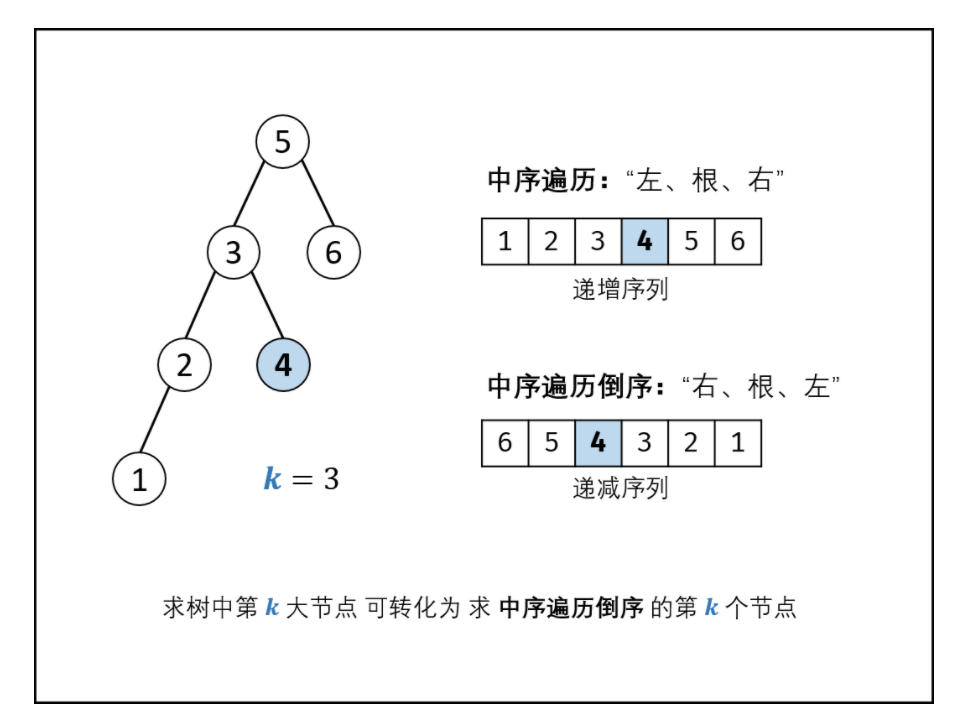
剑指offer-归类

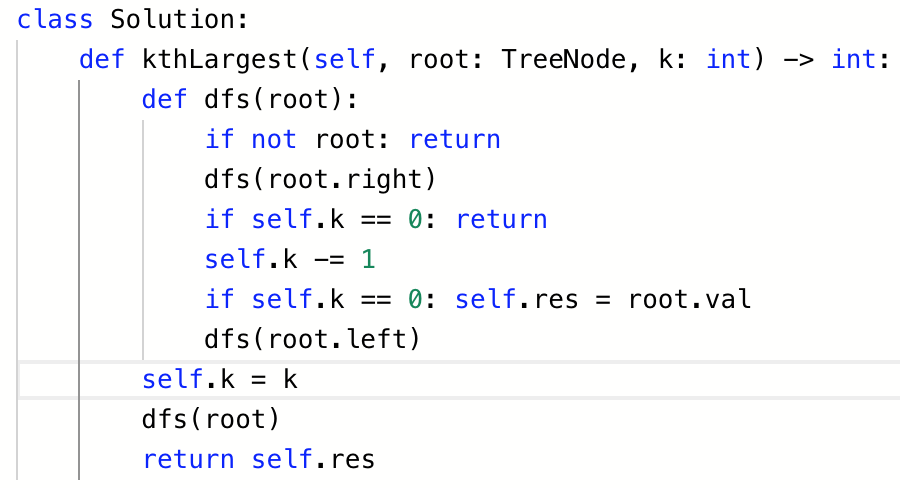
# 二叉树

## 1.[二叉搜索树的第k大节点](https://leetcode-cn.com/problems/er-cha-sou-suo-shu-de-di-kda-jie-dian-lcof/)

给定一棵二叉搜索树，请找出其中第k大的节点。

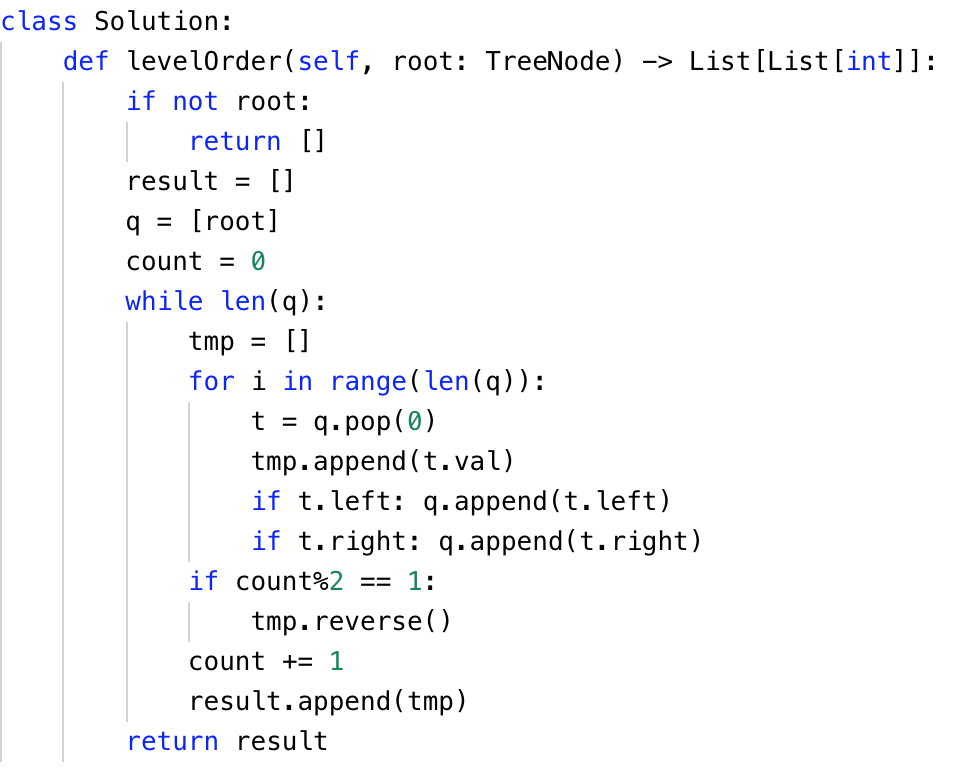
本文解法基于此性质：二叉搜索树的中序遍历为 递增序列 。





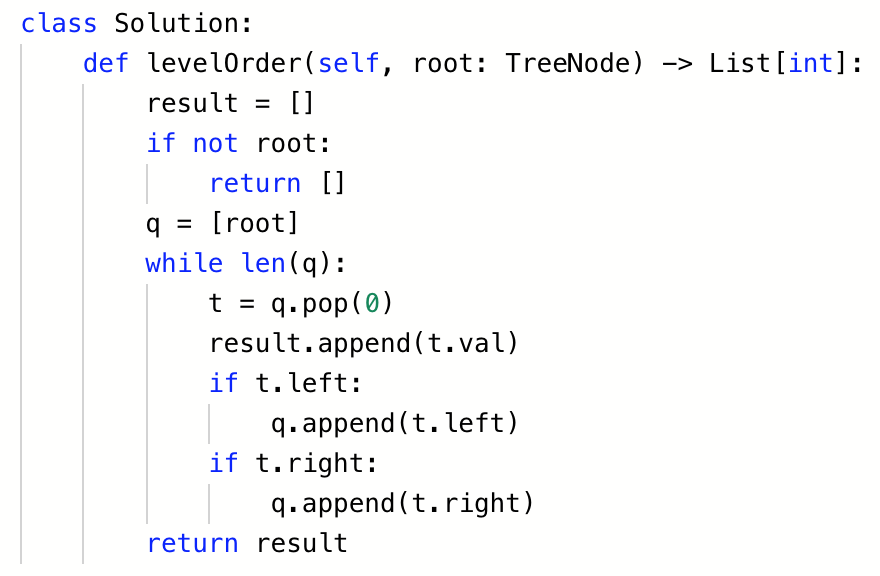
## 2. 之字形[从上到下打印二叉树 III](https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-iii-lcof/)

请实现一个函数按照之字形顺序打印二叉树，即第一行按照从左到右的顺序打印，第二层按照从右到左的顺序打印，第三行再按照从左到右的顺序打印，其他行以此类推。



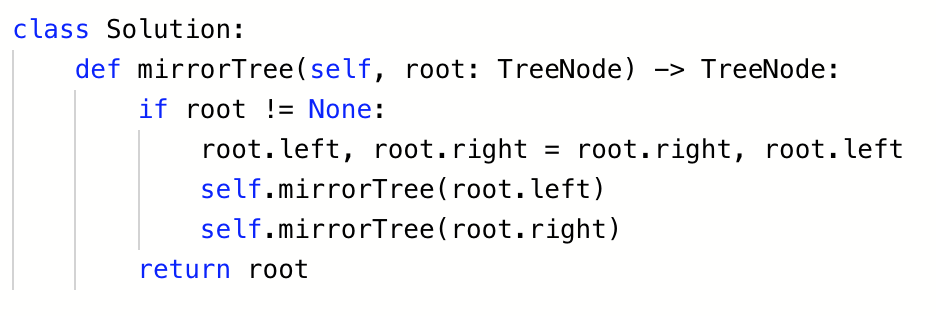
