HOA SEN UNIVERSITY

www.hoasen.edu.vn

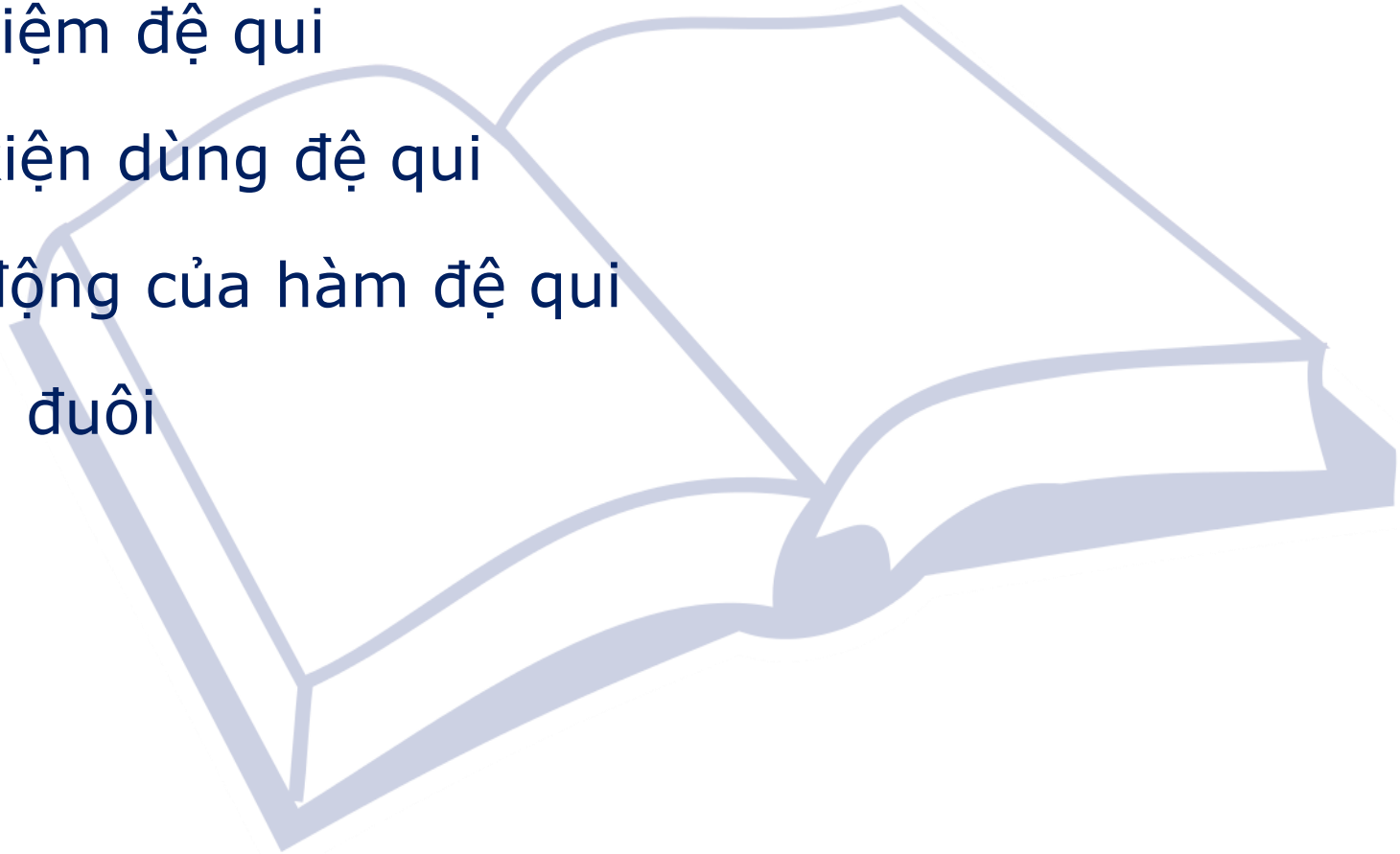
Cấu Trúc Dữ Liệu & Giải Thuật

Data Structures & Algorithms

GV: Phan Hồng Trung

Bài 2 – Đệ qui (Recursion)

- Khái niệm đệ qui
- Điều kiện dùng đệ qui
- Hoạt động của hàm đệ qui
- Đệ qui đuôi



