递推算法

黑猫编程 blackcat1995.com 什么是递推

所谓递推，是指从已知的初始条件出发，依据某种递推关系，逐次推出所要求的各中间结果及最后结果。其中初始条件或是问题本身已经给定，或是通过对问题的分析与化简后确定。

递推算法使用“步步为营”的方法，不断利用已有信息推导出新的信息。

顺推法： 是指从已知条件出发，逐步推算出要解决问题的方法。

逆推法：是从已知的结果出发，用迭代表达式逐步推算出问题开始的条件，即顺推法的逆过程。

无论顺推还是逆推，其关键是要找到递推式。这种处理问题的方法能使复杂运算化为若干步重复的简单运算，充分发挥出计算机擅长于重复处理的特点。

黑猫编程 blackcat1995.com 递推的特点

1、问题可以划分成多个状态；

2、除初始状态外，其它各个状态都可以用固定的递推关系式来表示。

在我们实际解题中，题目不会直接给出递推关系式，而是需要通过分析各种状态，找出递推关系式

算法介绍

递推算法的首要问题是得到相邻的数据项间的关系（即递推关系）。

递推算法避开了求通项公式的麻烦，把一个复杂的问题的求解，分解成了连续的若干步简单运算。

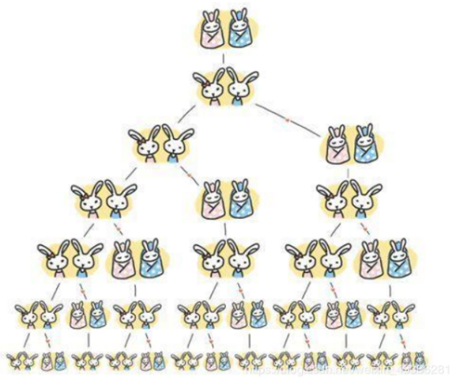
一般说来，可以将递推算法看成是一种特殊的迭代算法。

兔子数列

把雌雄各一的一对新兔子放入养殖场中。每只雌兔在出生两个月以后，每月产雌雄各一的一对新兔子。试问第n个月后养殖场中共有多少对兔子。

第一个月只有一对兔宝宝，1对兔子。
第二个月兔宝宝变成大兔子，1对兔子。
第三个月大兔子生了一对兔宝宝，一大一小2对兔子。
第四个月大兔子继续生一对兔宝宝，小兔子变成大兔子。两大一小3对兔子。
….
我们把这个数列列表

月份	1	2	3	4	5	6	7	8	9
小兔子	1	0	1	1	2	3	5	8	13
大兔子	0	1	1	2	3	5	8	13	21
总数	1	1	2	3	5	8	13	21	34



我们发现会发现以下几个规律：

前一个月的大兔子对数就是下一个月的小兔子对数。

前一个月的大兔子和小兔子对数的和就是下个月大兔子的对数。

按照这个表格，我们会发现无论是小兔子对数、大兔子对数还是总对数，除了最初几个数字不一样之外，后面都是按照1、1、2、3、5、8、13…变化的，这个数列就称为兔子数列或者斐波那契数列。

