Chương 2

Đệ quy và giải thuật đệ quy

Mục tiêu

- Đưa ra các khái niệm về đệ quy và giải thuật đệ quy
- Cách thiết kế giải thuật đệ quy
- Một số giải thuật đệ quy điển hình
- Đánh giá độ phức tạp giải thuật đệ quy

Nội dung

2.1 Khái niệm về đệ quy

2.2 Giải thuật đệ quy và thủ tục đệ quy

2.3 Thiết kế giải thuật đệ quy

2.4 Phân loại đệ quy

2.1 Khái niệm về đệ quy

- **Khái niệm**

  Một đối tượng gọi là đệ qui nếu nó nằm trong chính nó như một bộ phận (hay nói cách khác nó được định nghĩa qua chính nó).

- **Ví dụ:**

  Trong toán học ta hay gặp các định nghĩa đệ qui

  1. Số tự nhiên

     1 là số tự nhiên

     x là số tự nhiên nếu x-1 là số tự nhiên

2.1 Khái niệm về đệ quy

- **Ví dụ:**

  Hàm tính giai thừa

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

2.1 Khái niệm về đệ quy

- **Ví dụ:**

  Tính số thứ n của dãy Fibonaci

  1, 1, 2, 3, 5, 8,……..

$$\textbf{Fib}(n) = \begin{cases} 1 & \text{với } n<=2 \\ \textbf{Fib}(n-1) + \textbf{Fib}(n-2) & \text{với } n>2 \end{cases}$$

2.2 Giải thuật đệ quy

- **Khái niệm**
  - Lời giải T của bài toán được biểu diễn qua T' giống T thì lời giải T được gọi là ***lời giải đệ qui***.
  - Giải thuật cho lời giải đệ qui gọi là ***giải thuật đệ qui***

2.2 Giải thuật đệ quy

- **Ví dụ** Tìm từ trong quyển từ điển
  - Nếu từ điển bằng một trang thì tìm tuần tự trong đó
  - Nếu từ điển khác 1 trang thì
    - ✓ Chia đôi từ điển, xem từ cần tìm thuộc nửa đầu hay nửa cuối
    - ✓ Lập lại giải thuật tìm từ trong trang (chỉ còn 1/2)

2.2 Giải thuật đệ quy

- **Tính chất**
  - Có một trường hợp đặc biệt để kết thúc đệ qui (Suy biến) **n! = 1 với n=0**
  - Trường hợp tổng quát suy ra cách tính tương tự nhưng với ý nghĩa nhỏ đi và dần đến trường hợp suy biến. **n! = n * (n-1)! với n>0**

2.2 Giải thuật đệ quy

- **Ví dụ:**

Viết hàm đệ qui tính n!

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

2.2 Giải thuật đệ quy

- **Tính chất của giải thuật đệ quy**
  - Một giải thuật đệ qui chứa lời gọi đến chính giải thuật đó trong thân giải thuật
  - Có một trường hợp suy biến để kết thúc lời gọi đệ qui
  - Mỗi lần gọi thì lại dần đến trường hợp suy biến

2.3 Thiết kế giải thuật đệ quy

- **Ví dụ**

$$\textbf{Fib}(n) = \begin{cases} 1 & \text{với } n<=2 \\ \textbf{Fib}(n-1) + \textbf{Fib}(n-2) & \text{với } n>2 \end{cases}$$

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

2.3 Thiết kế giải thuật đệ quy

- Nếu đã có định nghĩa đệ quy thì chuyển thành giải thuật đệ quy ngay
- Nếu chưa xuất hiện định nghĩa đệ quy thì ta phải xây dựng định nghĩa đệ quy sau đó mới chuyển thành giải thuật đệ quy.

2.3 Thiết kế giải thuật đệ quy

- **Bài toán tháp Hà Nội**

  Có ba cột A, B, C. Cột A hiện đang gắn n đĩa có kích thước khác nhau, đĩa nhỏ ở trên đĩa lớn hơn ở dưới. Cần chuyển n đĩa từ cột A sang cột C (dùng cột B làm trung gian) với điều kiện mỗi lần chỉ được chuyển một đĩa, đĩa đặt trên bao giờ cũng nhỏ hơn đĩa đặt dưới.