Khái niệm đệ qui

- Đệ qui là một quá trình một hàm gọi chính nó một cách trực tiếp hoặc gián tiếp.

```
1  //Đệ qui trực tiếp
2  function f(){
3      ...
4      call f();
5      ...
6  }
```

```
1  //Đệ qui gián tiếp
2  function f(){
3      ...
4      call g();
5      ...
6  }
7  function g(){
8      ...
9      call f();
10     ...
11 }
```

Khái niệm đệ qui

- Ưu điểm:
  - Giúp giải quyết dễ dàng một lớp bài toán khó.
  - Chương trình rõ ràng, dễ hiểu.
- Nhược điểm:
  - Tốn nhiều bộ nhớ do lưu nhiều thông tin gọi hàm trong stack.
  - Hàm đệ qui chạy chậm hơn hàm không dùng đệ qui.
- Khử đệ qui là quá trình thay thế lập trình đệ qui bằng những kỹ thuật lập trình khác như dùng vòng lặp.

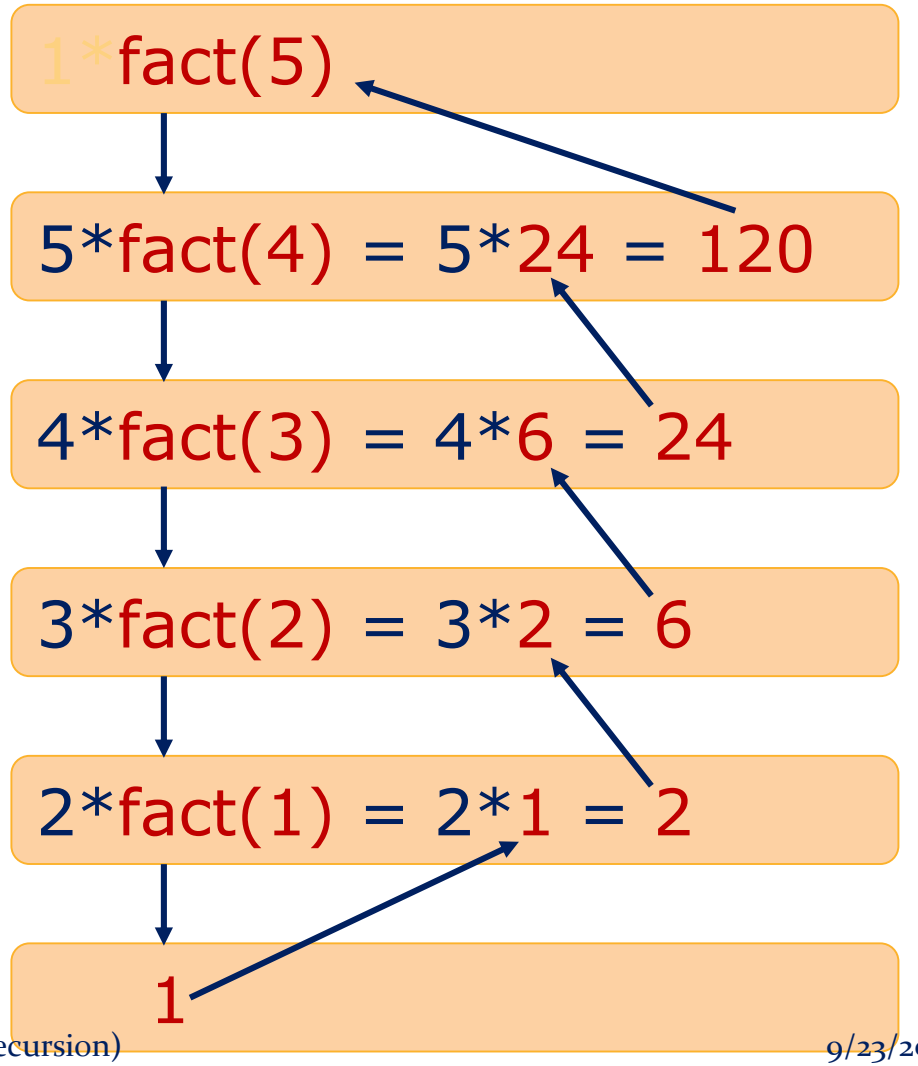
Điều kiện dùng đệ qui

- Để giải quyết bằng đệ qui, vấn đề phải được định nghĩa dưới dạng đệ qui.
- Định nghĩa đệ qui gồm hai phần:
  - Phần điều kiện dừng (Phần cơ sở): điều kiện dừng thuật toán.
  - Phần đệ qui: thu hẹp dần phạm vi bài toán hướng đến điều kiện dừng.
- Ví dụ bài toán giai thừa:
  - Định nghĩa thông thường: n! = 1*2*…*n
  - Định nghĩa đệ qui: $n! = \begin{cases} 1 & 0 \le n \le 1 \ (\text{điều kiện dừng}) \\ n * (n-1)! & n > 1 \quad (\text{phần đệ qui}) \end{cases}$

