数据结构与算法

疫情面前，让我们一起努力！

1

栈的定义-总结

上述过程中，我们可以总结：

栈中增加元素时，Top++；

栈中减少元素时，Top--。

换句话说，栈的基础操作的关键是Top指针。

栈的特性：后进先出

如何在实际问题中，运用栈的特性。

栈与递归

- **递归的定义** 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); }
```

24

栈与递归

✓有人送了我金、银、铜、铁、木五个宝箱，我想打开金箱子，却没有打开这个箱子的钥匙。

✓在金箱子上面写着一句话：“打开我的钥匙装在银箱子里。”

✓于是我来到银箱子前，发现还是没有打开银箱子的钥匙。

✓银箱子上也写着一句话：“打开我的钥匙装在铜箱子里。”

✓于是我再来到铜箱子前，发现还是没有打开铜箱子的钥匙。

✓铜箱子上也写着一句话：“打开我的钥匙装在铁箱子里。”

✓于是我又来到了铁箱子前，发现还是没有打开铁箱子的钥匙。

✓铁箱子上也写着一句话：“打开我的钥匙装在木箱子里。”

25

栈与递归

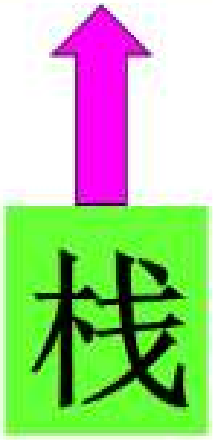
- ✓我来到木箱子前，打开了木箱，
- ✓并从木箱里拿出铁箱子的钥匙，打开了铁箱，
- ✓从铁箱里拿了出铜箱的钥匙，打开了铜箱，
- ✓再从铜箱里拿出银箱的钥匙打开了银箱，
- ✓最后从银箱里取出金箱的钥匙，打开了我想打开的金箱子。
- ✓晕吧……很啰嗦地讲了这么长一个故事。

