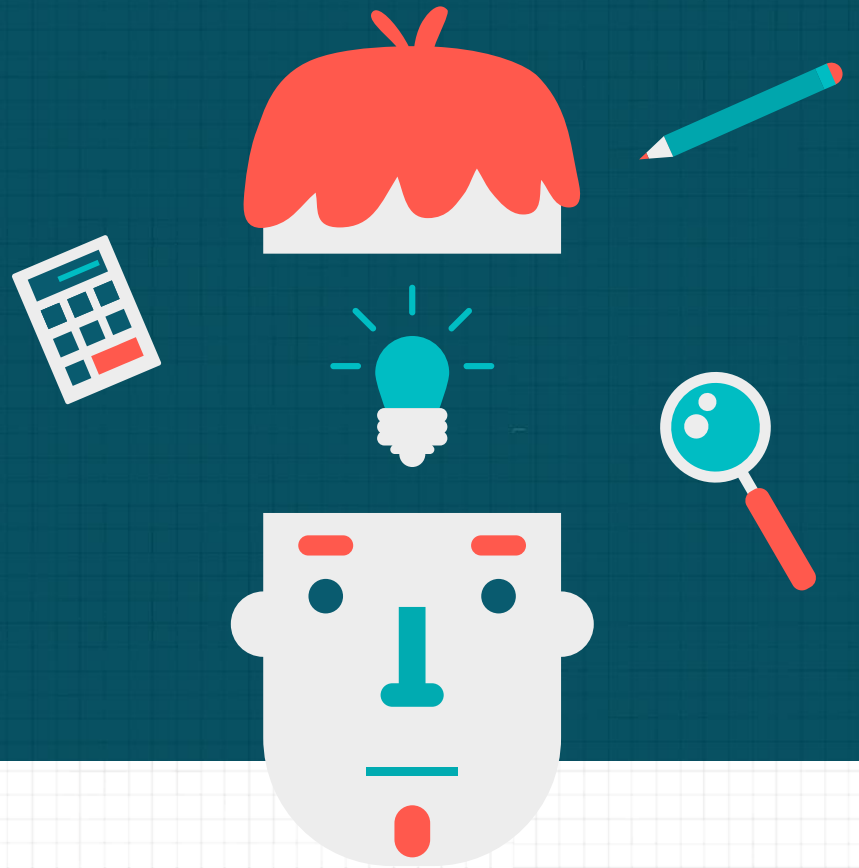
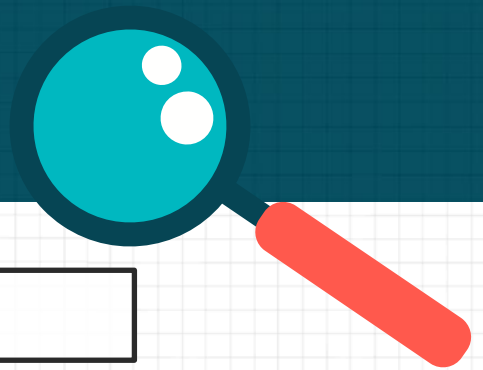
Bài 4
Lập trình đệ quy

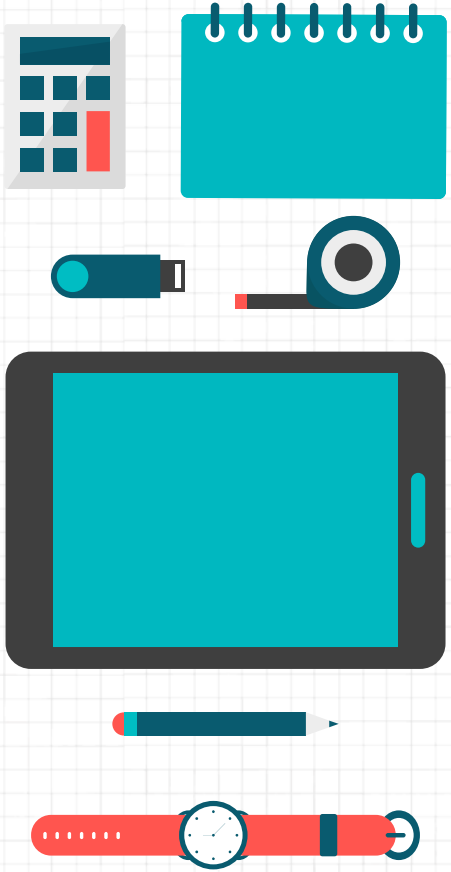


Ths. Phạm Minh Hoàng

Nội dung



- Các khái niệm về đệ quy
- Phân loại đệ quy
- Các kỹ thuật đệ quy
- Các bài toán đệ quy
- Các vấn đề khác

