第5讲 从两种不同的递归函数看规模化计算的规则表达与执行--递归与迭代

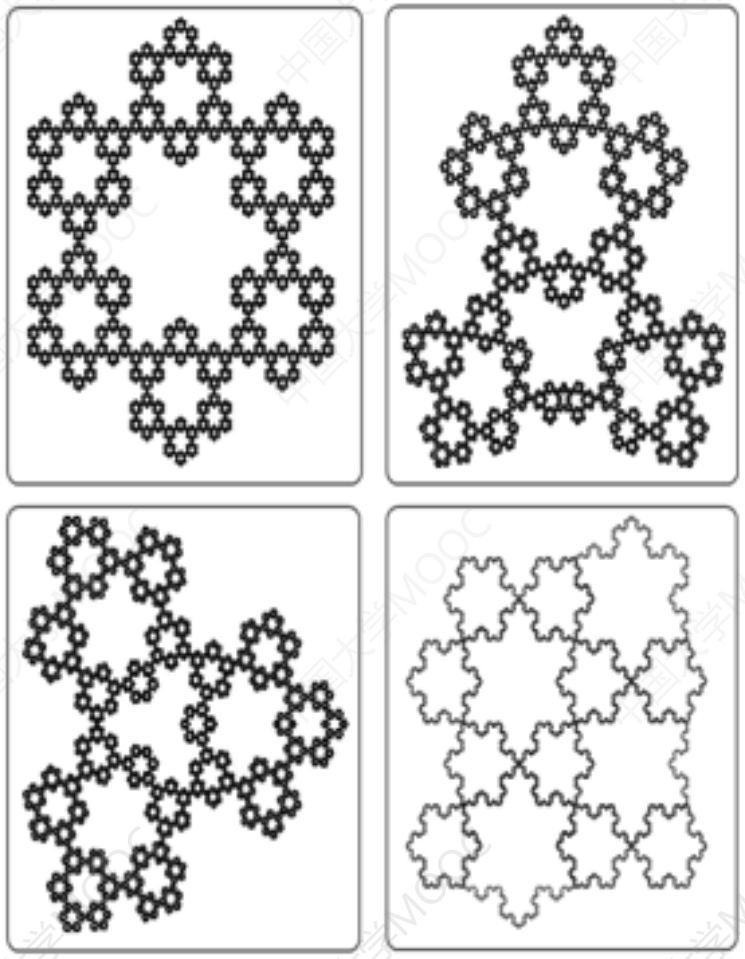
战德臣

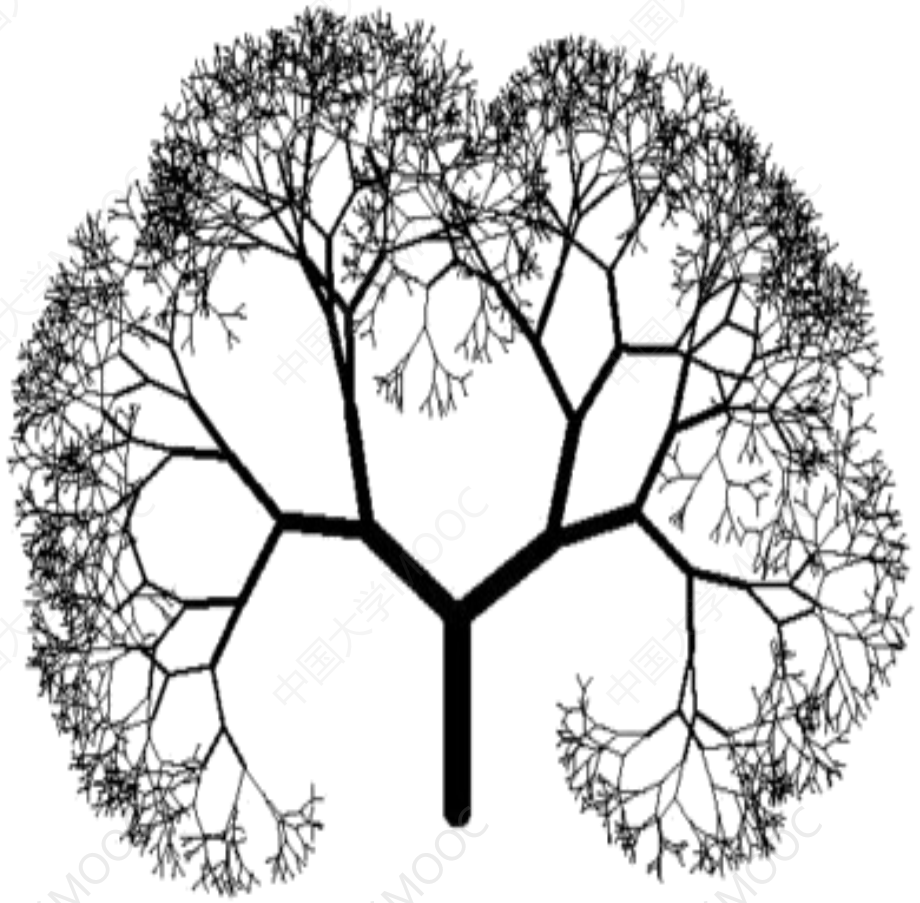
哈尔滨工业大学计算学部教学委员会主任
国家教学名师

18686783018, dechen@hit.edu.cn

递归的概念

递归的典型视觉形式---自相似性事物的无限重复性构造