```
void FindKey ( 箱子 ) {
    if ( 木箱子) return ;
    else FindKey ( 下面的箱子 ) }
```

栈与递归

当多个函数构成嵌套调用时，遵循

后调用先返回



27

栈与递归

- 以下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

20

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:

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

29

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

30

栈与递归

分治法求解递归问题算法的一般形式：

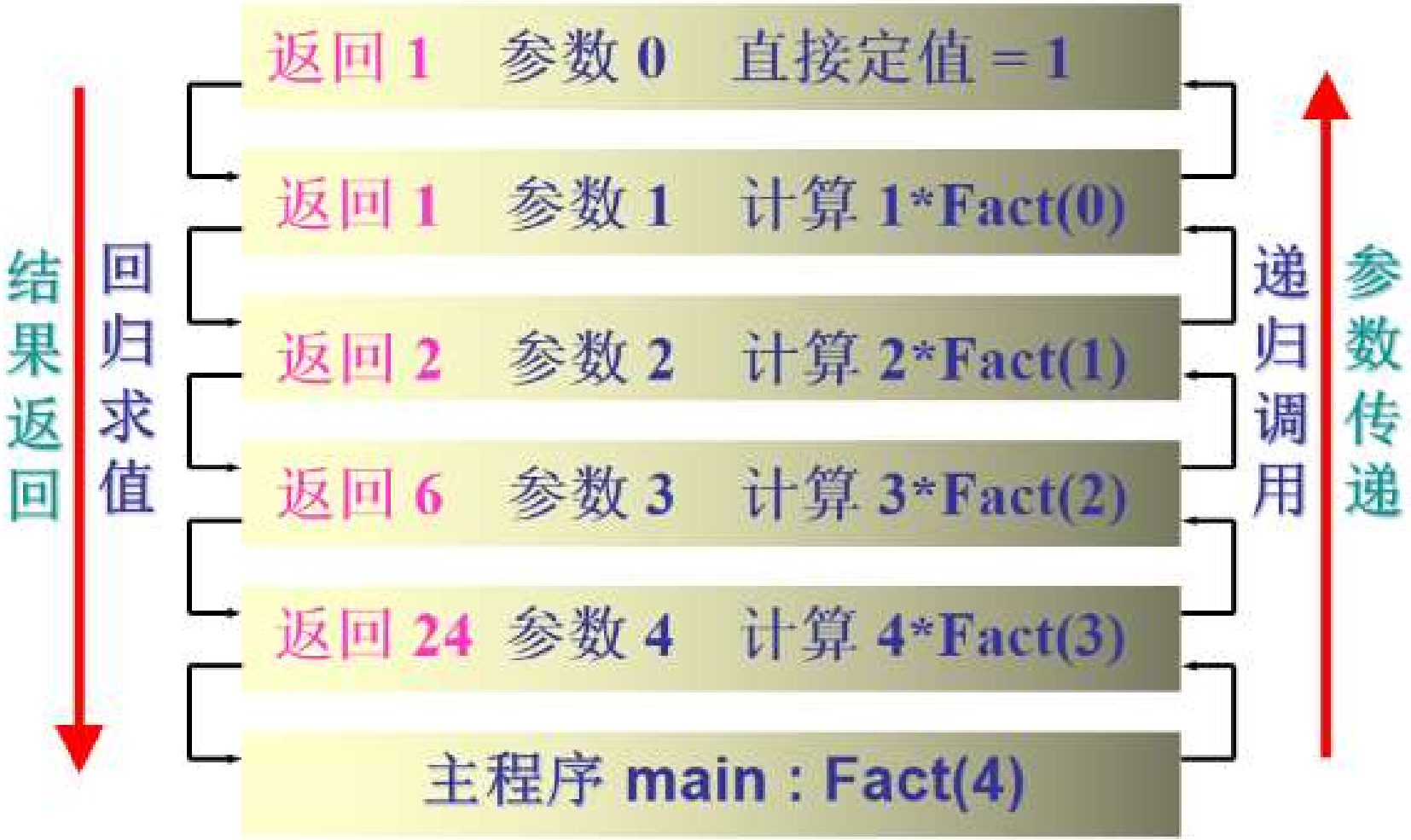
```
void   p (参数表) {
  if   （递归结束条件）可直接求解步骤；-----基本项
  else  p（较小的参数）；------归纳项
  }
```

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

31

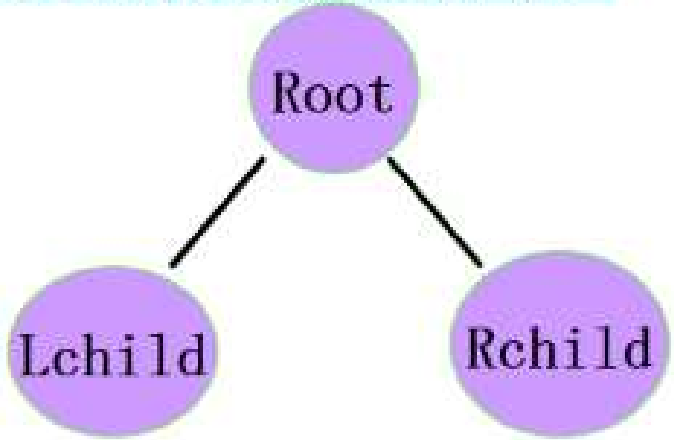
求解阶乘 n! 的过程

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



2. 具有递归特性的数据结构：

- 树

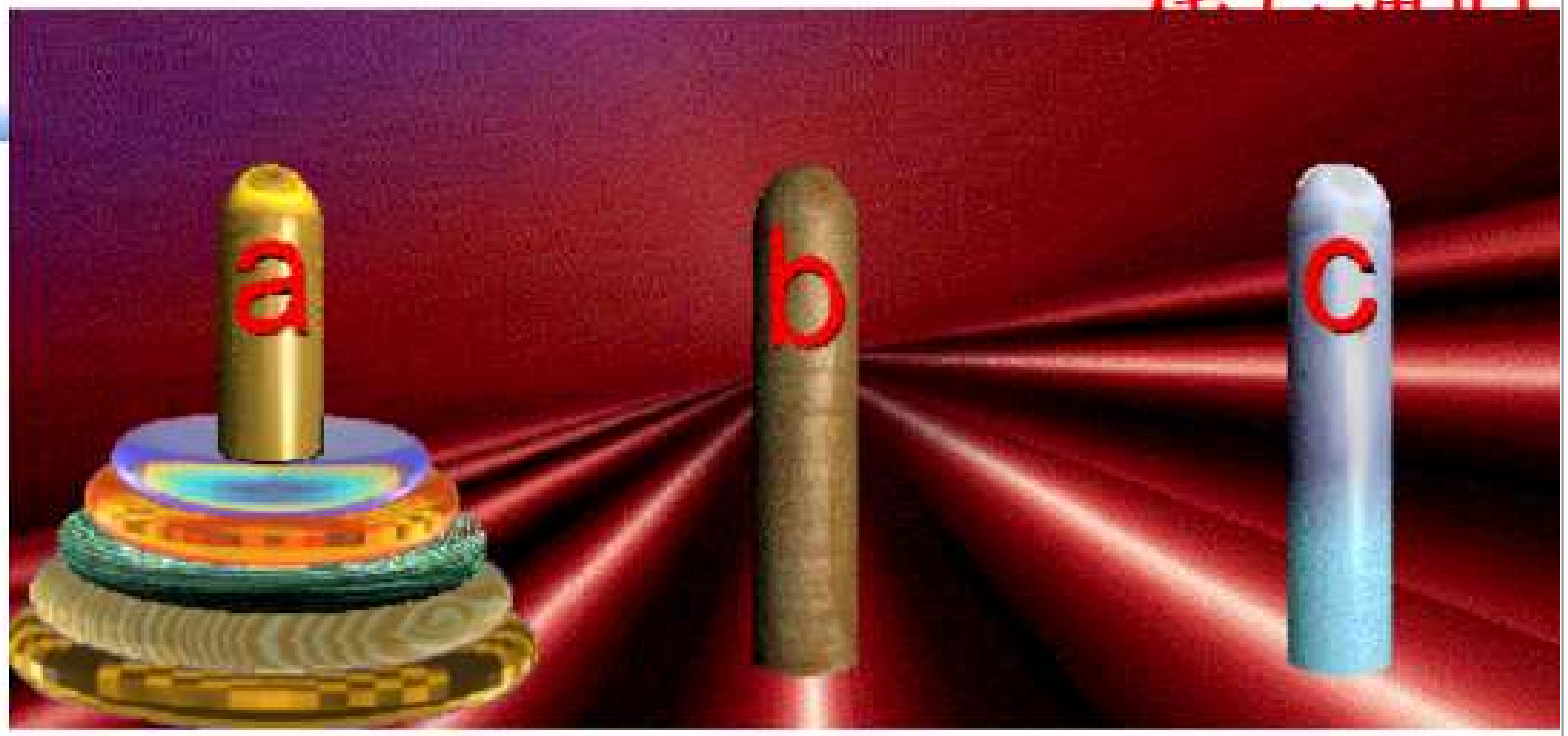


- 广义表　A=(a, A)

3. 可递归求解的问题：