Hoạt động của hàm đệ qui

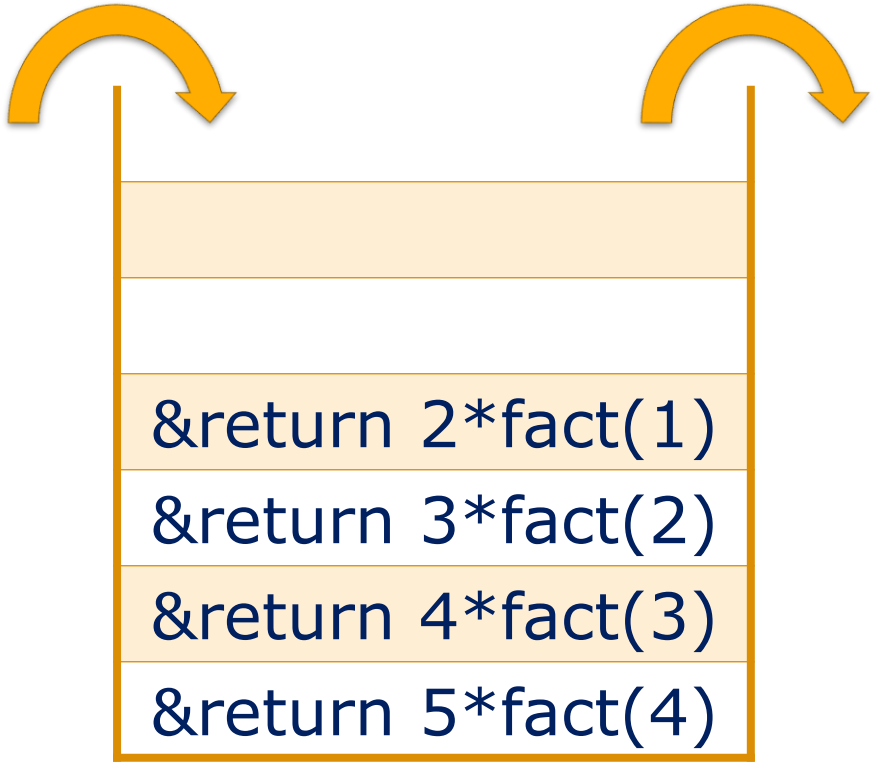
$$n! = \begin{cases} 1 & 0 \le n \le 1 \\ n * (n-1)! & n > 1 \end{cases}$$

```
1   #include <iostream>
2   using namespace std;
3   int fact(int n)
4   {
5       if (n <= 1) //điều kiện dừng
6           return 1;
7       else
8           return n * fact(n - 1);
9   }
10  int main()
11  {
12      cout << fact(5) << endl;
13      return 0;
14  }
```



Hoạt động của hàm đệ qui

```
1   #include <iostream>
2   using namespace std;
3   int fact(int n)
4   {
5       if (n <= 1) //điều kiện dừng
6           return 1;
7       else
8           return n * fact(n - 1);
9   }
10  int main()
11  {
12      cout << fact(5) << endl;
13      return 0;
14  }
```



- Stack được dùng để lưu thông tin gọi hàm.