兔子数列最大的特点就是前两项之和等于后一项，比如1+1=2、1+2=3、2+3=5、3+5=8、5+8=13…

我们用f(n)表示一个数列的第n项，那么斐波那契数列的规律就是

$f(n)=f(n-1)+f(n-2)$

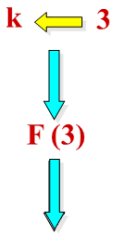
递归算法

递归求阶乘

黑猫编程 blackcat1995.com **递归算法求阶乘**

```
int  Factorial ( int  n ) {
        if  ( n = = 0 )      return  1 ;
        else       return   n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!



Output **3! = 6**

黑猫编程 blackcat1995.com **递归算法求阶乘**

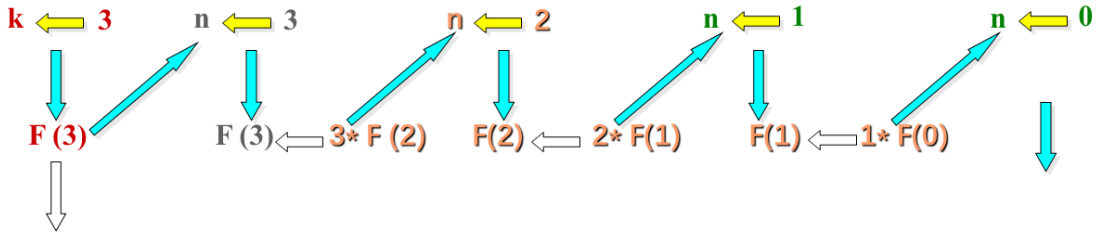
```
int  Factorial ( int  n ) {
        if  ( n = = 0 )      return  1 ;
        else       return   n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int  Factorial ( int  n ) {
        if  ( n = = 0 )      return  1 ;
        else       return   n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!



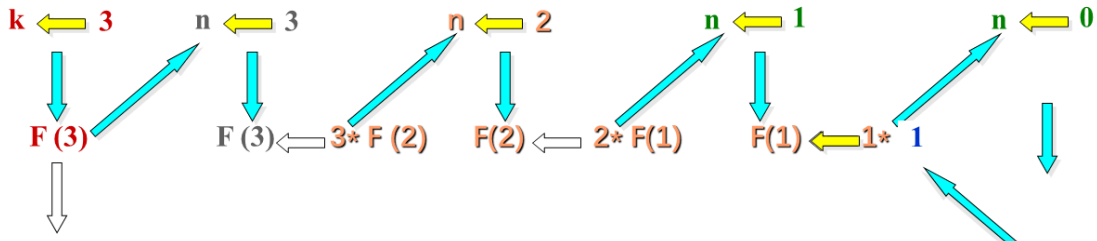
Output 3! = 6

F (0) ⇐ 1

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int  Factorial ( int  n ) {
        if  ( n = = 0 )      return  1 ;
        else       return   n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!

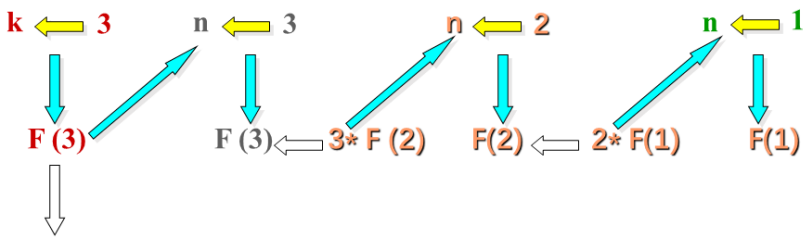


Output 3! = 6

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int  Factorial ( int  n ) {
        if  ( n = = 0 )      return  1 ;
        else       return   n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!

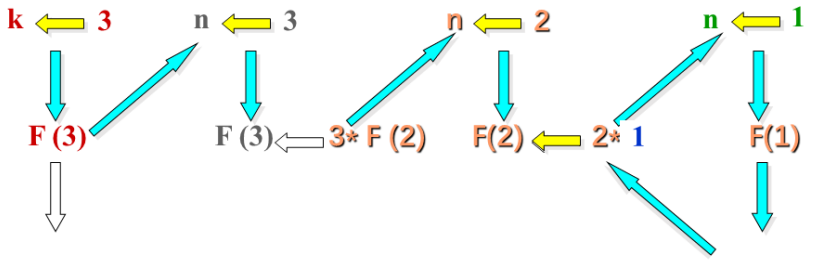


Output 3! = 6

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int Factorial ( int n ) {
        if ( n = = 0 )      return 1 ;
        else        return  n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!



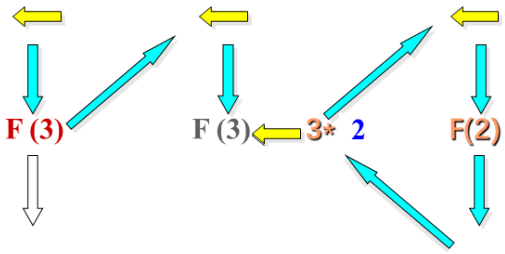
Output 3! = 6

F (1) = 1

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int Factorial ( int n ) {
        if ( n = = 0 )      return 1 ;
        else        return  n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!