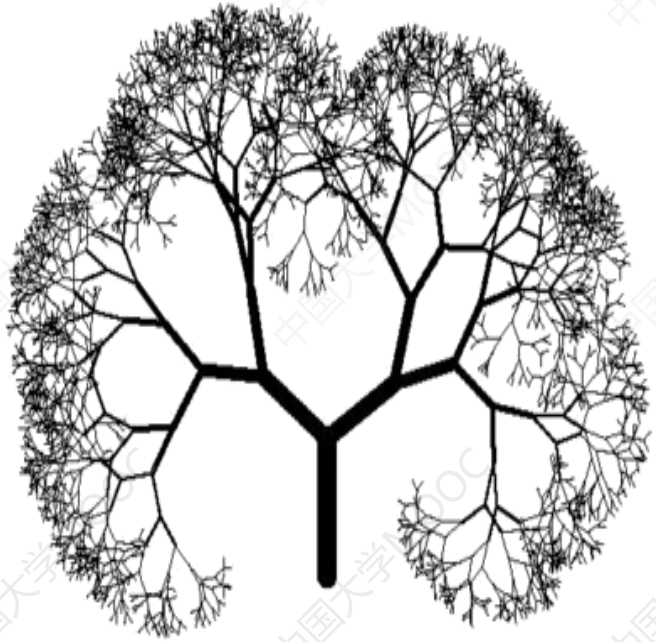


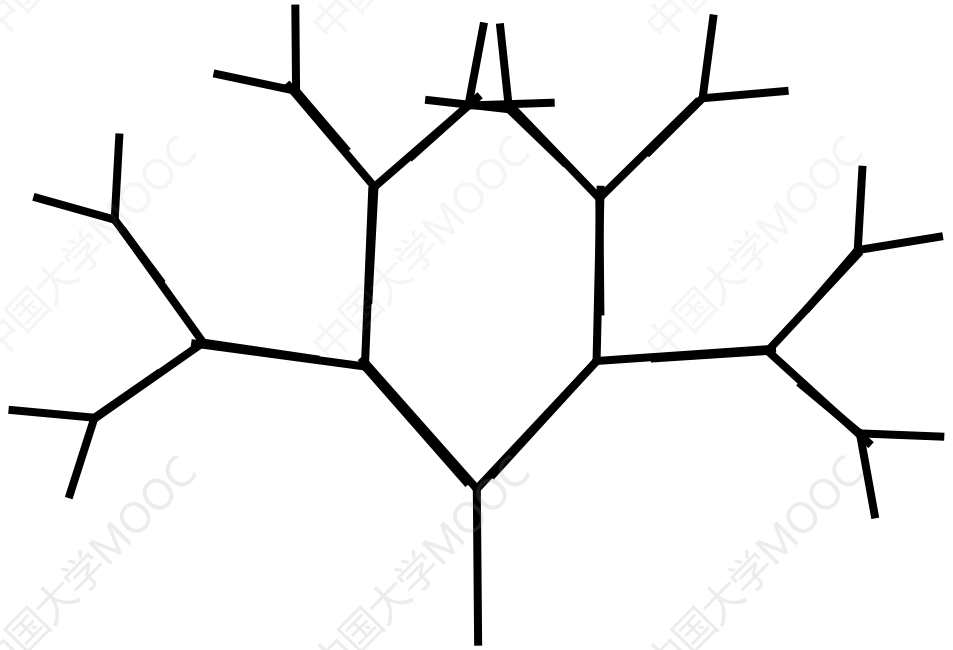
递归的概念

什么是递归?

递归是表达相似性对象及动作的无限性重复性构造的方法。

- **递归基础**：定义、构造和计算的起点，直接给出。即h(0)。
- **递归步骤**：由**第n项**或前n项**定义第n+1项**。即: **h(n+1) = g(h(n),n)**，g是需要明确给出的，以说明h(n+1)怎样由h(n)和n构造出来。