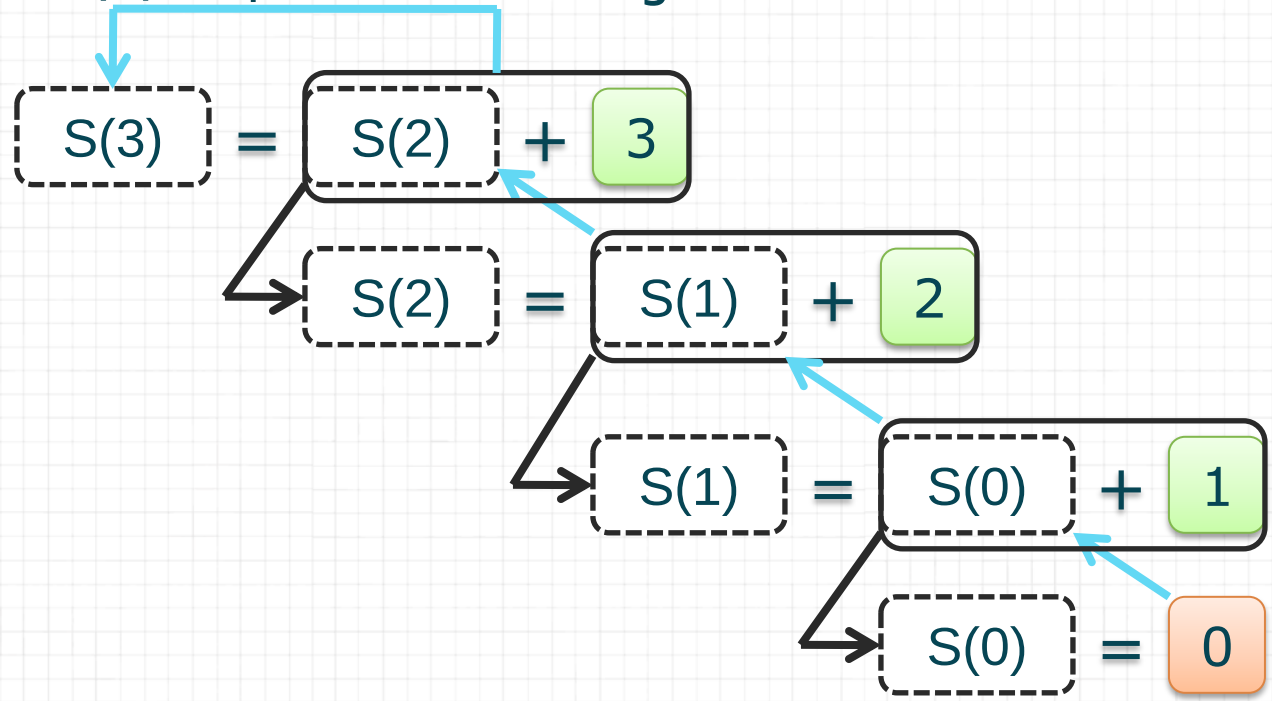


01

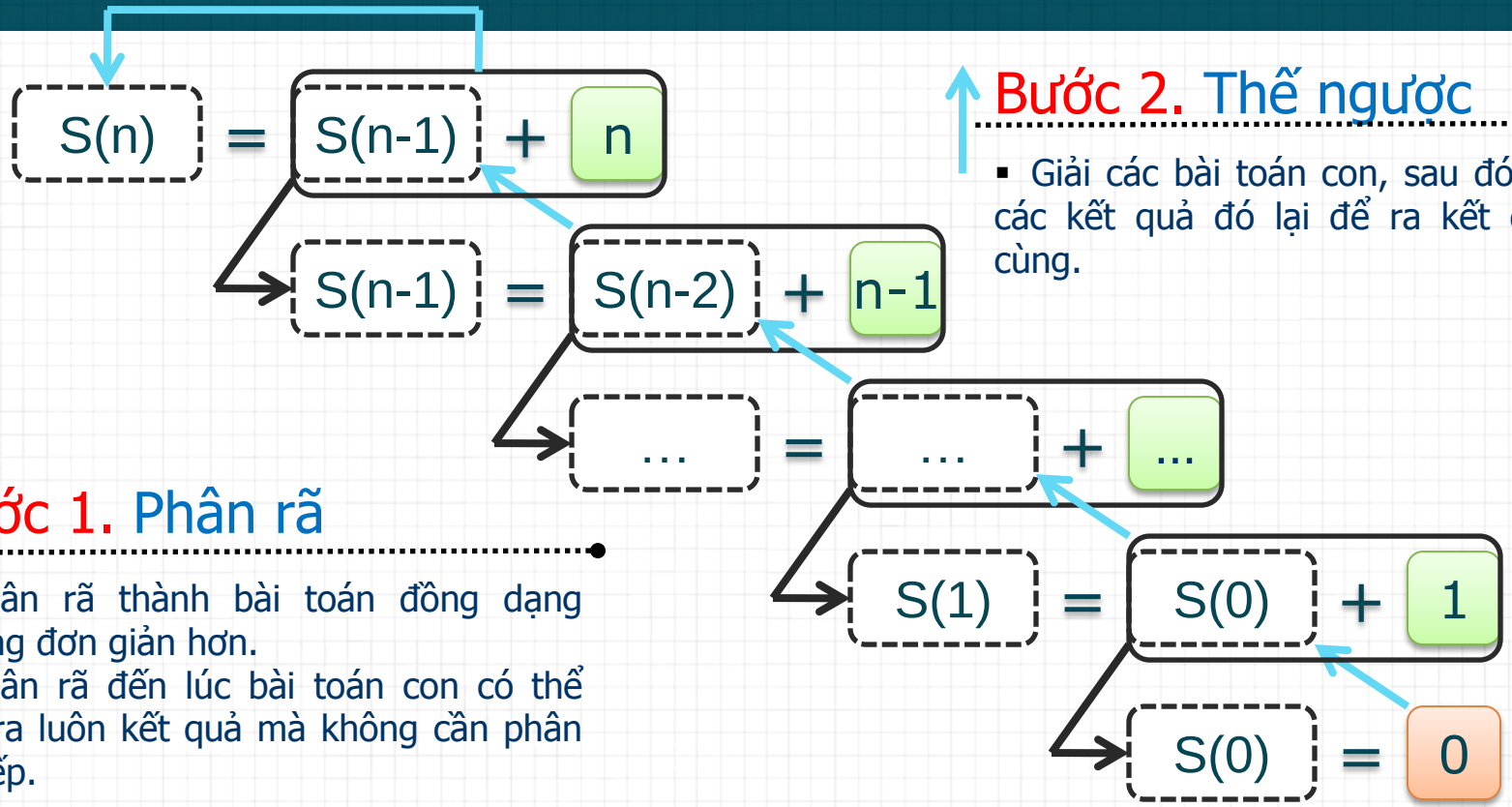
Các khái niệm về đệ quy

Đệ quy là gì?

- Cho dãy số S(0), S(1),...S(n) được tính theo công thức
  - S(0) = 0
  - S(n) = S(n-1) + n
- Tính S(3) = ?



Đệ quy là gì



Bước 1. Phân rã

- Phân rã thành bài toán đồng dạng nhưng đơn giản hơn.
- Phân rã đến lúc bài toán con có thể suy ra luôn kết quả mà không cần phân rã tiếp.

Bước 2. Thế ngược

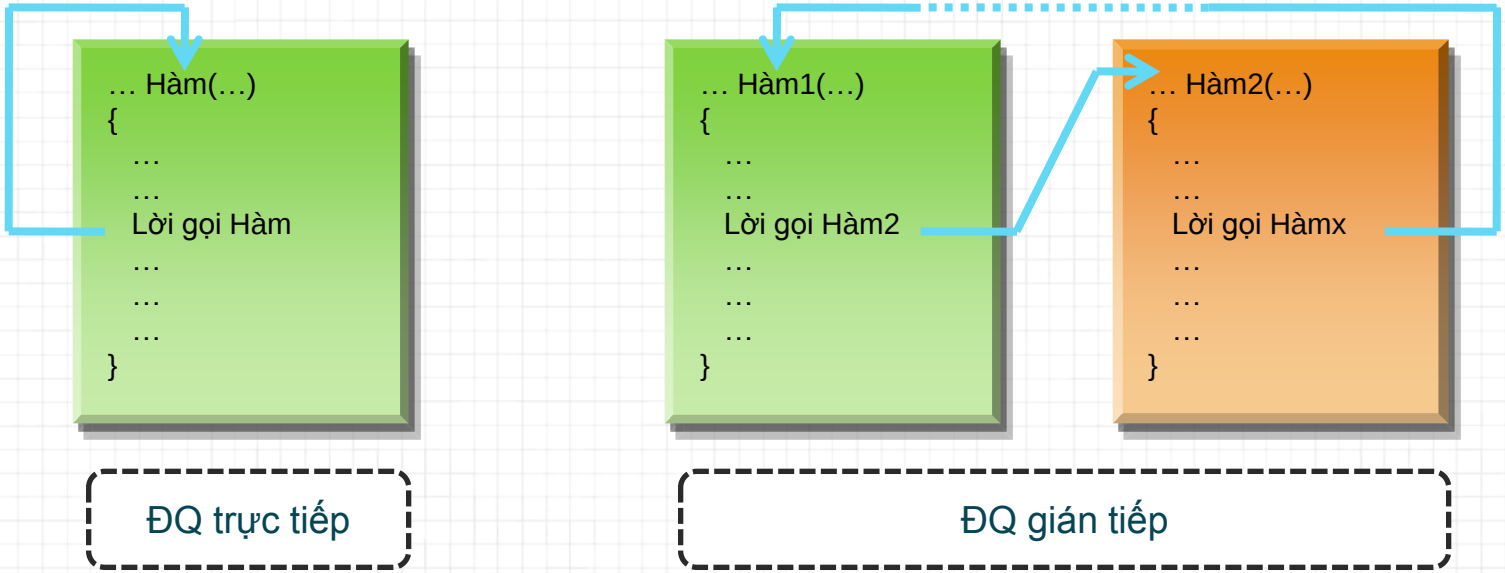
- Giải các bài toán con, sau đó kết hợp các kết quả đó lại để ra kết quả cuối cùng.