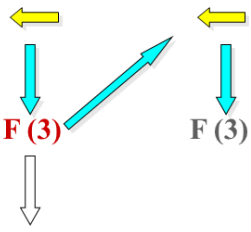
Output 3! = 6

F(2) = 2

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int Factorial ( int n ) {
        if ( n = = 0 )      return 1 ;
        else        return  n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!

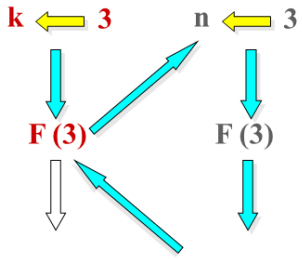


Output 3! = 6

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int Factorial ( int n ) {
        if ( n = = 0 )      return 1 ;
        else        return  n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!

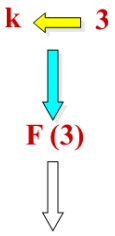


Output 3! = 6 **F(3) = 6**

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int Factorial ( int n ) {
        if ( n = = 0 )      return 1 ;
        else        return  n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!

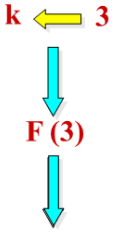


Output 3! = 6

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int Factorial ( int n ) {
        if ( n = = 0 )      return 1 ;
        else        return  n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!

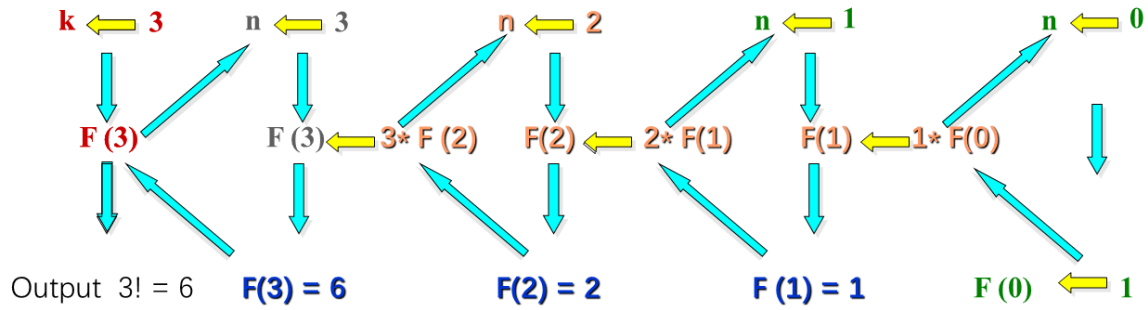


Output **3! = 6**

黑猫编程 blackcat1995.com 递归算法求阶乘

```
int  Factorial ( int  n ) {
        if  ( n = = 0 )      return  1 ;
        else        return   n * Factorial ( n - 1 ) ;
}
```

计算 Factorial (3) = 3!



递归求斐波那契数列

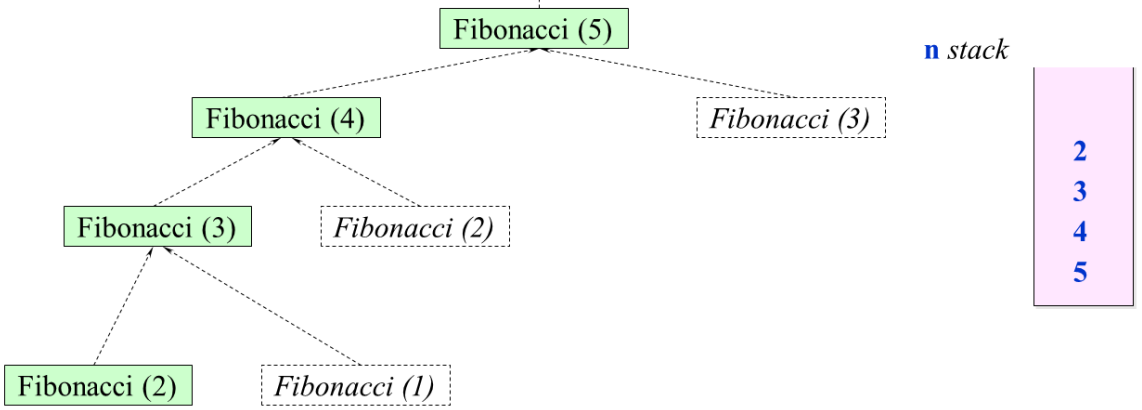
黑猫编程 blackcat1995.com 递归算法求斐波那契数列

$$F_n = \begin{cases} F_n = 1 & if \quad n = 1 \\ F_n = 1 & if \quad n = 2 \\ F_n = F_{n-1} + F_{n-2} & if \quad n > 2 \end{cases}$$

