



数据结构与算法（ Python ）-05/第6周

北京大学 陈斌

2021.04.13

线下课堂

- › 本周内容小结
- › 几个递归问题讨论
- › 问题答疑
- › 慕课作业讲解
- › 课堂练习



W05 : 递归 (上)

- › 401 什么是递归 13m20s
- › 402 递归的应用：任意进制转换 6m32s
- › 403 递归调用的实现 8m32s
- › 404 递归可视化：分形树 10m54s
- › 405 递归可视化：谢尔宾斯基三角 10m31s
- › 406 递归的应用：汉诺塔 10m37s
- › 407 递归的应用：探索迷宫 16m35s

401 什么是递归

❖ 递归是一种解决问题的方法，其精髓在于将问题分解为规模更小的相同问题，持续分解，直到问题规模小到可以用非常简单直接的方式来解决。

递归的问题分解方式非常独特，其算法方面的明显特征就是：在算法流程中调用自身。



递归 “三定律”

❖ 为了向阿西莫夫的“机器人三定律”致敬，递归算法也总结出“三定律”

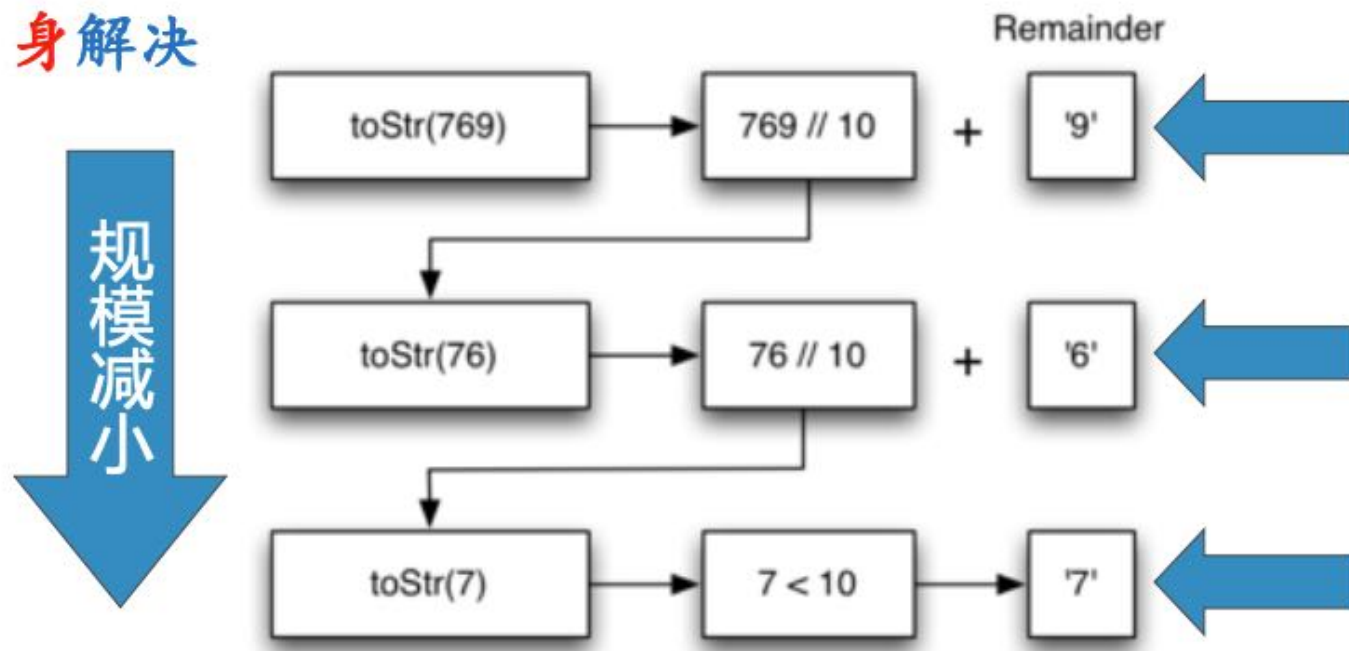
- 1，递归算法必须有一个基本结束条件（最小规模问题的直接解决）
- 2，递归算法必须能改变状态向基本结束条件演进（减小问题规模）
- 3，递归算法必须调用自身（解决减小了规模的不同问题）

