CHƯƠNG 2

ĐỆ QUY VÀ GIẢI THUẬT ĐỆ QUY

Nội dung

- Khái niệm đệ quy
- Giải thuật đệ quy và hàm đệ quy
- Thiết kế giải thuật đệ quy
- Hiệu lực của đệ quy
- Bài tập

1.1. Khái niệm đệ quy

- Ta nói một đối tượng là đệ quy nếu nó bao gồm chính nó như một bộ phận hoặc nó được định nghĩa dưới dạng của chính nó.
- Ví dụ: Trong toán học ta gặp các định nghĩa đệ quy sau:
  - *Số tự nhiên:*
    - 1 là số tự nhiên.
    - n là số tự nhiên nếu n-1 là số tự nhiên.
  - *Hàm n giai thừa: n!*
    - 0! = 1
    - Nếu n>0 thì n! = n(n-1)!

1.2. Giải thuật đệ quy và hàm đệ quy

1.2.1. Giải thuật đệ quy

- Nếu lời giải của của bài toán T được giải bằng lời giải của một bài toán T1, có dạng giống như T, thì lời giải đó được gọi là lời giải đệ quy.
- Giải thuật tương ứng với lời giải đệ quy gọi là giải thuật đệ quy.
- Ở đây T1 có dạng giống T nhưng theo một nghĩa nào đó T1 phải "nhỏ" hơn T.
- Chẳng hạn, với bài toán tính n!, thì tính n! là bài toán T còn tính (n-1)! là bài toán T1 ta thấy T1 cùng dạng với T nhưng nhỏ hơn (n-1 < n).

Giải thuật đệ quy (tt)

- Giải thuật của bài toán tìm từ trong từ điển

```
if (từ điển là một trang)
        tìm từ trong trang này
else
{       Mở từ điển vào trang "giữa";
        Xác định xem nửa nào của từ điển chứa từ cần tìm;
        if (từ đó nằm ở nửa trước)
                tìm từ đó ở nửa trước;
        else    tìm từ đó ở nửa sau;
}
```

- Giải thuật này được gọi là giải thuật đệ quy

Giải thuật đệ quy (tt)

- Nhận xét:
  - Sau mỗi lần từ điển được tách làm đôi thì một nửa thích hợp sẽ lại được tìm bằng một chiến thuật như đã dùng trước đó (nửa này lại được tách đôi).
  - Có một trường hợp đặc biệt, đó là sau nhiều lần tách đôi, từ điển chỉ còn một trang. Khi đó việc tách đôi ngừng lại và bài toán trở thành đủ nhỏ để ta có thể tìm từ mong muốn bằng cách tìm tuần tự. Trường hợp này gọi là *trường hợp suy biến*.

1.2.2. Hàm đệ quy

- Hàm đệ quy được cài đặt khi giải thuật được thiết kế là giải thuật đệ quy.

```
SEARCH (dict, word)          //Tìm từ 'word' trong từ điển 'dict'
{
    if (Từ điển chỉ còn là một trang)
            tìm từ word trong trang này
    else
    {       mở từ điển vào trang giữa
            xác định xem nửa nào của từ điển chứa từ word
            if (từ word nằm ở nửa sau của từ điển)
                    return SEARCH(dict\{nửa trước}, word);
            else    return SEARCH(dict\{nửa sau}, word);
    }
}
```


Hàm đệ quy (tt)

- **Đặc điểm của hàm đệ quy:**
  - Trong hàm đệ quy có lời gọi đến chính nó. Trong hàm SEARCH có lệnh: *return* SEARCH(dict\{nửa trước}, word);
  - Sau mỗi lần có lời gọi đệ quy thì kích thước của bài toán được thu nhỏ hơn trước.
  - Trường hợp suy biến là khi lời gọi hàm SEARCH với từ điển dict chỉ còn là một trang. Trường hợp này bài toán còn lại sẽ được giải quyết theo một cách khác hẳn (tìm từ word trong trang đó bằng cách tìm kiếm tuần tự) và việc gọi đệ quy cũng kết thúc.

1.3. Thiết kế giải thuật đệ quy

- Khi bài toán đang xét, hoặc dữ liệu đang xử lý được định nghĩa dưới dạng đệ quy, thì việc thiết kế các giải thuật đệ quy tỏ ra rất thuận lợi.
- Giải thuật đệ quy phản ánh rất sát nội dung của định nghĩa đó.
- Không có giải thuật đệ quy vạn năng cho tất cả các bài toán đệ quy, nghĩa là mỗi bài toán cần thiết kế một giải thuật đệ quy riêng.

1.3.1. Bài toán n!

- Bài toán được định nghĩa như sau:

  Nếu n=0 -> n! = 1

  Nếu n>0 -> n! = n(n-1)!*
- Giải thuật đệ quy được viết dưới dạng hàm

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

- Trong hàm trên lời gọi đến nó nằm ở câu lệnh gán sau *else*.
- Mỗi lần gọi đệ quy đến Factorial, thì giá trị của n giảm đi 1.
- Ví dụ, Factorial(4) gọi đến Factorial(3), gọi đến Factorial(2), gọi đến Factorial(1), gọi đến Factorial(0) đây là trường hợp suy biến, nó được tính theo cách đặc biệt Factorial(0) = 1.