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )
        return 1 ;
    else
        return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int Fibonacci ( int n ) {
    if ( n <= 2 )       return 1 ;
    else        return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

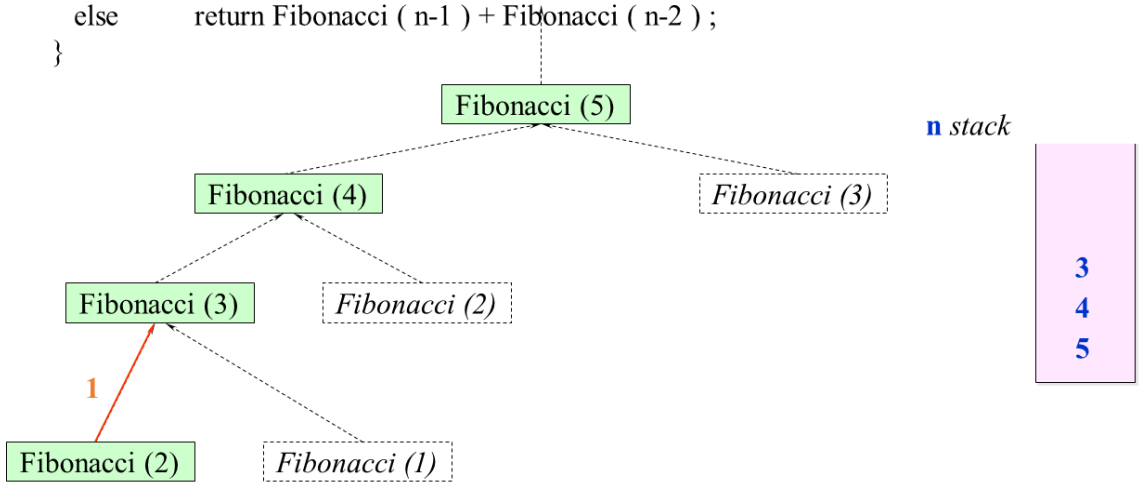


n *stack*

2
3
4
5

黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int Fibonacci ( int n ) {
    if ( n <= 2 )       return 1 ;
    else        return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