## 3.[从上到下打印二叉树](https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-lcof/)

从上到下打印出二叉树的每个节点，同一层的节点按照从左到右的顺序打印。



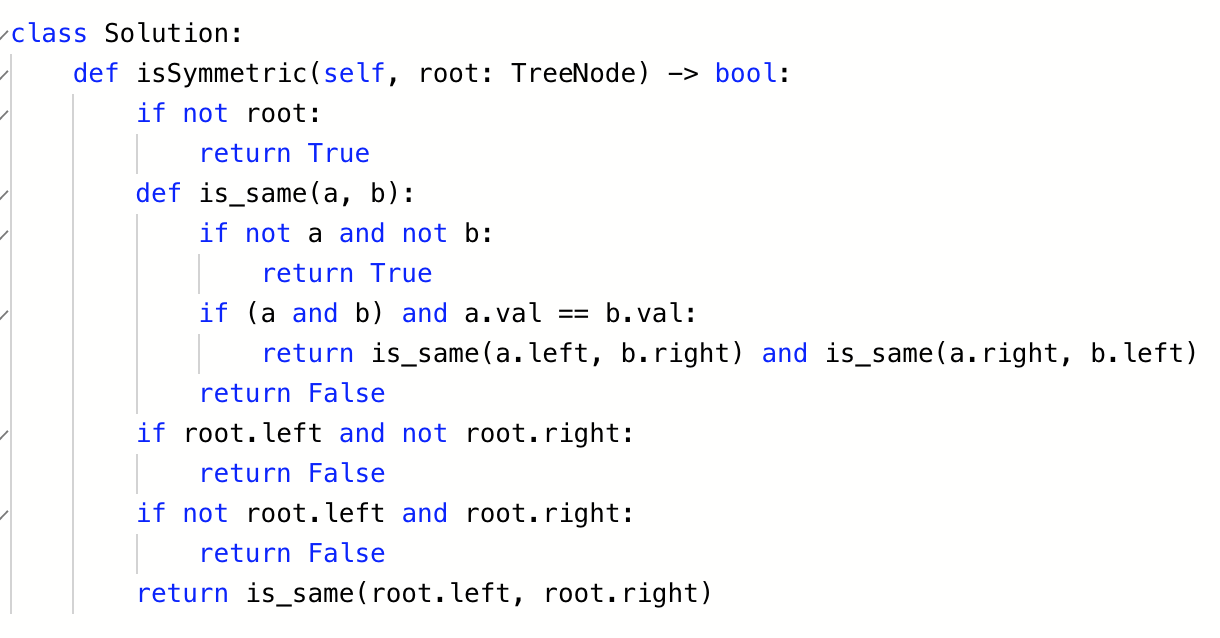
## 4.[二叉树的镜像](https://leetcode-cn.com/problems/er-cha-shu-de-jing-xiang-lcof/)

请完成一个函数，输入一个二叉树，该函数输出它的镜像。



## 5.[对称的二叉树](https://leetcode-cn.com/problems/dui-cheng-de-er-cha-shu-lcof/)

请实现一个函数，用来判断一棵二叉树是不是对称的。如果一棵二叉树和它的镜像一样，那么它是对称的。



## 6. [树的子结构](https://leetcode-cn.com/problems/shu-de-zi-jie-gou-lcof/)

输入两棵二叉树A和B，判断B是不是A的子结构。(约定空树不是任意一个树的子结构)

