# 1.1 算法基础

## 1. 时间复杂度

### 1.1常见的时间复杂度（按效率排序）

- O(1)<O(logn)<O(n)<O(nlogn)<O($n^2$)<O($n^2$logn)<O($n^3$)

### 1.2不常见的时间复杂度

- O(n!) O($2^n$) O($n^n$)

### 1.3 一眼能判断的时间复杂度

- 循环减半过程 O(logn)

  例子：O(logn)

  ```
  while n > 1:
      print(n)
      n = n //2
  ```

- 几次循环就是n的几次方的复杂度

  例子：O（$n^2$）

```
for i in range(n):
    for j in range(n):
        print("xxx")
```

## 2. 空间复杂度

例如开辟了一块n*n的空间， 则S（n) = O($n^2$)

## 3. 复习：递归

### 3.1 递归的两个特点：

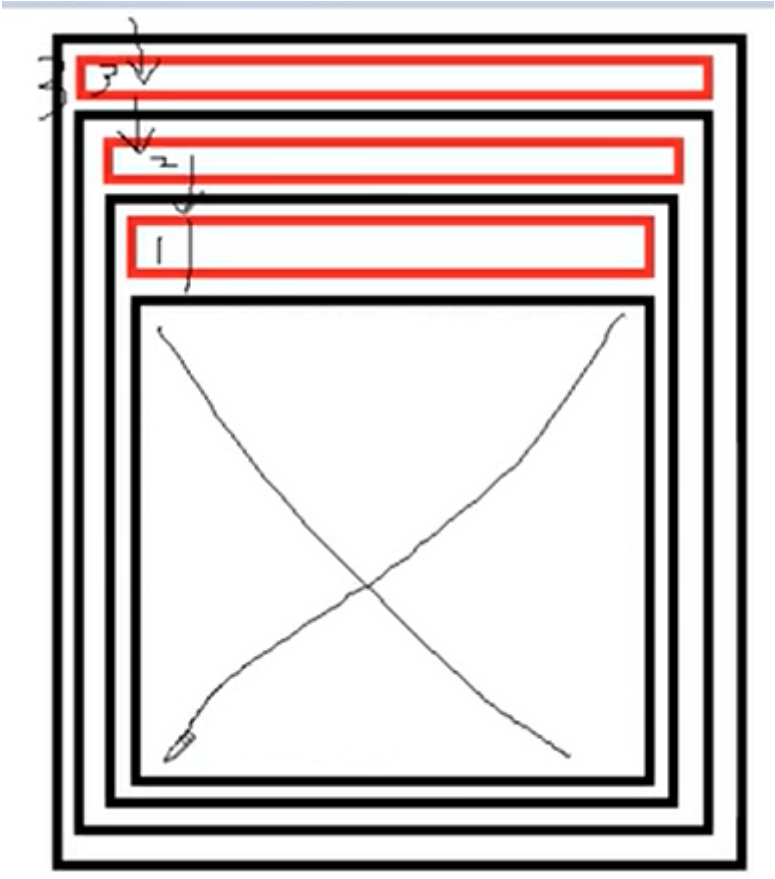
- 调用自身
- 结束条件

**例子：**

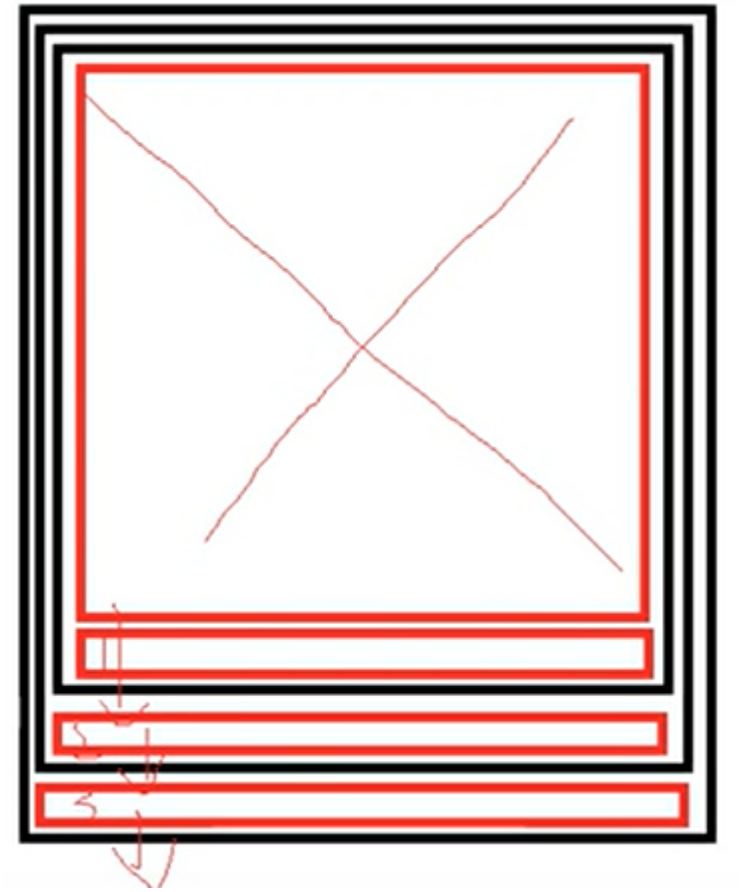
```
x = 3
def func3(x):
    if x > 0:
        print(x)
        func3(x-1) #  3 2 1

def func4(x):
    if x > 0:
        func4(x-1)
        print(x) #  1 2 3
```

**对例子的解释：**

func3：（红色部分为print（x），最外边的框是第一次）



func4：（红色部分为print（x），最外边的框是第一次，最里面的大红色框应该为黑色）



## 3.2 斐波那契数列

斐波那契数列　例如：1 1 2 3 5 8.....