Đệ quy là gì

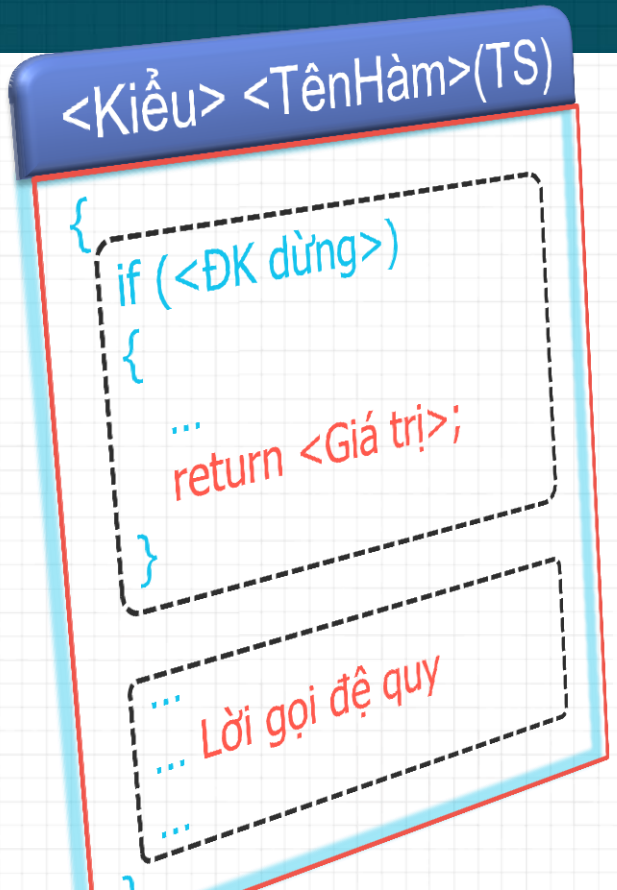
- Trong khoa học, bài toán đệ quy là bài toán được phân rã thành các bài toán nhỏ hơn, đơn giản hơn và có cùng dạng với bài toán ban đầu
- Hai điều kiện của đệ quy
  - Biểu diễn đệ quy: Phải xây dựng được cách biểu diễn bài toán thành bài toán nhỏ hơn có cùng dạng
  - Phải tồn tại điều kiện dừng
- Ví dụ: tính giai thừa của số nguyên n
  - Biểu diễn đệ quy: n! = n.(n-1)! với n > 0
  - Điều kiện dừng khi n = 0 thì 0! = 1

Đệ quy là gì

- Trong lập trình, đệ quy là quá trình một hàm bên trong thân hàm có lời gọi đến chính nó một cách trực tiếp hay gián tiếp



Cấu trúc hàm đệ quy



Phần điều kiện dừng (Base step)

- Là trường hợp đặc biệt dùng để kết thúc việc gọi lại chính hàm đó

Phần đệ quy (Recursion step)

- Sử dụng lời gọi lại chính bản thân của hàm

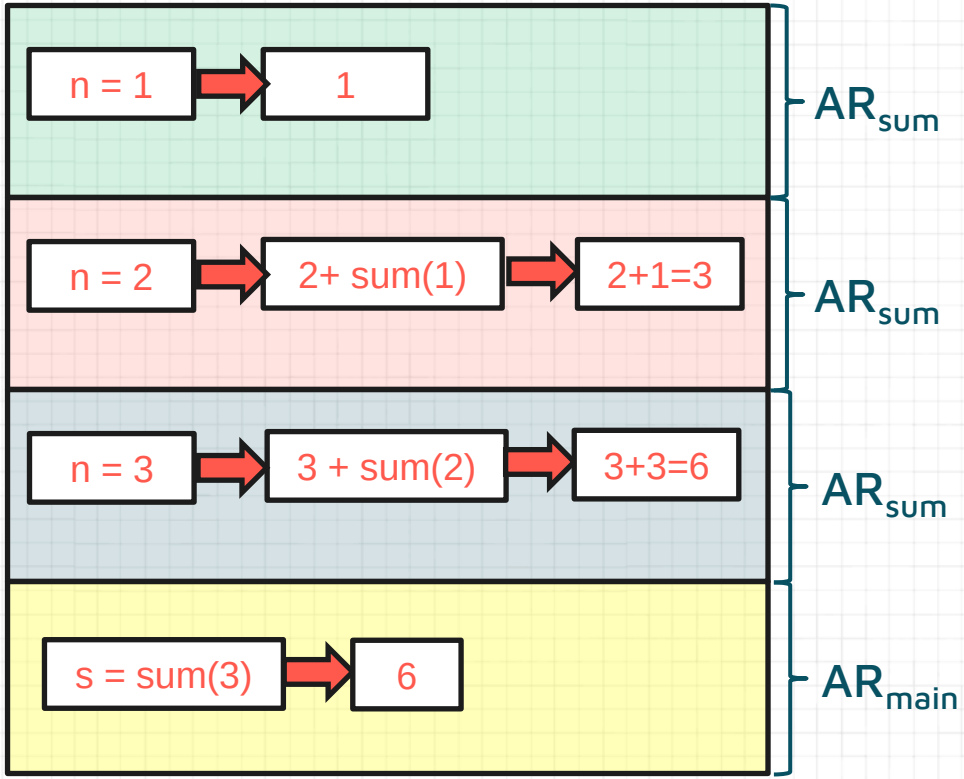
Cấu trúc hàm đệ quy

- Cho n nguyển dương. Tính tổng S = 1 + 2 + … + n
- Biểu diễn đệ quy: S(n) = S(n-1) + n
- Điều kiện dừng: khi n = 1 → S = 1

```
int sum(int n){
    if(n == 1)
        return 1;
    return n + sum(n-1);
}
```

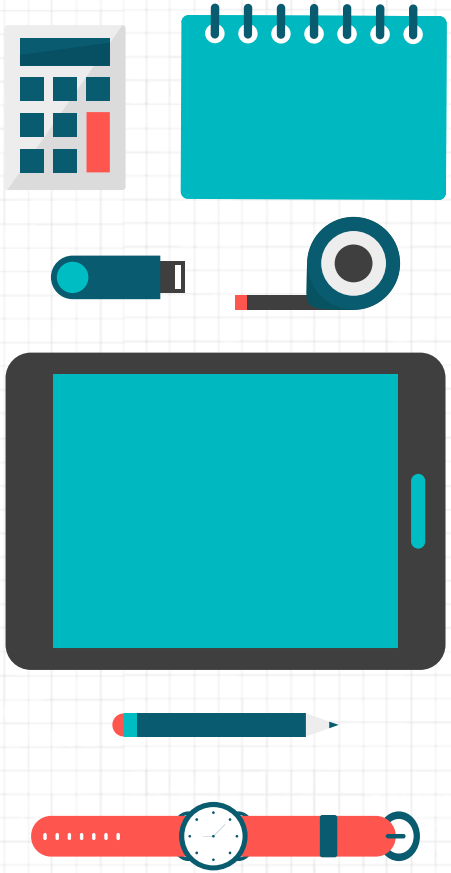
Cơ chế gọi hàm đệ quy

```
int sum(int n){
    if(n == 1)
        return 1;
    return n + sum(n-1);
}

void main(){
    int s = sum(3);
}
```