B是A的子结构， 即 A中有出现和B相同的结构和节点值

class Solution:

    def isSubStructure(self, A: TreeNode, B: TreeNode) -> bool:

        if not B or not A:

            return False

        return self.is\_sub(A, B) or self.isSubStructure(A.left, B) or self.isSubStructure(A.right, B)

    def is\_sub(self, A, B):

        if not B:

            return True

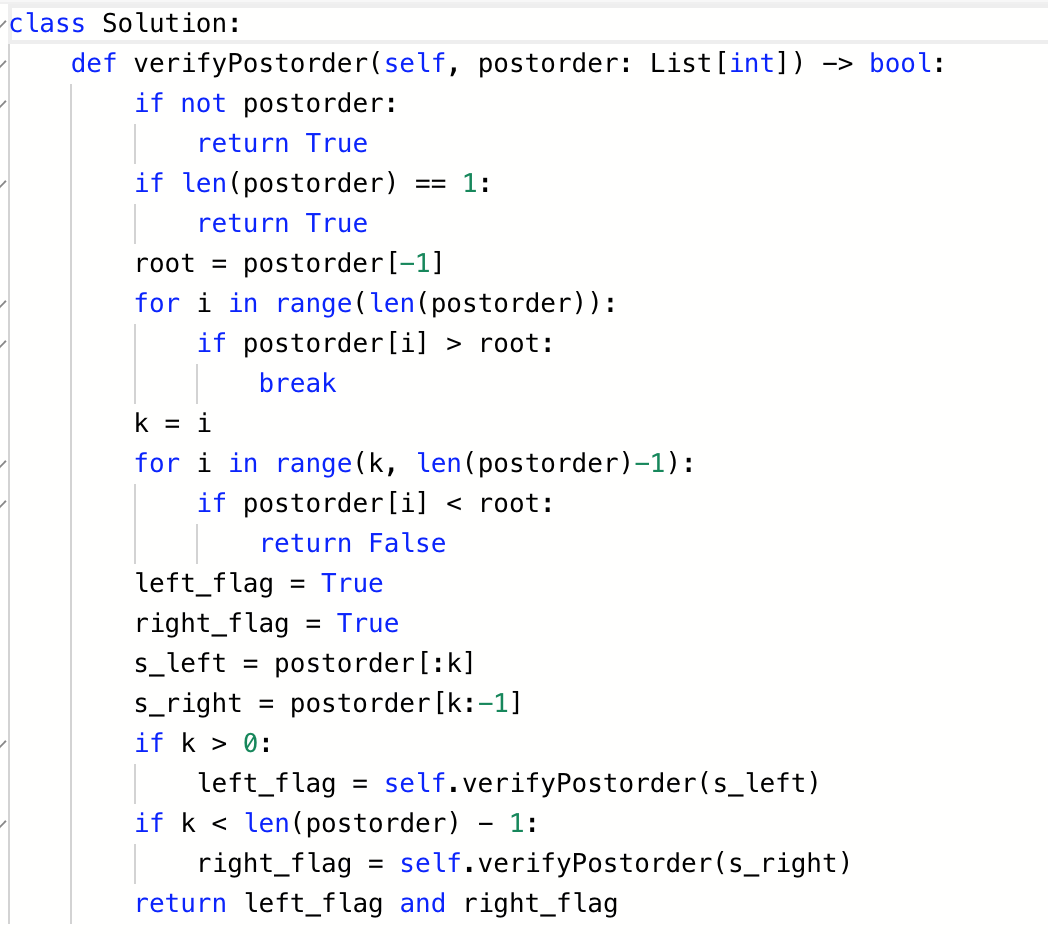
        if not A or A.val != B.val:

            return False

        return self.is\_sub(A.left, B.left) and self.is\_sub(A.right, B.right)

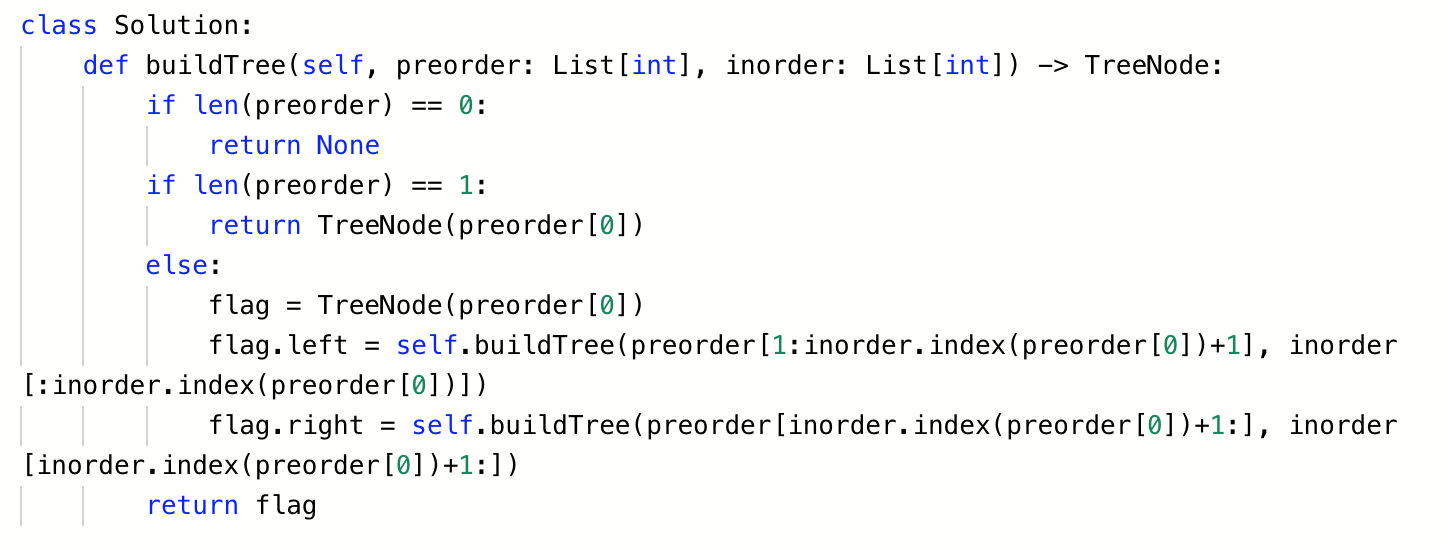
## 7.[二叉搜索树的后序遍历序列](https://leetcode-cn.com/problems/er-cha-sou-suo-shu-de-hou-xu-bian-li-xu-lie-lcof/)

输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历结果。如果是则返回 true，否则返回 false。假设输入的数组的任意两个数字都互不相同。



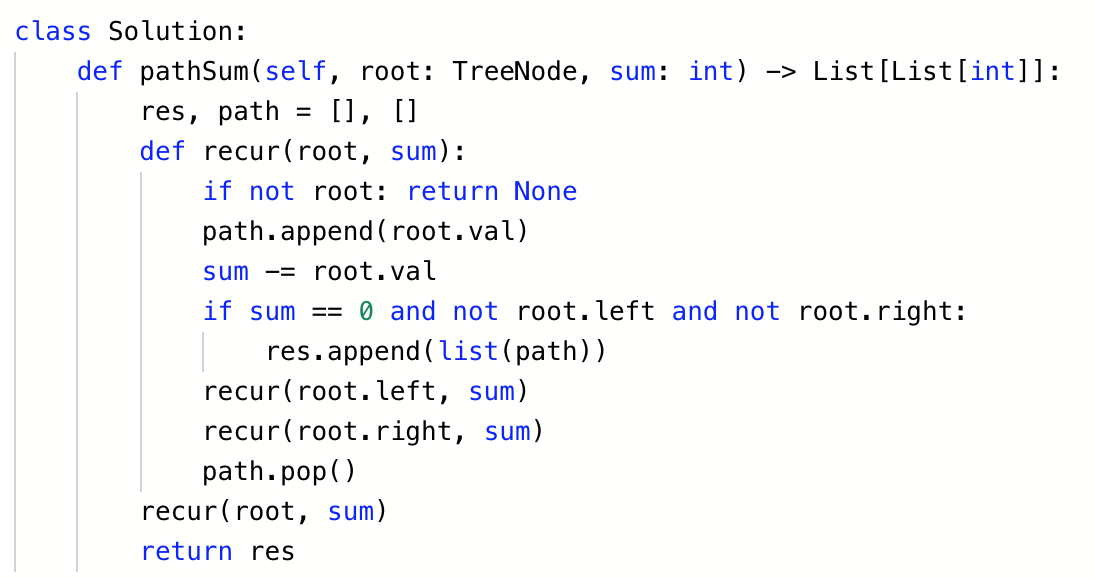
## 8.[重建二叉树](https://leetcode-cn.com/problems/zhong-jian-er-cha-shu-lcof/)