402 递归的应用：任意进制转换

❖ 问题就分解为：

余数总小于“进制基base”，是“基本结束条件”，可直接进行查表转换

整数商成为“更小规模”问题，通过递归调用自身解决

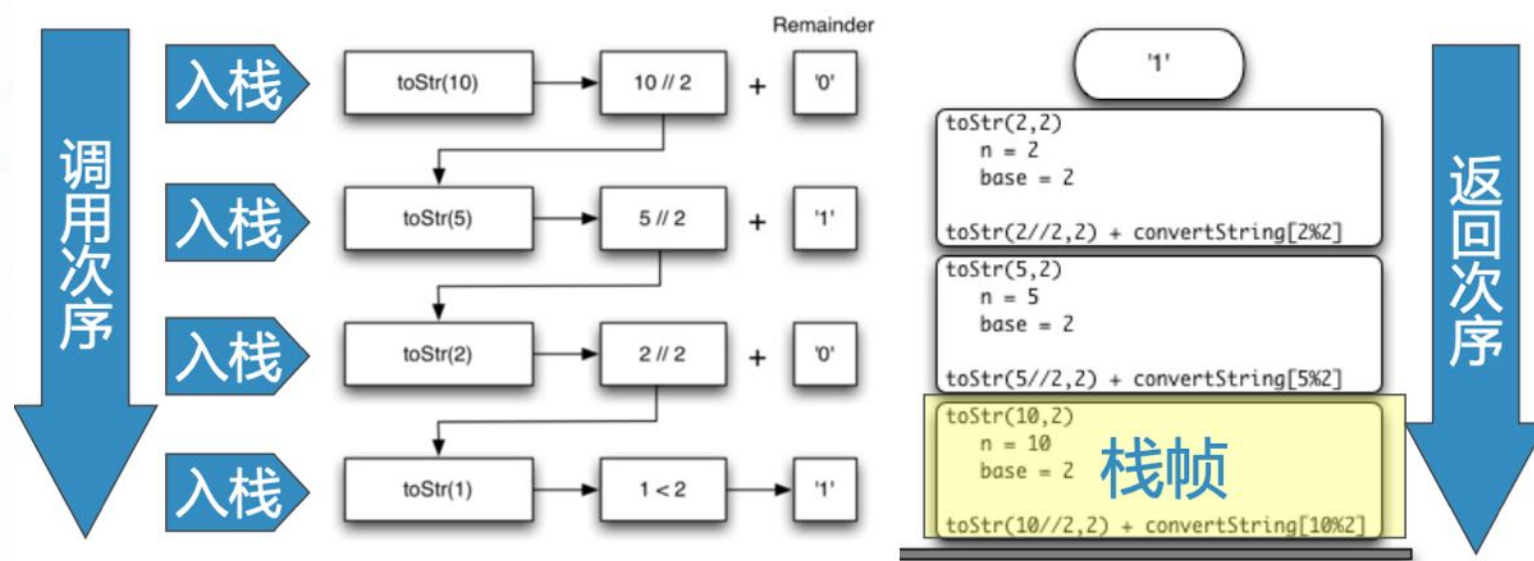


递归调用的实现

❖ 当一个函数被调用的时候，系统会把调用时的**现场数据**压入到**系统调用栈**

每次调用，压入栈的现场数据称为**栈帧**

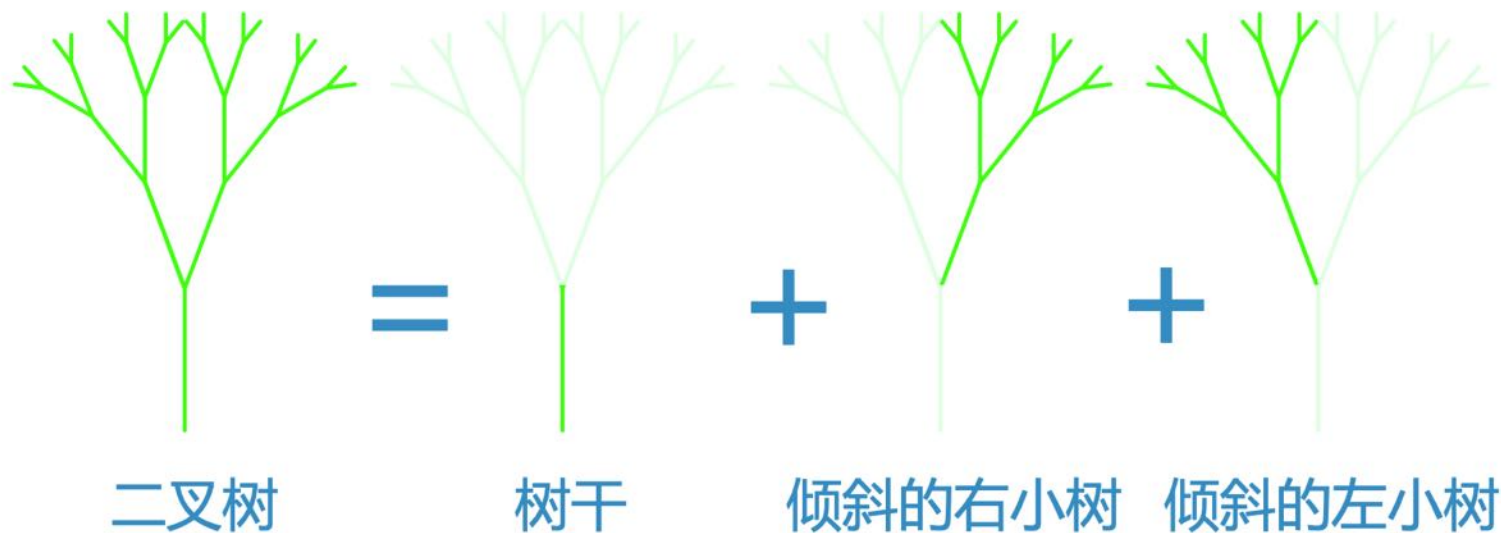
当函数返回时，要从调用栈的栈顶取得返回地址，恢复现场，弹出栈帧，按地址返回。



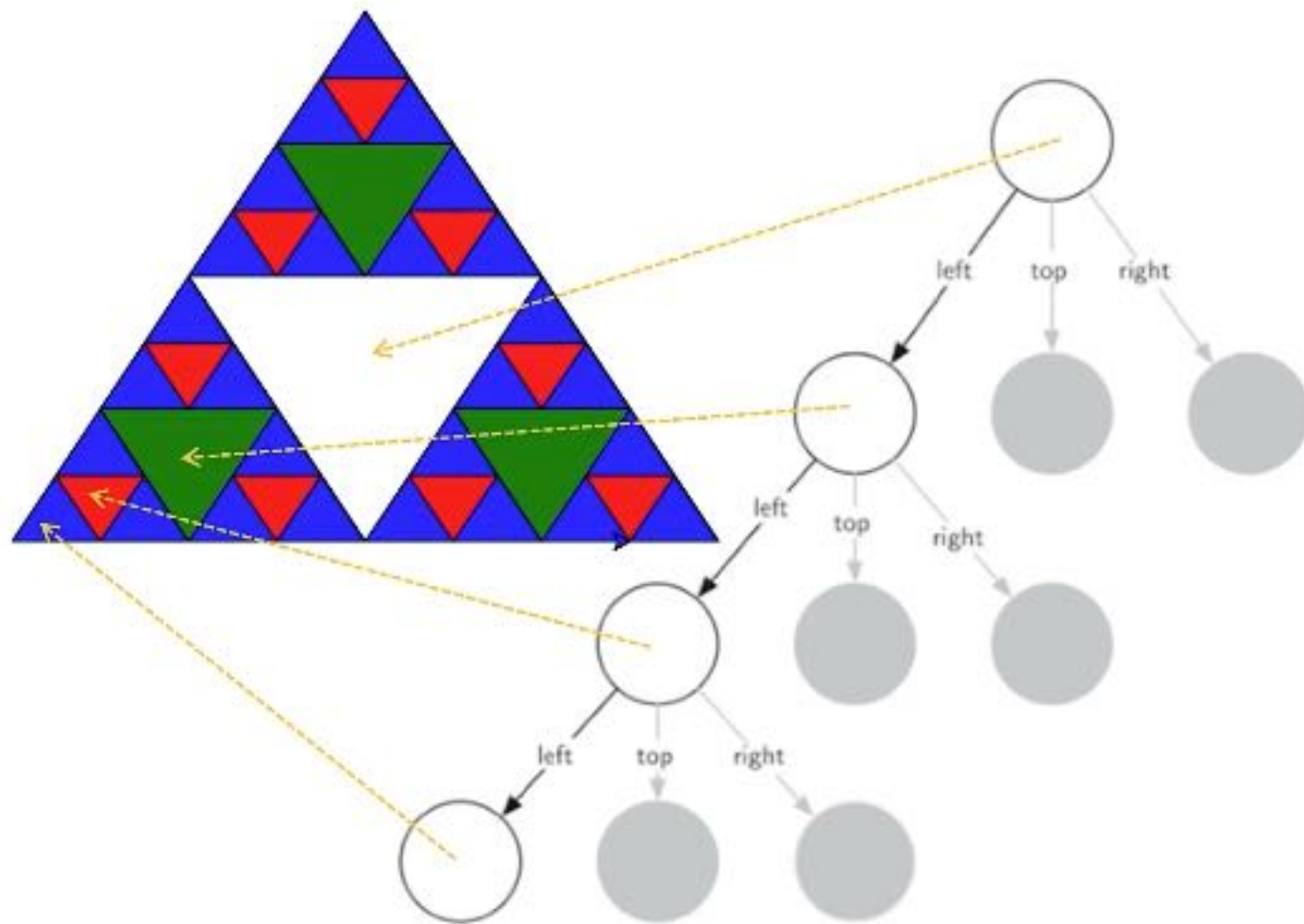
404 递归可视化：分形树