1.3.2. Bài toán dãy số Fibonaci

- Dãy số Fibonacci bắt nguồn từ bài toán cổ về việc sinh sản của các cặp thỏ. Bài toán được đặt ra như sau:
  - *Các con thỏ không bao giờ chết.*
  - *Hai tháng sau khi được sinh ra một cặp thỏ mới sẽ sinh ra một cặp thỏ con.*
  - *Khi đã sinh sản, thì cứ sau mỗi tháng chúng lại sinh được một cặp con mới.*
- Giả sử bắt đầu từ một cặp thỏ mới được sinh ra, hỏi đến tháng thứ n sẽ có bao nhiêu cặp?

Bài toán dãy số Fibonaci

- Tính trực tiếp số cặp thỏ trong các tháng đầu tiên

  Chẳng hạn với n=6, ta tính được
  - Tháng thứ 1: 1 cặp *(cặp ban đầu)*
  - Tháng thứ 2: 1 cặp *(cặp ban đầu vẫn chưa sinh con)*
  - Tháng thứ 3: 2 cặp *(đã có thêm 1 cặp con do cặp ban đầu đẻ sinh ra)*
  - Tháng thứ 4: 3 cặp (*cặp ban đầu vẫn sinh thêm*)
  - Tháng thứ 5: 5 cặp (*cặp con bắt đầu sinh*)
  - Tháng thứ 6: 8 cặp (*cặp con vẫn sinh tiếp*)

Bài toán dãy số Fibonaci (tt)

- Suy ra định nghĩa cách tính số cặp thỏ từ việc tính trực tiếp:
  - Đặt F(n) là số cặp thỏ ở tháng thứ n.
  - Ta thấy ở tháng thứ 1 (n=1), và tháng thứ 2 (n=2) luôn có 1 cặp
  - Chỉ những cặp thỏ đã có ở tháng thứ n-2 mới sinh con ở tháng thứ n, nên số cặp thỏ ở tháng thứ n là:

    F(n) = F(n-2) + F(n-1).

Bài toán dãy số Fibonaci (tt)

- Vì vậy F(n) được định nghĩa dạng ĐỆ QUI như sau:

$$F(n) = \begin{cases} 1 \text{ nếu } n=1 \text{ hoặc } n=2 \\ F(n-2) + F(n-1) \text{ nếu } n>2 \end{cases}$$

Dãy số F(n) ứng với các giá trị của n = 1, 2, 3, 4, 5, 6, 7, 8..., có dạng: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55... được gọi là dãy số Fibonacci.

Bài toán dãy số Fibonaci (tt)

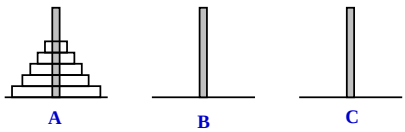
- Giải thuật đệ quy thể hiện việc tính F(n) được biểu diễn dưới dạng hàm.

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

- Ở đây trường hợp suy biến ứng với 2 giá trị F(1) = 1 và F(2) = 1.

1.3.3. Bài toán Tháp Hà Nội

- Bài toán này mang tính chất là một trò chơi, nội dung như sau:
  - Có n đĩa, kích thước nhỏ dần, mỗi đĩa có lỗ ở giữa.
  - Có thể xếp chồng chúng lên nhau xuyên qua một cọc, đĩa to ở dưới, đĩa nhỏ ở trên để cuối cùng có một chồng đĩa

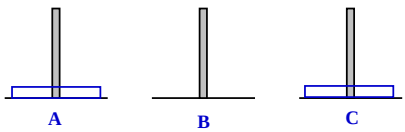


Bài toán Tháp Hà Nội (tt)

- Yêu cầu đặt ra là:
  - Chuyển chồng đĩa từ cọc A sang cọc khác, chẳng hạn cọc C, theo những điều kiện:
    - Mỗi lần chỉ được chuyển một đĩa.
    - Không khi nào có tình huống đĩa to ở trên đĩa nhỏ (dù là tạm thời).
    - Được phép sử dụng một cọc trung gian, chẳng hạn cọc B để đặt tạm đĩa.

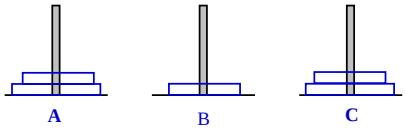
Bài toán Tháp Hà Nội (tt)

- Để đi tới cách giải tổng quát, trước hết ta xét vài trường hợp đơn giản.
  - *Trường hợp có 1 đĩa:*
    - Chuyển 1 đĩa từ cọc A sang cọc C.