输入某二叉树的前序遍历和中序遍历的结果，请重建该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。



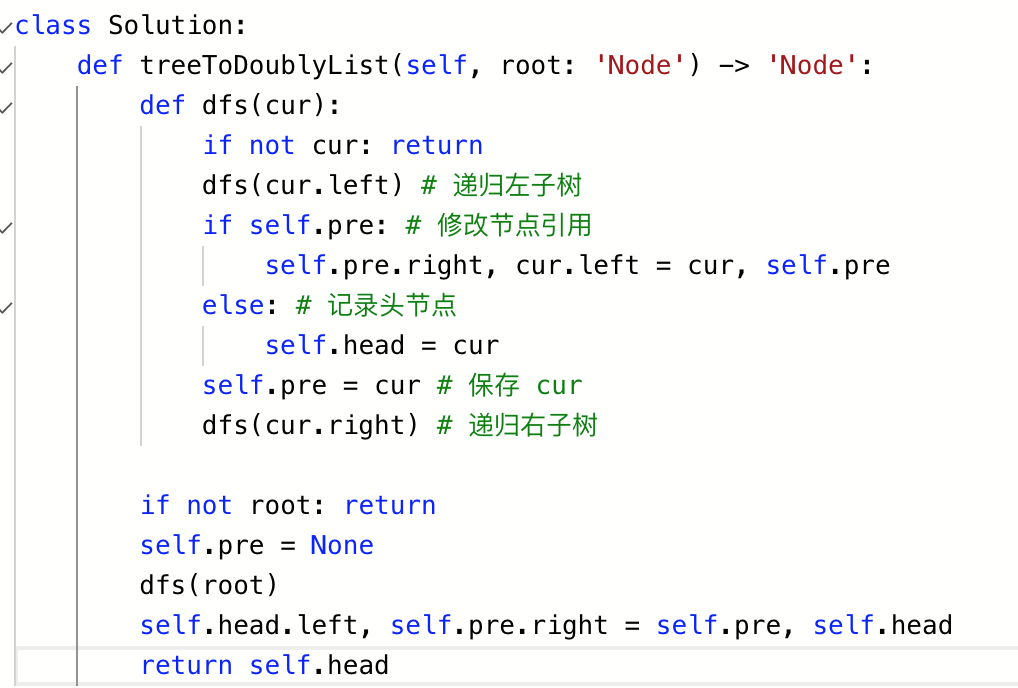
## 9.[二叉树中和为某一值的路径](https://leetcode-cn.com/problems/er-cha-shu-zhong-he-wei-mou-yi-zhi-de-lu-jing-lcof/)

输入一棵二叉树和一个整数，打印出二叉树中节点值的和为输入整数的所有路径。从树的根节点开始往下一直到叶节点所经过的节点形成一条路径。



## 10.[二叉搜索树与双向链表](https://leetcode-cn.com/problems/er-cha-sou-suo-shu-yu-shuang-xiang-lian-biao-lcof/)

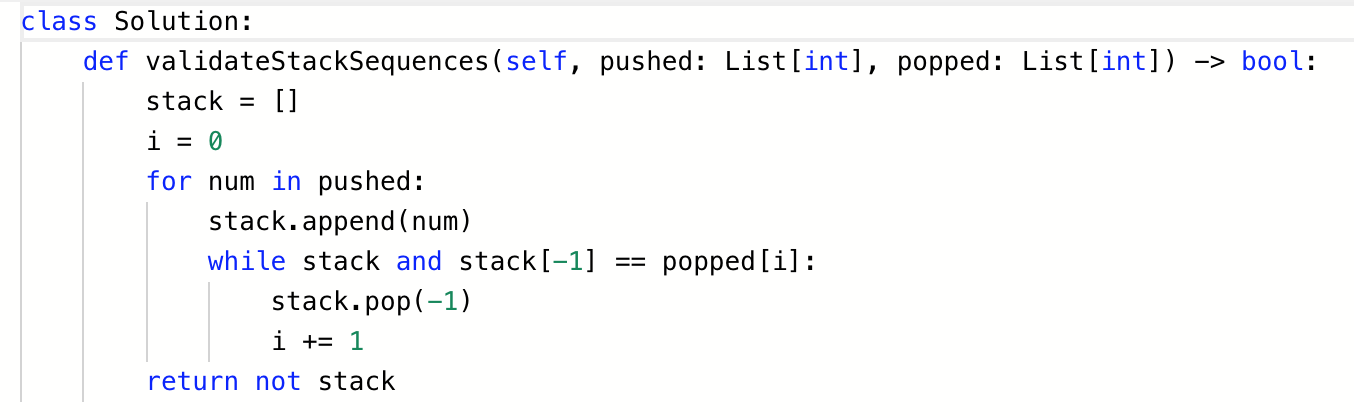
输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的循环双向链表。要求不能创建任何新的节点，只能调整树中节点指针的指向。



# 栈

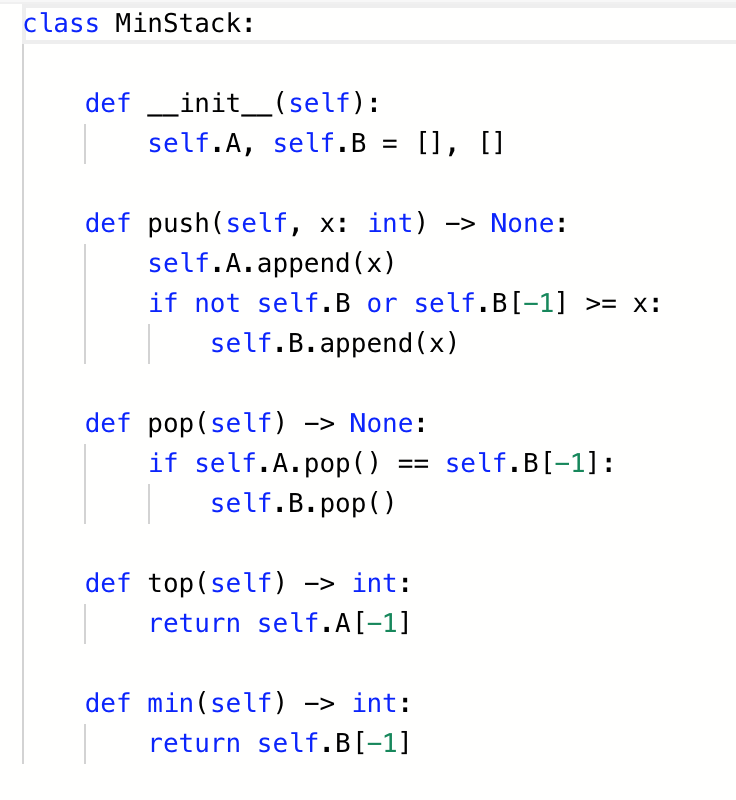
## 1.[栈的压入、弹出序列](https://leetcode-cn.com/problems/zhan-de-ya-ru-dan-chu-xu-lie-lcof/)