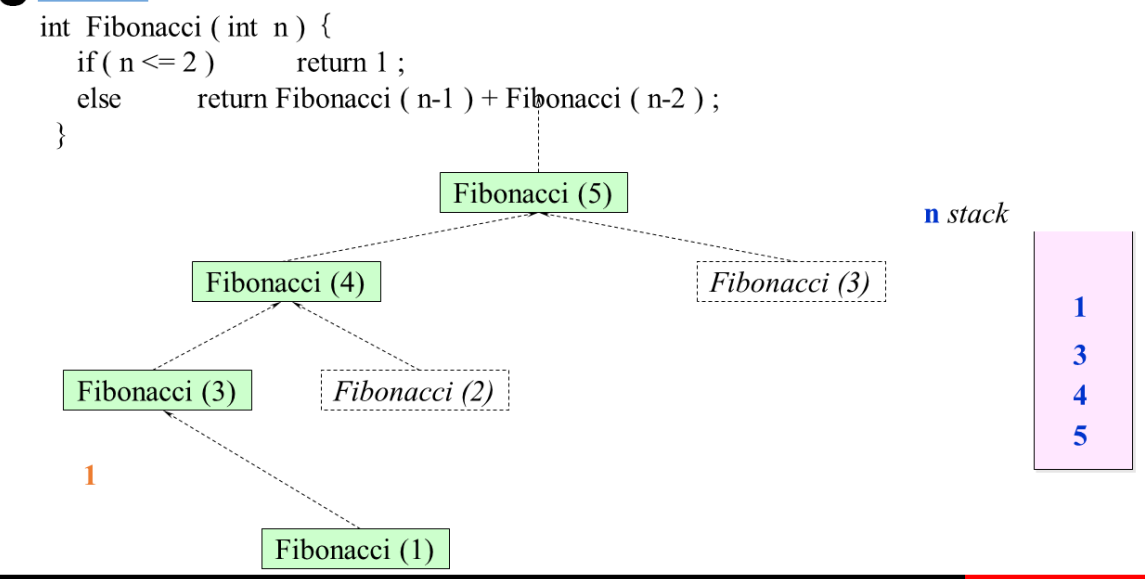



n *stack*

3
4
5

黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int Fibonacci ( int n ) {
    if ( n <= 2 )       return 1 ;
    else        return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

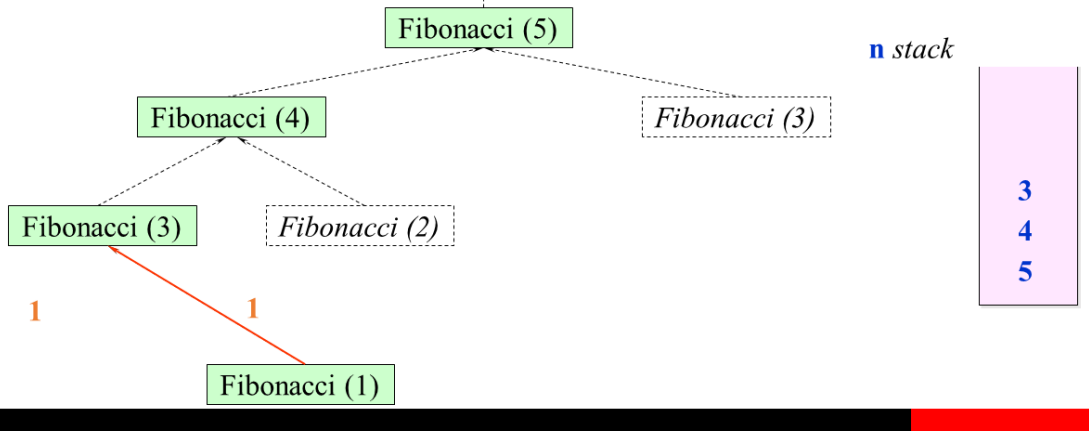



n *stack*

1
3
4
5

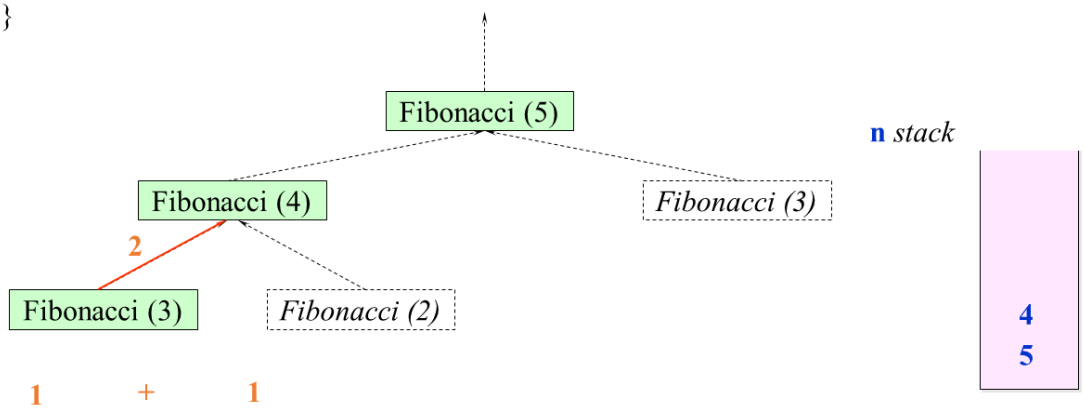
黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int Fibonacci ( int n ) {
    if ( n <= 2 )       return 1 ;
    else        return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```