2.3 Thiết kế giải thuật đệ quy

- **Bài toán tháp Hà Nội**

Định nghĩa đệ quy

- **Điều kiện dừng**

  Nếu cột A chỉ còn 1 đĩa thì chuyển 1 đĩa từ A sang C

- **Bước đệ quy**

  Nếu cột A còn n đĩa (n>1) thì gọi đệ quy như sau:

  Chuyển n đĩa từ A→C
  - Chuyển n-1 đĩa từ A → B
  - Chuyển đĩa cuối từ A → C
  - Chuyển n-1 đĩa từ B → C

2.3 Thiết kế giải thuật đệ quy

- **Bài toán tháp Hà Nội**

Định nghĩa đệ quy

Với n = 1: Chuyển đĩa từ A → C

Với n>1:

Chuyển n-1 đĩa từ A → B

Chuyển đĩa cuối từ A → C

Chuyển n-1 đĩa từ B → C

2.3 Thiết kế giải thuật đệ quy

- **Bài toán tháp Hà Nội**

```
void  hanoi(int  n, char   A, char   C, char   B)
  {
      if  (n = = 1)
      {     cout<<"Chuyen 1 dia tu A sang C"<<endl;
            return;
      }
  hanoi(n-1, A, B, C);
  cout<<"Chuyen 1 dia tu A sang C"<<endl;
  hanoi(n-1, B, C, A);
  }
```

2.4 Phân loại đệ quy

2.4.1 Đệ quy tuyến tính

2.4.2 Đệ quy nhị phân

2.4.3 Đệ quy phi tuyến

2.4.4 Đệ quy tương hỗ

2.4.1 Đệ quy tuyến tính

Khái niệm:

Trong thân hàm có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.

Cú pháp:

```
<kiểu trả về>   <tên hàm> (<tham số>)
{
          if (<điều kiện dừng>)
         {
               …return <giá trị>;
          }
          …<tên hàm> (<đối số>);…
}
```

2.4.1 Đệ quy tuyến tính

- **Ví dụ:**

Viết hàm đệ qui tính n!

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

2.4.2 Đệ quy nhị phân

Khái niệm

Đệ quy nhị phân là loại đệ quy trong thân hàm có hai lời gọi lại chính nó một cách tường minh

Cú pháp:

```
<kiểu trả về>  <tên hàm> (<tham số>)
{
        if (<điều kiện dừng>)
        {
            …return <giá trị>;
        }
        …<tên hàm> (<đối số>);…
        …<tên hàm> (<đối số>);…
}
```

2.4.2 Đệ quy nhị phân

- **Ví dụ**

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

2.4.3 Đệ quy phi tuyến

Khái niệm:

- Hàm được gọi là đệ qui phi tuyến nếu bên trong thân hàm có lời gọi lại chính nó được đặt bên trong thân của vòng lặp.
- Thực chất đệ quy phi tuyến là việc tạo các vòng lặp for lồng nhau, trong đó những vòng lặp for càng ở sâu bên trong thì chịu sự ràng buộc về điều kiện do các vòng lặp for bên ngoài tạo nên.

2.4.3 Đệ quy phi tuyến

Cú pháp:

```
<kiểu trả về>   <tên hàm> (<tham số>)
{     if (<điều kiện dừng>)
            {
                …return <giá trị>;
            }
            …
            Vòng lặp
           {
           …<tên hàm> (<đối số>);
           }
           …
}
```

2.4.3 Đệ quy phi tuyến

Ví dụ:

$$x(n) = \begin{cases} 1 & \text{với } n=0 \\ n^2x(0)+(n-1)^2x(1)+\ldots+2^2x(n-2)+1^2x(n-1) & \text{với } n>0 \end{cases}$$

2.4.3 Đệ quy phi tuyến

```
longint  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;
}
```

2.4.4 Đệ quy tương hỗ

Khái niệm:

- Hàm được gọi là đệ qui tương hỗ nếu có hai hàm đệ quy, bên trong thân hàm này chứa lời gọi đến hàm kia và ngược lại.

