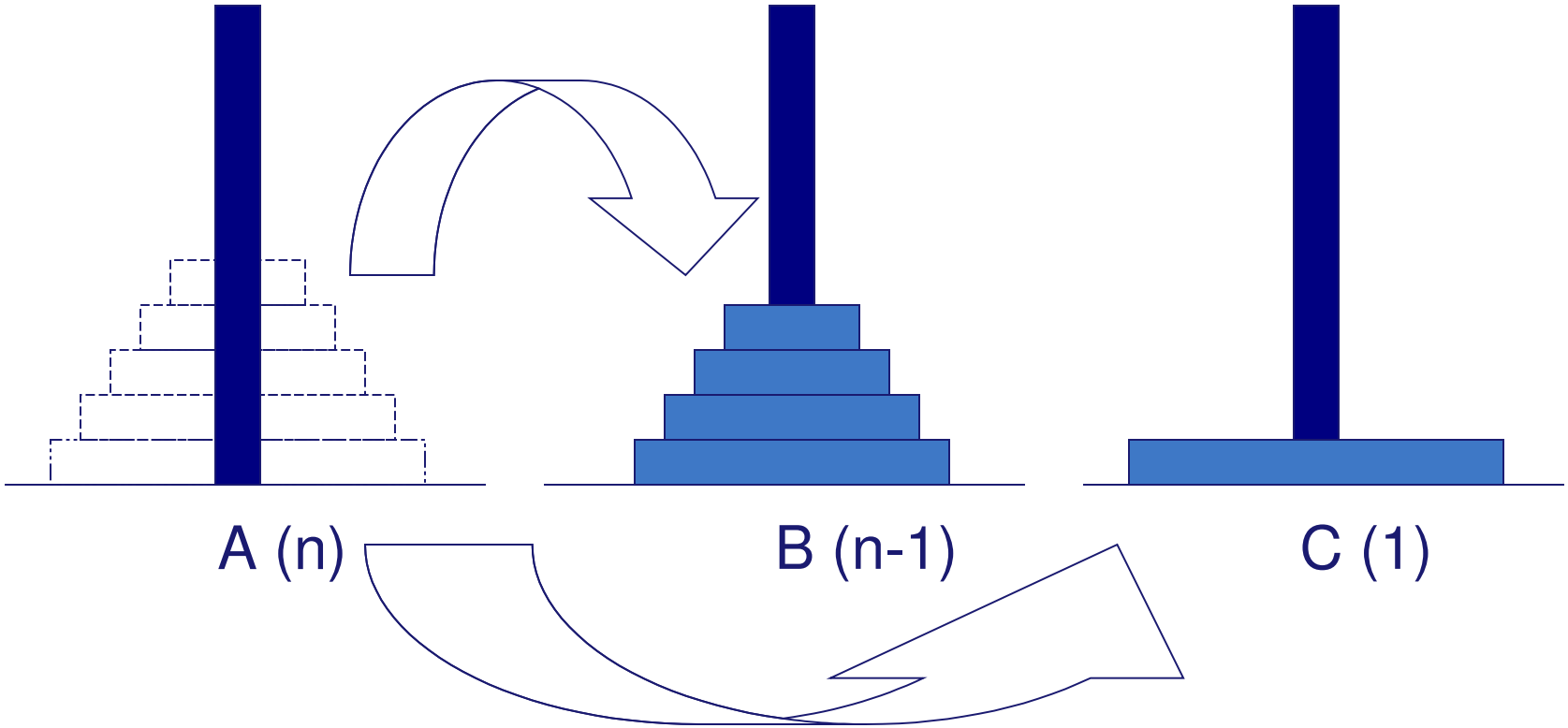
- Trường hợp n = 1
  - Chuyển từ A sang C
- Trường hợp n > 1
  - Chuyển (n-1) đĩa từ A sang B, C trung gian
  - Chuyển đĩa n từ A sang C
  - Chuyển (n-1) đĩa từ B sang C, A làm trung gian