输入两个整数序列，第一个序列表示栈的压入顺序，请判断第二个序列是否为该栈的弹出顺序。假设压入栈的所有数字均不相等。例如，序列 {1,2,3,4,5} 是某栈的压栈序列，序列 {4,5,3,2,1} 是该压栈序列对应的一个弹出序列，但 {4,3,5,1,2} 就不可能是该压栈序列的弹出序列。



## 2.[包含min函数的栈](https://leetcode-cn.com/problems/bao-han-minhan-shu-de-zhan-lcof/)

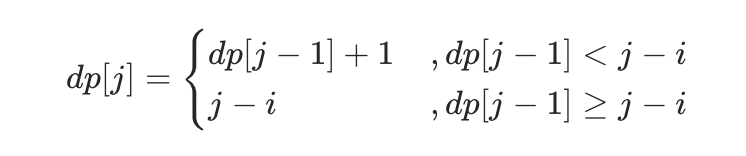
定义栈的数据结构，请在该类型中实现一个能够得到栈的最小元素的 min 函数在该栈中，调用 min、push 及 pop 的时间复杂度都是 O(1)。

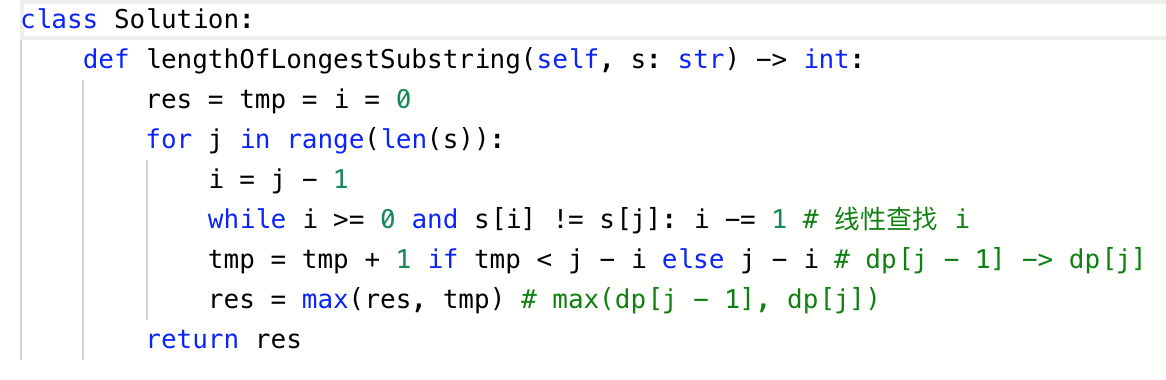


# 动态规划

## 1.[最长不含重复字符的子字符串](https://leetcode-cn.com/problems/zui-chang-bu-han-zhong-fu-zi-fu-de-zi-zi-fu-chuan-lcof/)

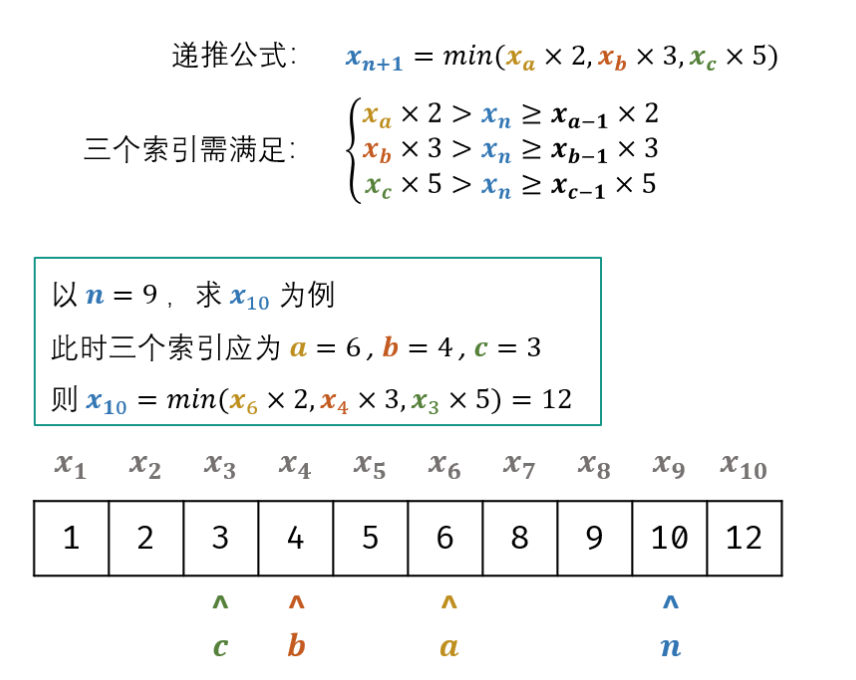