- 迷宫问题　Hanoi塔问题

汉诺塔



在印度圣庙里，一块黄铜板上插着三根宝石针。
主神梵天在创造世界时，在其中一根针上穿好了由大到小的64片金片，这就是汉诺塔。
僧侣不停移动这些金片，一次只移动一片，小片必在大片上面。
当所有的金片都移到另外一个针上时，世界将会灭亡。

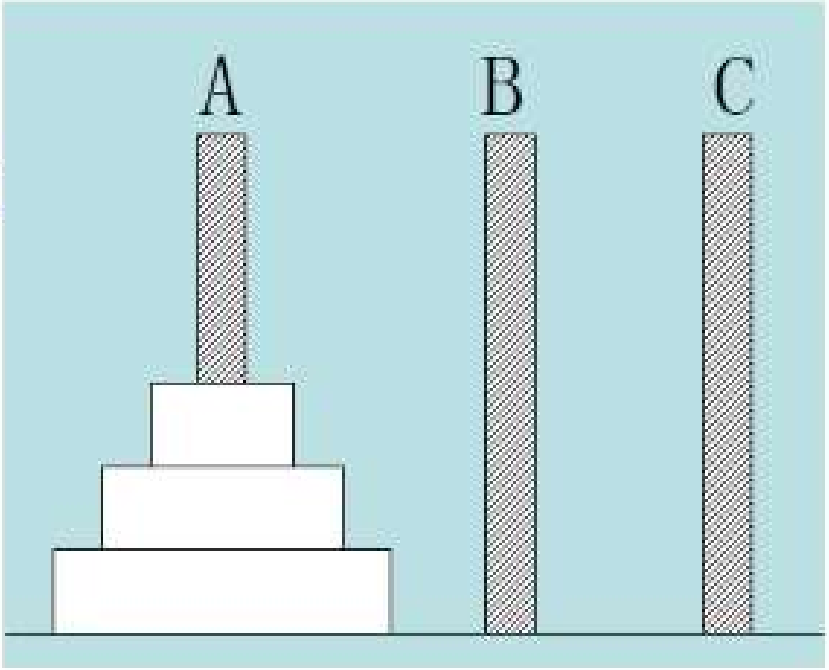
Hanoi塔问题

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



35

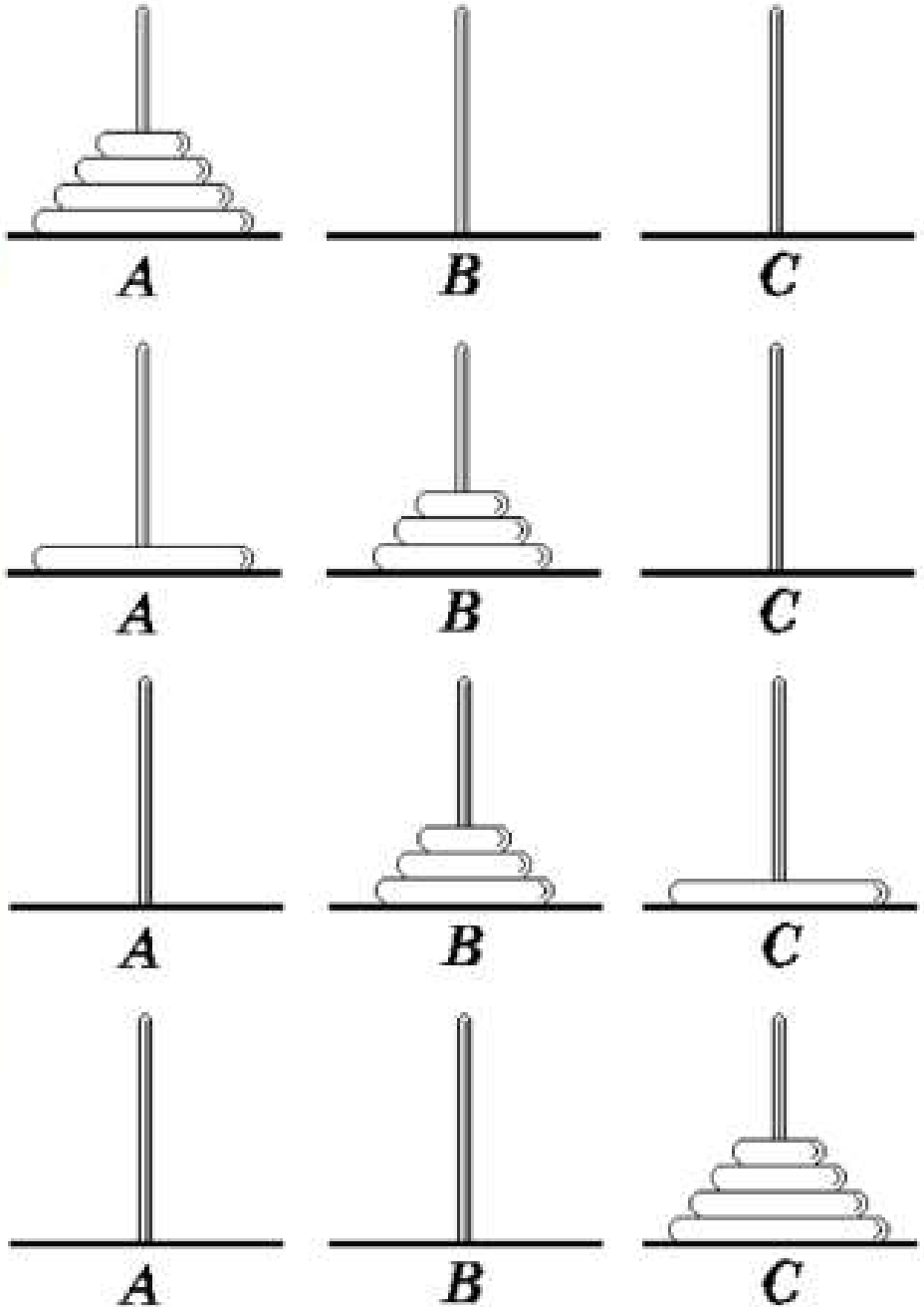
Hanoi塔问题

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



跟踪程序，给出下列程序的运行结果，以深刻地理解递归的调用和返回过程