### 3.2.1递归表达式

$$F(n) = F(n-1) + F(n-2), (with F(0) = 1, F(1) = 1), n = 2, 3, 4...$$

```
def fibnacci(n):
    if n == 0 or n == 1:
        return 1
    else:
        return fibnacci(n-1) + fibnacci(n-2)

print(fibnacci(4)) # 5
```

**递归粗略时间**

时间装饰器:

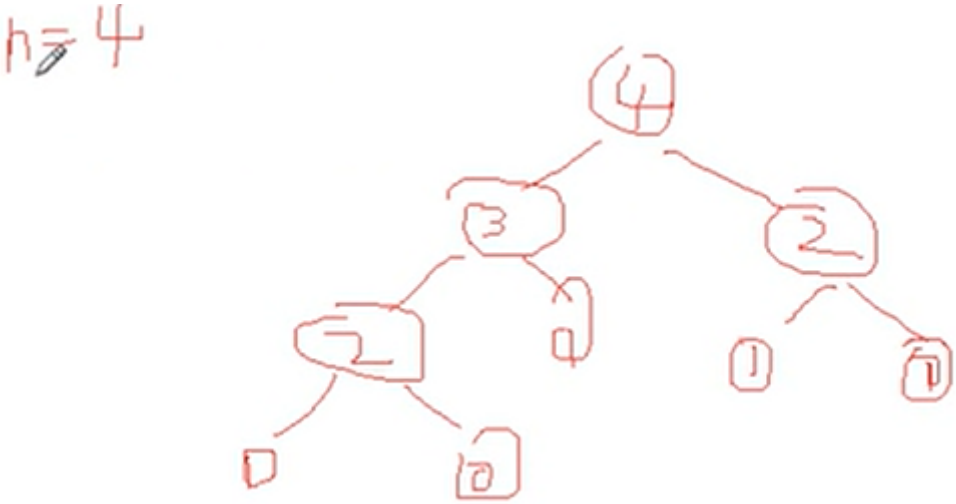
cal_time.py

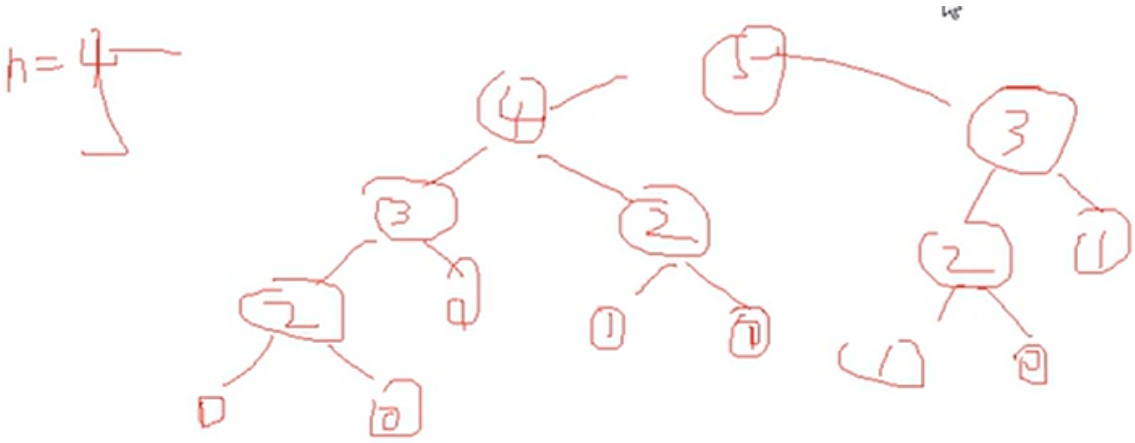
```
import time


def cal_time(func):
    def wrapper(*args, **kwargs):
        t1 = time.time()
        result = func(*args, **kwargs)
        t2 = time.time()
        print("%s runing time: %s secs." % (func.__name__, t2-t1))
        return result
    return wrapper
```

recursion.py

```
![1.1.4](C:\Users\45169\Desktop\在写博文\数据结构与算法\1.1.4.png)from cal_time
import *



def fibnacci(n):
    if n == 0 or n == 1:
        return 1
    else:
        return fibnacci(n-1) + fibnacci(n-2)

@cal_time
def fib(n):
    return fibnacci(n)

print(fib(34)) # fib running time: 2.619 secs
```

**递归算法的时间复杂度O（$2^n$）**





从上面两张图可知， n增加1，所需计算量翻倍，所以时间复杂度是O（$2^n$）

### 3.2.2 非递归

**普通**

```
@cal_time
def fib2(n):
    li = [1, 1]
    for i in range(2, n+1):
    li.append(li[-1]+li[-2])
    return li[n]


print(fib2(100))
```

时间复杂度O（n），空间复杂度O（n）

**优化**

我们不需要完整的li，空间有浪费，我们可优化空间复杂度，代码：

```
@cal_time
def fib3(n):
    a = 1
    b = 1
    c = 0
    for i in range(2, n + 1):
        c = a + b
        a = b
        b = c
    return c


print(fib3(100))
```

时间复杂度O（n），空间复杂度O（1）

### 3.2.3 通项表达式

$a_n = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n]$, n = 1, 2, 3...

时间复杂度O（1），空间复杂度O（1）

用Python代码运算时， 因为根号不精确， 当 n >= 70 之后， $a_n$ 误差会明显

## 3.3 递归：面试题

一段有n个台阶组成的楼梯，小明从楼梯的最底层向最高处前进， 它可以选择一次迈一级台阶或者一次迈两级台阶。问： 他有多少种不同的走法?

F（n）=F（n-1）+F（n-2）

F(1) = 1, F(2) = 1 , F(3) = 2 ...