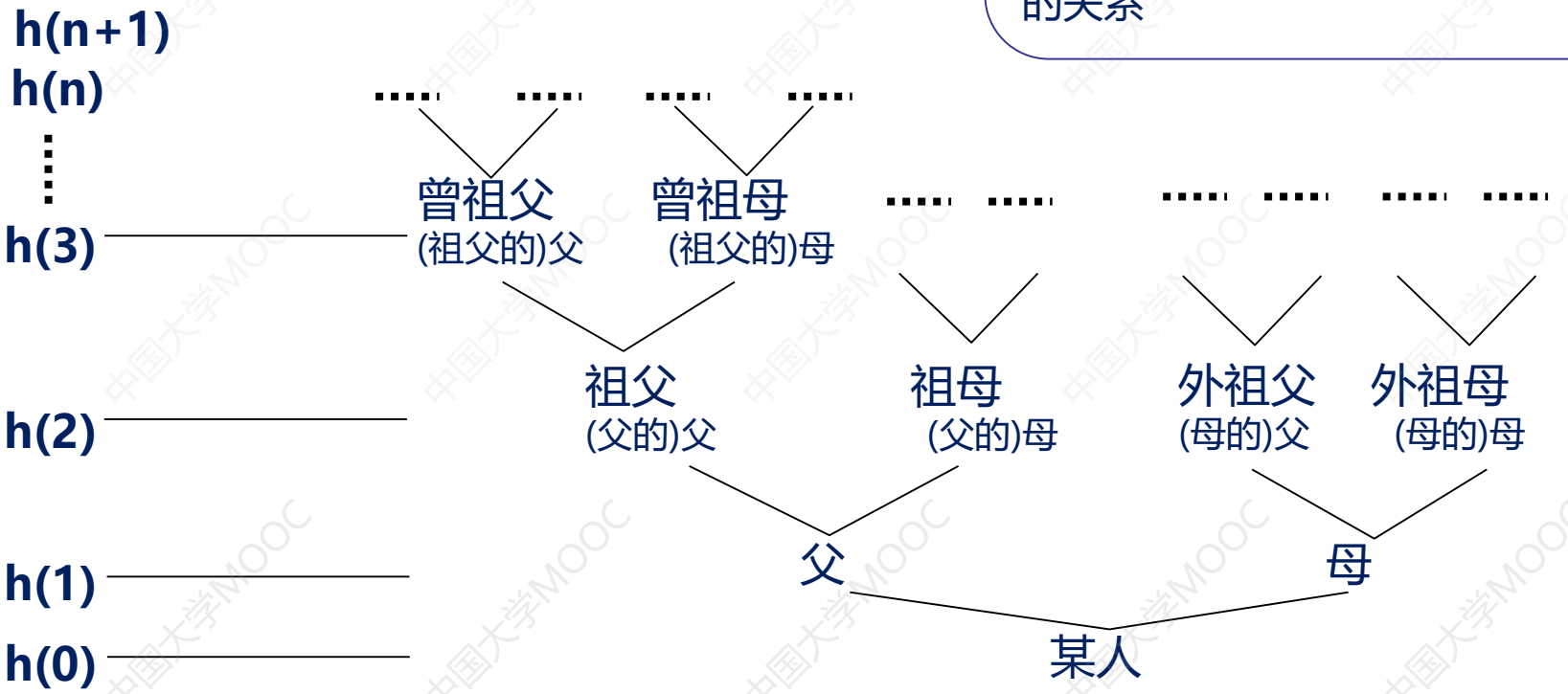


用递归进行定义

怎样递归?

【示例】怎样定义你的祖先?

(1)某人的双亲(父母)是他的祖先（递归基础）
--- h(1)直接给出
(2)某人祖先的双亲(父母)同样是某人的祖先（递归步骤/公式）
--- h(n+1) = g(h(n) , n)
--- n: 第几代；h(n)是第几代祖先；g是h(n+1)与h(n)和n的关系



递归的理解比较难

递归思维是计算机最重要的特征

想学计算机就要理解递归思维

体验两种不同的递归函数

一种类型的递归函数：定义是递归的，但执行可以是递归的也可以是迭代的

- Fibonacci数列，无穷数列1，1，2，3，5，8，13，21，34，55，……，

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

递归定义

计算规则的表达—告诉执行者要递归计算

F(0)=1;

F(1)=1;

F(2)=F(1)+F(0)=2;

F(3)=F(2)+F(1)= 3;

F(4)=F(3)+F(2)= 3+2=5;… …

递推计算/迭代计算/迭代执行

执行者要一步一步地完成计算获得结果

体验两种不同的递归函数

另一种类型的递归函数：定义是递归的，但执行也是递归的（用迭代不容易完成）

- **阿克曼递归函数**