```
#include<iostream.h>
int c=0;
void move(char x,int n,char z)
{cout<<++c<<","<<n<<","<<x<<","<<z<<endl;}
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);  }}
void main(){Hanoi(3,'a','b','c');}
```

```
1,1,a,c
2,2,a,b
3,1,c,b
4,3,a,c
5,1,b,a
6,2,b,c
7,1,a,c
```

37

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

30

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

39

大家还记得？

数制转换是计算机实现计算和处理的基本问题。

比如，将十进制数N转换为j进制的数，其解决的方法很多，其中一个常用的算法是除j取余法。

其算法原理是：

$N = (N \text{ div } j) * j + N \bmod j$

其中： div为整除，mod为求余

40

栈的应用

例如：（1348)$_{10}$ = (2504)$_8$，

其运算过程如下：

计算顺序 ↓

n	n div 8	n mod 8
1348	168	4
168	21	0
21	2	5
2	0	2

↑ 输出顺序

由于上述计算过程与打印输出的过程相反。因此，若将计算过程中得到的八进制数的各位顺序进栈，则按出栈序列打印输出的即为与输入对应的八进制数。

栈的应用

如何实现上述过程？

关键问题：

记录每次所得余数；
倒序输出这些余数。

栈的“后进先出”恰恰解决了这个问题。

42

栈的应用

1．算法思想如下：

（1）若N<>0，则将N % j 取得的余数压入栈s中 ，执行（2）；

若N=0，将栈s的内容依次出栈，算法结束。

（2）用N / j 代替 N;

（3）当N>0，则重复步骤（1）、（2）。

主观题 0.5分

请大家尝试用类C语言描述上述算法。

栈的应用

2．算法的实现：

```
（1） void conversion( ) { //十进制转换为等值的八进制
（2） Initstack(S);
（3）  scanf (“%d”,N);
（4） while(N){
（5）    Push(S,N%8);
（6）    N=N/8;
（7）    }
（8） while(! StackEmpty(S)){
（9）  Pop(S,e);
（10）  printf(“%d”,e);
    }
  }
```

栈的应用

栈的其他应用

括号匹配的检验

表达式求值

栈的应用，离不开“后进先出”。

– 后进先出（Last In First Out）