

6. 递归的实现、特性以及思维要点

为什么树的面试题解法一般都是递归?

1. 树节点数据结构的定义就是用递归的方式来进行的;
2. 树、二叉树、二叉搜索树在定义其数据结构和算法时, 也是有所谓的重复性 (自相似性) .

递归 Recursion

- 定义:

递归 - 循环, 即, 通过函数体来进行的循环

- 举例说明:

- 例子1:

1. 从前有个山
2. 山里有个庙
3. 庙里有个和尚讲故事
4. 返回 1

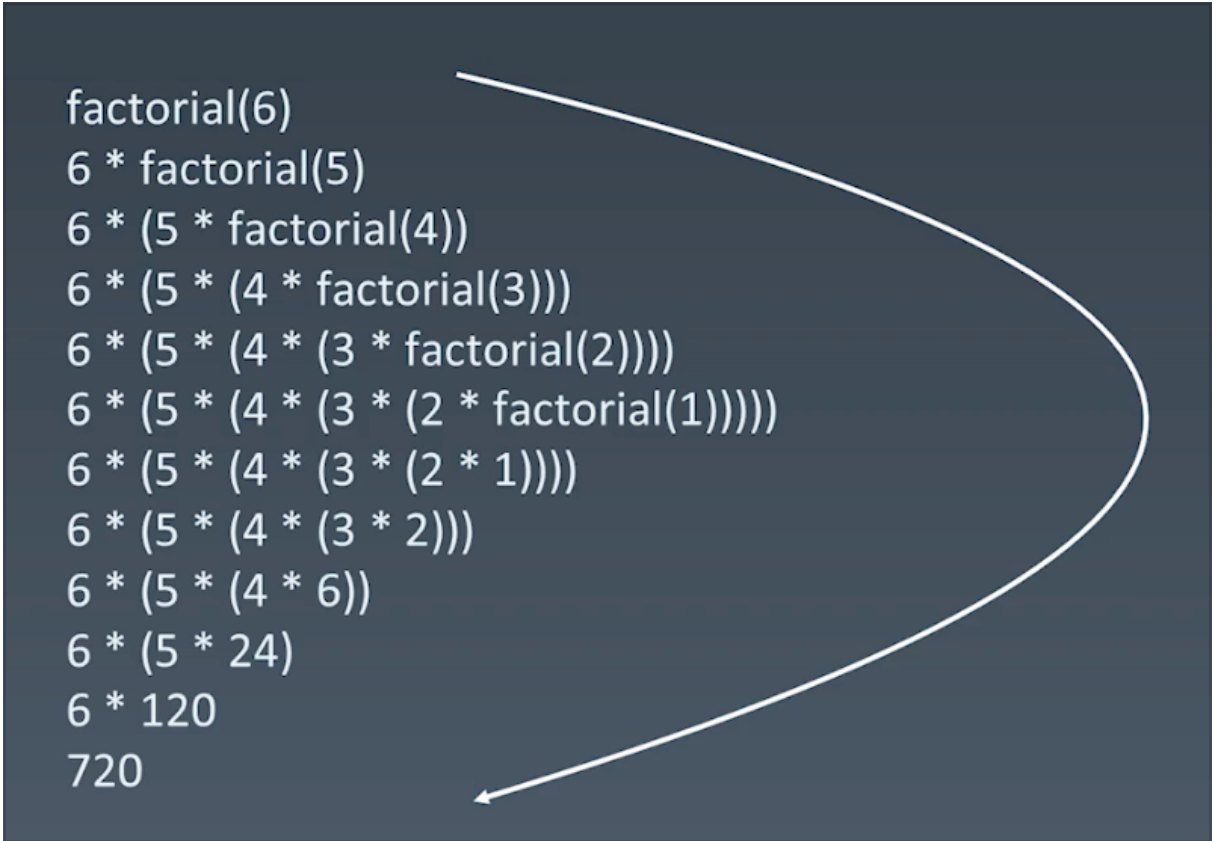
- 例子2: 盗梦空间

1. 向下进入到不同梦境中;
2. 向上又回到原来一层;
3. 通过声音同步回到上一层;
4. 每一层的环境和周围的人都是一份拷贝、主角等几个人穿越不同层级的梦境 (发生和携带变化) .

- 阶乘例子:

$$n! = 1 * 2 * 3 * \dots * n$$

```
def Factorial(n):  
    if n <= 1:  
        return 1  
    return n * Factorial(n-1)
```



```
factorial(6)
6 * factorial(5)
6 * (5 * factorial(4))
6 * (5 * (4 * factorial(3)))
6 * (5 * (4 * (3 * factorial(2))))
6 * (5 * (4 * (3 * (2 * factorial(1)))))
6 * (5 * (4 * (3 * (2 * 1))))
6 * (5 * (4 * (3 * 2)))
6 * (5 * (4 * 6))
6 * (5 * 24)
6 * 120
720
```

- 递归代码模板

```
def recursion(level, param1, param2, ...):

    # recursion terminator
    # 递归终止条件
    if level > MAX_LEVEL:
        # process_result
        # 处理结果
        return

    # process logic in current level
    # 处理当前层逻辑
    process(level, data...)

    # drill down
    # 下探到下一层
```

```
self.recursion(level+1,p1,...)
```

```
# reverse the current level status if needed
```

```
# 清理当前层
```

```
public void recursion(int level, int param){
```

```
    // terminator
```

```
    if(level>MAX_LEVEL){
```

```
        // process result
```

```
        return;
```

```
    }
```

```
    // process current logic
```

```
    process(level,param);
```

```
    // drill down
```

```
    recursion(level:level+1,newParam);
```

```
    // restore current status
```

```
}
```

- 思维要点：

1. 不要人肉递归（最大误区）；
2. 找到最近最简方法，将其拆解成可重复解决的问题（重复子问题）；
3. 数学归纳法的思维。
4. 当前层只考虑当前层的问题，不下下探太多，人脑不擅长人肉递归。

#Algorithm/Part II : Theory/Data Structure#