Bài toán tháp Hà Nội



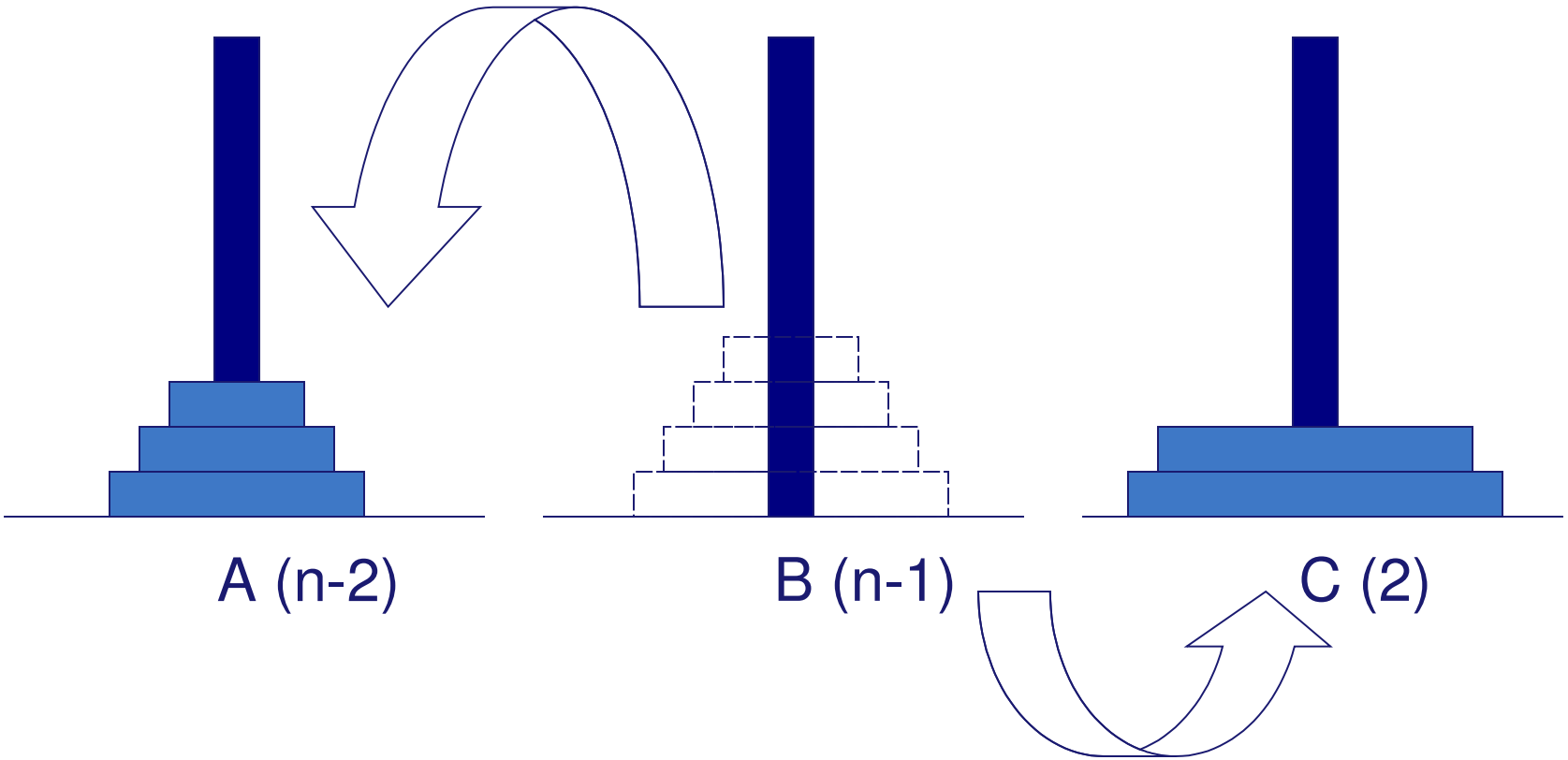
Bài toán tháp Hà Nội