请从字符串中找出一个最长的不包含重复字符的子字符串，计算该最长子字符串的长度。

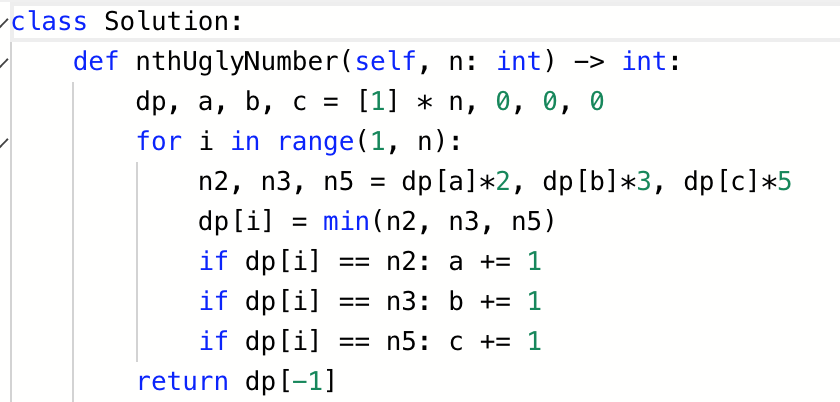




## 2.丑数

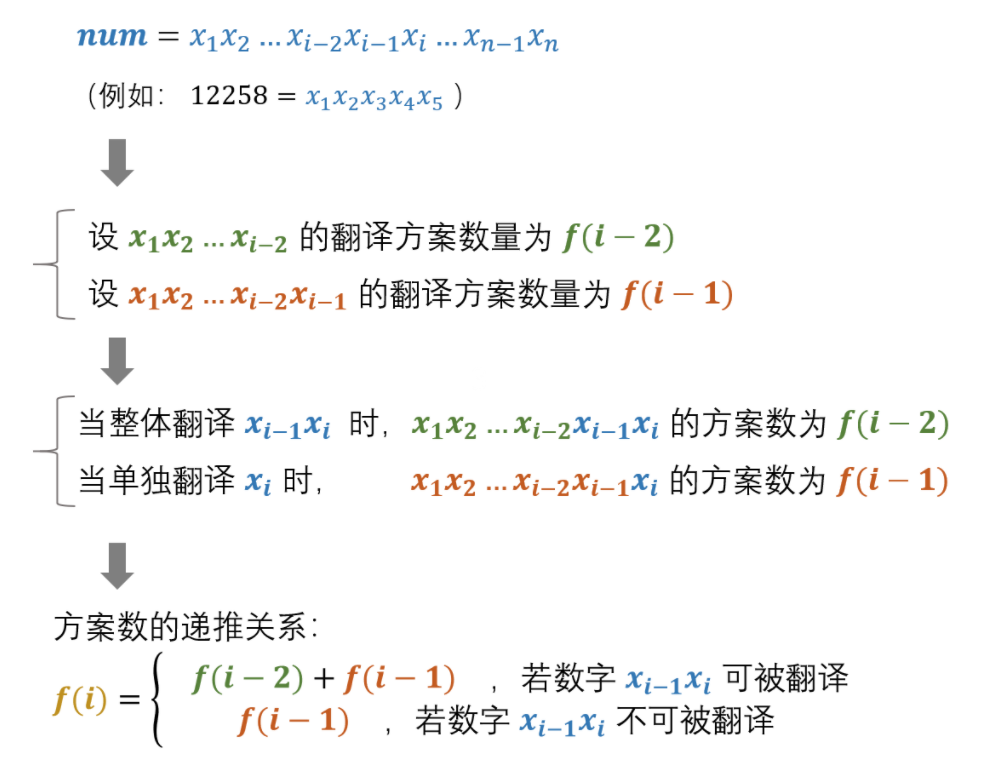
我们把只包含质因子 2、3 和 5 的数称作丑数（Ugly Number）。求按从小到大的顺序的第 n 个丑数。

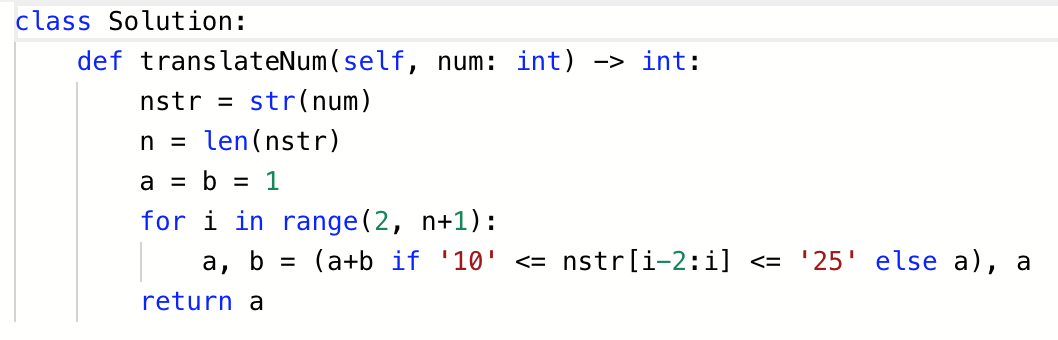




## 3.[把数字翻译成字符串](https://leetcode-cn.com/problems/ba-shu-zi-fan-yi-cheng-zi-fu-chuan-lcof/)

给定一个数字，我们按照如下规则把它翻译为字符串：0 翻译成 “a” ，1 翻译成 “b”，……，11 翻译成 “l”，……，25 翻译成 “z”。一个数字可能有多个翻译。请编程实现一个函数，用来计算一个数字有多少种不同的翻译方法。