❖ 这样，我们可以把树分解为三个部分：树干、左边的小树、右边的小树

分解后，正好符合递归的定义：对自身的调用



405 递归可视化：谢尔宾斯基三角



406 递归的应用：汉诺塔

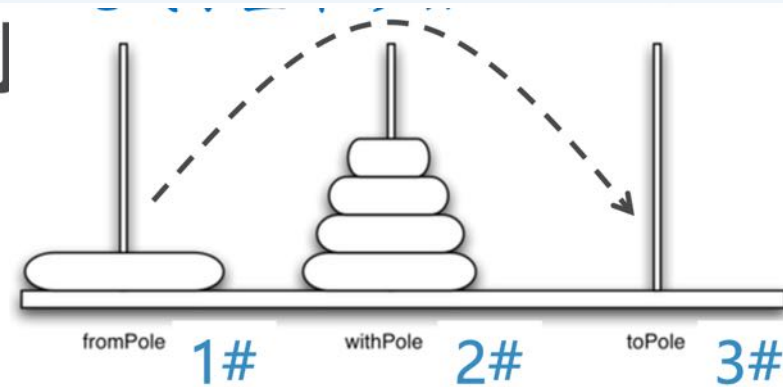
❖ 将盘片塔从**开始柱**，经由**中间柱**，移动到**目标柱**：

首先将上层 $N-1$ 个盘片的盘片塔，从**开始柱**，经由**目标柱**，移动到**中间柱**；

然后将第 N 个（最大的）盘片，从**开始柱**，移动到**目标柱**；

最后将放置在**中间柱**的 $N-1$ 个盘片的盘片塔，经由**开始柱**，移动到**目标柱**。

❖ 基本结束条件，也就是最小规模问题是： 1个盘片的移动问题



407 递归的应用：探索迷宫

❖ 这样，探索迷宫的递归算法思路如下：