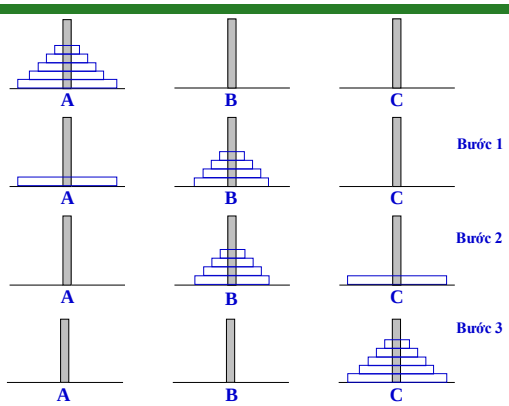
Bài toán Tháp Hà Nội (tt)

- **Trường hợp 2 đĩa:**
  - Chuyển đĩa thứ nhất từ cọc A sang cọc B.
  - Chuyển đĩa thứ hai từ cọc A sang cọc C.
  - Chuyển đĩa thứ nhất từ cọc B sang cọc C.



Bài toán Tháp Hà Nội (tt)

- **Tổng quát với n đĩa (n>2)**
  - Ta thấy với trường hợp n đĩa (n>2) nếu coi n-1 đĩa ở trên, đóng vai trò như đĩa thứ nhất thì có thể xử lý giống như trường hợp 2 đĩa được, nghĩa là:
    - Chuyển n-1 đĩa (phía trên) từ cọc A sang cọc B.
    - Chuyển 1 đĩa (dưới cùng) từ cọc A sang cọc C.
    - Chuyển n-1 đĩa từ cọc B sang cọc C.



Bài toán Tháp Hà Nội (tt)

- **Nhận xét:**
  - Bài toán *"Tháp Hà Nội"* với n đĩa đã được dẫn đến bài toán tương tự với kích thước nhỏ hơn, chẳng hạn từ chỗ chuyển n đĩa từ cọc A sang cọc C nay là chuyển n-1 đĩa từ cọc A sang cọc B và ở mức này thì giải thuật lại là:
    - Chuyển n-2 đĩa từ cọc A sang cọc C.
    - Chuyển 1 đĩa từ cọc A sang cọc B.
    - Chuyển n-2 đĩa từ cọc C sang cọc B.
  - và cứ như thế cho tới khi trường hợp suy biến xảy ra, đó là trường hợp ứng với bài toán chuyển 1 đĩa (n=1).

Bài toán Tháp Hà Nội (tt)

- **Giải thuật đệ quy**

```
//Chuyển n đĩa từ cọc A sang cọc C với B là cọc trung gian
void Chuyen(n, A, B, C)
{
    if (n==1)
        Chuyển 1 đĩa từ A sang C;
    else
    {
        Chuyen(n-1, A, C, B);
        Chuyen(1, A, B, C);
        Chuyen(n-1, B, A, C);
    }
}
```


1.4. Hiệu lực của đệ quy

- **Đệ quy là một kỹ thuật giải quyết bài toán khá hữu dụng.**
- **Việc thiết kế giải thuật cũng đơn giản vì nó khá giống với định nghĩa lời giải bài toán.**
- **Tuy nhiên:**
  - Sử dụng đệ quy rất tốn bộ nhớ và thời gian
  - Nên sử dụng giải thuật lặp thay thế nếu được (khử đệ quy)
- **Vẫn có những bài toán sử dụng đệ quy khá hữu ích: Giải thuật sắp xếp Quick Sort, Các phép duyệt cây…**

1.5. Bài tập

Bài 1:

- Xét định nghĩa đệ qui:

$$\text{Acker}(m,n) = \begin{cases} n+1 \text{ nếu } m = 0 \\ \text{Acker}(m-1,1) \text{ nếu } n = 0 \\ \text{Acker}(m-1, \text{Acker}(m, n-1)) \text{ còn lại} \end{cases}$$

- Hãy xác định Acker(1, 2)
- Viết hàm đệ qui thực hiện tính giá trị của hàm này.

Bài tập

Bài 2:

- Xét định nghĩa đệ qui:

$$F(x) = \begin{cases} \cos(x) \text{ nếu } x = 0 \\ x \text{ nếu } x < 0 \\ F(x-\pi) + F\left(x - \frac{\pi}{2}\right) \text{ còn lại} \end{cases}$$

- Hãy xác định F(3π/2)
- Viết hàm đệ qui thực hiện tính giá trị của hàm này.

Bài tập

Bài 3:

- Giải thuật tính ước số chung lớn nhất của hai số nguyên dương p, q (p>q) được cho như sau:
  - Gọi r là số dư trong phép chia p cho q.
  - Nếu r = 0 thì ước số chung lớn nhất là q
  - Nếu r ≠ 0 thì gán cho p giá trị q, gán cho q giá trị của r và lặp lại quá trình.

a) Hãy xây dựng định nghĩa đệ qui cho hàm USCLN(p, q).

b) Viết một giải thuật đệ qui và một giải thuật lặp thể hiện hàm đó.

Thank you…!