- A C, B trung gian



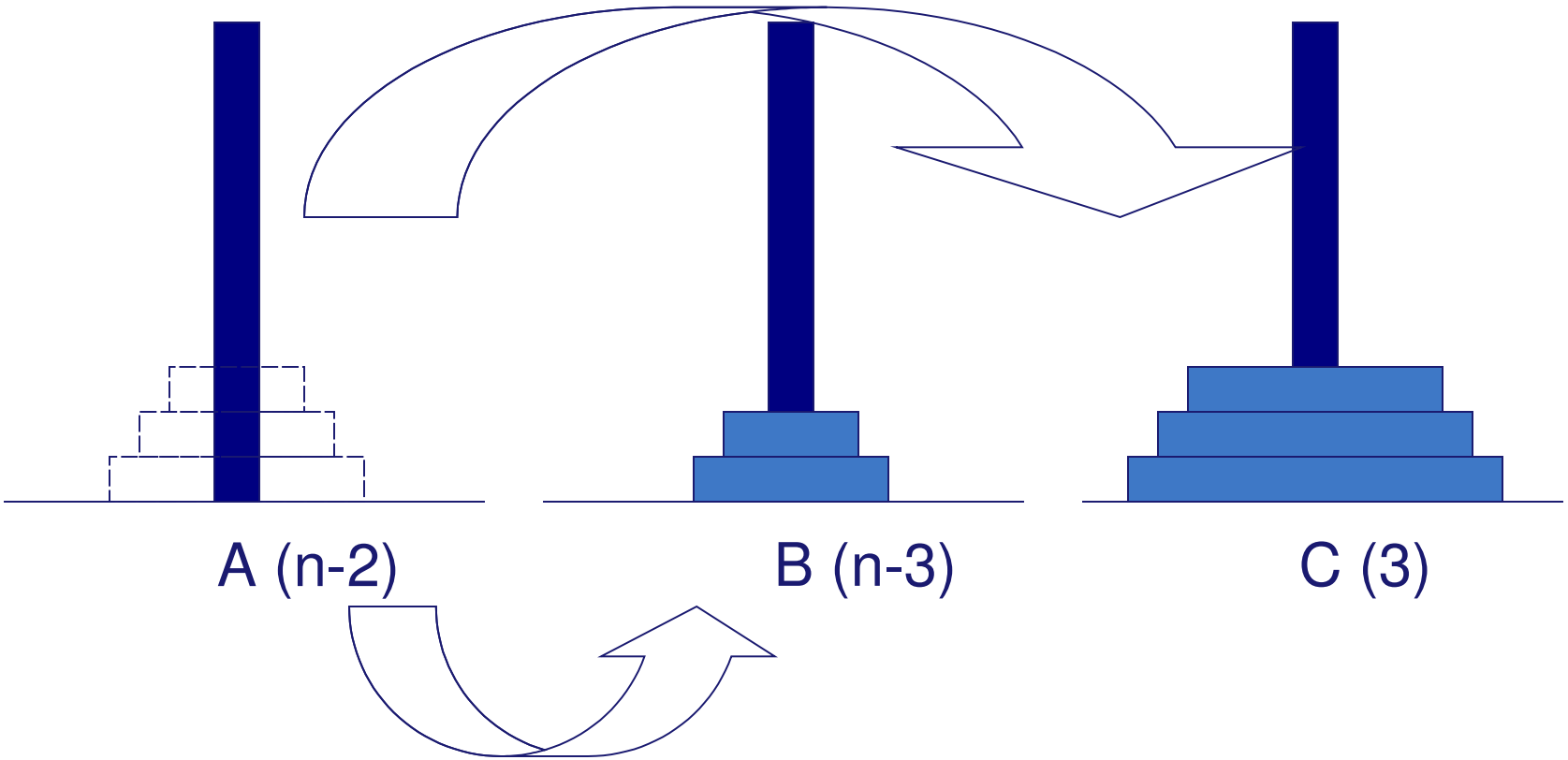
Bài toán tháp Hà Nội

- B C (A trung gian)