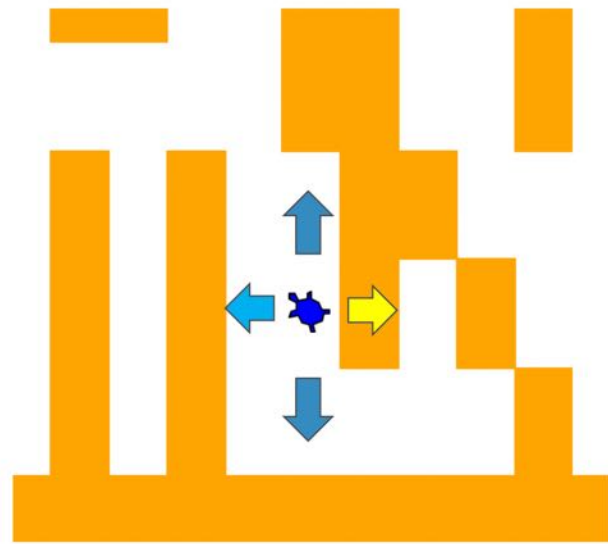
将海龟从原位置向北移动一步，以新位置递归调用探索迷宫寻找出口；

如果上面的步骤找不到出口，那么将海龟从原位置向南移动一步，以新位置递归调用探索迷宫；

如果向南还找不到出口，那么将海龟从原位置向西移动一步，以新位置递归调用探索迷宫；

如果向西还找不到出口，那么将海龟从原位置向东移动一步，以新位置递归调用探索迷宫；

如果上面四个方向都找不到出口，那么这个迷宫没有出口！



407 递归的应用：探索迷宫

❖ 递归调用的“基本结束条件”归纳如下：

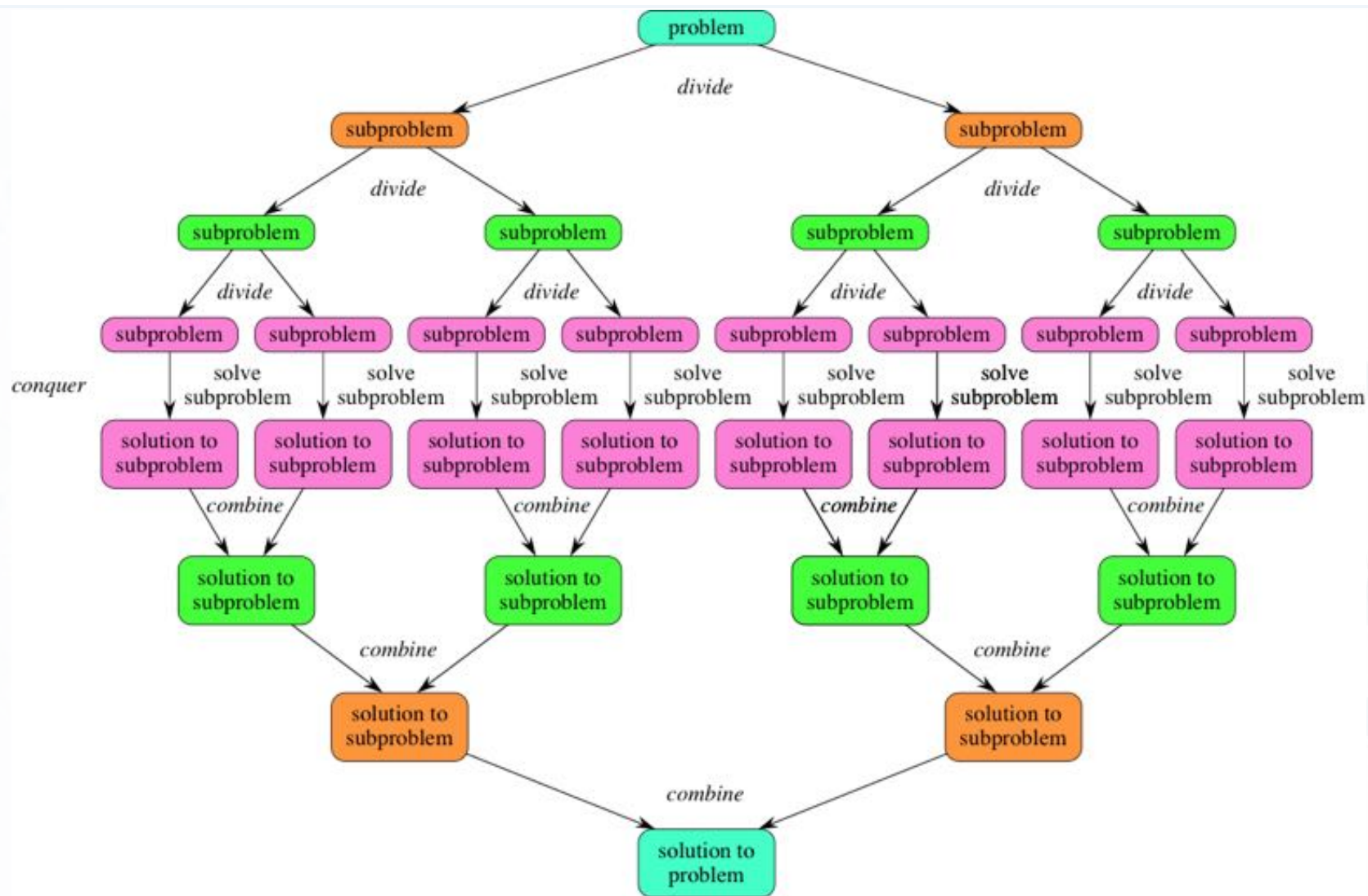
海龟碰到“**墙壁**”方格，递归调用结束，返回**失败**；