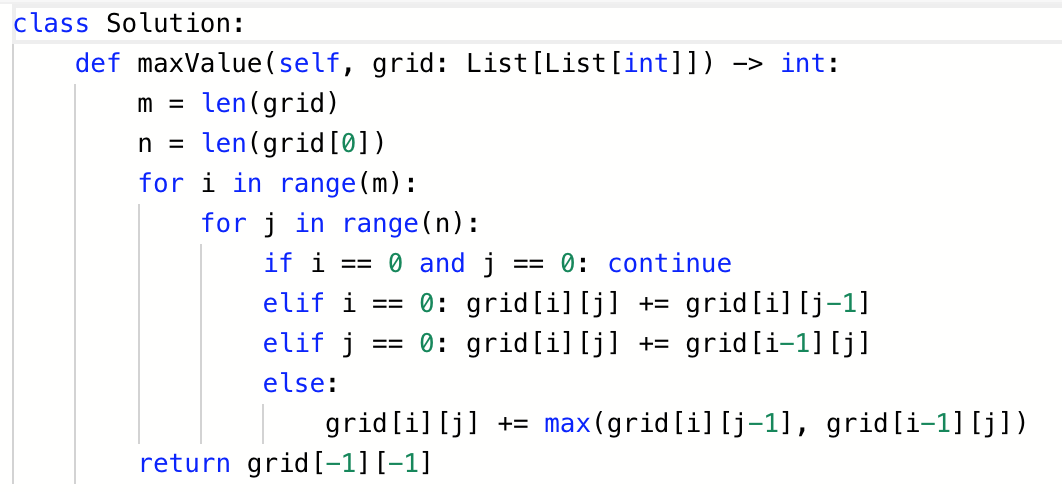




## 4.[礼物的最大价值](https://leetcode-cn.com/problems/li-wu-de-zui-da-jie-zhi-lcof/)

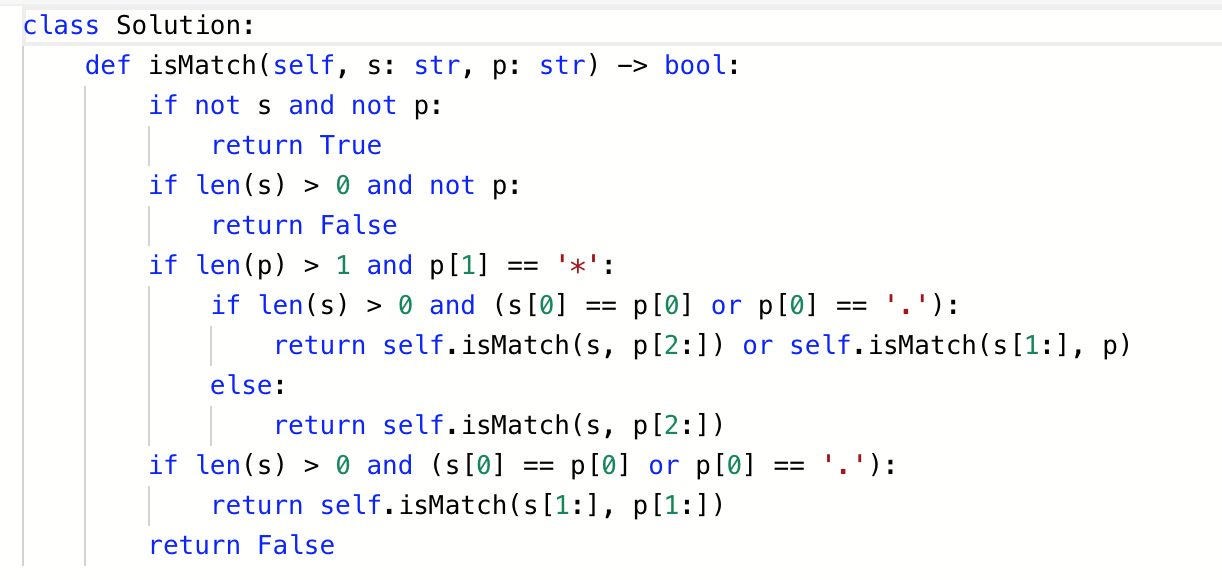
在一个 m\*n 的棋盘的每一格都放有一个礼物，每个礼物都有一定的价值（价值大于 0）。你可以从棋盘的左上角开始拿格子里的礼物，并每次向右或者向下移动一格、直到到达棋盘的右下角。给定一个棋盘及其上面的礼物的价值，请计算你最多能拿到多少价值的礼物？





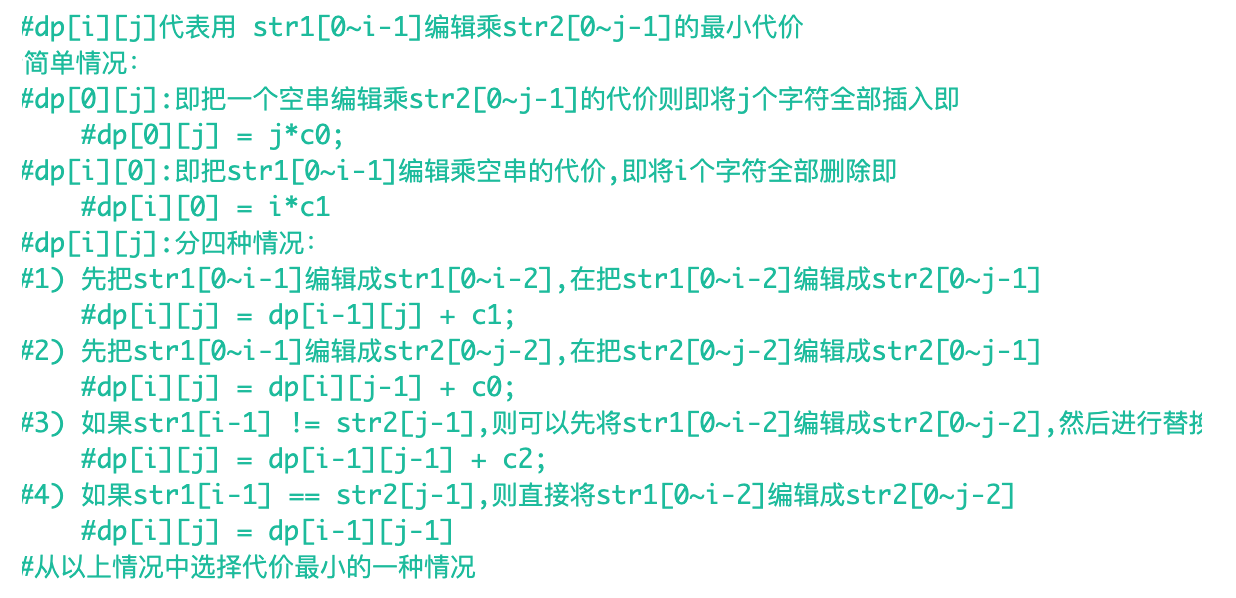
## 5. [正则表达式匹配](https://leetcode-cn.com/problems/zheng-ze-biao-da-shi-pi-pei-lcof/)

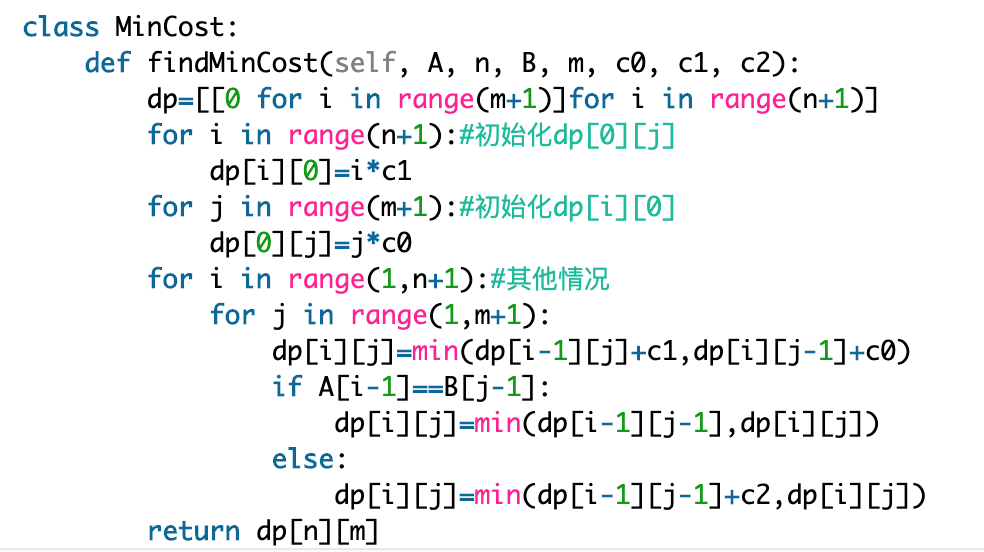
请实现一个函数用来匹配包含'. '和'\*'的正则表达式。模式中的字符'.'表示任意一个字符，而'\*'表示它前面的字符可以出现任意次（含0次）。在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab\*ac\*a"匹配，但与"aa.a"和"ab\*a"均不匹配。



## 6.最小编辑代价

对于两个字符串A和B，我们需要进行插入、删除和修改操作将A串变为B串，定义c0，c1，c2分别为三种操作的代价，请设计一个高效算法，求出将A串变为B串所需要的最少代价。