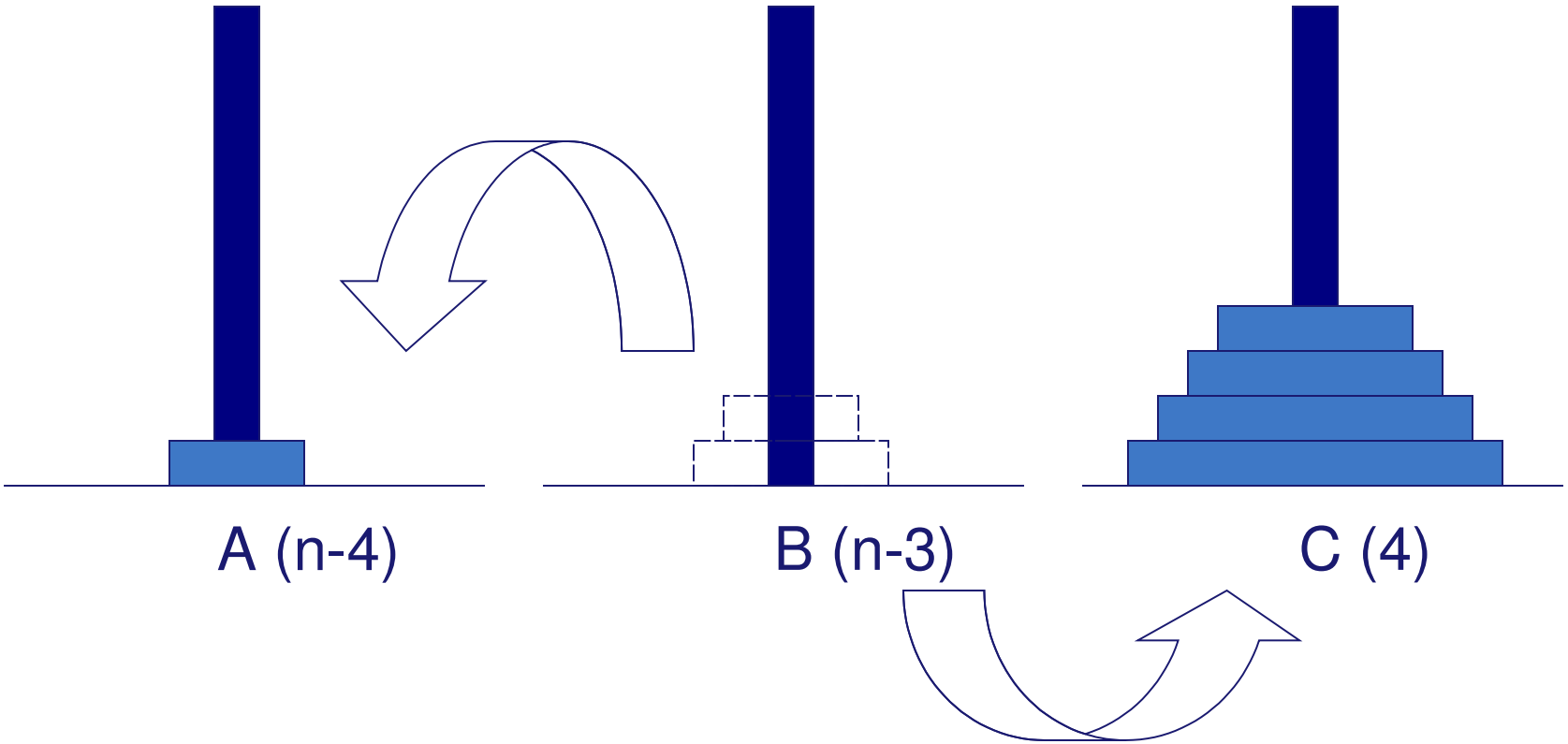
Bài toán tháp Hà Nội

- A C (B trung gian)