$$A(m,n)=\begin{cases} n+1 & 若m=0 \\ A((m-1),1) & 若n=0 \\ A(m-1,A(m,n-1)) & 若m,n>0 \end{cases}$$

递归定义

函数本身是递归的，
函数的变量也是递归的

计算规则的表达—告诉执行者要递归计算

A(1,n) =A(0, A(1,n-1))=A(0. <...代入前式计算过程>)=A(0,n+1)=n+2。

A(2,1) = A(1， A(2， 0)) --A(2,1)按m,n>0公式代入
= A(1， A(1， 1)) --A(2,0)按n=0公式代入
=A(1， A(0， A(1， 0))) --A(1,1)按m,n>0公式代入
= A(1， A(0， A(0， 1))) --A(1,0)按n=0公式代入
= A(1， A(0， 2)) --A(0,1)按m=0公式代入
= A(1， 3) --A(0,2)按m=0公式代入
=A(0， A(1， 2)) --A(1,3)按m,n>0公式代入
= A(0， A(0， A(1， 1))) --A(1,2)按m,n>0公式代入
= A(0， A(0， A(0， A(1， 0)))) --A(1,1)按m,n>0公式代入
= A(0， A(0， A(0， A(0,1))) --A(1,0)按n=0公式代入
= A(0， A(0， A(0， 2))) --A(0,1)按m=0公式代入
= A(0， A(0， 3)) = A(0， 4) = 5。 --依次按m=0公式代入