海龟碰到“**面包屑**”方格，表示此方格已访问过，递归调用结束，返回**失败**；

海龟碰到“**出口**”方格，即“位于边缘的通道”方格，递归调用结束，返回**成功**！

海龟在**四个方向上探索都失败**，递归调用结束，返回**失败**

递归和分治策略



几个递归问题分析：二分查找

› 有序表OrderedList中的查找可以采用“二分查找”递归算法

› 前提：列表中的数据项是有序的

› 基本结束条件

如果列表中没有数据项，返回“没找到”

› 缩小规模

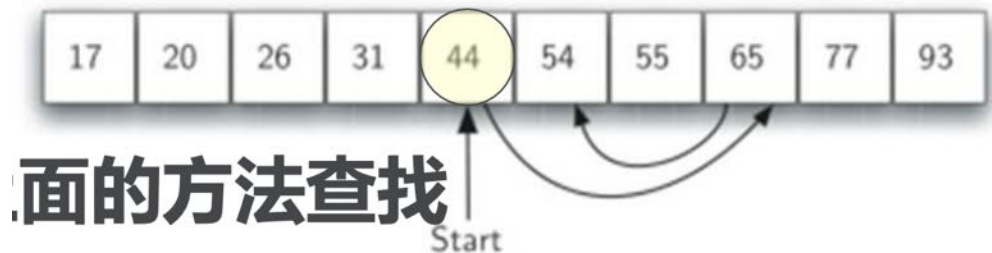
与列表中央的数据项对比，如果相等，返回“找到”

如果查找项较小，规模缩小到前半列表；

如果查找项较大，规模缩小到后半列表；

› 递归调用

对缩小后的列表进行“二分查找”



