CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Bài 02: Các thuật toán sắp xếp cơ bản

Nguyễn Thị Tâm
nguyenthitam.hus@gmail.com

Ngày 17 tháng 9 năm 2025

Nội dung

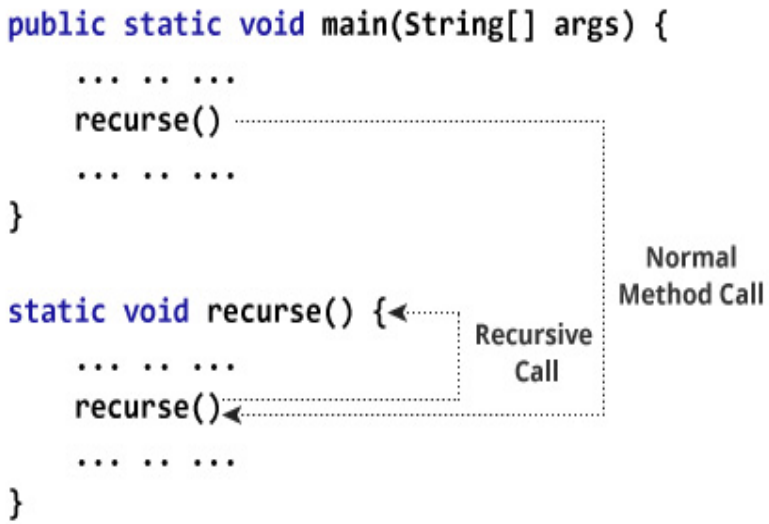
Đệ quy

Nội dung

- Khái niệm đệ quy
- So sánh lặp và đệ quy

Khái niệm Đệ quy

- Cơ chế cho phép một hàm/phương thức gọi đến chính nó.



Ví dụ chương trình đệ quy

- Với $n \in N$, $n \geq 1$, tính giai thừa

$$n! = n \times (n-1) \ldots \times 2 \times 1.$$

- Sử dụng vòng lặp ???
- Đệ quy

```
public static long fact(int n)
{
   if (n > 1)
      return n * fact(n-1);
   else
      return 1;
}
```

