ĐỆ QUI (RECURSION)

- Khái niệm, định nghĩa
- Thiết kế giải thuật đệ qui
- Một số ví dụ
- Bài tập

Khái niệm

- Định nghĩa đệ qui:
  - Nó hoặc 1 phần của nó được định nghĩa qua chính nó
- Định nghĩa số tự nhiên:
  - 0 là số tự nhiên.
  - Nếu k là số tự nhiên thì k+1 cũng là số tự nhiên
- Định nghĩa xâu ký tự bằng đệ qui:
  - Xâu rỗng là 1 xâu ký tự.
  - Một chữ cái bất kỳ ghép với 1 xâu sẽ tạo thành 1 xâu mới.
- Định nghĩa hàm giai thừa, n!.
  - Khi n=0, định nghĩa 0!=1
  - Khi n>0, định nghĩa n!=(n-1)! x n

Khái niệm

- Chương trình đệ qui:
  - Có lời gọi chính nó
- Đặc điểm chương trình đệ qui:
  - Có thể gọi chính nó.
  - Giải quyết vấn đề tương tự nhưng nhỏ hơn (tham số nhỏ hơn)
  - Chương trình được gọi nhỏ dần đến khi tự giải quyết đc mà ko cần gọi chính nó nữa
- Ưu điểm chương trình đệ qui:
  - Thực hiện một số lượng lớn các thao tác tính toán thông qua 1 đoạn chương trình ngắn gọn

Điều kiện chương trình đệ qui

- Vấn đề cần xử lý phải được giải quyết 1 cách đệ qui
- Ngôn ngữ dùng để viết chương trình phải hỗ trợ đệ qui (hỗ trợ hàm/thủ tục)
- Hạn chế khai báo biến, hằng, thủ tục con trong hàm đệ qui
- Đệ qui trực tiếp
  - Gọi chính nó
- Đệ qui gián tiếp
  - Gọi chính nó thông qua một thủ tục trung gian

Khi nào không nên dùng đệ qui

- Khi chương trình có thể xử lý bằng các cấu trúc lệnh khác (lặp) mà không quá phức tạp.
- Khi gọi hàm đệ qui không đảm bảo tham số sẽ nhỏ đi.
  - n chẵn: =n/2
  - n lẻ: = 3*n+1
- Khi gọi hàm đệ qui có thể tạo các tính toán thừa.
  - Ví dụ: Tính dãy Fibonaci
  - F(0) = 0
  - F(1) =1
  - Với n >1 thì F(n) = F(n-1) + F(n-2)

Khi nào không nên dùng đệ qui

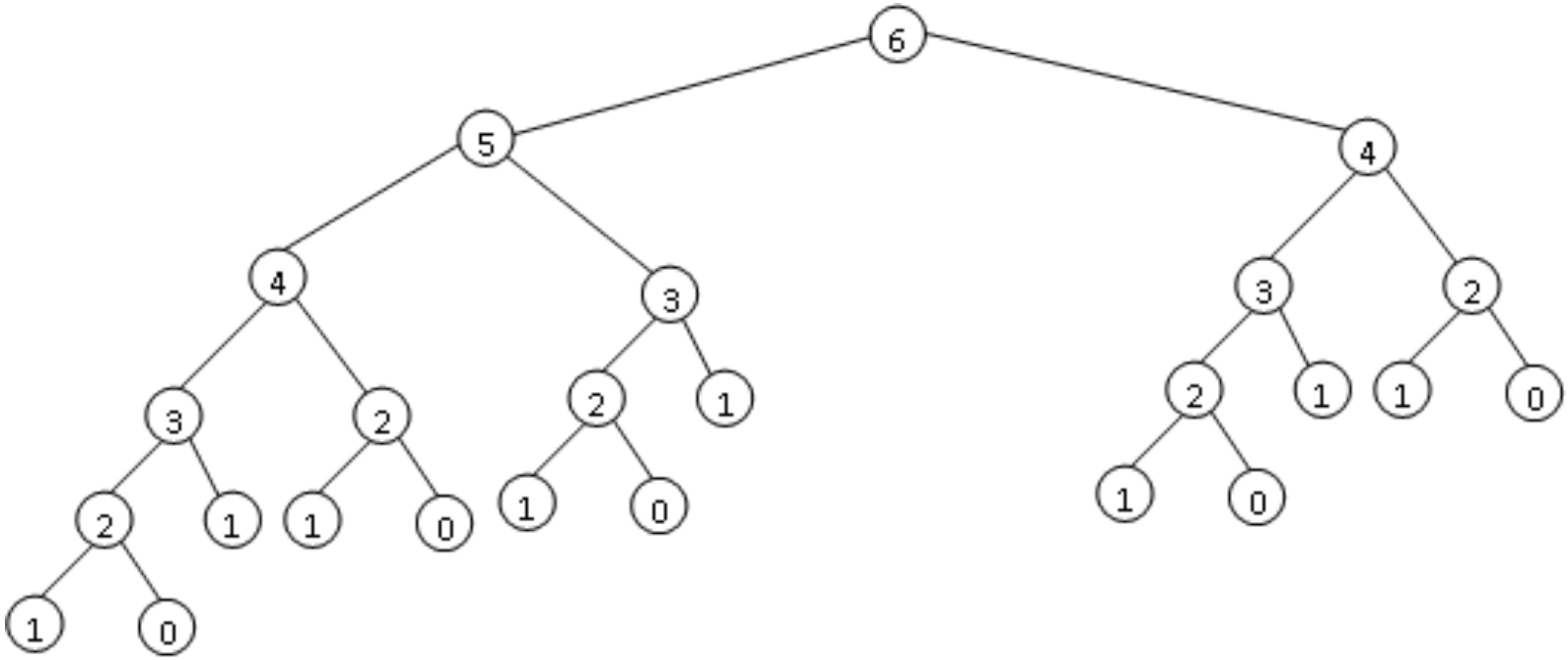
- Hàm đệ qui tính dãy fibonaci

```
int Fibonaci(int i){
          if (i==0) return 0;
          if (i==1) return 1;
          return Fibonaci(i-1) + Fibonaci (i-2)
}
```