Cơ chế gọi hàm đệ quy

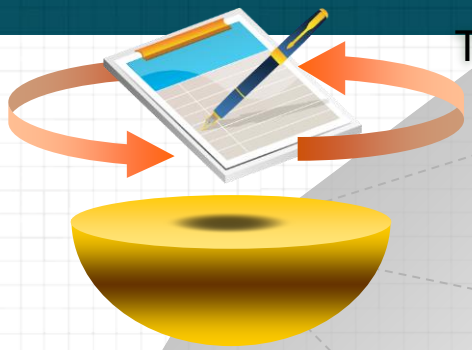
- Cơ chế gọi hàm dùng stack phù hợp cho đệ quy vì
  - Lưu thông tin còn dang dở mỗi khi gọi đệ quy
  - Thực hiện xong 1 lần gọi cần khôi phục trạng thái trước khi gọi
  - Lần gọi cuối cùng sẽ hoàn tất đầu tiên
- Một sỗ lỗi thường gặp khi làm đệ quy
  - Không biểu diễn đệ quy được
  - Không xác định được điều kiện dừng
  - Số lần gọi đệ quy quá lớn làm tràn stack



02

Phân loại đệ quy

Phân loại đệ quy



TUYẾN TÍNH 1

NHỊ PHÂN 2

HỖ TƯƠNG 3

PHI TUYẾN 4

Đệ quy tuyến tính

Đệ quy tuyến tính

```
<Kiểu> TênHàm(<TS>) {
  if (<ĐK đừng>) {
    …
    return <Giá Trị>;
  }
  … TênHàm(<TS>); …
}
```

- Đệ quy tuyến tính trong thân hàm chỉ có duy nhất 1 lời gọi đệ quy

```
int sum(int n){
    if(n == 1)
        return 1;
    return n + sum(n-1);
}
```

Đệ quy nhị phân

Đệ quy nhị phân

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

- Đệ quy nhị phân trong thân hàm chứa 2 lời gọi đệ quy

```
int fibo(int n){
    if(n == 1 || n == 0)
        return 1;
    return fibo(n-1) + fibo(n-2);
}
```

Đệ quy hỗ tương

Đệ quy hỗ tương

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

- Trong hàm thứ 1 gọi đến hàm thứ 2, và trong hàm thứ 2 lại gọi đến hàm thứ 1
- Ví dụ Tính số hạng thứ n của dãy:

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

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

  $y(n) = 3*x(n-1) + 2*y(n-1)$
- Điều kiện dừng
  - $x(0) = 1, y(0) = 0$

Đệ quy hỗ tương

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

Đệ quy phi tuyến

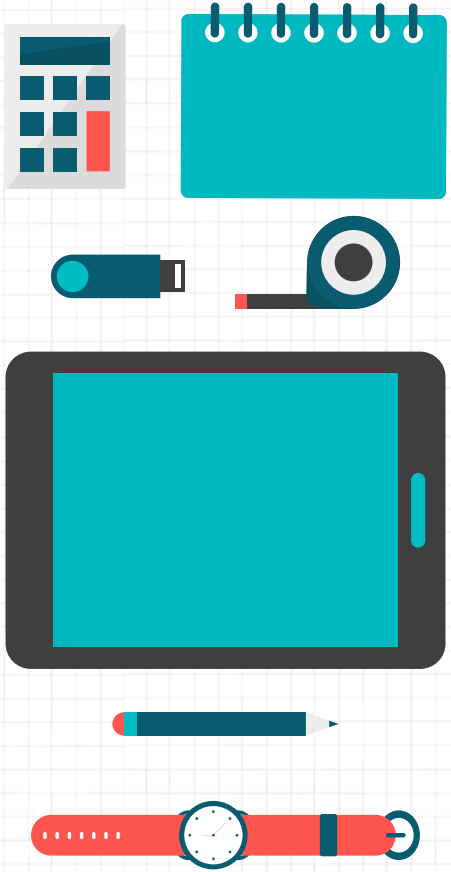
Đệ quy phi tuyến

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

- Thâm hàm chứa lời gọi đệ quy được đặt trong vòng lặp
- Ví dụ Tính phần tử n của dãy:
  - $x(0) = 1$
  - $x(n) = n^2x(0) + (n-1)^2x(1) + \ldots + 2^2x(n-2) + 1^2x(n-1)$
- Điều kiện dừng
  - $x(0) = 1$

Đệ quy phi tuyến

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



03

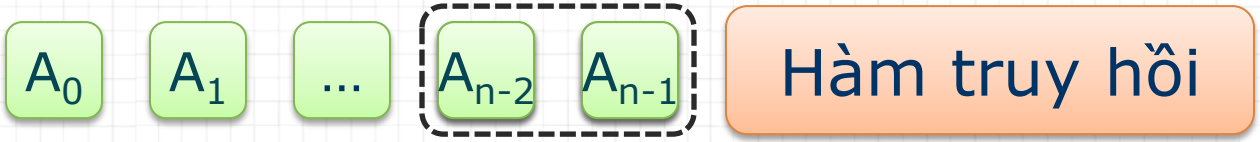
Các kỹ thuật đệ quy

Kỹ thuật đệ quy

- Một số kỹ thuật đệ quy thông dụng
  - Kỹ thuật truy hồi
  - Kỹ thuật chia để trị
  - Kỹ thuật quay lui

Kỹ thuật truy hồi

- Biểu diễn đệ quy dưới dạng các hệ thức truy hồi
- Hệ thức truy hồi là công thức biểu diễn A_n thông qua 1 hay nhiều phần tử trước đó A_{n-1}, A_{n-2},...



Kỹ thuật truy hồi

- Ví dụ 1:
  - Vi trùng cứ sau 1h lại nhân đôi. Hỏi sau 5h sẽ có bao nhiêu con vi trùng nếu ban đầu có 2 con?
- Lời giải
  - Gọi V_h là số vi trùng tại thời điểm h
  - Biểu diễn đệ quy bằng công thức truy hồi: $V_h = 2V_{h-1}$
  - Điều kiện dừng: khi h = 0 → $V_0 = 2$

Kỹ thuật truy hồi

```
int tinhViTrung(int h)
{
    if (h == 0)
        return 2;

    return 2*tinhViTrung(h-1);
}
```

Kỹ thuật truy hồi

- Ví dụ 2:
  - Gửi ngân hàng 1000USD, lãi suất 12%/năm. Sau 30 năm, số tiền có được là bao nhiêu?
- Lời giải
  - Gọi T_n là số tiền có được sau n năm
  - Biểu diễn đệ quy bằng công thức truy hồi:
    - $T_n = T_{n-1} + 0.12T_{n-1} = 1.12T_{n-1}$.
  - Điều kiện dừng: khi n = 0 → $T_0 = 1000$

