数据结构与算法（三）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

第3章　栈与队列

- 栈
- **栈的应用**
  - **递归到非递归的转换**
- 队列

递归转非递归

- 递归函数调用原理
- 机械的递归转换
- 优化后的非递归函数

递归的再研究

- 阶乘 $f(n)=\begin{cases} n\times f(n-1) & n\geq 1 \\ 1 & n=0 \end{cases}$
- **递归出口**
  - 递归终止的条件，即最小子问题的求解
  - 可以允许多个出口
- **递归规则**（递归体 + 界函数）
  - 将原问题划分成子问题
  - 保证递归的规模向出口条件靠拢

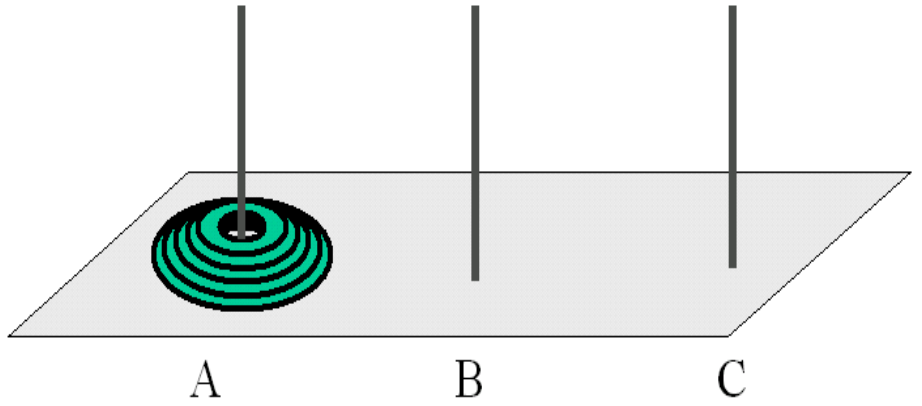
递归算法的非递归实现

$$f(n)=\begin{cases} n\times f(n-1) & n\geq 1 \\ 1 & n=0 \end{cases}$$

- 阶乘的非递归方式
  - 建立迭代
  - 递归转换为非递归
- 以Hanoi塔为例呢？

河内塔问题的递归求解程序

http://www.17yy.com/f/play/89425.html

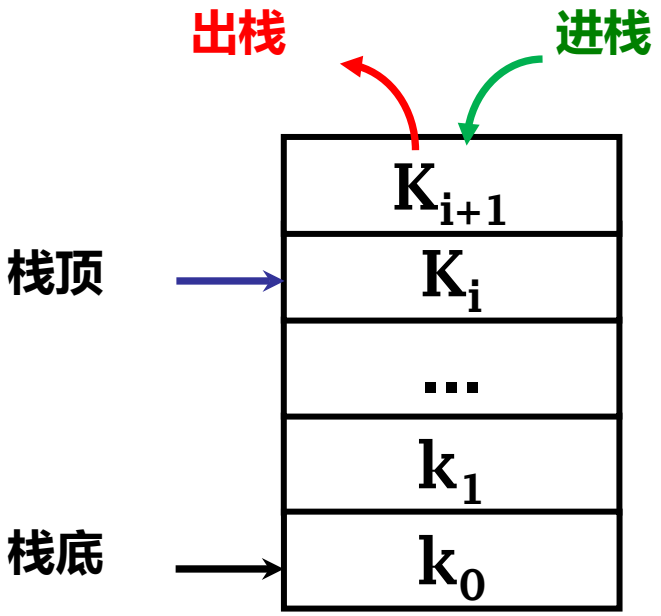


- hanoi(n,X,Y,Z)
  - 移动n个槃环
  - X柱出发，将槃环移动到Z 柱
  - X、Y、Z都可以暂存
    - 大盘不能压小盘
- 例如， hanoi(2, ‘B’, ‘C’ , ‘A’)
  - B柱上部的2个环槃移动 到A 柱

3.1.3 递归转非递归

```
void hanoi(int n, char X, char Y, char Z) {
    if (n <= 1)
        move(X,Z);
    else  { // X上最大的不动，其他 n - 1个环槃移到Y
        hanoi(n-1,X,Z,Y);
        move(X,Z);          // 移动最大环槃到Z，放好
        hanoi(n-1,Y,X,Z); // 把 Y上的n - 1个环槃移到Z
    }
}
void move(char X, char Y)   // 把柱X的顶部环槃移到柱Y
{
  cout  << "move"  <<  X  << "to" << Y  << endl;
}
```