几个递归问题分析：归并排序

基本结束条件

如果列表中只有1个数据项，则已经排好序

缩小规模

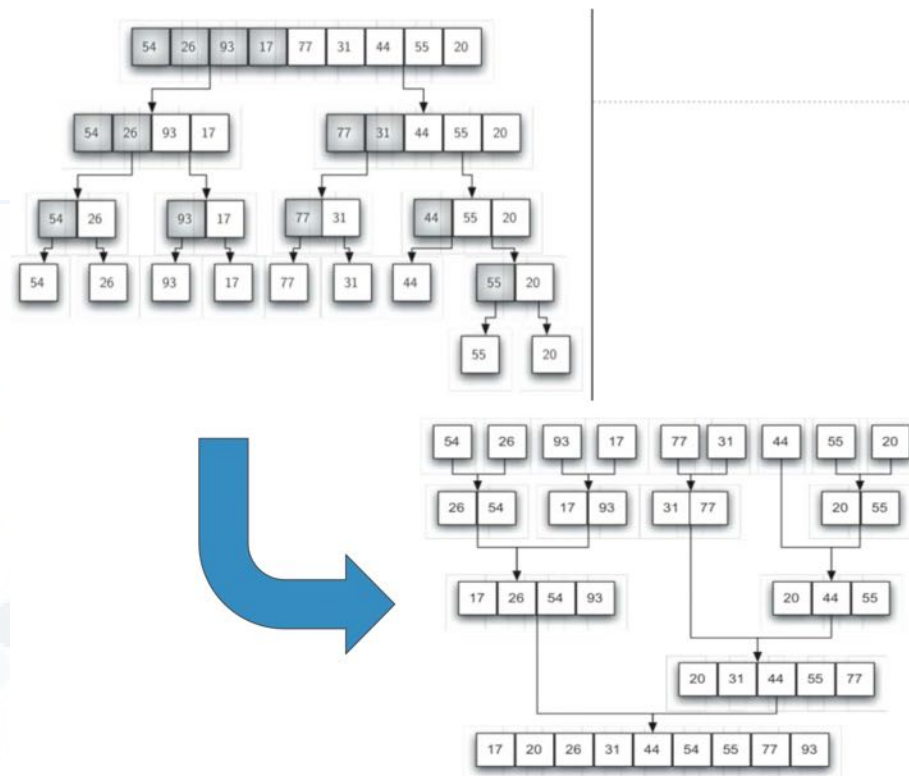
把列表分为前半和后半

递归调用

对前半进行“归并排序”

对后半进行“归并排序”

将排好序的两半合并起来



问题答疑

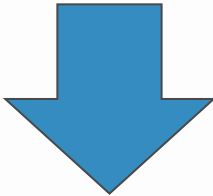
- › 希望老师详细讲解一下四柱汉诺塔的解析，看了一些资料，还是不是很明白
- › 同样想问一下四柱汉诺塔要怎么优化才能过掉最后一个用例
- › 请问ASCII谢尔宾斯基地毯的思路是啥，似乎没法正常递归？
- › 想知道绘制谢尔宾斯基方形时，若采用坐标方法判定某点是否为空格，该如何确定空格坐标（ x,y ）的判定条件

问题答疑

› 为什么这里无法使用递归呢？

```
def __getitem__(self, start=None, end=None, step=None):  
    if start==None:  
        start=0  
    if end==None:  
        end=self.length-1  
    if step==None:  
        step=1  
    ans=DoublyLinkedList()  
    cr=self.head  
    pos=start  
    while pos+step<=end:  
        ans.append(self.get_one(pos))  
        return __getitem__(self, pos+step, end, step)  
    else:  
        return ans
```

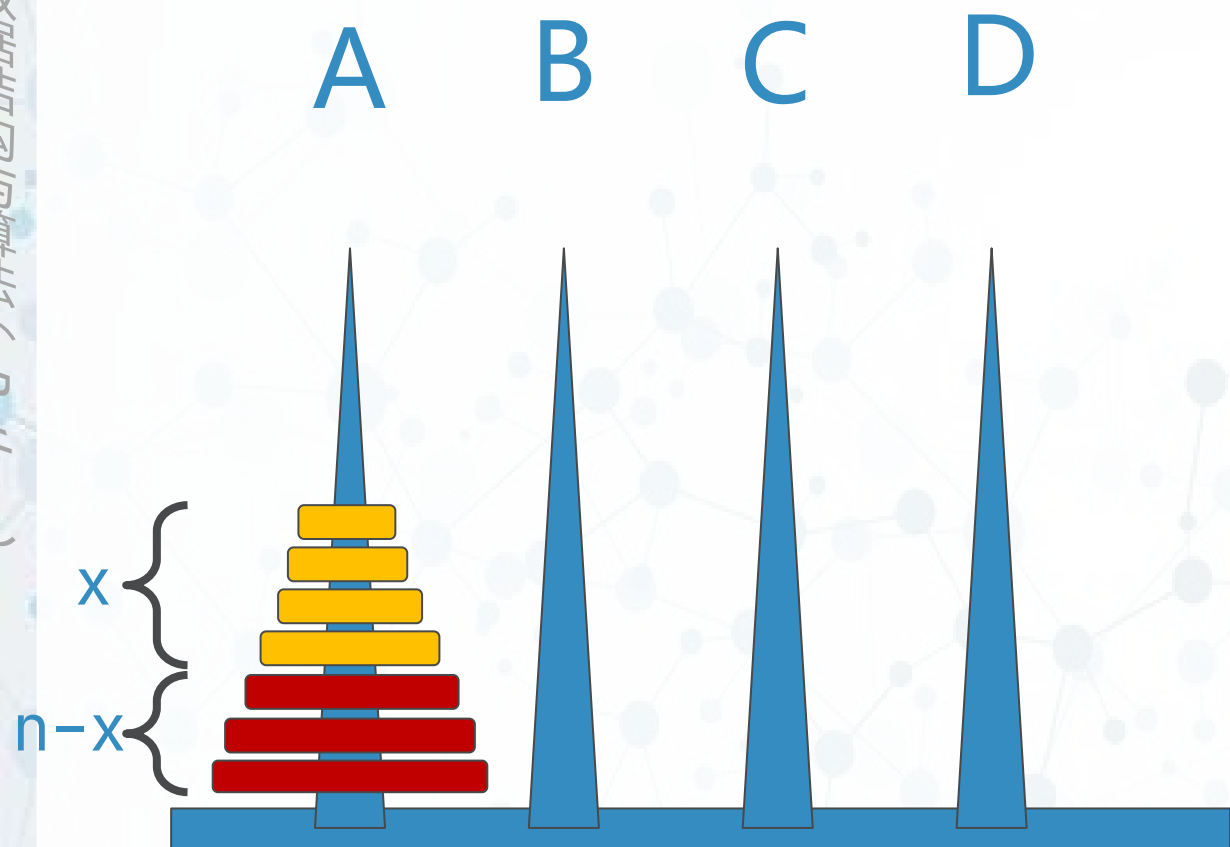
self[pos+step:end:step]
或
self.__getitem__(pos+step, end, step)
或
DoublyLinkedList.__getitem__(self, pos...)