Kỹ thuật truy hồi

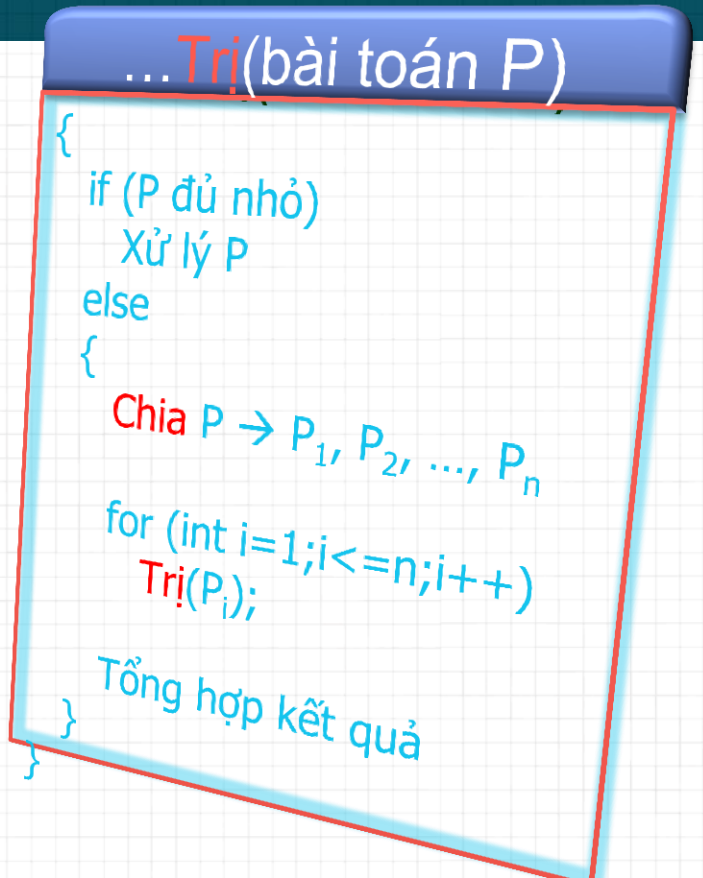
```
double tinhTienGui(int n, double M0, double q)
{
    if (h == 0)
        return M0;

    return (1+q)*tinhTienGui(n-1, M0, q);
}
```

Kỹ thuật truy hồi

- Bài tập:
  - Tính giai thừa của n
  - Tính tổng bình phương từ 1 đến n
  - Tìm phần tử nhỏ nhất trong mảng
  - Kiểm tra mảng có thứ tự tăng dần hay không
  - Đếm số lượng mảng con tăng liên tục trong mảng số nguyên
  - Sắp xếp mảng tăng

Kỹ thuật chia để trị

- Các bước chia để trị
  - Nếu bài toán đủ nhỏ: giải quyết trực tiếp
  - Ngược lại
    - Chia bài toán thành nhiều bài toán con
    - Giải quyết từng bài toán con bằng gọi đệ quy
    - Tổng hợp kết quả từng bài toán con ra lời giải cuối cùng



Kỹ thuật chia để trị

- Ví dụ 1: Tìm phần tử lớn nhất trong mảng số nguyên
  - Trường hợp bài toán đủ nhỏ: khi mảng có 1 phần tử
    → max = phần tử duy nhất của mảng
  - Chia đôi mảng thành 2 nửa trái và phải
    - Gọi đệ quy tìm max mảng bên trái (max_left)
    - Gọi đệ quy tìm max mảng bên phải (max_right)
  - Tổng hợp kết quả cuối cùng
    - max_final = max(max_left, max_right)

Kỹ thuật chia để trị

```
int getMax(int a[], int l, int r)
{
    if (l >= r)
        return a[l];
    int mid = (l + r)/2;
    int lmax = getMax(a, l, mid);
    int rmax = getMax(a, mid + 1, r);
    return max(lmax, rmax);
}
```

Kỹ thuật chia để trị

- Ví dụ 2: Tìm phần tử nhỏ thứ k trong mảng số nguyên
  - Trường hợp bài toán đủ nhỏ: khi mảng có 1 phần tử → phần tử cần tìm = phần tử duy nhất của mảng
  - Chia mảng theo pivot. Chọn phần tử tại vị trí p bất kỳ làm pivot. Chia mảng thành các phần
    - Các phần tử nhỏ hơn pivot (mảng 1)
    - Các phần tử lớn hơn pivot (mảng 2)
  - Tổng hợp kết quả cuối cùng: Gọi m là số phần tử mảng 1
    - Nếu k < m: trả về kết quả đệ quy trên mảng 1
    - Nêu k > m: trả về kết quả đệ quy trên mảng 2
    - Nếu k = m: trả về pivot

Kỹ thuật chia để trị

```
int getKthMin(int a[], int l, int r, int k){
    if (l >= r) return arr[l];
    int p = (l + r) / 2;
    int m = partition(arr, l, r, p);
    if (k < m) {
        return getKthMin(arr, l, m - 1, k);
    }
    else if (k > m) {
        return getKthMin(arr, m + 1, r, k);
    }
    else {
        return arr[m];
    }
}
```

Kỹ thuật chia để trị

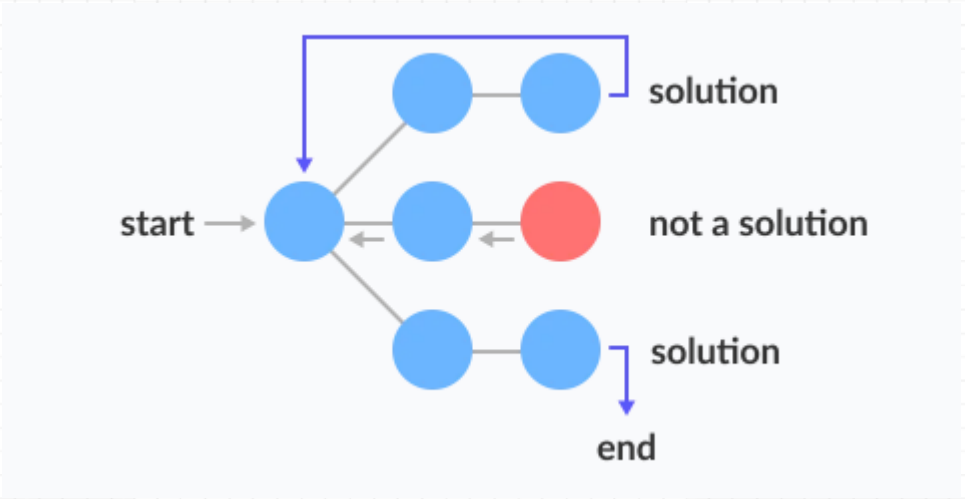
```
int partition(int a[], int l, int r, int p)
{
   swap(a[p], a[r]);
   int i = l, j = r - 1;
   while (i <= j) {
        while(a[i] < a[r])
           i++;
        while(a[j] > a[r])
           j--;
        if(i < j)
           swap(a[i], a[j]);
   }
   swap(a[i], a[r]);
   return i;
}
```

Kỹ thuật chia để trị

- Bài tập: cho mảng số nguyên có n phần tử
  - Tìm phần tử x
  - Đếm các số nguyên tố trong mảng
  - Tính tổng các phần tử là số lẻ
  - Đảo ngược mảng