执行者要一步一步地完成计算获得结果

递归计算/递归执行

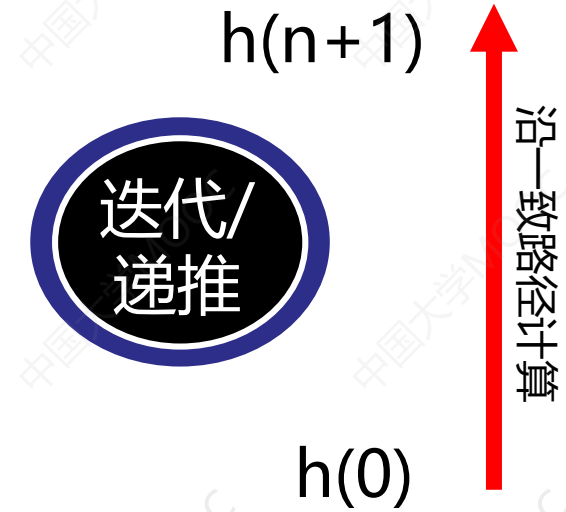
由后向前代入，
再由前向后计算

两种不同的递归函数：递归与迭代

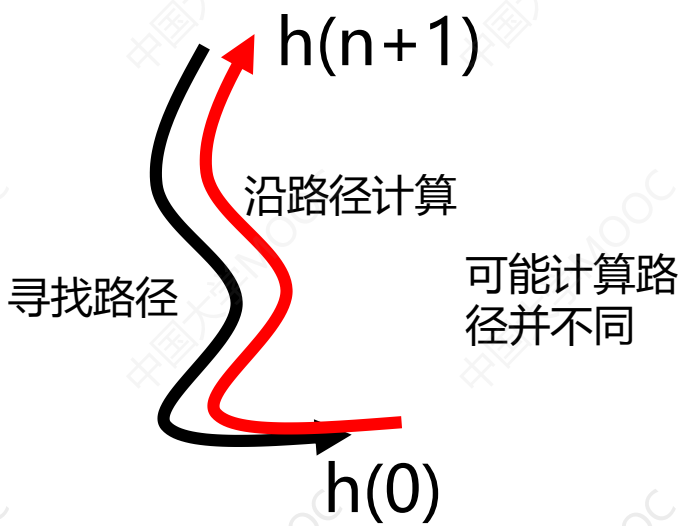
递归与迭代的差异

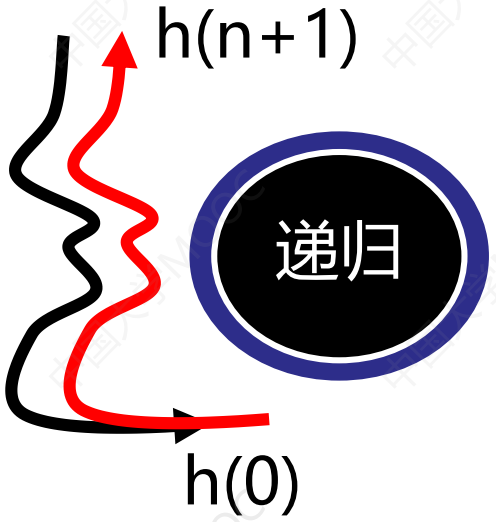
$$h(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ h(n-1)+h(n-2) & n>1\end{cases}$$

$$h(m,n)=\begin{cases}n+1 & \text{若}m=0\\ h((m-1),1) & \text{若}n=0\\ h(m-1,h(m,n-1)) & \text{若}m,n>0\end{cases}$$