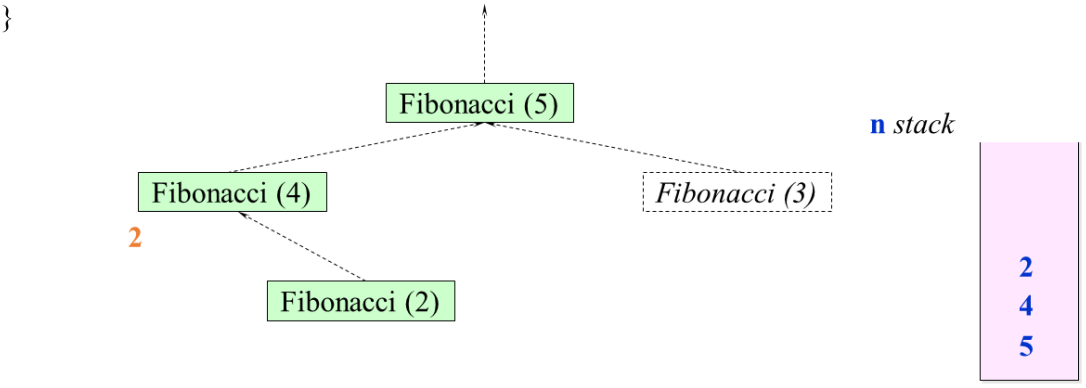
黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int Fibonacci ( int n ) {
    if ( n <= 2 )       return 1 ;
    else        return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```



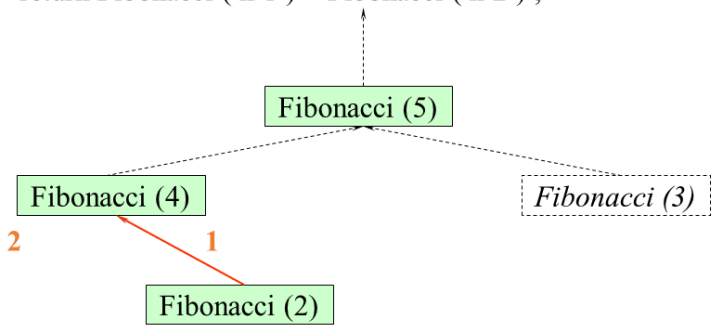
黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int Fibonacci ( int n ) {
    if ( n <= 2 )       return 1 ;
    else        return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