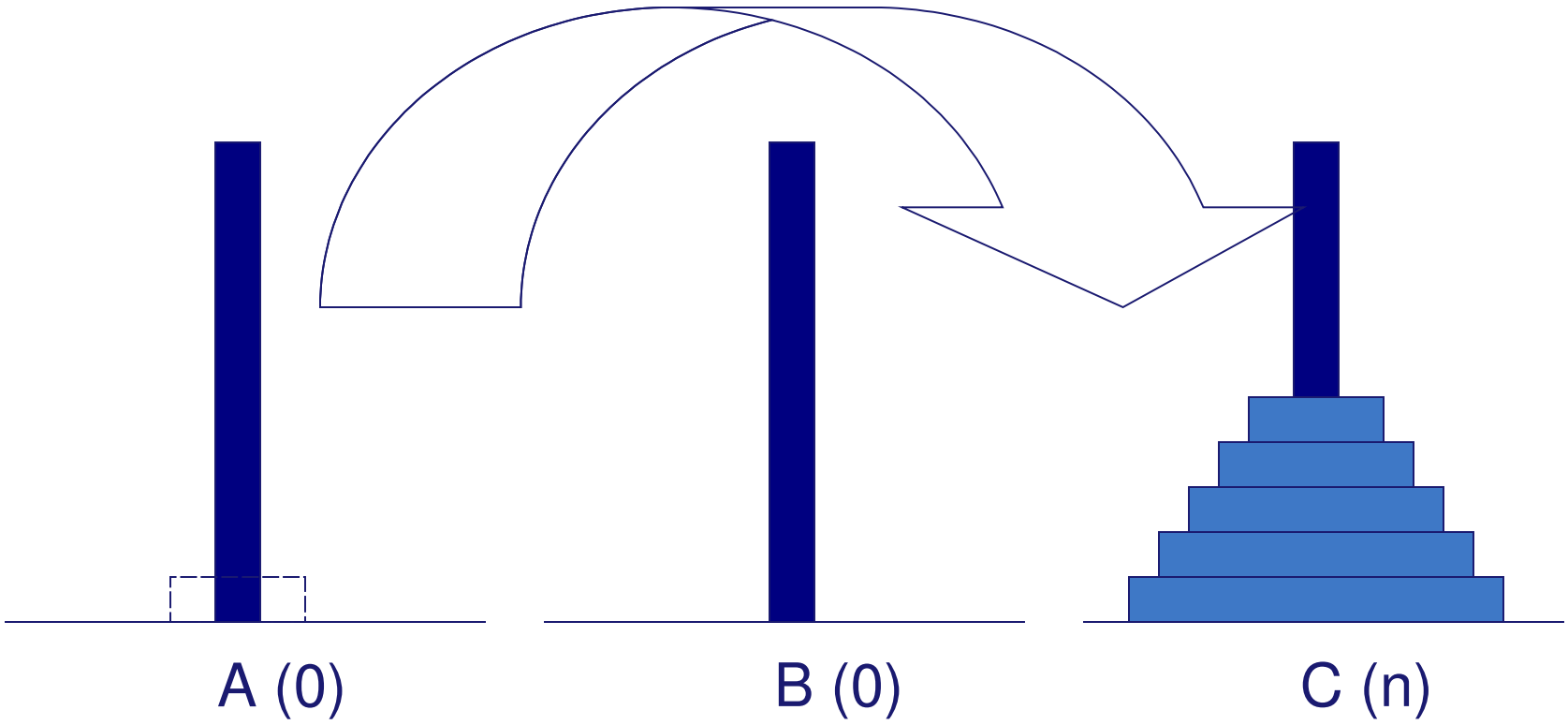
Bài toán tháp Hà Nội

- B C (A trung gian)



Bài toán tháp Hà Nội

- A C



Bài toán tháp Hà Nội

```
Void HANOI(int n, char A,B,C){
    if (n==1)
    cout << “chuyển đĩa từ” << A <<“sang”<< C
    else {
        HANOI(n-1,A,C,B);
        HANOI(1,A,B,C);
        HANOI(n-1,B,A,C);
    }
}
```

Bài toán chia thưởng

- Tìm số cách chia m phần thưởng cho n đối tượng học sinh giỏi có thứ tự 1, 2, ..,n. thỏa nguyên tắc
  - Học sinh A giỏi hơn học sinh B, thì số phần thưởng của A sẽ lớn hơn hoặc bằng B
  - Tất cả m phần thưởng đều chia hết cho học sinh
  - Hàm Part(int m,n) là số cách chia
    - Nếu m = 0, có 1 cách chia, tất cả học sinh đều có 0 phần thưởng
    - Nếu n = 0, không có cách chia nào cả

Bài toán chia thưởng

- Khi m < n, thì có n-m học sinh cuối không có phần thưởng, Part(m,n) = Part(m,m)
- Khi m>n, ta xét hai trường hợp
  - Khi học sinh cuối cùng không nhận được phần thưởng nào, dó đó Part(m,n) = Part(m, n-1)
  - Khi học sinh cuối cùng nhận được ít nhất 1 phần thưởng, do đó số cách chia là Part(m-n, n)
  - Tóm lại m > n, có Part(m,n) = Part(m, n-1) + Part(m-n, n)

Bài toán chia thưởng

```
int PART( int m , int n )
{
    if (m == 0 ) return 1 ;
    else
    if (n == 0 ) return 0 ;
    else
    if(m < n ) return ( PART(m , m )) ;
    else
    return ( PART(m , n -1 ) + PART(m -n , n ) ) ;
}
```

Phương pháp quay lui (back tracking)

- Đặc trưng : là các bước hướng tới lời giải cuối cùng của bài toán hoàn toàn được làm thử.
- Tại mỗi bước
  - Nếu có một lựa chọn được chấp nhận thì ghi nhận lại lựa chọn này và tiến hành các bước thử tiếp theo.
  - Ngược lại, không có lựa chọn nào thích hợp thì làm lại bước trước, xóa bỏ sự ghi nhận và quay về chu trình thử các lựa chọn còn lại

Phương pháp quay lui (back tracking)