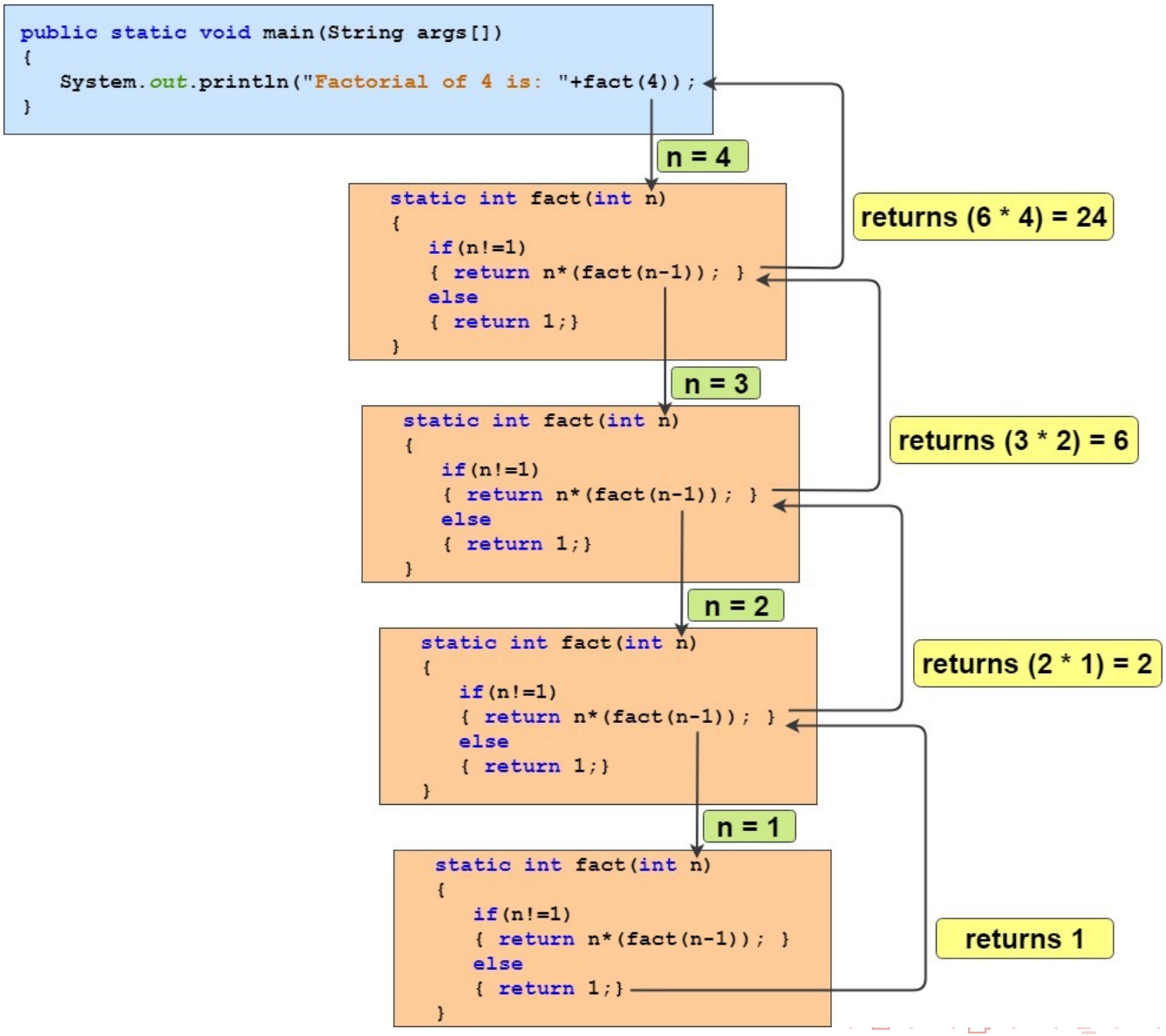
Ví dụ chương trình đệ quy

- Truy ngược vết của chương trình đệ quy

```
fact(5)
fact(4)
fact(3)
fact(2)
fact(1)
return 1
return 2*1 = 1
return 3*2 = 6
return 4*6 = 24
return 5*24 = 120
```

.

Ví dụ chương trình đệ quy



Luyện tập 1: Dãy Fibonacci

- Định nghĩa dãy Fibonacci

$$F_0 = 0; \quad F_1 = 1; \quad F_n = F_{n-1} + F_{n-2} \quad \forall n > 1.$$

- Sử dụng vòng lặp (tự thực hiện!)
- Đệ quy

```
public static long fibonacci(int n)
{
   if (n <= 1)
      return n;
   else
      return fibonacci(n-1) + fibonacci(n-2);
}
```

Nhận xét

- Gần như mọi vấn đề giải quyết được bằng đệ quy thì có thể giải quyết được bằng cấu trúc lặp.
- Một số trường hợp, đệ quy cho phép chương trình dễ hình dung - dễ hiểu, dễ thiết lập hơn.
- Vòng lặp hiệu quả thực hiện tốt hơn!

Luyện tập 2: Biểu diễn nhị phân của số tự nhiên

- Với $n \in N$, hãy in ra xâu ký tự biểu diễn nhị phân của n.

Luyện tập 2: Biểu diễn nhị phân của số tự nhiên

- Với $n \in N$, hãy in ra xâu ký tự biểu diễn nhị phân của n.
- Phân tích:
  - Nếu $n = 0$ hoặc $n = 1$: Biểu diễn là chữ số n.
  - Nếu $n > 1$: Là xâu biểu diễn của $n/2$ (chia nguyên) ghép với chữ số $n\%2$ (lấy dư). Ví dụ

$$\begin{aligned} 35_{10} = 100011_2 &= \text{"10001"} + \text{"1"} = [17]_2 + \text{"1"} \\ &= [35/2]_2 + [35\%2]_2 \end{aligned}$$

  - Sử dụng String.valueOf(n) để đổi n thành xâu ký tự tương ứng.
- Sử dụng vòng lặp (tự thực hiện!)
- Đệ quy?