慕课W05作业：W05-1进制转换

```
1 # W05-1, 进制转换
2 m, n = [int(x) for x in input().split()]
3 k = input()
4 chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
5
6 def nbase2int(k, n): # n进制转为整数, k是字符串
7     if len(k) == 1:
8         return chars.index(k)
9     else:
10        return nbase2int(k[:-1], n) * n + chars.index(k[-1])
11
12 def int2nbase(k, n): # 整数转为n进制, k是整数
13     if k < n:
14         return chars[k]
15     else:
16         return int2nbase(k // n, n) + chars[k % n]
17
18 print(int2nbase(nbase2int(k, m), n))
```

慕课W05作业：W05-2四柱汉诺塔



- 1, x 个盘子 $A-(B/D) \rightarrow C$
- 2, $n-x$ 个盘子 $A-(B) \rightarrow D$
- 3, x 个盘子 $C-(A/B) \rightarrow D$

让 x 从 1 到 $n-1$, 步数取最小值

慕课W05作业：W05-2四柱汉诺塔

› 递归基本结束条件

$\text{hanoi4}(0)=0, \text{hanoi4}(1)=1, \text{hanoi4}(2)=3$

› 缩小规模

对于取定的 x 值（小于 n ）

第一步步数就是 $\text{hanoi4}(x)$

第二步步数就是 $\text{hanoi3}(n-x)=2^{n-x}-1$

第三步步数还是 $\text{hanoi4}(x)$

三步相加的 $H(x)=2*\text{hanoi4}(x)+2^{n-x}-1$

› 递归调用

$\text{hanoi4}(n) = \min([H(x) \text{ for } x \text{ in range}(1,n)])$

慕课W05作业：W05-2四柱汉诺塔

```
1 # W05-2 四柱汉诺塔
2 n = int(input())
3
4 def hanoi4(n):
5     if n == 0:
6         return 0
7     elif n == 1:
8         return 1
9     elif n == 2:
10        return 3
11    else:
12        H = []
13        for x in range(1, n):
14            H.append(2 * hanoi4(x) + 2 ** (n - x) - 1)
15        return min(H)
16
17 print(hanoi4(n))
```

最后一个用例会超时？

慕课W05作业：W05-2四柱汉诺塔

```
3 h4cache = [None for x in range(n + 1)]
4
5 def hanoi4(n):
6     if h4cache[n]:
7         return h4cache[n]
8     if n == 0:
9         h4cache[n] = 0
10    elif n == 1:
11        h4cache[n] = 1
12    elif n == 2:
13        h4cache[n] = 3
14    else:
15        H = []
16        for x in range(1, n):
17            H.append(2 * hanoi4(x) + 2 ** (n - x) - 1)
18        h4cache[n] = min(H)
19    return h4cache[n]
```

