数据结构

（C语言版）（第2版）

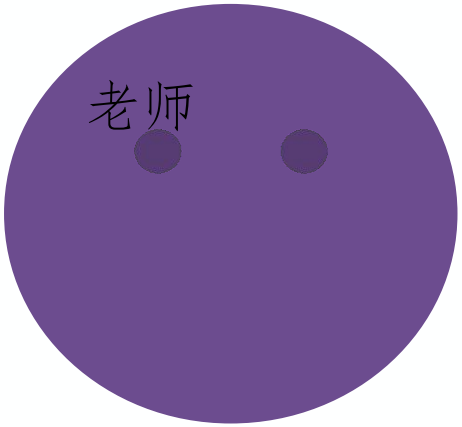
栈和队列

栈与递归

主讲教师：汪红松

教学内容 Contents

一、递归的概念
二、递归使用的情形
三、递归实现的原理



一、递归的概念

若一个对象部分地包含它自己，或用它自己给自己定义，则称这个对象是递归的；若一个过程直接地或间接地调用自己，则称这个过程是递归的过程。

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

一、递归的概念

三种情况常常用到递归方法

递归定义的数学函数

具有递归特性的数据结构

可递归求解的问题

二、递归使用的情形

1. 递归定义的数学函数

阶乘函数：
$$Fact(n)=\begin{cases}1 & \text{若n}=0\\ n\cdot Fact(n-1) & \text{若n}>0\end{cases}$$

2阶Fibonaci数列：
$$Fib(n)=\begin{cases}1 & \text{若n}=1\text{或}2\\ Fib(n-1)+Fib(n-2) & \text{其它}\end{cases}$$

二、递归使用的情形　　求解阶乘 n! 的过程

```
if ( n == 0 ) return 1;
else return n * Fact (n-1);
```

二、递归使用的情形　　用分治法求解递归问题

分治法：对于一个较为复杂的问题，能够分解成几个相对简单的且解法相同或类似的子问题来求解

必备的三个条件

1. 能将一个问题转变成一个新问题，而新问题与原问题的解法相同或类同，不同的仅是处理的对象，且这些处理对象是变化有规律的

2. 可以通过上述转化而使问题简化

3. 必须有一个明确的递归出口，或称递归的边界

二、递归使用的情形

分治法求解递归问题算法的一般形式：

```
void   p (参数表)
{
      if   （递归结束条件）可直接求解步骤；-----基本项
      else  p（较小的参数）；------归纳项
}
```

```
long Fact ( long n )
{
   if ( n == 0) return 1;//基本项
   else return n * Fact (n-1); //归纳项
}
```

二、递归使用的情形

2. 具有递归特性的数据结构

- 链表

链表结点LNode的定义由数据域data和指针域next组成. 而指针next则由LNode定义。

- 可将一个表头指针为L的单链表定义为一个递归结构，即:

(1)一个结点,其指针域为NULL,是一个单链表;

(2)一个结点,其指针域指向单链表, 仍是一个单链表。

输出链表中各个结点的递归算法

```
void TraverseList(LinkList p) {
        if(p==NULL)  return ;
        else{
                cout<< p->data<<endl;
                 TraverseList(p->next);}
}
```

二、递归使用的情形

3. 可递归求解的问题

迷宫问题　Hanoi塔问题

Hanoi
塔问题

假设有3个分别命名为A. B和C的塔座，在塔座A上插有n个直径大小各不相同，依小到大编号为1 .2.... ，n的圆盘，现要求将塔座A上的n个圆盘移至塔座C上，并仍按同样顺序叠排。