  - Lời gọi F(n)-> F(n-1) và F(n-2)
  - Lời gọi F(n-1) -> F(n-2) và F(n-3)
  - Lời gọi F(n-2) -> F(n-3) và F(n-4)
  - …

Khi nào không nên dùng đệ qui

- Số lời gọi đệ qui khi gọi Fibonaci(6)



Khi nào không nên dùng đệ qui

- Tính dãy fibonaci bằng vòng lặp

```
F[0]=0;
F[1]=1;
for (i=2; i<n-1; i++)
          F[i] = F[i-1] + F [i-2];
```

- Chỉ sử dụng khi đảm bảo ko có tính toán thừa
- Một số bài toán rất phù hợp với đệ qui (dễ hiểu)
- Các giải thuật đệ qui đều có thể chuyển về lặp
  - Có thể làm chương trình phức tạp hơn nhiều

Ví dụ 1

- Tính giá trị n!:
  - n=0: n! = 1
  - n>0: n! = (n-1)! * n
- Hàm đệ qui tính n!:

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

  - Điểm dừng thuật toán khi n=0
  - Điểm dừng không thỏa mãn: gọi hàm với tham số n-1

Ví dụ 2

- Thuật toán Euclid tính USCLN của (m,n)
  - Đơn giản:
    - Duyệt từ min(m,n)->1, nếu chia hết số nào thì đó là USCLN(m,n)
  - Thuật toán Euclid hiệu quả hơn:
    - USCLN(m, n) = USCLN(m mod n, n) với m>n
  - Hàm đệ qui tính USCLN(m,n)

```
int USCLN(int m, int n){
        if (n==0) return m;
        else return USCLN(n, m % n);
}
```

Ví dụ 2

- Thuật toán Euclid tính USCLN của (m,n)
  - Điểm dừng thuật toán khi n=0: USCLN(m,0)=m
  - Khi n>0, gọi đệ qui USCLN(n,m%n)
  - Sau mỗi lần gọi các tham số sẽ nhỏ dần đi: n<m, m%n<n
  - Sau 1 số hữu hạn lời gọi tham số sẽ = 0 ->dừng
    - USCLN(108, 45) 108 chia 45 dư 18, do đó tiếp theo gọi
    - USCLN(45, 18) 45 chia 18 dư 9, do đó tiếp theo gọi
    - USCLN(18, 9) 18 chia 9 dư 0, do đó tiếp theo gọi
    - USCLN(9, 0) tham số thứ 2 = 0, do đó kết quả là tham số thứ nhất, tức là 9.

Bài tập

- Viết chương trình bằng C/C++ thực hiện các thuật toán đệ qui đã trình bày.
- Sử dụng đệ qui để xác định số chữ số của 1 số tự nhiên (bằng cách chia liên tục cho 10).
  - ✓ Phác thảo thuật toán đệ qui cho vấn đề
  - ✓ Viết hàm đệ qui giải quyết bài toán trên.
- Một chuỗi ký tự được gọi là chuỗi palindrome nếu nó đọc xuôi hoặc ngược đều như nhau (ví dụ madam).
  - ✓ Phác thảo thuật toán đệ qui cho vấn đề
  - ✓ Viết hàm đệ qui kiểm tra 1 chuỗi có phải là palindrome không?