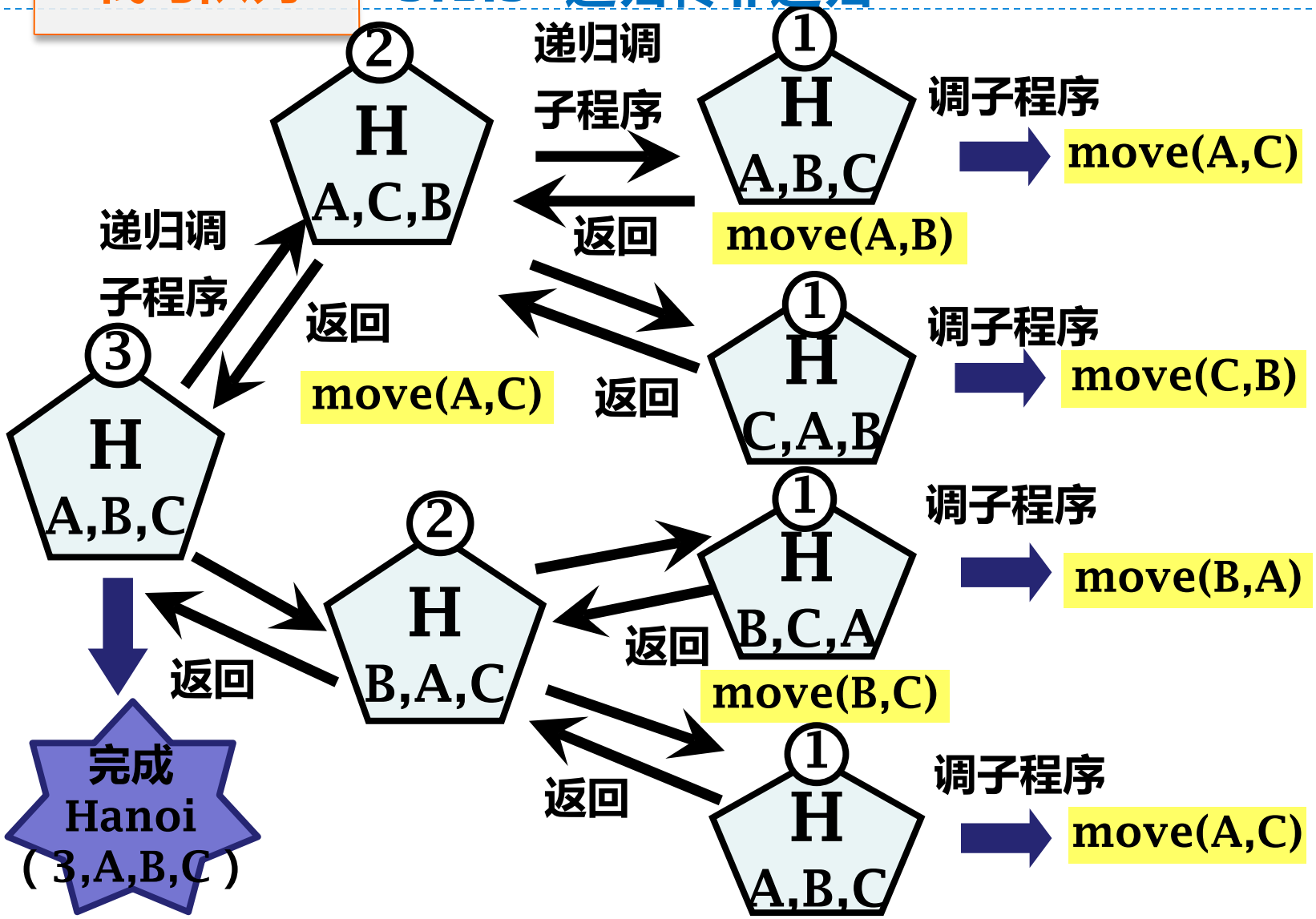
Hanoi递归子程序的运行示意图



执行hanoi程序的指令流

通过内部栈和子程序交换信息

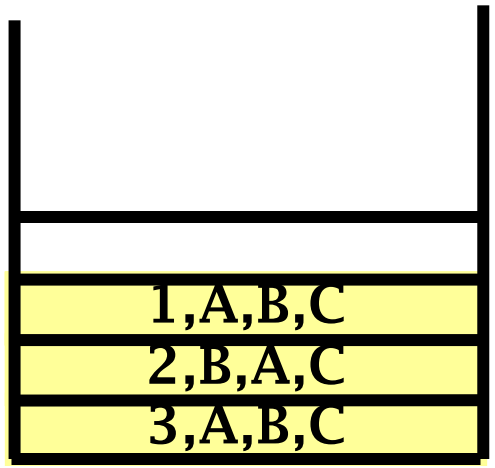
3.1.3 递归转非递归


②
H
A,C,B
递归调
子程序
①
H
A,B,C
调子程序
move(A,C)
返回
move(A,B)
递归调
子程序
返回
move(A,C)
③
H
A,B,C
①
H
C,A,B
调子程序
move(C,B)
返回
②
H
B,A,C
①
H
B,C,A
调子程序
move(B,A)
返回
返回
move(B,C)
①
H
A,B,C
调子程序
move(A,C)
返回
完成
Hanoi
(3,A,B,C)

3.1.3 递归转非递归

递归运行时,堆栈的进退以及通过堆栈传递参数

hanoi(1,A,B,C)
hanoi(1,B,C,A)
hanoi(2,B,A,C)
hanoi(1,C,A,B)
hanoi(1,A,B,C)
hanoi(2,A,C,B)
hanoi(3,A,B,C)



执行move(A,C)