在前次结果基础上进行计算
可采用**循环**结构实现





通常，只能由**函数**结构实现

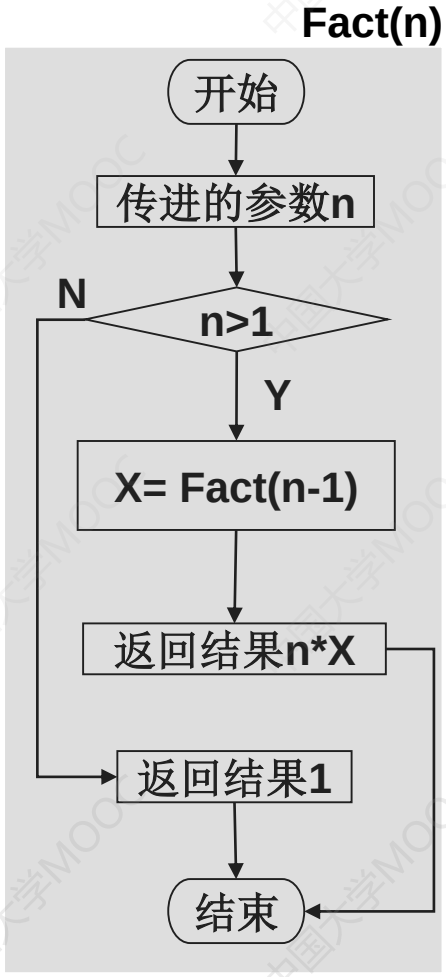
递归程序的构造与执行

运用递归进行程序构造：函数结构，自身调用自身，高阶调用低阶

【示例】求n!的算法或程序--用递归方法构造：函数调用

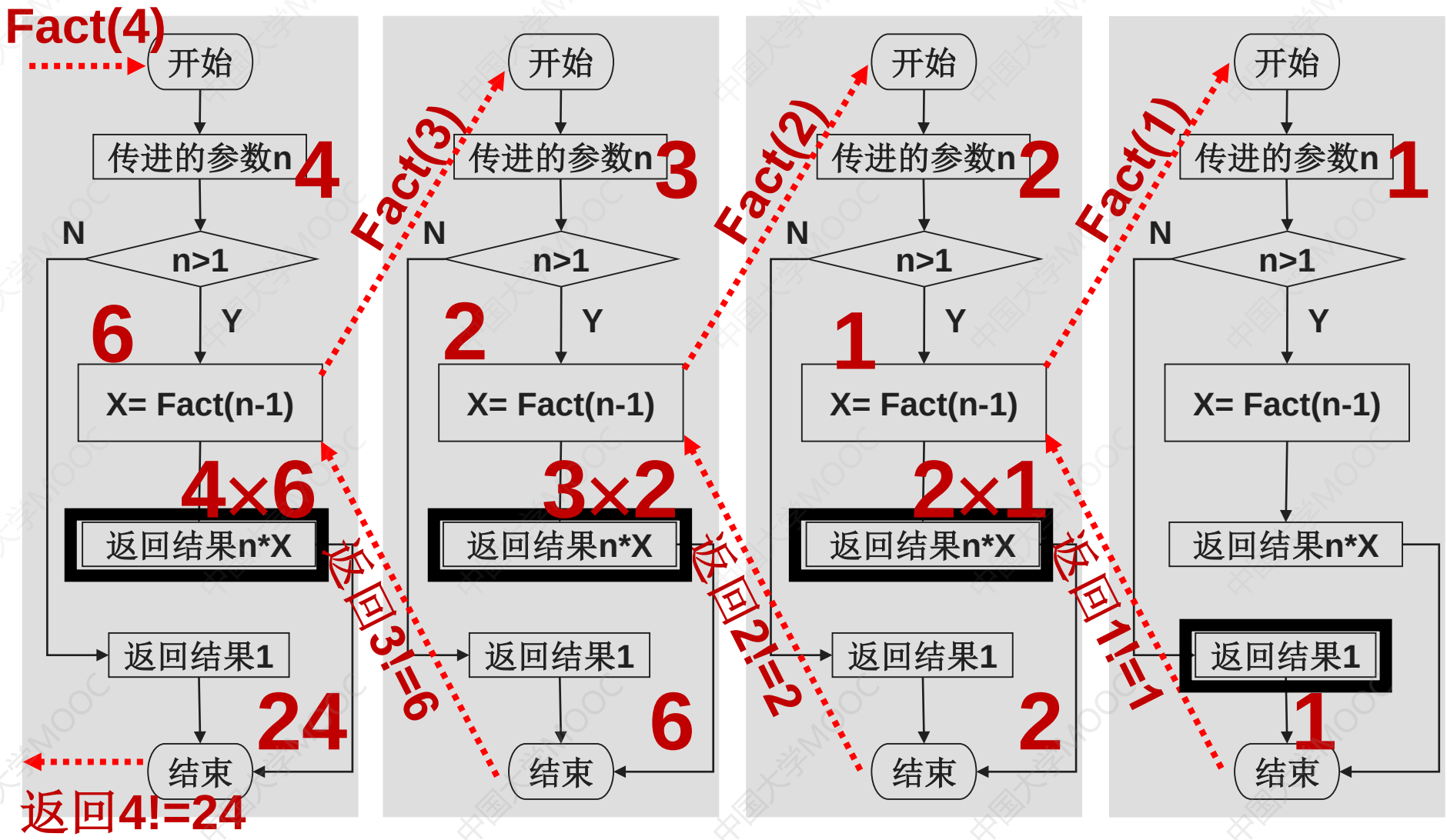
$$n\,! = \begin{cases} 1 & \text{当} n \le 1 \text{ 时} \\ n \times (n-1)! & \text{当} n > 1 \text{ 时} \end{cases}$$

```
long int Fact(int n)
{    long int x;
     If (n > 1)
     { x = Fact(n-1);
  /*递归调用*/
        return n*x;   }
     else return 1;
  /*递归基础*/
}
```