2.4.4 Đệ quy tương hỗ

Cấu trúc hàm:

```
<kiểu trả về>    <tên hàm1> (<tham số>)
{     if (<điều kiện dừng>)
            {
                …return <giá trị>;
            }
            …
             …<tên hàm1> (<đối số>);
      …<tên hàm2> (<đối số>);
           …
}
```

2.4.4 Đệ quy tương hỗ

Cấu trúc hàm:

```
<kiểu trả về>    <tên hàm2> (<tham số>)
{     if (<điều kiện dừng>)
            {
                …return <giá trị>;
            }
            …
             …<tên hàm1> (<đối số>);
             …<tên hàm2> (<đối số>);
          …
}
```

2.4.4 Đệ quy tương hỗ

Ví dụ:

$$X(n) = \begin{cases} 0 & \text{với } n = 1 \\ 3*X(n-1) + Y(n-1) & \end{cases}$$

$$Y(n) = \begin{cases} 1 & \text{với } n = 1 \\ X(n-1) - 2*Y(n-1) & \end{cases}$$

Câu hỏi củng cố bài

1. Phương pháp định nghĩa bằng đệ qui là:

A. Phương pháp định nghĩa đối tượng thông qua chính nó.

B. Phương pháp xác định đối tượng thông qua chính nó.

C. Phương pháp xác định đối tượng thông qua các đối tượng khác.

D. Phương pháp định nghĩa đối tượng thông qua các đối tượng khác.

Câu hỏi củng cố bài

2. Khi xây dựng một hàm đệ qui ta phải:

A. Kiểm tra điều kiện ngừng đệ qui

B. Kiểm tra điều kiện gọi đệ qui

C. Kiểm tra điều kiện gọi hoặc điều kiện ngừng đệ qui

D. Kiểm tra bộ nhớ

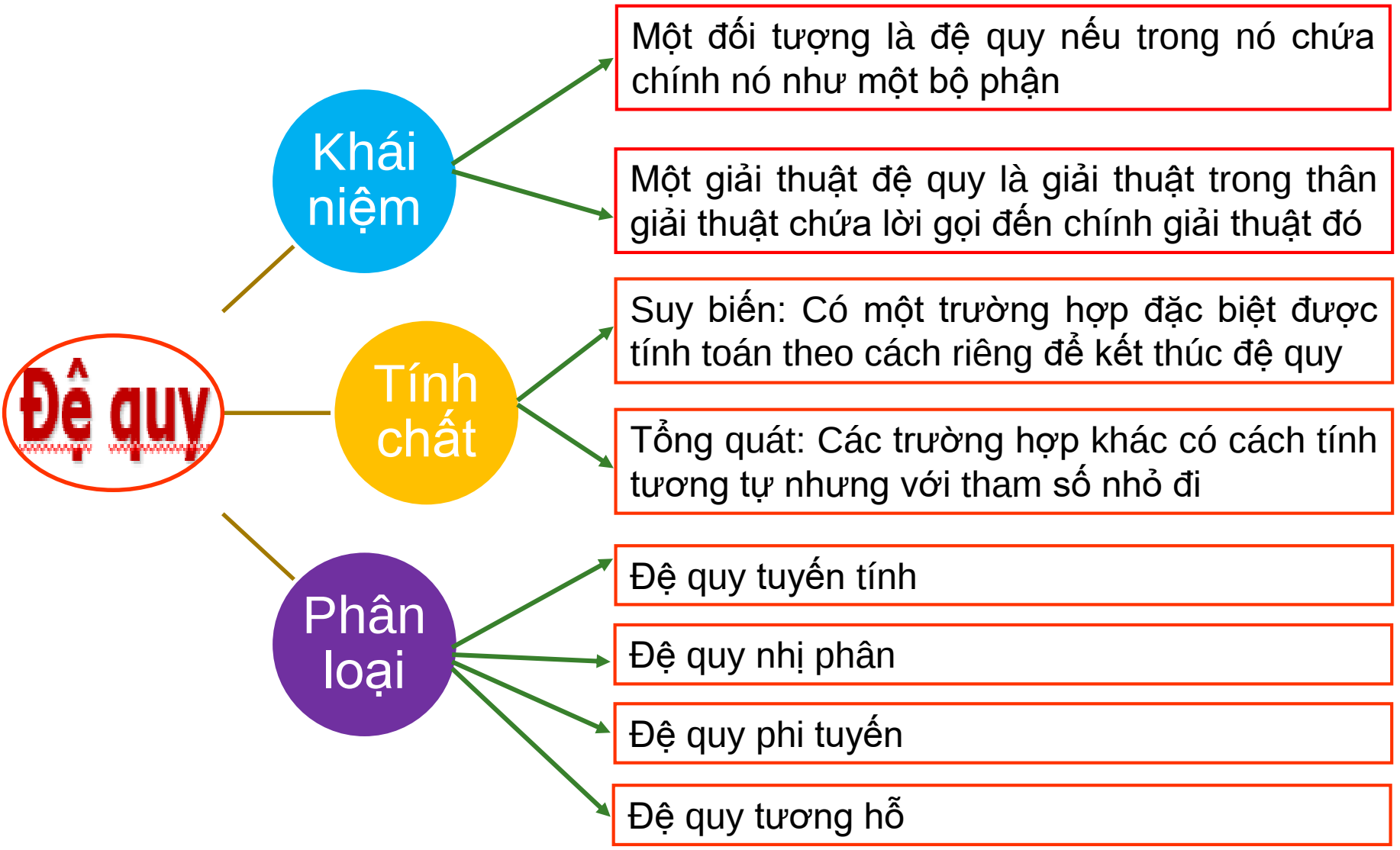
Câu hỏi củng cố bài

3. Tất cả các hàm đệ quy đều phải có một điều kiện dừng (hay trường hợp suy biến) đúng hay sai?

Câu hỏi củng cố bài

4. Khái niệm nào dùng để đo lường tính hiệu quả của một giải thuật

Tổng kết


Đệ quy
Khái niệm
Một đối tượng là đệ quy nếu trong nó chứa chính nó như một bộ phận
Một giải thuật đệ quy là giải thuật trong thân giải thuật chứa lời gọi đến chính giải thuật đó
Tính chất
Suy biến: Có một trường hợp đặc biệt được tính toán theo cách riêng để kết thúc đệ quy
Tổng quát: Các trường hợp khác có cách tính tương tự nhưng với tham số nhỏ đi
Phân loại
Đệ quy tuyến tính
Đệ quy nhị phân
Đệ quy phi tuyến
Đệ quy tương hỗ

Bài tập