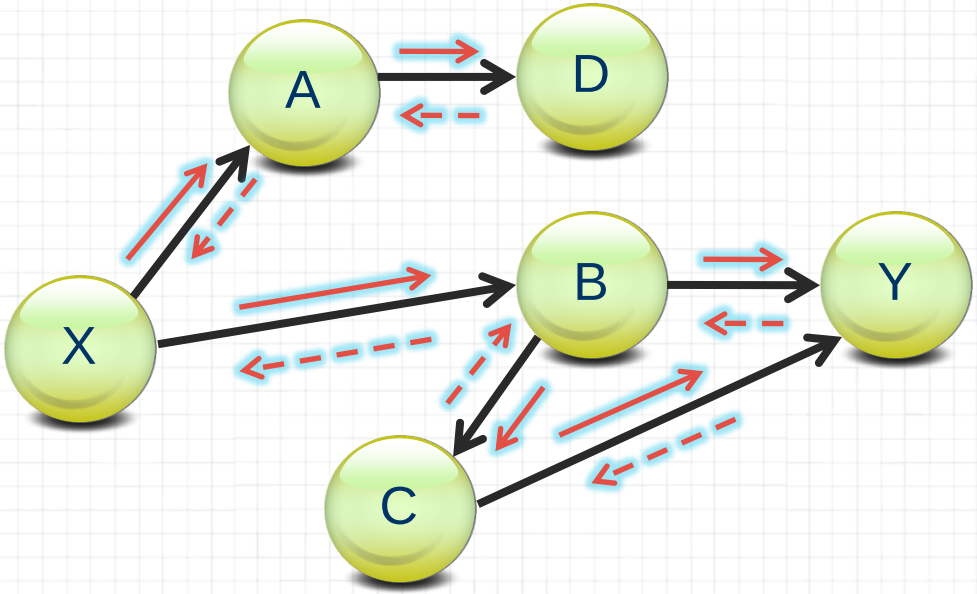
Kỹ thuật quay lui

- Xét bước hiện tại, nếu gặp lời giải thì ghi nhận lại
- Ngược lại
  - Chọn đi bước khác, thử xem bước đó có phải lời giải không
  - Nếu không còn đi được, thi quay lại bước trước đó



Kỹ thuật quay lui

- Ví dụ: tìm đường đi từ X đến Y



Kỹ thuật quay lui

- Bài toán: Cho mảng số nguyên có N phần tử và số nguyên T. Tìm tất cả bộ phần tử có tổng bằng T
- Ví dụ
  - Mảng A = {1, 2, 4, 5, 9}
  - T = 15
  - Kết quả: có 2 bộ phần tử (1, 5, 9) và (2, 4, 9) có tổng là 15

Kỹ thuật quay lui

```
void findSubArray(int a[], int n, int start, int T, int sum, bool track[]){
    if (sum == T)
        printSolution(a, track, start);
    for (int i = start; i < n; i++) {
        sum += a[i]; track[i] = true;
        findSubArray(a, n, i + 1, T, sum, track);
        sum -= a[i]; track[i] = false;
    }
}
```

```
void printSolution(int a[], int track[], int m){
    for(int i = 0; i < m; i++){
        if(track[i] == true)
            cout << a[i] << ", ";
    }
    cout << endl;
}
```

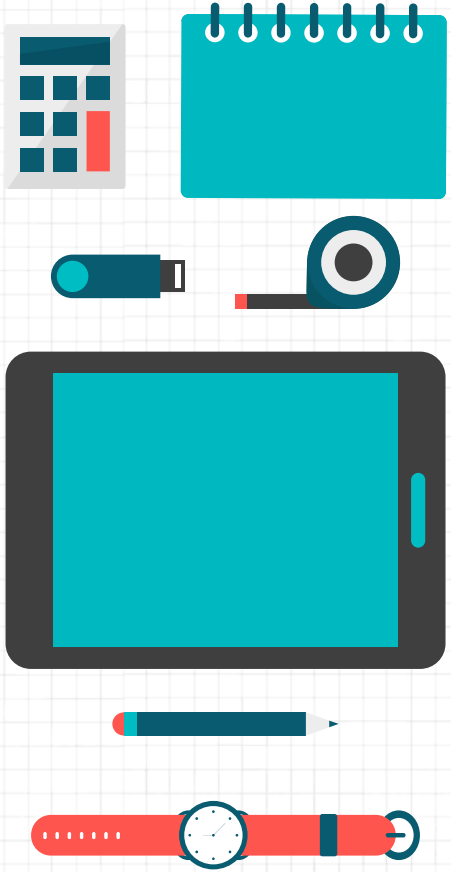
Kỹ thuật quay lui

- Bài toán 2: Cho N số nguyên từ 0 đến N - 1. Tìm tất cả hoán vị của N phần tử này
- Hoán vị của N phần tử là một cách sắp xếp vị trí của N phần tử
- Ví dụ: 3 phần tử {0, 1, 2} sẽ có 3! = 6 hoán vị gồm:
  - {0, 1, 2},
  - {0, 2, 1},
  - {1, 0, 2},
  - {1, 2, 0},
  - {2, 0, 1},
  - {2, 1, 0}

Kỹ thuật quay lui

```
int permute(int A[], int n, bool track[], int k){
    for(int i = 0; i < N; i++){
        if(!track[i]){
            A[i] = i;
            track[i]=true;
            if(k == N-1)
                printSolution(A, N);
            else
                permute(A, n, track, k+1);
            track[i]=false;
        }
    }
}
```

```
void printSolution(int h[], int m){
    for(int i = 0; i < m; i++){
        cout << h[i] << ", ";
    }
    cout << endl;
}
```



04

Các bài toán đệ quy kinh điển

Các bài toán đệ quy kinh điển

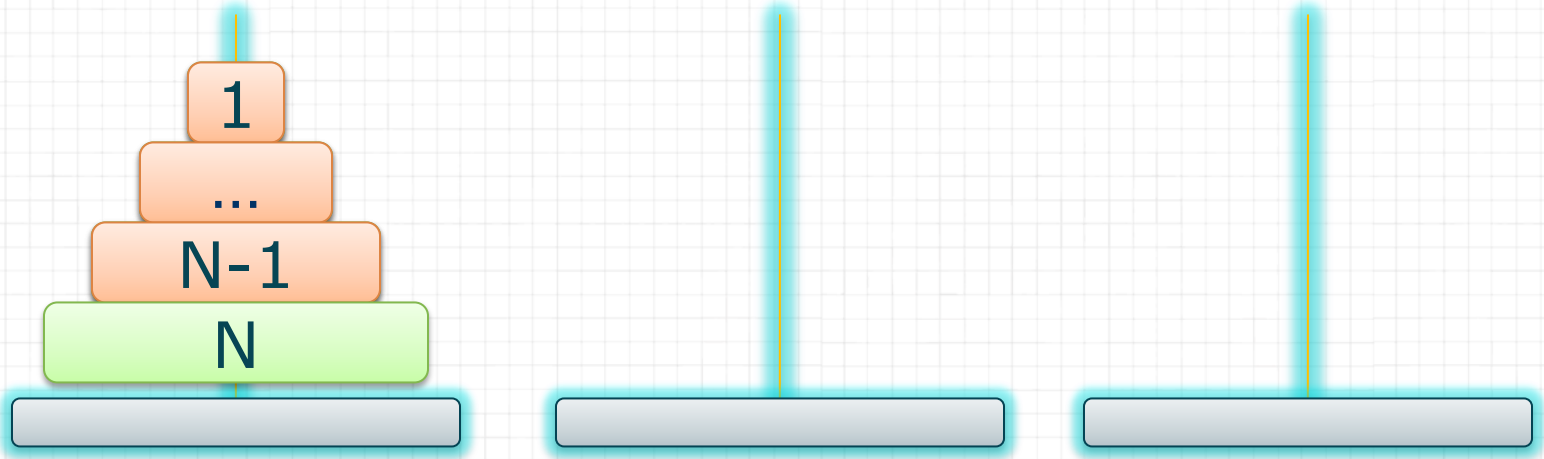
- Bài toán Tháp Hà Nội
- Bài toán Tám hậu
- Bài toán Mã đi tuần