Luyện tập 2: Biểu diễn nhị phân của số tự nhiên

```
public static String toBinary(int n) {
   if (n <= 1 ) {
      return String.valueOf(n);
   }
   return toBinary(n / 2) + String.valueOf(n % 2);
}
```

Luyện tập 3: Tìm ước chung lớn nhất

- Tìm ước chung lớn nhất (*greatest common divisor* - gcd) của hai số $p, q \in N$ dương $\text{gdc}(p, q)$.
  - Nếu $q = 0$: $\text{gdc}(p, q) = p$.
  - Nếu $p > q$: $\text{gdc}(p, q) = \text{gdc}(q, p \mod q)$ và ngược lại.
- Sử dụng vòng lặp (tự thực hiện!)
- Đệ quy?

Luyện tập 3: Tìm ước chung lớn nhất

```
public static int gcd(int p, int q) {
   if (q == 0) return p;
   else return gcd(q, p % q);
}
```

Truy ngược theo vết:

```
gcd(1440, 408)
gcd(408, 216)
gcd(216, 192)
gcd(192, 24)
gcd(24, 0)
return 24
return 24
return 24
return 24
return 24
```

Ý tưởng thuật toán chia để trị

Phương pháp thiết kế dựa trên hai thao tác chính:

- Chia (*Divide*): phân rã bài toán ban đầu thành các bài toán con có kích thước nhỏ hơn, có cùng cách giải.
- Trị (*Conquer*): giải từng bài toán con (theo cách tương tự bài toán đầu - đệ qui) rồi tổng hợp các lời giải để nhận kết quả của bài toán ban đầu.

Việc "phân rã" được thực hiện trên miền dữ liệu (chia miền dữ liệu thành các miền nhỏ hơn tương đương với một bài toán con)

Mô hình - Lược đồ giải thuật

Mô hình

- Xét bài toán trên miền dữ liệu R
- *D&C(R)* là hàm thể hiện cách giải bài toán trên miền dữ liệu R theo phương pháp chia để trị.
- Nếu R có thể phân rã thành n miền con: $R = R_1 \cup R_2 \cup ... \cup R_n$

Lược đồ giải thuật

```
D&C(R)
    if (R = R0)
        Giải D&C(R0)
    else {
        Chia miền R thành R1, R2, ..., Rn
        for ( i = 1; i<=n;i++)
            Giải D&C(Ri);
        Tổng hợp kết quả;
    }
```

Nội dung

Bài toán tìm kiếm nhị phân trên mảng được sắp

Bài toán 1

Cho mảng a có n phần tử được sắp theo thứ tự tăng dần và một giá trị x bất kỳ. Kiểm tra xem phần tử x có trong mảng hay không?

Bài toán tìm kiếm nhị phân trên mảng được sắp

Bài toán 1

Cho mảng a có n phần tử được sắp theo thứ tự tăng dần và một giá trị x bất kỳ. Kiểm tra xem phần tử x có trong mảng hay không?

Ý tưởng: Chia đôi mảng, mỗi lần so sánh phần tử giữa với x, nếu phần tử x nhỏ hơn thì xét nửa trái, ngược lại xét nửa phải

Bài toán tìm kiếm nhị phân trên mảng được sắp

Lược đồ thuật toán

```
public static int binarySearch(int[] a, int x, int l, int r) {
    if (l == r) {
            return (x == a[l]) ? l : -1;
        } else {
            int m = (l + r) / 2;
            if (x == a[m]) {
                return m;
            } else {
                if (x < a[m]) {
                    return BinarySearch(a, x, l, m - 1);
                } else {
                    return BinarySearch(a, x, m + 1, r);
                }
            }
        }
}
```

Bài toán MinMax

Bài toán 3

Cho mảng a có n phần tử. Tìm giá trị lớn nhất (max), giá trị nhỏ nhất (min) trong đoạn $a_l ... a_r$ của mảng.