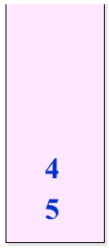


黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

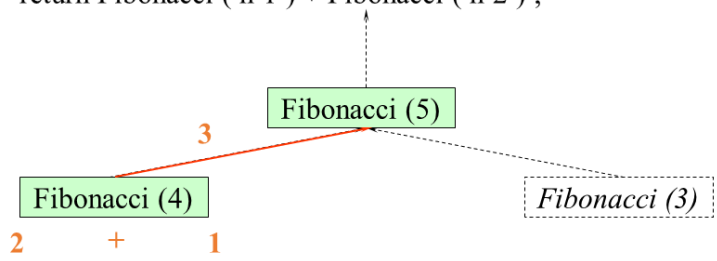


n *stack*

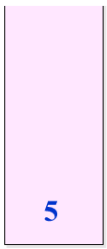


黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

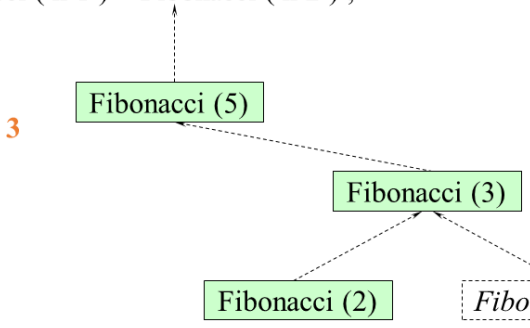


n *stack*

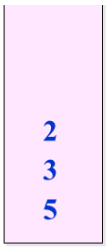


黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

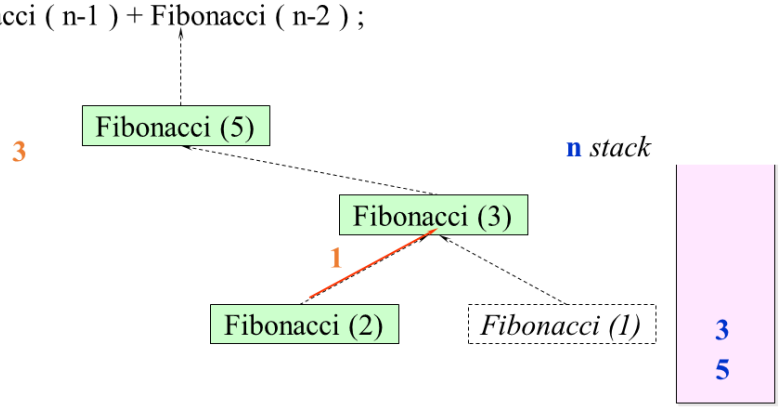


n *stack*



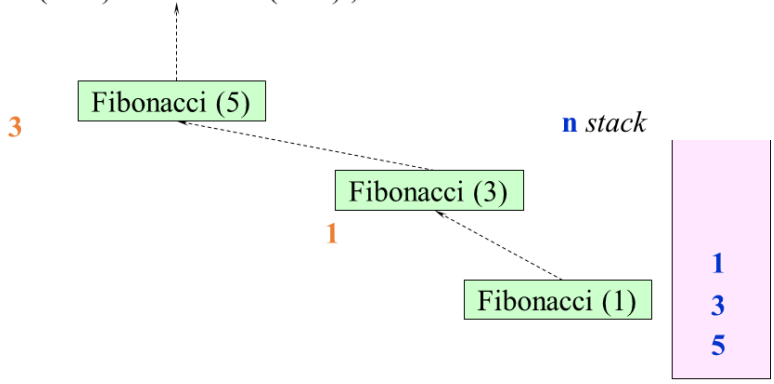
黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```



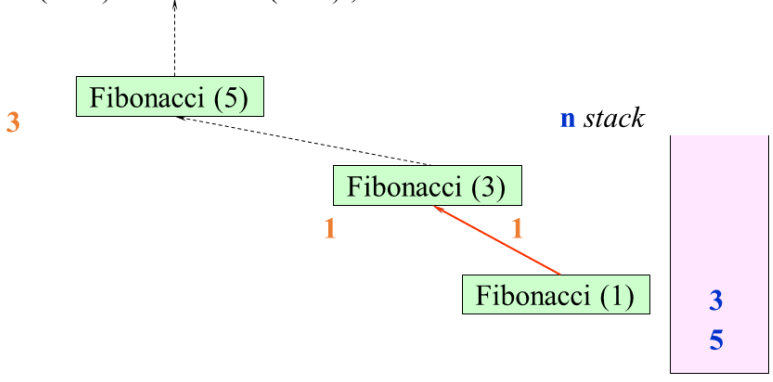
黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

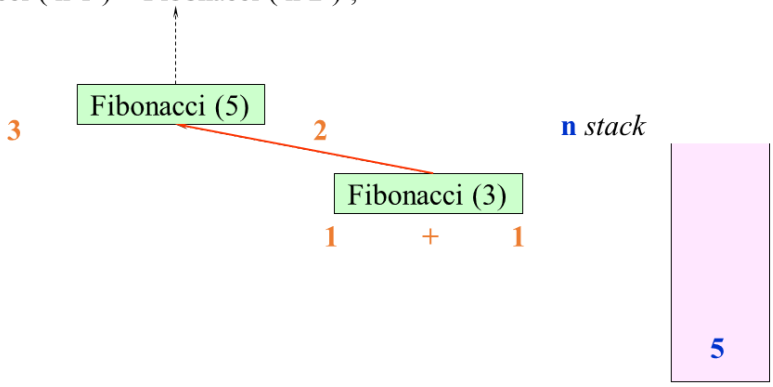
黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```