- Mô hình bài toán:
  - Tìm $X=(x_1, x_2, ..,x_n)$ thỏa B.
  - Để chỉ ra lời giải X, ta phải dựng dần các thành phần lời giải x_i

Phương pháp quay lui (back tracking)

- Phương pháp: Ta xây dựng $x_1, x_2, .., x_n$ theo cách sau
  - Đầu tiên, Tập T_1 các ứng cử viên có thể là thành phần đầu tiên của vectơ nghiệm chính là x_1
  - Chọn x_1 T_1,
  - Giả sử, đã xác định được k-1 phần tử đầu tiên của dãy đó là $x_1, x_2, .., x_{k-1}$. Cần xác định phần tử kế tiếp x_k
  - Xác định T_k là tập tất cả các ứng viên mà x_k có thể nhận được, có hai khả năng

Phương pháp quay lui (back tracking)

- Nếu T_k không rỗng, ta chọn x_i T_k và ta có được nghiệm bộ $(x_1,x_2,\ldots,x_{k-1},x_k)$, đồng thời loại x_k đã chọn khỏi T_k. Sau đó ta lại tiếp tục mở rộng bộ $(x_1,x_2,\ldots,x_k)$ bằng cách áp dụng đệ quy thủ tục mở rộng nghiệm.
- Nếu T_k rỗng, tức là không thể mở rộng bộ $(x_1,x_2,\ldots,x_{k-2},x_{k-1})$, thì ta quay lại chọn phần tử mới x'_{k-1} trong T_{k-1} làm thành phần thứ k-1 của vectơ nghiệm

- **Chú ý**: phải có thêm thao tác "Trả lại trạng thái cũ cho bài toán" để hỗ trợ cho bước quay lui

Phương pháp quay lui (back tracking)

```
Thu(k) {
if (k==n) <ghi nhận bộ giá trị mới>
else
for ( j = 1 → n_k) // Mỗi j thuộc tập T_k
    if ( j chấp nhận được){
            <Xác định x_k theo khả năng j>
            <Ghi nhận trạng thái mới>
             Thu(k+1);
            <Trả lại trạng thái cũ cho bài toán>;
     }
```

Phương pháp quay lui (back tracking)

- Quan tâm:
  - Làm thế nào để xác định được tập T_k, tức là tập tất cả các khả năng mà phàn tử thứ k của dãy $x_1, x_2, .., x_n$ có thể nhận
  - Khi đã có tập $T_{k,}$ để xác định x_k, thấy rằng x_k phụ thuộc vào chỉ số j mà còn phụ thuộc vào $x_1, x_2, .., x_{k-1}$

Bài toán: Liệt kê tất cả các hoán vị của n số tự nhiên đầu tiên

- Đặt N= {1, 2, ..,n}. Hoán vị của n số tự nhiên đầu tiên là một bộ x[0], x[1],..,x[n-1]. Trong đó x[i] x[j], i,j và x[i] {0..n-1}.
- T_1 = N, giả sử đã xác định được x[0], x[1], .., x[k-1]. khi đó, T_k = {1..n}- {x[0], x[1], .., x[k-1]}.
- Ghi nhớ tập T_k , k = 0..n-1, ta cần sử dụng một mảng b[0..n-1] là các giá trị 0, 1 sao cho b[i] = 1 khi và chỉ khi i thuộc T_k

Bài toán: Liệt kê tất cả các hoán vị của n số tự nhiên đầu tiên

```
Thu(int k){
   if (k== n) inkq();
   else
      for (int j=1; j<=n; j++)
            if (b[j])
            {
               x[k] = j;
               b[j] = 0;
               Thu(k+1);
               b[j] = 1;
            }
   }
```

```
int main()
{
   b[j] = 1; // j=1..n-1
   Thu (0);
   return 0;
}
```