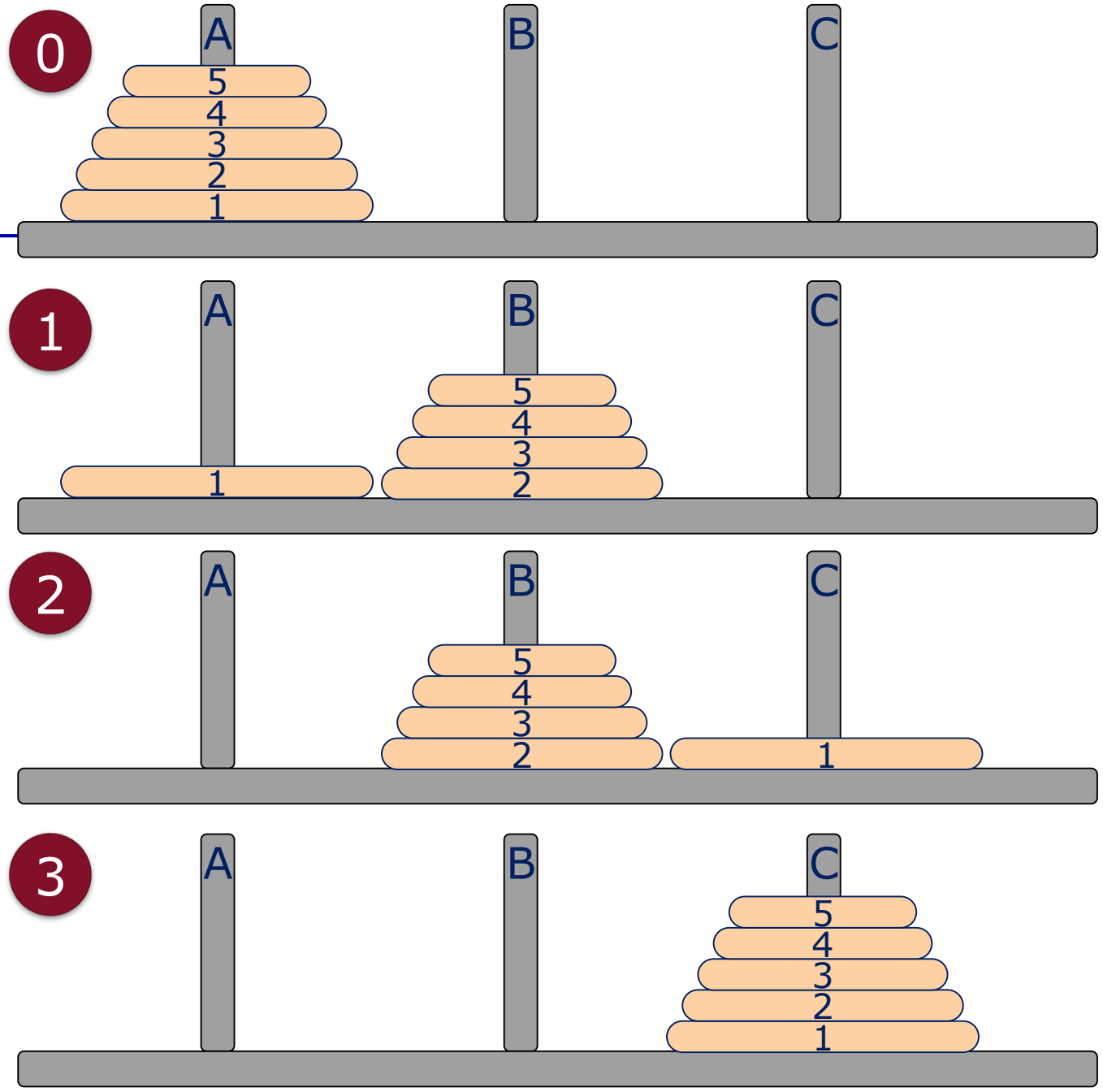
Các ví dụ

- Tính U_n của dãy Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55…
  - $U_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ U_{n-1} + U_{n-2} & n \geq 2 \end{cases}$
- Tính tổng: $S = 1 + 2 + \cdots + n \ (n \geq 1)$
- Tính lũy thừa: $a^n = \underbrace{a * a * \cdots * a}_{n \text{ lần } n} \ (\text{a} \neq 0, n \geq 0)$
- Tính tổ hợp chập k của n phần tử: $C_n^k = \begin{cases} 0 & k > n \\ \frac{n!}{k!(n-k)!} & k \leq n \end{cases}$