一个递归数学公式

$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$

递归函数示例

$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$

```
int f(int n) {
    if (n<2)
        return n+1;
    else
        return f(n/2) * f(n/4);
}
```

递归函数示例(转换一下)

```
void exmp(int n, int& f) {
    int u1, u2;
    if (n<2)
        f = n+1;
    else {
        exmp((int)(n/2), u1);
        exmp((int)(n/4), u2);
        f = u1*u2;
    }
}
```

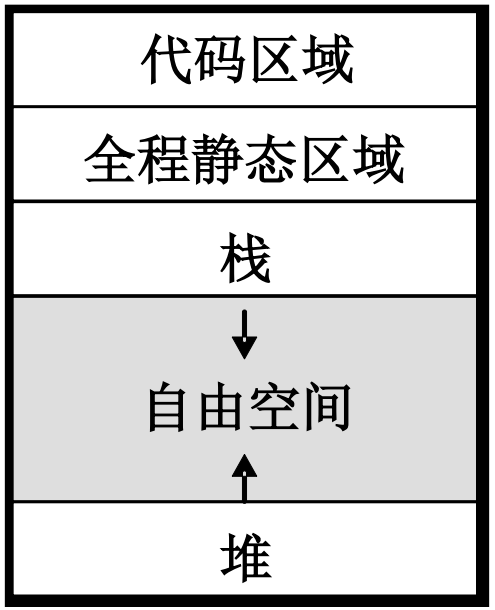
$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$

