

第6章 函数

——Hanoi塔问题的递归求解

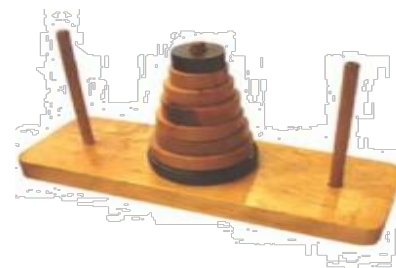
什么情况下考虑使用递归？

- 通常下面三种情况需要使用递归：
 - * **数学定义**是递归的
 - * 如计算阶乘，最大公约数和Fibonacci数列
 - * **数据结构**是递归的
 - * 如队列、链表、树和图
 - * **问题的解法**是递归的
 - * 如Hanoi塔，骑士游历、八皇后问题（回溯法）

一个经典的递归问题

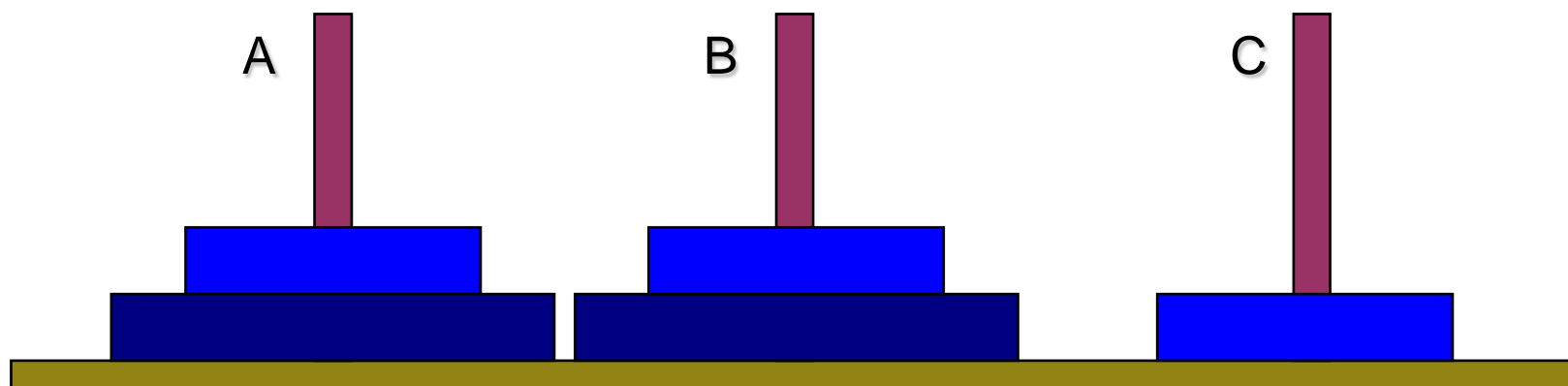
■ 汉诺塔（Hanoi）问题

- * 印度神话，上帝创造世界时作了三根金刚石柱子，第一根上从下往上按大小顺序摞着64片黄金圆盘，上帝命令婆罗门把圆盘从下开始按大小顺序重新摆放到第二根上，规定每次只能移动一个圆盘，在小圆盘上不能放大圆盘
- * 当 $n=64$ 时，需移动多少次呢？
 - * 18446744073709551615，即1844亿亿次
 - * 若按每次耗时1微秒计算，则64个圆盘的移动需60万年