好起来了！

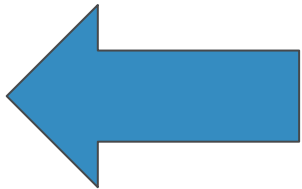
函数值缓存：lru_cache装饰器

- › 来自functools模块的lru_cache装饰器
- › 对产生大量重复计算的递归函数自动提供函数值缓存

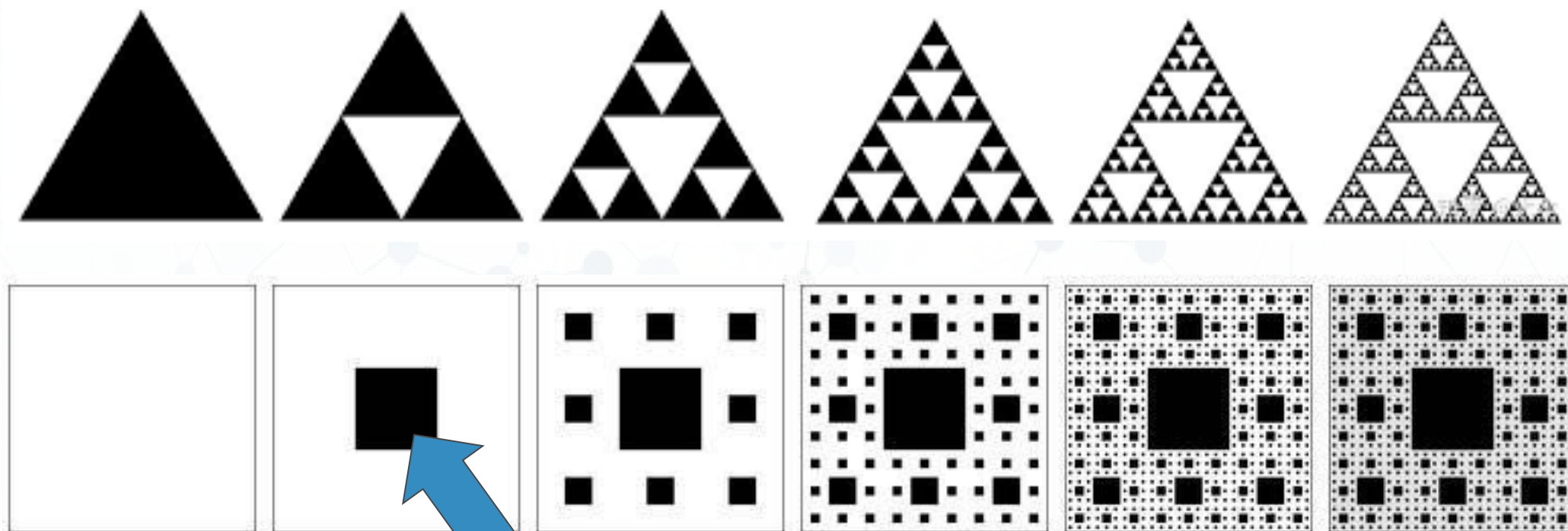
```
1 from functools import lru_cache
2
3 @lru_cache(maxsize=128) ←
4 def fibonacci(n):
5     if n == 1:
6         return 0
7     elif n == 2:
8         return 1
9     else:
10         return fibonacci(n - 1) + fibonacci(n - 2)
11
12 print(fibonacci(100))
```

W05-2四柱汉诺塔/缓存版

```
4 from functools import lru_cache
5
6 @lru_cache(maxsize=128)
7 def hanoi4(n):
8     if n == 0:
9         return 0
10    elif n == 1:
11        return 1
12    elif n == 2:
13        return 3
14    else:
15        H = []
16        for x in range(1, n):
17            H.append(2 * hanoi4(x) + 2 ** (n - x) - 1)
18        return min(H)
```



慕课W05作业：W05-3 ASCII谢尔宾斯基地毯

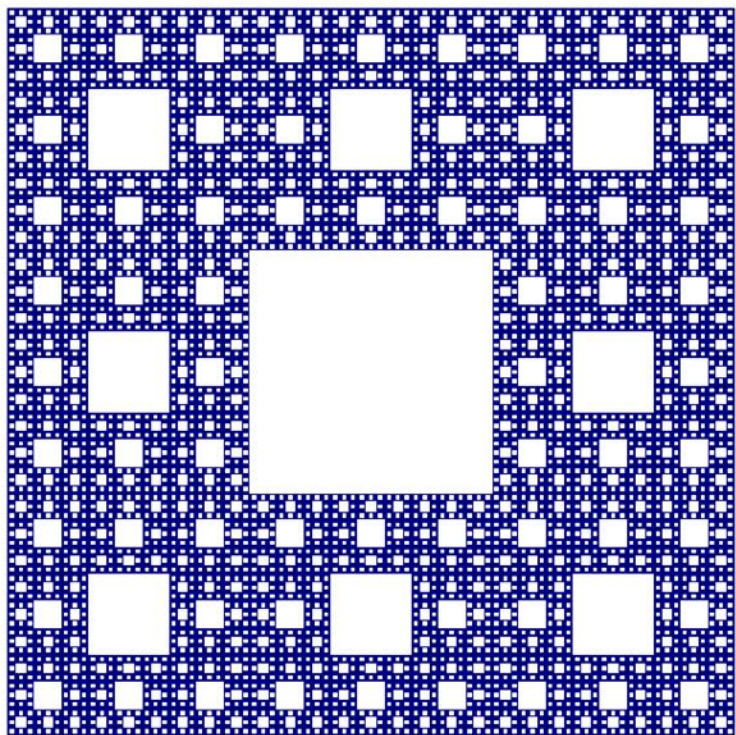


挖掉

```
1 # W05-3 ASCII谢尔宾斯基地毯
2 n = int(input()) # 阶数
3 ch = input() # 构成字符
4 blank = " " * len(ch) # 构成空白
5 pic = [[ch for col in range(n)] for row in range(n)] # 画板
6
7 def spski(n, top, left): # n阶, 左上角的行列数
8     if n == 1: # 基本结束条件
9         return
10    # 分为3行3列, 挖掉中心, 其余递归n//3
11    for row in range(3):
12        for col in range(3):
13            if row == 1 and col == 1: # 挖空
14                for r1 in range(n // 3):
15                    for c1 in range(n // 3):
16                        pic[top + n // 3 + r1][left + n // 3 + c1] = blank
17            else: # 递归n//3
18                spski(n // 3, top + row * n // 3, left + col * n // 3)
19
20 spski(n, 0, 0) # 挖n阶
21 for r in range(n):
22     print("".join(pic[r]))
```

【K05】课堂作业：绘制谢尔宾斯基地毯

- › 写出一个递归算法，用海龟作图绘制谢尔宾斯基地毯。
- › 将代码和运行结果截图做成文档提交。



下课！