Liệt kê dãy nhị phân độ dài n

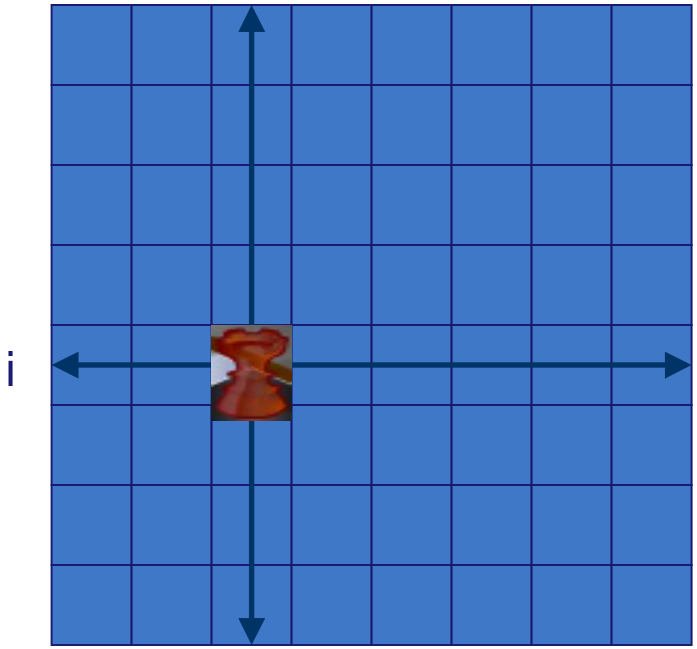
- Chuỗi nhị phân độ dài n có dạng x[0], x[1],..,x[n-1], Đặt B={0,1}
- T_1=B, Giả sử đã xác định được x[0], x[1], .., x[k-1]. Thấy rằng T_k = B

Liệt kê dãy nhị phân độ dài n

```
int x[20] ;
int n, d;
void Thu(int k)
   {
    if (k==n) inkq();
     else
                for (int j = 0; j <=1; j++)
                {
                x[k] = j;
                Thu(k+1);
                }
      }
```

Bài toán 8 quân xe

- Sắp xếp 8 quân xe trên bàn cờ 8x8 sao cho chúng không ‘ăn’ lẫn nhau (mỗi hàng, mỗi cột, có đúng một quân)



Bài toán 8 quân xe

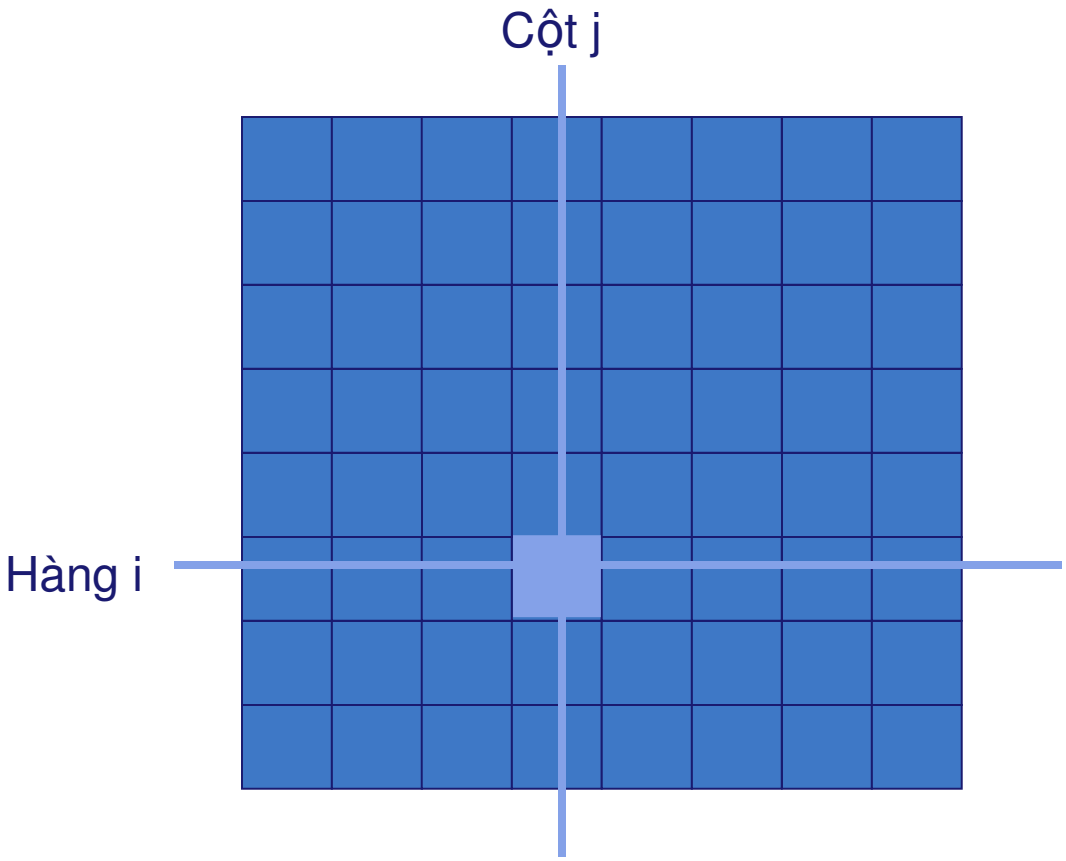
- Đặt quân xe thứ i vào cột thứ j sao cho nó không bị ‘ăn’ bởi i-1 quân xe hiện có trên bàn cờ
- Mỗi hàng chỉ có 1 quân xe, Nên việc chọn vị trí quân xe thứ i, chỉ nằm trên hàng i