1. Các hàm đệ qui sau tính cái gì?

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

Bài tập

1. Các hàm đệ qui sau tính cái gì?

```
b. float    f(float  x, int   n)
        {
          if (n == 0)
                        return      1;
        else
                return  x * f(x,n-1);
        }
```

Bài tập

1. Các hàm đệ qui sau tính cái gì?

c. int f(intn)

```
{
  if  (n < 2)
            return    0;
else
        return  1 + f(n/2);
}
```

Bài tập

2. Cho định nghĩa đệ quy

$$\text{Acker}(m,n) = \begin{cases} m+1 & \text{nếu } m = 0 \\ \text{Acker}(m-1,1) & \text{nếu } n=0 \\ \text{Acker}(m-1, \text{Acker}(m,n-1)) & \text{với các t/h khác} \end{cases}$$

a. Xác định Acker(1, 2)

b. Viết một giải thuật đệ quy tính giá trị hàm này

Bài tập

3. Giải thuật tính ước số chung lớn nhất của hai số nguyên dương p và q (p > q) được mô tả như sau:

Gọi r là số dư trong phép chia p cho q

- Nếu r = 0 thì q là USCLN
- Nếu r ≠ 0 thì gán cho p giá trị của q, gán cho q giá trị của r rồi lặp lại quá trình

a./ Xây dựng định nghĩa đệ qui cho hàm USCLN(p,q)

b./ Viết giải thuật đệ qui thể hiện hàm đó.

Bài tập

4. Viết một thủ tục đệ qui chuyển một số từ cơ số 10 sang cơ số 2

5. Hàm C(n,k) với n, k là các giá trị nguyên không âm và k<n được định nghĩa

C(n,n) = 1

C(n,0) = 1

C(n,k) = C(n-1,k-1) + C(n-1,k) nếu 0 < k < n

Viết một giải thuật đệ qui tính giá trị C(n,k) khi biết n, k

6. Viết một giải thuật đệ quy in ngược một dòng cho trước ví dụ dòng “PASCAL” thì in thành “LACSAP”

Tài liệu tham khảo

[1]. Giáo trình Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh, NXB Đại học quốc gia TP HCM, 2013

[2]. Cấu trúc dữ liệu & thuật toán, Đỗ Xuân Lôi, NXB Đại học quốc gia Hà Nội, 2010.

[3]. Trần Thông Quế, *Cấu trúc dữ liệu và thuật toán (phân tích và cài đặt trên C/C++)*, NXB Thông tin và truyền thông, 2018

[4]. Robert Sedgewick, *Cẩm nang thuật toán*, NXB Khoa học kỹ thuật, 2004.

[5]. PGS.TS Hoàng Nghĩa Tý, *Cấu trúc dữ liệu và thuật toán*, NXB xây dựng, 2014