BÀI 11.
THUẬT TOÁN ĐỆ QUY

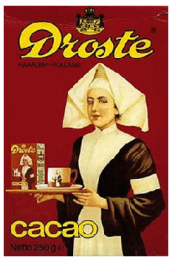
Nội dung

- Thuật toán đệ quy và hàm đệ quy là gì?
- Thuật toán đệ quy hoạt động như thế nào?
- Một số thuật toán đệ quy đơn giản

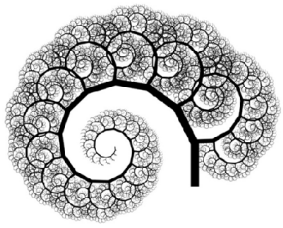
1. THUẬT TOÁN ĐỆ QUY LÀ GÌ?

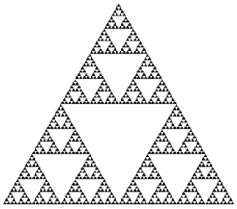
3

Khái niệm "đệ quy"



Droste effect




Sierpinksi triangle

- Đối tượng đệ quy: là đối tượng mà một phần hoặc toàn bộ đối tượng được định nghĩa thông qua chính nó
  - Quy nạp toán học
- Quá trình đệ quy: là quá trình mà một phần hoặc toàn bộ quá trình tự lặp lại theo cùng một cách

4

Đệ quy – Ví dụ

- Định nghĩa số tự nhiên:
  - 0 là số tự nhiên
  - n là số tự nhiên nếu n-1 cũng là số tự nhiên
- Dãy số:
  - Dãy số là một số
  - Dãy số là một số và sau đó là một dãy số
- Một số thuật ngữ:
  - PHP = PHP: Hypertext Preprocessor
  - GNU = GNU's Not Unix

5

Thuật toán đệ quy

- Thuật toán đệ quy là thuật toán mà trong các bước thực hiện tự thực hiện lại chính nó với đầu vào nhỏ hơn(kích thước, giá trị, mức độ phức tạp…)
- Tư tưởng của thuật toán đệ quy là đưa bài toán cần giải về bài toán đồng dạng nhưng ở mức độ thấp hơn

Ví dụ: Tính dãy số Fibonacci, n!

- Tại sao dùng đệ quy:
  - Một số thuật toán trong cách thức thực hiện mặc nhiên có tính đệ quy
  - Một số thuật toán rất khó tìm ra lời giải có thể sử dụng kỹ thuật đệ quy để giải quyết. Ví dụ: Bài toán tháp Hà Nội
  -

6

Thuật toán đệ quy(tiếp)

- Để xây dựng thuật toán đệ quy, cần xác định:
  - Trường hợp cơ bản: (Các) trường hợp không cần thực hiện lại thuật toán, có thể xác định được ngay kết quả đầu ra
  - Phần tổng quát: Có yêu cầu gọi đệ quy
    - Cần xác định nguyên lý đưa trường hợp tổng quát về trường hợp cơ bản
    - Đảm bảo tính dừng của giải thuật đệ quy - chắc chắn từ trường hợp tổng quát sẽ đến được trường hợp cơ bản
- Hàm/Phương thức đệ quy: Hàm/Phương thức có lời gọi tới chính nó

7

Các thức hoạt động của thuật toán đệ quy

- Để hiểu các thức thực hiện của thuật toán đệ quy, xem xét ví dụ tính n! sau

$$n! = \begin{cases} 1, & n = 0 \\ n \times (n-1) \times \cdots \times 2 \times 1, & n > 0 \end{cases}$$

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

Sử dụng vòng lặp

```
int fact(int n) {
  int result = 1;
  for (int i=1;i<=n;i++)
    result *= i;
  return result;
}
```

Sử dụng đệ quy

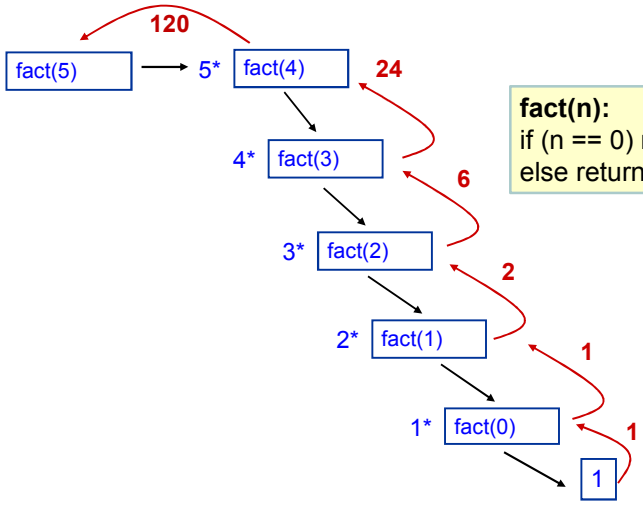
```
int fact(int n) {
  if (n == 0)
    return 1;
  else
    return n * fact(n-1);
}
```

Lời gọi đệ quy

Trường hợp cơ sở

8

Đệ quy tính n!