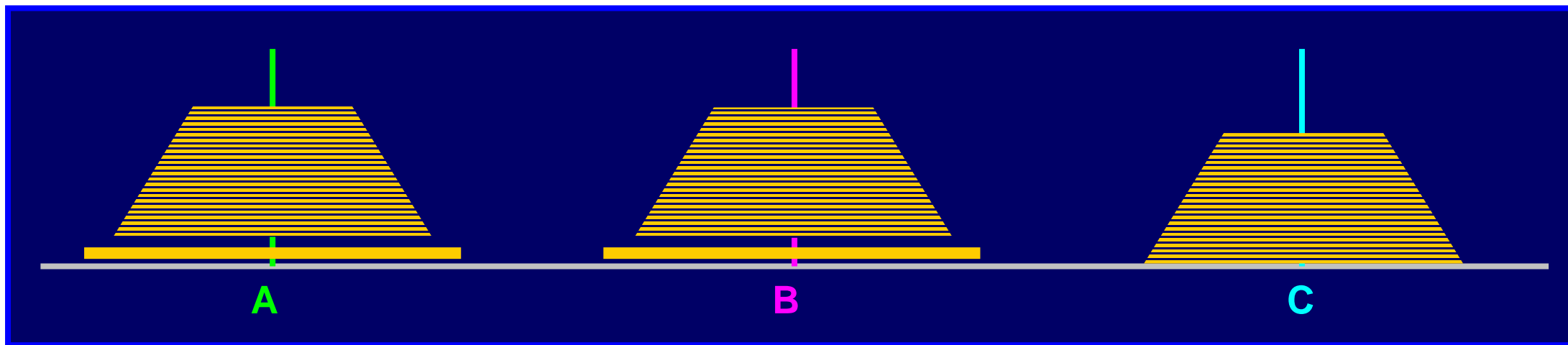
汉诺塔（Hanoi）问题

■ 较为简单的情形—— $n=2$



- * 将1号圆盘从A移到C
- * 将2号圆盘从A移到B
- * 将1号圆盘从C移到B

汉诺塔问题的递归求解



■ 数学归纳法

- 假设 $n-1$ 个圆盘的汉诺塔问题已解决
- 将“上面 $n-1$ 个圆盘”看成一个整体

移动 n 个圆盘



移动 $n-1$ 个圆盘

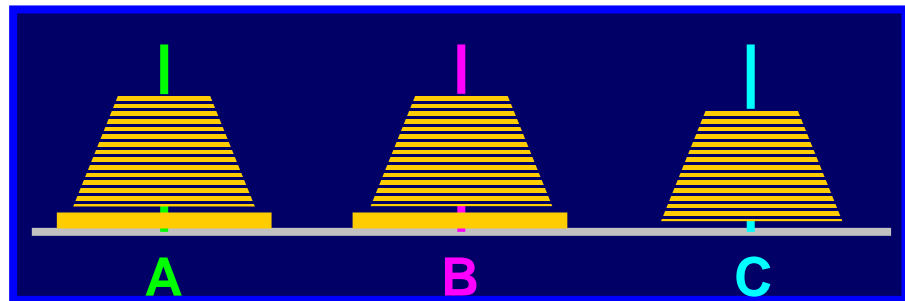
- ✓ 将“上面 $n-1$ 个圆盘”从A移到C
- ✓ 将第 n 号圆盘从A移到B
- ✓ 将“上面 $n-1$ 个圆盘”从C移到B



汉诺塔问题的递归函数实现

✓ 将“n个圆盘”借助于C从A移到B

```
void Hanoi(int n, char a, char b, char c)
{
    if (n == 1)
    {
        Move(n, a, b);
    }
    else
    {
        Hanoi(n-1, a, c, b);
        Move(n, a, b);
        Hanoi(n-1, c, b, a);
    }
}
```



将“上面n-1个圆盘”从A移到C
将第n号圆盘从A移到B
将“上面n-1个圆盘”从C移到B

汉诺塔问题的递归函数实现

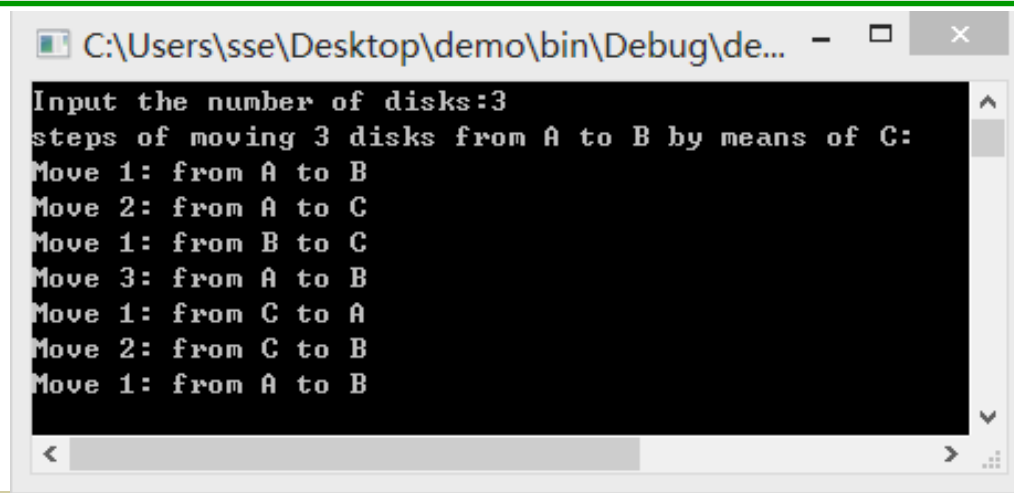
✓ 将“n个圆盘”借助于C从A移到B

```
void Hanoi(int n, char a, char b, char c)
{
    if (n == 1)
    {
        Move(n, a, b);
    }
    else
    {
        Hanoi(n-1, a, c, b);
        Move(n, a, b);
        Hanoi(n-1, c, b, a);
    }
}
```

```
void Move(int n, char a, char b)
{
    printf("Move %d: from %c to %c\n", n, a, b);
}
```

将“上面n-1个圆盘”从A移到C
将第n号圆盘从A移到B
将“上面n-1个圆盘”从C移到B

```
#include <stdio.h>
void Hanoi(int n, char a, char b, char c);
void Move(int n, char a, char b);
int main()
{
    int n;
    printf("Input the number of disks:");
    scanf("%d", &n);
    printf("steps of moving %d disks from A to B by means of C:\n", n);
    Hanoi(n, 'A', 'B', 'C');
    return 0;
}
```



```
C:\Users\sse\Desktop\demo\bin\Debug\de...
Input the number of disks:3
steps of moving 3 disks from A to B by means of C:
Move 1: from A to B
Move 2: from A to C
Move 1: from B to C
Move 3: from A to B
Move 1: from C to A
Move 2: from C to B
Move 1: from A to B
```


讨论

- 汉诺塔问题可以用非递归的方法求解吗？