函数运行时的动态存储分配

- **栈（stack）**用于分配后进先出LIFO的数据
  - 如函数调用
- **堆（heap）**用于不符合LIFO的
  - 如指针所指向空间的分配

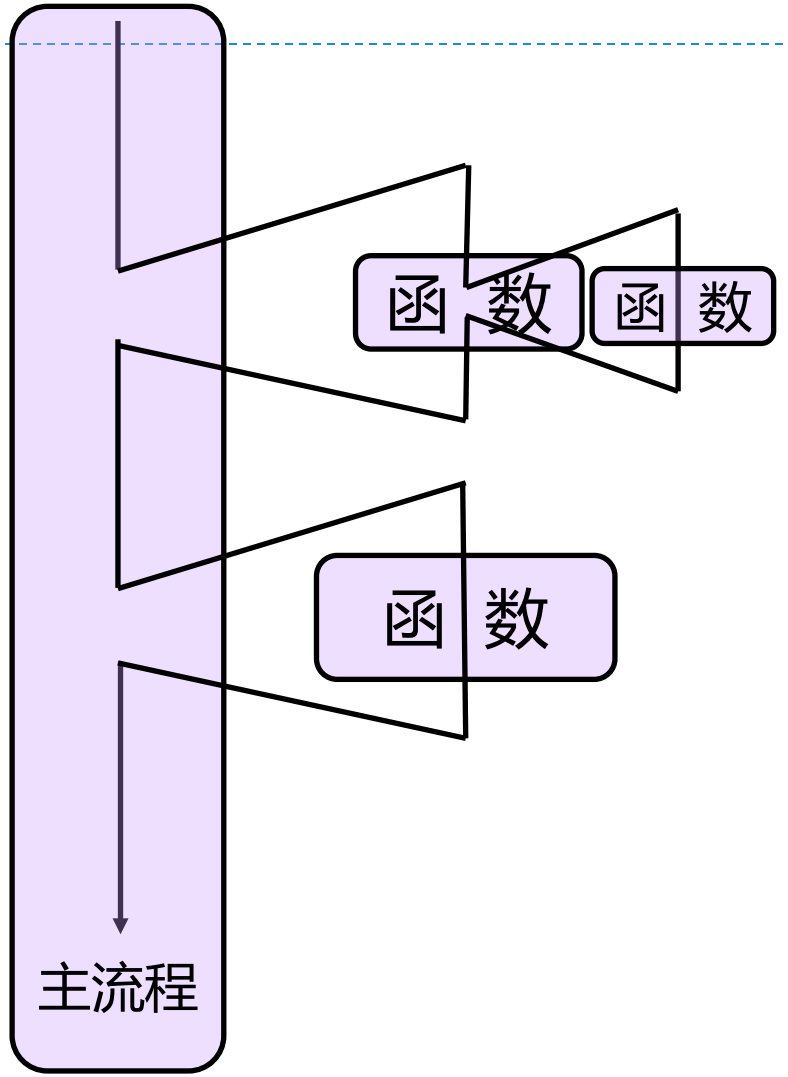
```
int A[1000000];
```
x

```
int *B = new int[1000000];
```
√