fact(n):
if (n == 0) return 1;
else return n * fact(n-1);

9

Tính n!

```
public class Factorial{
   public int fact(int n){
       if n == 0 return 1;
       else return n*fact(n-1);
   }

   public static void main(){
       System.out.println(n + "! = " + fact(n));
   }
}
```

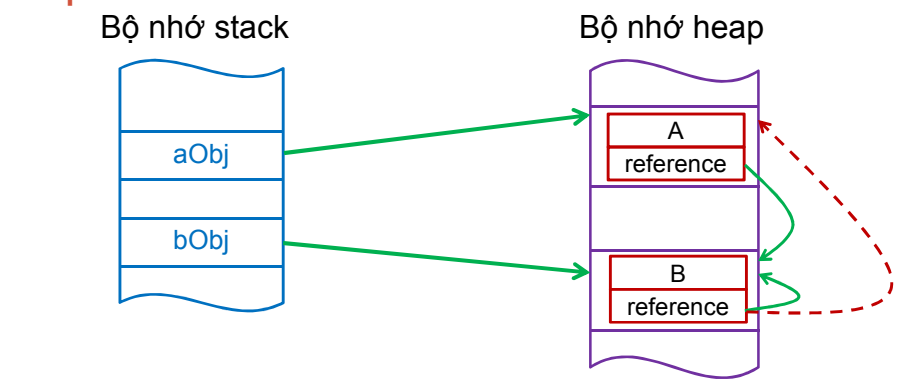
10

Lớp chứa tham chiếu có kiểu chính nó

```
public class MyClass<E>{
    private E element;
    private MyClass<E> reference;
    public MyClass(E item){
        element = item;
        reference = null
    }
    public setReference(MyClass<E> ref){
        reference = ref;
    }
    public MyClass<E> getReference(){
        return reference;
    }
    public void showElement(){
        System.out.println(element.toString());
    }
}
```

11

Lớp chứa tham chiếu có kiểu chính nó




```
MyClass<String> aObj = new MyClass<String>("A");
MyClass<String> bObj = new MyClass<String>("B");
aObj.setReference(bObj);
bObj.setReference(bObj);
bObj.setReference(aObj);   //---
```

12

Lớp chứa tham chiếu có kiểu chính nó

```
public class TestMyClass {
   public static void main(String[] args){
       MyClass<String> aObj = new MyClass<String>("A");
       MyClass<String> bObj = new MyClass<String>("B");
       aObj.setReference(bObj);
       aObj.getReference().showElement();
       bObj.setReference(aObj);
       bObj.getReference().getReference().showElement();
       bObj.setReference(bObj);
       bObj.getReference().showElement();
       bObj.getReference().setReference(
                    bObj.getReference().getReference());
       bObj.getReference().showElement();
   }
}//Chú ý: Không có lời gọi nào ở trên là đệ quy
```

Kết quả hiển thị là gì?

13

2. MỘT SỐ VÍ DỤ

14

Tính số Fibonacci

- Dãy số Fibonacci: 1, 1, 2, 3, 5 ,8, 13…
- Tính số Fibonacci thứ n:

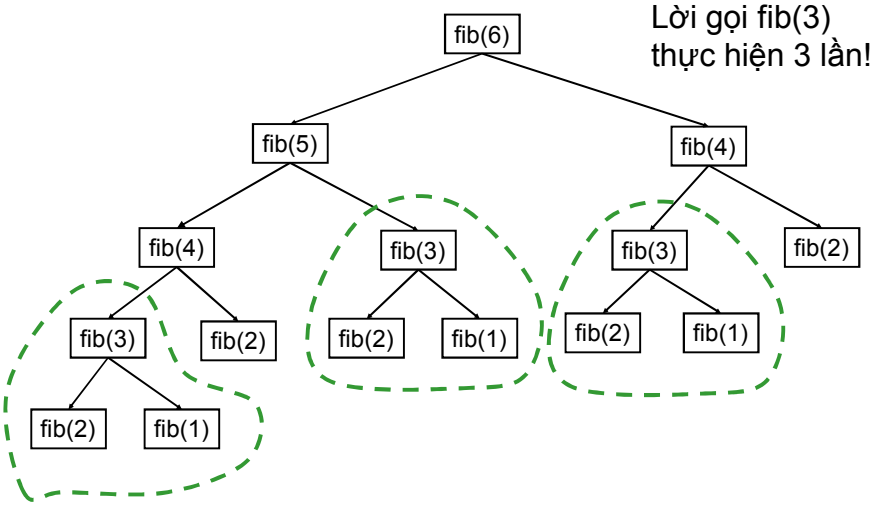
$F_n = 1$ nếu $n \leq 2$

$F_n = F_{n-1} + F_{n-2}$ nếu $n > 2$

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

15

Quá trình thực hiện



Lời gọi fib(3) thực hiện 3 lần!