Bài toán Tháp Hà Nội

- Mô tả
  - Có 3 cột A, B, C và cột A đang có N đĩa xếp chồng nhau theo quy tắc đĩa lớn nằm dưới đĩa nhỏ
  - Tìm cách chuyển N đĩa từ cột A sang cột C sao cho
    - Mỗi lần chỉ chuyển 1 đĩa
    - Đĩa lớn luôn nằm dưới đĩa nhỏ
    - Có thể dùng cột A, B, C làm các cột trung gian

Bài toán Tháp Hà Nội

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



Cột nguồn A Cột trung gian B Cột đích C

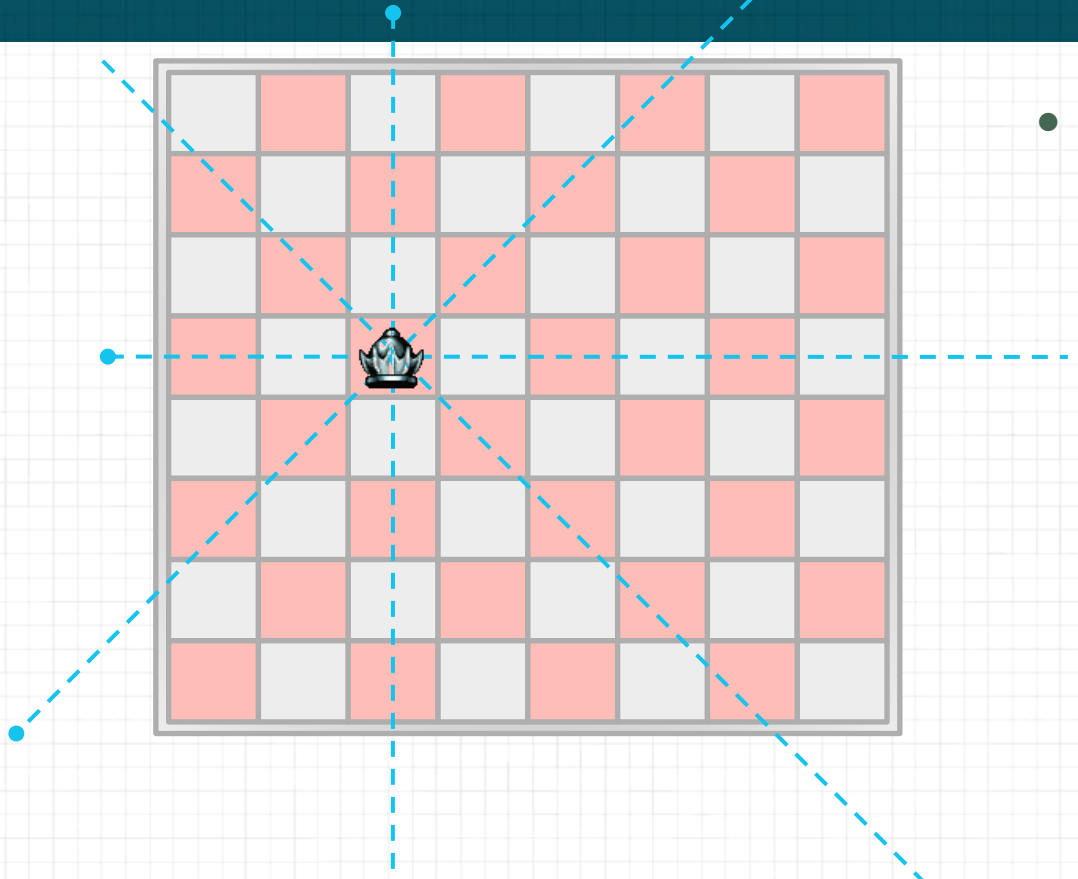
Bài toán Tháp Hà Nội

- Ý tưởng giải
  - Dùng kỹ thuật truy hồi
  - Chuyển N – 1 đĩa bên trên từ cột A sang cột B
  - Chuyển đĩa thứ N từ cột A sang cột C
  - Chuyển N – 1 đĩa từ cột B sang cột C
  - Điều kiện dừng: khi N = 1 → chỉ có 1 đĩa duy nhất ở cột A, thì chuyển đĩa đó sang cột C

Bài toán tám hậu

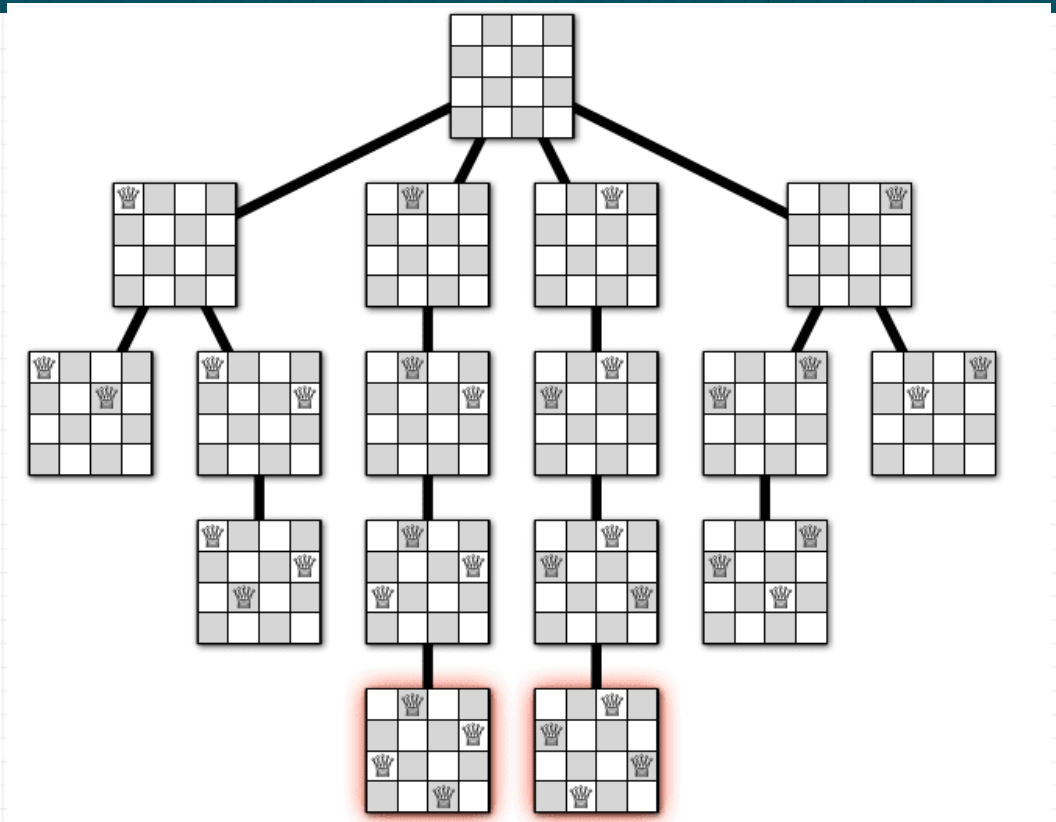
- Mô tả bài toán
  - Cho bàn cờ vua kích thước 8x8
  - Hãy đặt 8 quân hậu lên bàn cờ này sao cho không có quân hậu nào "ăn" nhau:
    - Không nằm trên cùng dòng, cùng cột
    - Không nằm trên cùng đường chéo xuôi, ngược.