Bài toán MinMax

Bài toán 3

Cho mảng a có n phần tử. Tìm giá trị lớn nhất (max), giá trị nhỏ nhất (min) trong đoạn $a_l ... a_r$ của mảng.

Ý tưởng:

- Tại mỗi bước chia đôi đoạn cần tìm rồi tìm min, max của từng đoạn. Sau đó, tổng hợp lại kết quả;
- Nếu đoạn chỉ có 1 phần tử thì $min = max$ và bằng phần tử đó.

Bài toán MinMax

```
public static int min(int[] a, int l, int r) {
   if (l == r) {
      return a[l];
   } else {
      int m = (l + r) / 2;
      int min1 = min(a, l, m);
      int min2 = min(a, m + 1, r);
      return (min1 < min2) ? min1 : min2;
   }
}
```

Bài toán nhân ma trận vuông

Thuật toán nhân ma trận vuông thông thường

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

trong đó:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

Bài toán nhân ma trận vuông

Thuật toán nhân ma trận Strassen

$$
\begin{aligned}
C_{11} &= P_5 + P_4 - P_2 + P_6 \\
C_{12} &= P_1 + P_2 \\
C_{21} &= P_3 + P_4 \\
C_{22} &= P_1 + P_5 - P_3 - P_7
\end{aligned}
$$

trong đó:

$$
\begin{aligned}
P_1 &= A_{11}(B_{12} - B_{22}) \\
P_2 &= (A_{11} + A_{12})B_{22} \\
P_3 &= (A_{21} + A22)B_{11} \\
P_4 &= (A_{22})(B_{21} - B_{11}) \\
P_5 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
P_6 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\
P_7 &= (A_{11} - A_{21})(B_{11} + B_{12})
\end{aligned}
$$

Thuật toán nhân ma trận Strassen

Algorithm 2 Thuật toán nhân ma trận Strassen

function STRASSEN(A, B, n)
  if $n = 1$ **then return** $A \times B$
  else
    Tính $A_{11}, B_{11}, ..., A_{22}, B_{22}$
    $P_1 \leftarrow strassen(A_{11}, B_{12} - B_{22}, n/2)$
    $P_2 \leftarrow strassen(A_{11} + A_{12}, B_{22}, n/2)$
    $P_3 \leftarrow strassen(A_{21} + A_{22}, B_{11}, n/2)$
    $P_4 \leftarrow strassen(A_{22}, B_{21} - B_{11}, n/2)$
    $P_5 \leftarrow strassen(A_{11} + A_{22}, B_{11} + B_{22}, n/2)$
    $P_6 \leftarrow strassen(A_{12} - A_{22}, B_{21} + B_{22}, n/2)$
    $P_7 \leftarrow strassen(A_{11} - A_{21}, B_{11} + B_{12}, n/2)$
    $C_{11} \leftarrow P_5 + P_4 - P_2 + P_6$
    $C_{12} \leftarrow P_1 + P_2$
    $C_{21} \leftarrow P_4 + P_3$
    $C_{22} \leftarrow P_1 + P_5 - P_3 - P_7$
    return C
  end if
end function

Bài toán lát gạch

Bài toán 5

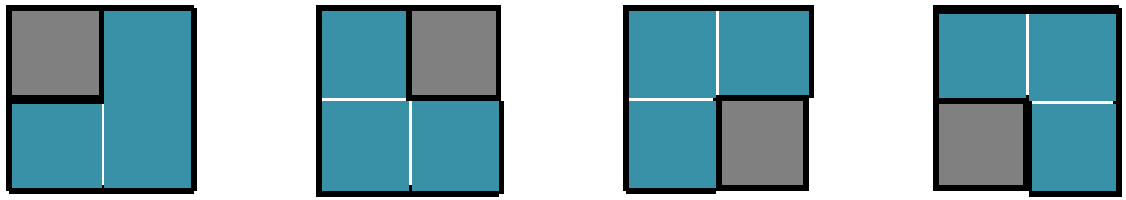
Cho một nền nhà hình vuông, kích thước $2^n \times 2^n$. Người ta dành riêng một ô để thoát nước. Hãy tìm cách xếp những viên gạch hình chữ L trên nền nhà, sao cho nền nhà được lát kín gạch (trừ ô vuông được dùng để thoát nước).