16

Fibonacci – Khử đệ qui với vòng lặp

```
int fib(int n) {
  if (n <= 2)
    return 1;
  else {
    int prev1=1, prev2=1, curr;
    for (int i=3; i<=n; i++) {
      curr = prev1 + prev2;
      prev2 = prev1;
      prev1 = curr;
    }
    return curr;
  }
}
```

17

Chuyển đổi cơ số

- Chuyển một giá trị nguyên dương N từ hệ thập phân sang hệ đếm khác với cơ số $2 \le b \le 10$
- Cách thực hiện:
  - **Bước 1:**Lấy phần nguyên của $N_{(10)}$ chia cho b, ta được thương là T_1 số **dư d_1**.
  - **Bước 2:** Nếu T_1 khác 0, Lấy T_1 chia tiếp cho b, ta được thương số là T_2, số **dư là d_2**

  (Cứ làm như vậy cho tới bước thứ n, khi ta được T_n =0)
  - **Bước n:** Nếu T_{n-1} khác 0, lấy T_{n-1} chia cho b, ta được thương số là T_n =0, **số dư là d_n**
  - Kết quả ta được số $N_{(b)}$ là số tạo bởi các số dư (***được viết theo thứ tự ngược lại***) trong các bước trên

  Phần nguyên của $N_{(10)} = d_n d_{n-1} \ldots d_{1\,(b)}$

18

Chuyển đổi cơ số

```
public static void displayInBase(int n, int base) {
  if (n > 0) {
    displayInBase(n / base, base);
    System.out.print(n % base);
  }
}
```

Ví dụ 1:
n = 123, base = 10

123/10 =12 123 % 10 = 3
12/10 = 1 12 % 10 = 2
1/10 = 0 1 % 10 = 1
Kết quả: 123

Ví dụ 2:
n = 123, base = 8

123/8 = 15 123 % 8 = 3
15/8 = 1 15 % 8 = 7
1/8 = 0 1 % 8 = 1
Kết quả: 173

19

Bài toán tháp Hà Nội

- Có 3 cọc A, B, C.
- Trên cọc A có một chồng đĩa, tìm cách di chuyển sang chồng đĩa khác.
- Luật: đĩa lớn không được đặt lên đĩa nhỏ

A **B** **C**

20

Bài toán tháp Hà Nội

- Trường hợp cơ sở là gì?
  - A: 1 đĩa
  - B: 0 đĩa
- Bước đệ quy?
  - A: chuyển n-1 đĩa ở trên sang cọc khác
  - B: chuyển n-1 đĩa ở dưới sang cọc khác
- Cần gọi bao nhiêu bước đệ quy?
  - A: 1
  - B: 2
  - C: 3

21

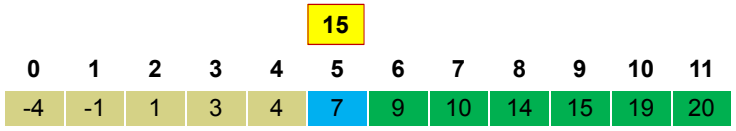
Bài toán tháp Hà Nội

```
public static void Towers(int numDisks, char A, char C,
char B) {
 if (numDisks == 1) {
   System.out.println("Move top disk from pole " + A +
" to pole " + C);
 } else {
   Towers(numDisks - 1, A, B, C);  // Gọi đệ quy lần 1
   Towers(1, A, C, B);             // Gọi đệ quy lần 2
   Towers(numDisks - 1, B, C, A);  // Gọi đệ quy lần 3
  }
}
```

22

Tìm kiếm nhị phân trên mảng

- Giả sử mảng arr đã được sắp xếp tăng dần
- Phương thức tìm kiếm nhị phân có thể gọi như sau: so sánh phần tử đang duyệt với phần tử trung vị
  - Trường hợp cơ sở 1: phần tử đang duyệt bằng khóa đang tìm kiếm
  - Trường hợp cơ sở 2: không tìm thấy khóa
  - Thực hiện lời gọi đệ quy:
    - Nếu khóa lớn hơn phần tử đang duyệt: tìm kiếm ở nửa trái
    - Nếu khóa nhỏ hơn phần tử đang duyệt: tìm kiếm ở nửa phải



23

Tìm kiếm nhị phân trên mảng

```
public int binarySearch (int [] a, int x){
   return binarySearch(int [] a, int x, 0, a.length-1);
}

private int binarySearch(int [] a, int x,
                         int low, int high) throws ItemNotFound  {
   // low: index of the low value in the subarray
   // high: index of the highest value in the subarray
  if (low > high)      // Base case 1: item not found
    throw new ItemNotFound("Not Found");

  int mid = (low + high) / 2;

  if (x > a[mid])
    return binarySearch(a, x, mid + 1, high);

  else if (x < a[mid])
    return binarySearch(a, x, low, mid - 1);

  else
    return mid;        // Base case 2: item found
}
```

24

Bài tập

Sử dụng thuật toán đệ quy để thực hiện các phương thức sau:

- Tìm UCLN của 2 số
- Tính tổ hợp chập k của n C_k^n

25

Tài liệu tham khảo

- Bài giảng sử dụng hình ảnh và mã nguồn minh họa từ bài giảng của Đại học QG Singapore (NUS)

26