	1	2	3	4	5	6	7	8
1								
2								
3								
4								
5								
6								
7								
8								

Bài toán 8 quân xe

- Qui ước x[i]: chỉ quân xe thứ i năm ở hàng i
- X[i] = j, quân xe thứ i đặt ở cột j
- Để quân xe i (hàng i) chấp nhận cột j, thì cột j phải tự do.

Bài toán 8 quân xe



Bài toán 8 quân xe

- Do đó ta sẽ chọn các mảng Boole 1 chiều để biểu diễn các trạng thái này
  - a[j] = 1 : Có nghĩa là không có quân xe nào ở cột j.
  - 1<= i, j <=8

Bài toán 8 quân xe

int x[8], a[8],

- Với các dữ liệu đã cho, thì lệnh đặt quân xe sẽ thể hiện bởi :
  - x[i] = j: đặt quân xe thứ i trên cột j.
  - a[j] = 0: Khi đặt xe tại cột j

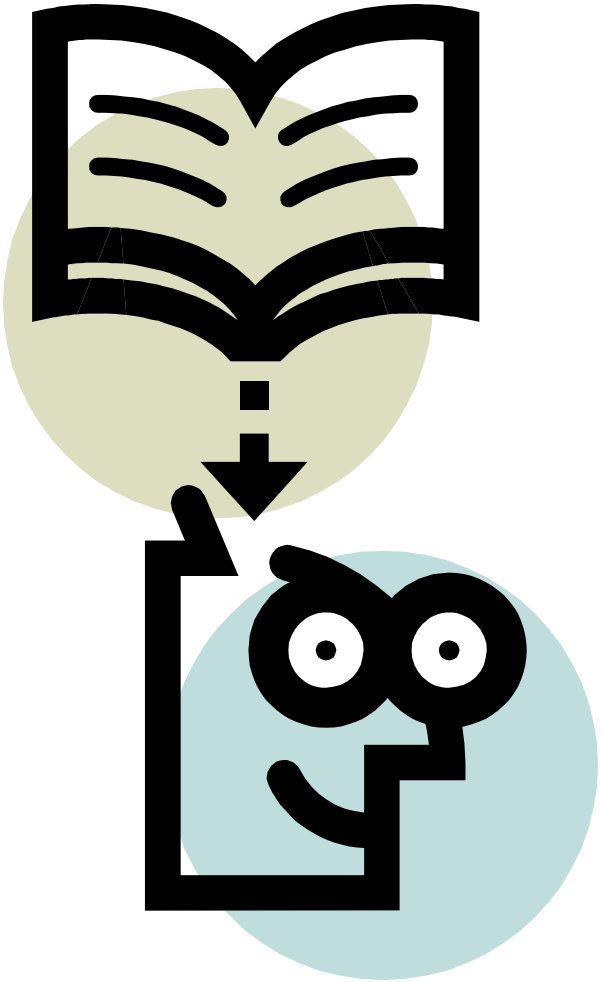
Bài toán 8 quân xe

- Lệnh dời quân xe là
  - a[j] = 1 ; // cột j tự do
- Còn điều kiện an toàn là ô có tọa độ (i,j) nằm ở cột chưa bị chiếm:
  - (a[j] == 1)

Bài toán 8 quân xe

```
Thu (i){
    If (i >8)
      Xuat (X)
    else
   for (j = 1; j <= 8; j++)
      if (a[j]) {
            x[i] = j;
             a[j] = 0;
           Thu (i+1);
           a[ j ] = 1
      }
   }
```

Đệ quy và quy nạp toán học

- Dùng đệ quy để giải các bài toán truy hồi
- Dùng quy nạp toán học để chứng minh tính đúng đắn, xác định độ phức tạp của giải thuật đệ quy

Q&A