Bài toán lát gạch

Bài toán 5

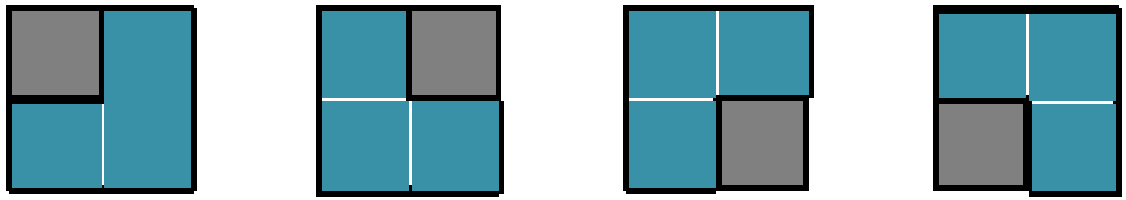
Cho một nền nhà hình vuông, kích thước $2^n \times 2^n$. Người ta dành riêng một ô để thoát nước. Hãy tìm cách xếp những viên gạch hình chữ L trên nền nhà, sao cho nền nhà được lát kín gạch (trừ ô vuông được dùng để thoát nước).

Bài toán lát gạch



Hình 1: Lát nền nhà kích thước 2×2

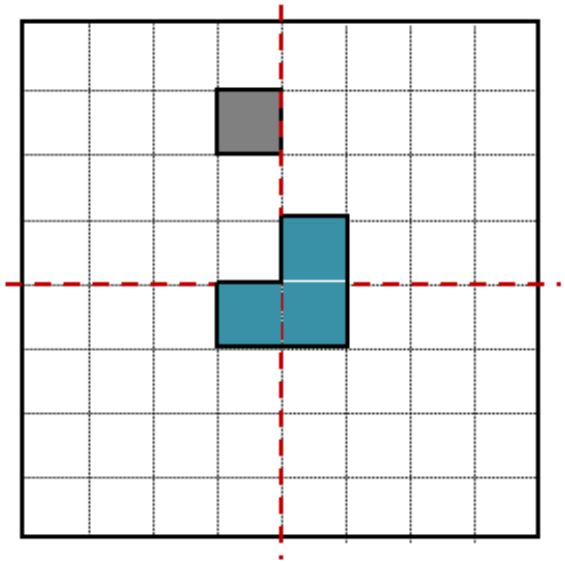
Bài toán lát gạch



Hình 1: Lát nền nhà kích thước 2×2

Ý tưởng: Làm cách nào đó, giảm kích thước nền nhà về kích thước 2×2 với 1 ô trống để giải nó.

Bài toán lát gạch



Hình 2: Lát nền nhà kích thước 2×2

Chèn một viên gạch vào trung tâm, sau đó, coi những ô đã có gạch như những ô thoát nước (không chèn được gạch nữa). Chúng ta sẽ đưa bài toán ban đầu về 4 bài toán nhỏ hơn. Quá trình tiếp tục cho đến khi gặp kích thước nhỏ nhất là 2×2.

Bài toán lát gạch

Algorithm 3 Thuật toán lát nền

function TILE(n,L)
  if n=1 **then**
    Lát nền kích thước 2×2 với 1 ô trống
  end if
  Chia nền nhà thành bốn hình vuông có kích thước bằng nhau
  Tìm góc phần tư chứa ô trống (kí hiệu là L_1)
  $Tile(n-1, L_1)$
  Đặt viên gạch vào trung tâm sao cho mỗi góc phần tư còn lại sẽ chứa một phần của viên gạch mới đặt. Coi như những ô này là những ô trống.
  Đặt L_2, L_3, L_4 là vị trí của lỗ trống
  $Tile(n-1, L_2)$
  $Tile(n-1, L_3)$
  $Tile(n-1, L_4)$
end function