Bài toán Tám hậu



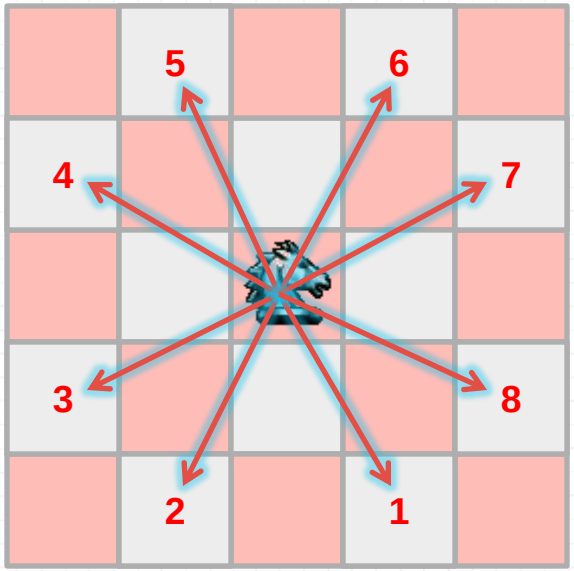
- Quân hậu tại vị trí (i, j) có thể khống chế các đường
  - Dòng: có chỉ số dòng bằng chỉ số dòng i
  - Cột: có chỉ số cột bằng chỉ số cột j
  - Hai đường chéo đi qua ô (i, j)

Bài toán tám hậu



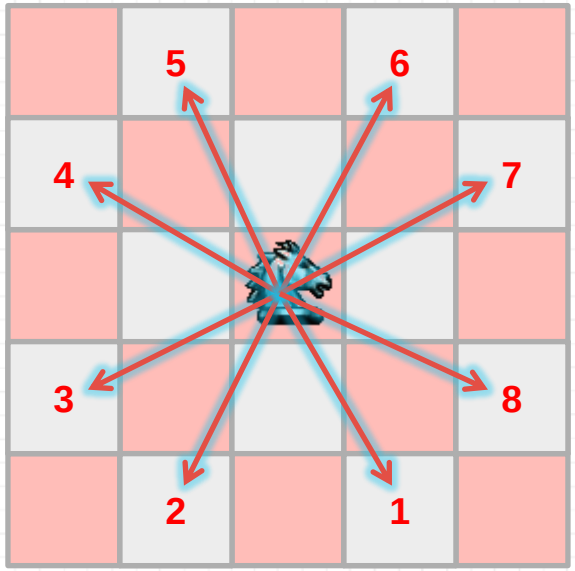
Bài toán Mã đi tuần

- 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:

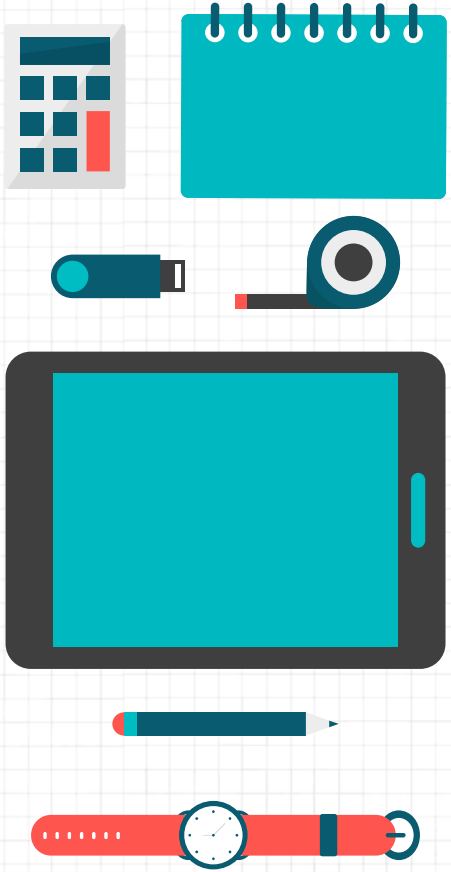


Bài toán Mã đi tuần

- Quân mã đặt tại vị trí (i, j) có thể đi đến ô nào



1: (i + 2, j + 1)
2: (i + 2, j – 1)
3: (i + 1, j – 2)
4: (i – 1, j – 2)
5: (i – 2, j – 1)
6: (i – 2, j + 1)
7: (i – 1, j + 2)
8: (i + 1, j + 2)



05

Các vấn đề khác

Đệ quy đuôi

- Đệ quy đuôi là khi lời gọi đệ quy là câu lệnh cuối cùng trong hàm được thực thi
- Ưu điểm đệ quy đuôi: Vì lời gọi đệ quy được thực hiện cuối cùng nên hàm hiện tại có thể kết thúc ngay khi hoàn thành câu lệnh đệ quy

Đệ quy đuôi

- Ví dụ

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

```
int gt(int n, int ret){
    if(n == 0)
        return ret;
    return gt(n-1, ret*n);
}
```

Khử đệ quy

- Ưu điểm đệ quy
  - Thuật toán rõ ràng
  - Chương trình ngắn gọn

- Khuyết điểm đệ quy
  - Tốn bộ nhớ, xử lý chậm
  - Dễ gây tràn stack
  - Khó debug
  - Nhiều bài toán không thể giải/khó giải/giải khó hiểu bằng đệ quy

Khử đệ quy

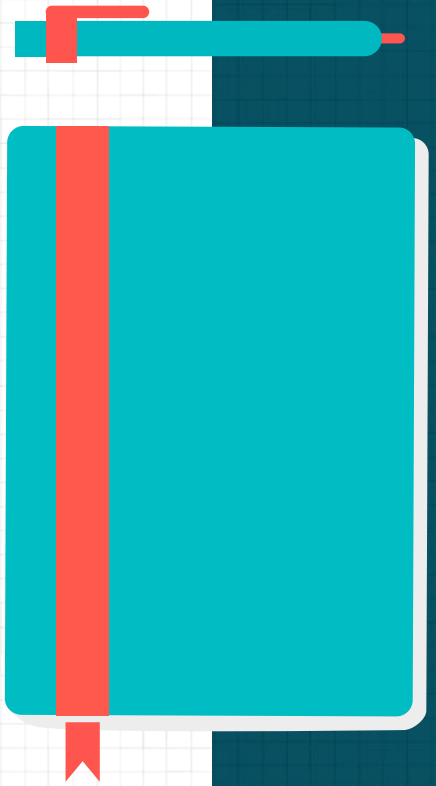
- Cách khử đệ quy 1: dùng vòng lặp
- Cách khử đệ quy 2: mô phỏng cách thực thi hàm đệ quy bằng cách dùng cấu trúc ngăn xếp (stack)
  - Lưu trạng thái hiện tại vào stack
  - Lấy trạng thái ra khỏi stack để xử lý

Bài tập 1

- Cho số nguyên dương N
  - Tính tổng các chữ số chẵn
  - Tìm chữ số lớn nhất/nhỏ nhất
  - Tính logarit cơ số 2 của N
  - Đổi N sang hệ nhị phân
  - In ra tam giác Pascal có chiều cao N
  - Tính lũy thừa a mũ N

Bài tập 2

- Viết hàm đệ quy cho bài toán
  - Tính chỉnh hợp chập k, tổ hợp chập k của n phần tử
  - Bài toán tháp Hà Nội
  - Bài toán tám hậu
  - Bài toán mã đi tuần



The End!

Do you have any questions?