递归算法求斐波那契数列

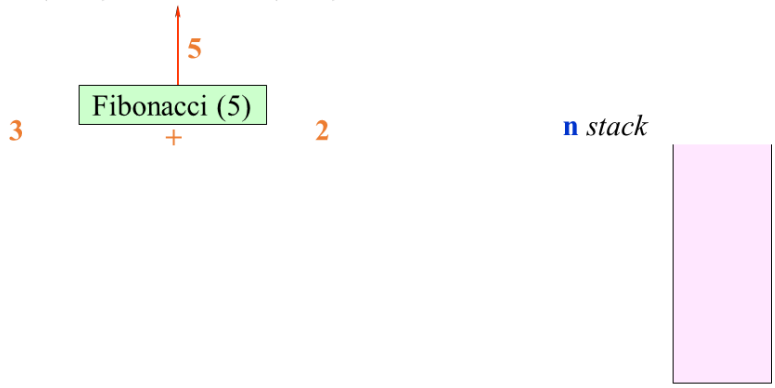
```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```



黑猫编程 blackcat1995.com

递归算法求斐波那契数列

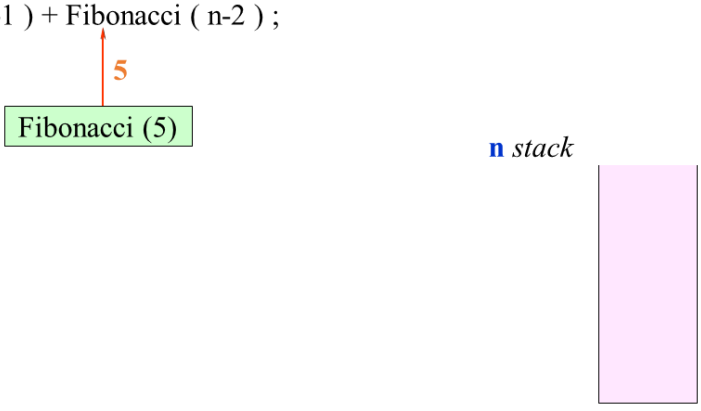
```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```



黑猫编程 blackcat1995.com

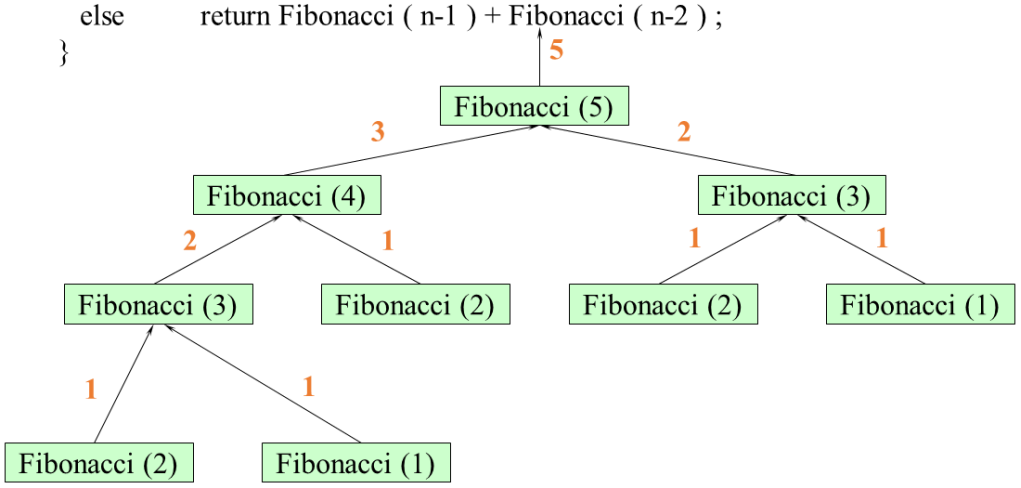
递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```



黑猫编程 blackcat1995.com 递归算法求斐波那契数列

```
int  Fibonacci ( int  n ) {
    if ( n <= 2 )          return 1 ;
    else          return Fibonacci ( n-1 ) + Fibonacci ( n-2 ) ;
 }
```



快乐刷题

- P19 斐波那契数列
- P63 兔子繁殖
- P7 走楼梯
- P64 平面分割
- P391 骨牌铺法
- P392 位数问题
- P399 倒序数
- P398 求最大公约数
- P397 十进制数转换成八进制数