HOA SEN UNIVERSITY

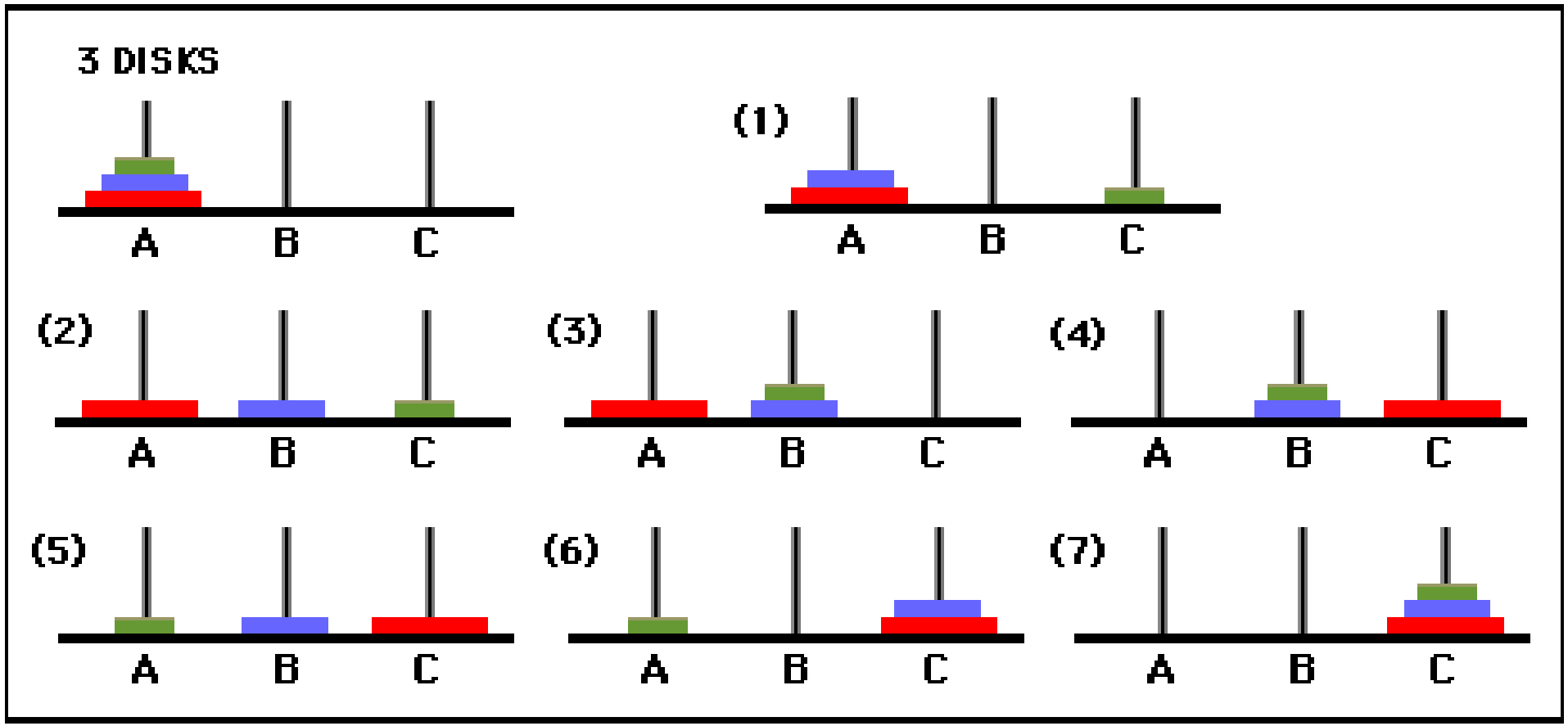
Tháp Hà Nội

- Di chuyển n đĩa từ cột A sang cột C dùng cột B làm trung gian.
- Nếu n=1:
  - Di chuyển 1 đĩa từ A sang C.
- Nếu n>1:
  1. Di chuyển (n-1) đĩa từ A sang B.
  2. Di chuyển 1 đĩa từ A sang C.
  3. Di chuyển (n-1) đĩa từ B sang C.



Tháp Hà Nội

- Di chuyển 3 đĩa từ cột A sang cột C dùng cột B làm trung gian.



Đệ qui đuôi (Tail Recursion)

- Một hàm đệ qui là đệ qui đuôi khi lệnh gọi đệ qui là lệnh cuối cùng được thực thi trong hàm.

```
1  int fact(int n)
2  {
3      if (n <= 1)
4          return 1;
5      else
6          return n * fact(n - 1);
7  }
```

```
1  int factTR(int n, int result = 1)
2  {
3      if (n <= 1)
4          return result;
5      else
6          return factTR(n - 1, n * result);
7  }
```

- fact() không là hàm đệ qui đuôi vì sau lệnh gọi đệ qui fact(n – 1) còn phải tính biểu thức n * fact(n – 1). factTR() là hàm đệ qui đuôi.

- Đệ qui đuôi là phiên bản tối ưu của đệ qui thông thường, giúp tiết kiệm bộ nhớ do không cần dùng stack để lưu thông tin gọi hàm.


Q
A