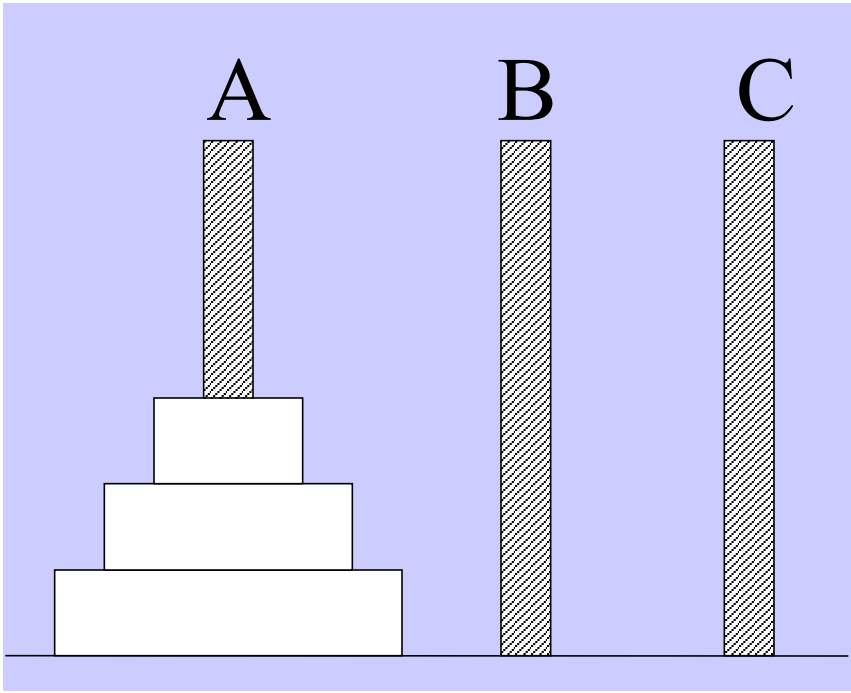
Hanoi塔问题

圆盘移动时必须遵循下列规则:

(1) 每次只能移动一个圆盘；

(2) 圆盘可以插在A,B和C中的任一塔座上；

(3) 任何时刻不可将较大圆盘压在较小圆盘之上。



二、递归使用的情形

3. 可递归求解的问题

Hanoi塔问题

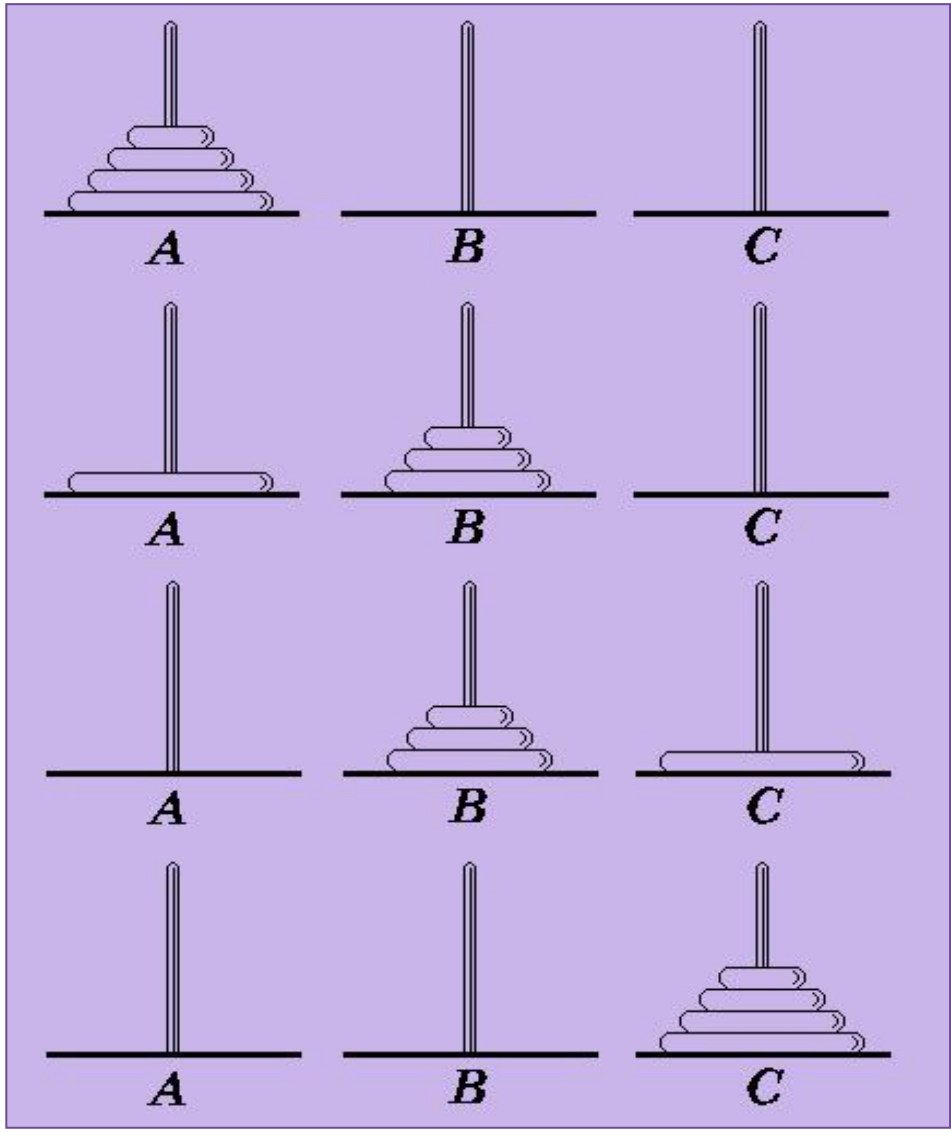
n = 1，则直接从 A 移到 C。

否则

(1)用 C 柱做过渡，将 A 的(n-1)个移到 B；

(2)将 A 最后一个直接移到 C；

(3)用 A 做过渡，将 B 的 (n-1) 个移到 C。



Hanoi塔问题

用递归法求解汉诺塔问题的算法描述：

```
void Hanoi(int n,char A,char B,char C)
{
    if(n==1)
        move(A,1,C);
    else {
        Hanoi(n-1,A,C,B);
        move(A,n,C);
        Hanoi(n-1,B,A,C);
        }
}
```

三、递归实现的原理

1.函数调用过程

调用前, 系统完成:

(1)将实参,返回地址等传递给被调用函数；

(2)为被调用函数的局部变量分配存储区；

(3)将控制转移到被调用函数的入口。

调用后, 系统完成:

(1)保存被调用函数的计算结果；

(2)释放被调用函数的数据区；

(3)依照被调用函数保存的返回地址将控制转移到调用函数。

三、递归实现的原理

2.递归函数调用的实现

“层次”

主函数	0层
第1次调用	1层
第 i 次调用	i 层

“递归工作栈”

“工作记录” → 实在参数,局部变量,返回地址

三、递归实现的原理

3.递归算法的效率分析

空间效率

- 阶乘问题、Fibonacci数列和Hanoi塔问题递归算法的空间复杂度均为O(n)。

时间效率

- 递归求解阶乘问题的时间复杂度为O(n)；
- 计算Fibonacci数列和Hanoi塔问题的递归算法时间复杂度均为O(2^n)。

三、递归实现的原理

4.递归的优缺点

优点：结构清晰，程序易读。

缺点：每次调用要生成工作记录，保存状态信息，入栈；返回时要出栈，恢复状态信息。时间开销大。

递归⇨非递归

三、递归实现的原理

5.借助栈改写递归

1. 设置一个工作栈存放递归工作记录（包括实参、返回地址及局部变量等)。

2. 进入非递归调用入口（即被调用程序开始处）将调用程序传来的实在参数和返回地址入栈。

3. 进入递归调用入口：当不满足递归结束条件时，逐层递归，将实参、返回地址及局部变量入栈，这一过程可用循环语句来实现(模拟递归分解的过程)。

4. 递归结束条件满足，将到达递归出口的给定常数作为当前的函数值。

5. 返回处理：在栈不空的情况下，反复退出栈顶记录，根据记录中的返回地址进行题意规定的操作，即逐层计算当前函数值，直至栈空为止(模拟递归求值过程)。

小结

1. 递归的概念
2. 递归使用的情形
   - » 采用递归定义的问题
   - » 采用递归的数据结构
   - » 问题的解采用递归描述
3. 递归实现的原理