递归程序的构造与执行

递归程序的执行过程：利用【堆栈】保存待计算的数据



1	Fact(4)的程序断点
2	Fact(3)的程序断点
3	Fact(4)的程序断点
4	
5	
6	

堆栈：后进先出

迭代程序的构造与执行

迭代程序的执行过程?

【示例】求n!的算法或程序--用迭代方法构造：循环-替代-递推

$$n!=\begin{cases}1 & \text{当} n\le 1\text{时}\\ 1\times 2\times\ldots(n-1)\times n & \text{当} n>1\text{时}\end{cases}$$

```
long int Fact(int n)
{   int  counter；
    long product=1;
    for  counter=1 to n step 1
    { product = product * counter；  }
  /*迭代*/
return product;
}
```

Fact(5)的执行过程

	Counter	Product
初始值		1
循环第1次	1	1
循环第2次	2	2
循环第3次	3	6
循环第4次	4	24
循环第5次	5	120
退出循环	6	120

递归与迭代

小结

递归是表达相似性对象及动作的无限性重复性构造的方法。

■递归基础：定义、构造和计算的起点，直接给出。即h(0)。

■递归步骤：由第n项或前n项定义第n+1项的一个或一组公式。

- 递归是一种关于抽象的表达方法---用递归定义无限的相似事物
- 递归是一种算法或程序的构造技术---自身调用自身，高阶调用低阶，构造无限的计算步骤
- 递归是一种典型的计算/执行过程---由后向前代入，直至代入到递归基础，再由递归基础向后计算直至计算出最终结果，即由前向后计算
- 数学上用递归定义，但算法上有些可由递归算法/程序实现，有些可由迭代算法/程序实现。
- 迭代算法/程序的执行效率要远高于递归算法/程序，因此能用迭代实现的尽量不用递归。
- 能用迭代算法实现的，都能用递归算法实现；而能用递归算法实现的，却未必都能用迭代算法实现。
- 递归算法是用【函数】结构来表达的，而迭代算法是用【循环】结构来表达的。
- 递归算法执行需要【堆栈】（一种后进先出的数据结构）予以支持。