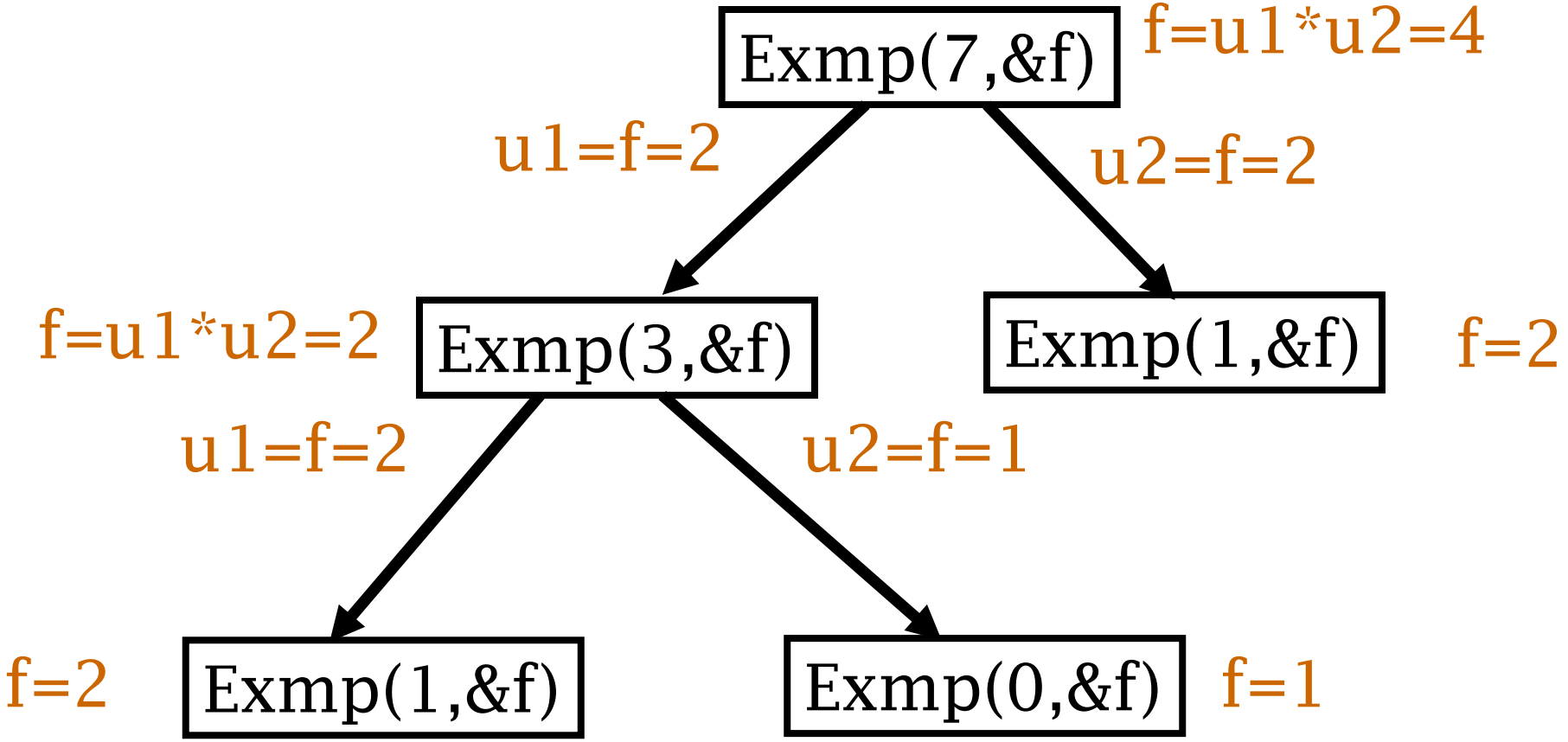
函数调用及返回的步骤

- **调用**
  - 保存调用信息（参数，返回地址）
  - 分配数据区（局部变量）
  - 控制转移给被调函数的入口
- **返回**
  - 保存返回信息
  - 释放数据区
  - 控制转移到上级函数（主调用函数）



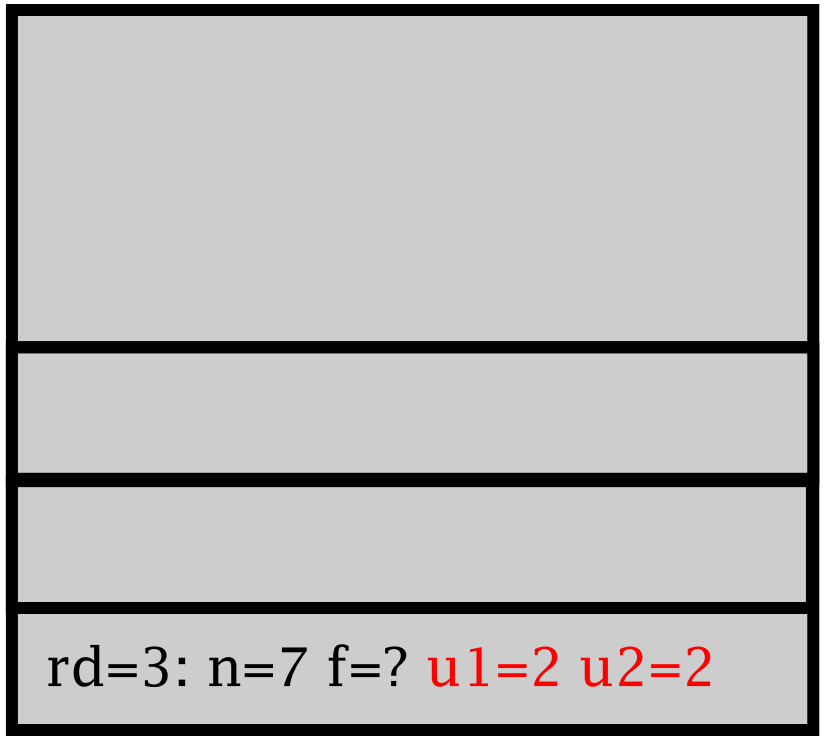
函数执行过程图解

$$fu(n)=\begin{cases} n+1 & \text{当}n<2\text{时} \\ fu(\lfloor n/2 \rfloor)*fu(\lfloor n/4 \rfloor) & n\geq 2\text{时} \end{cases}$$



用栈模拟递归调用过程

- 后调用，先返回（LIFO），所以用栈




```
void exmp(int n, int& f) {
    int u1, u2;
    if (n<2) f = n+1;
    else {
        exmp((int)(n/2), u1);
        exmp((int)(n/4), u2);
        f = u1*u2;
    }
}
```

思考

- 对以下函数，请画出n=4情况下的递归树，并用栈模拟递归的调用过程。
  - 阶乘函数
  - 2阶斐波那契函数

$$f_0=0,\ f_1=1,\ f_n = f_{n-1}+ f_{n-2}$$

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材