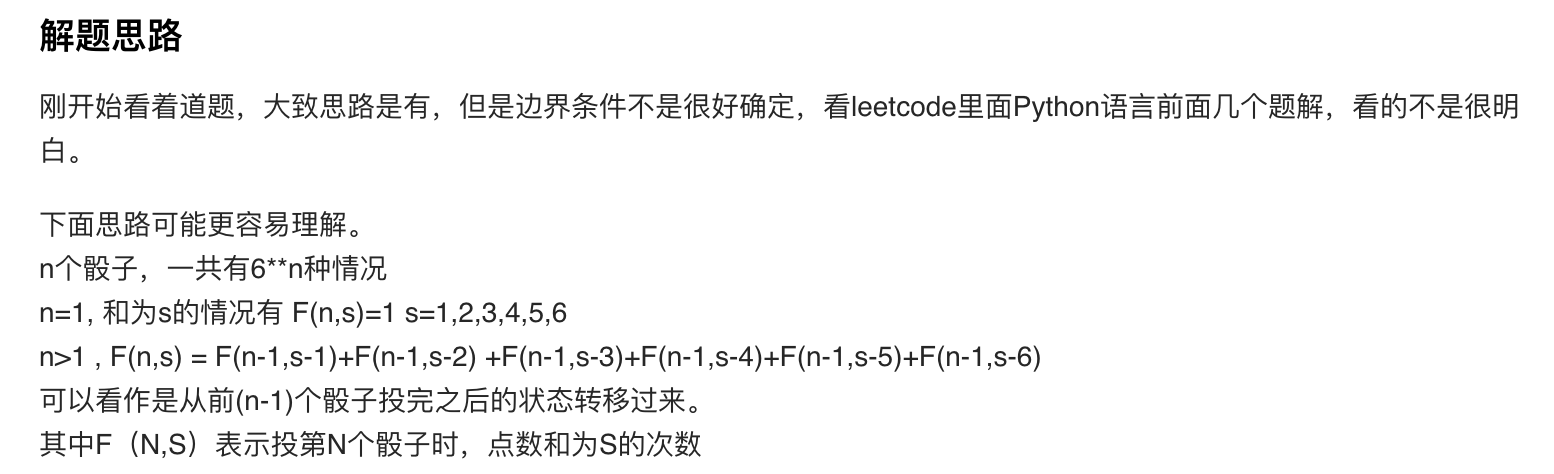


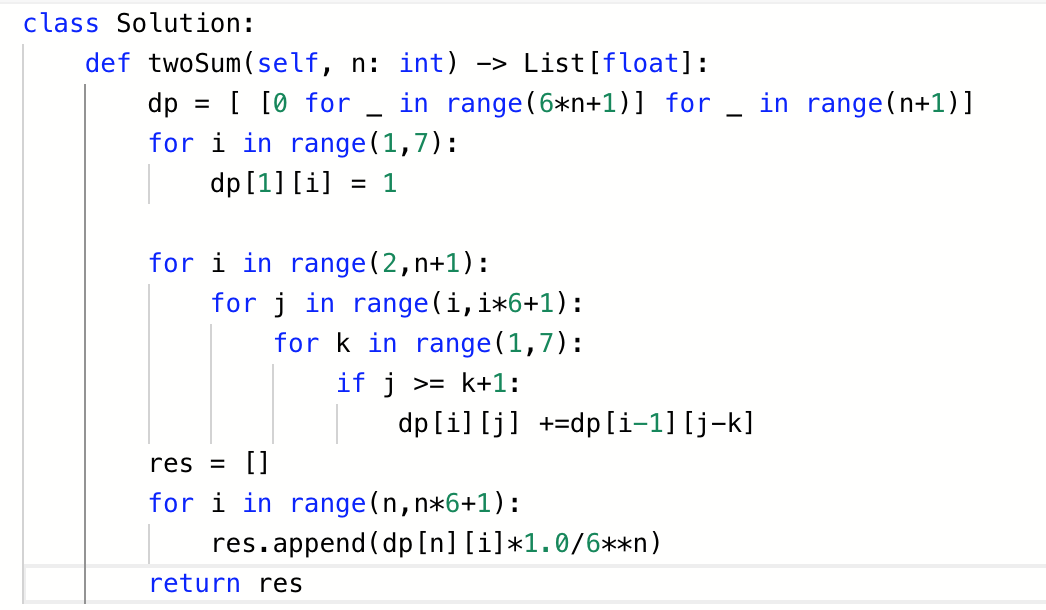


## 7.[n个骰子的点数](https://leetcode-cn.com/problems/nge-tou-zi-de-dian-shu-lcof/)

把n个骰子扔在地上，所有骰子朝上一面的点数之和为s。输入n，打印出s的所有可能的值出现的概率。

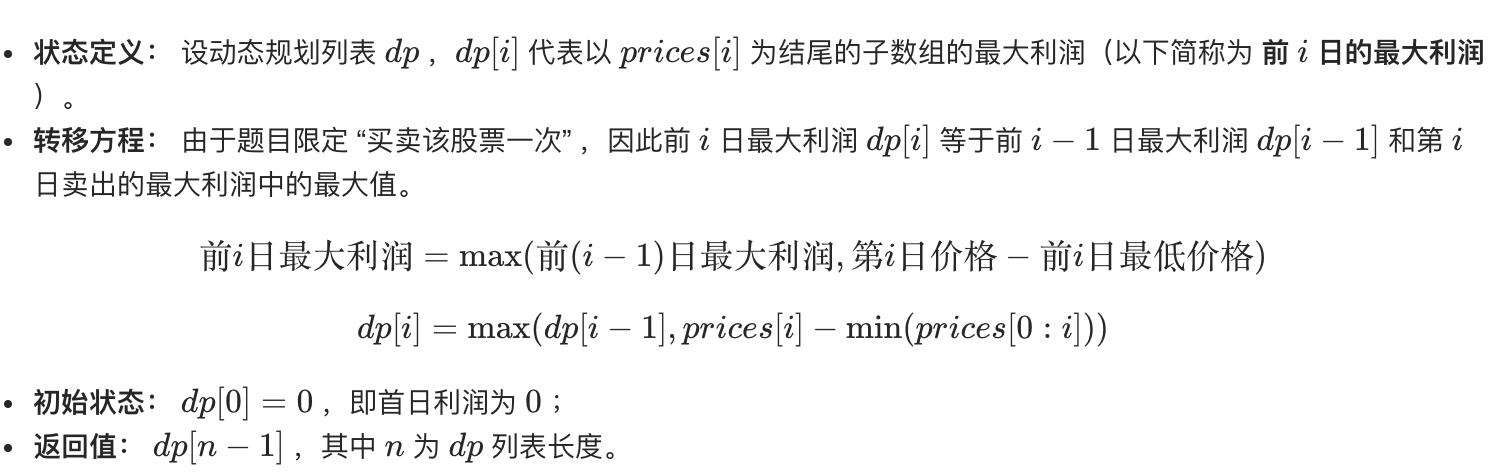
你需要用一个浮点数数组返回答案，其中第 i 个元素代表这 n 个骰子所能掷出的点数集合中第 i 小的那个的概率

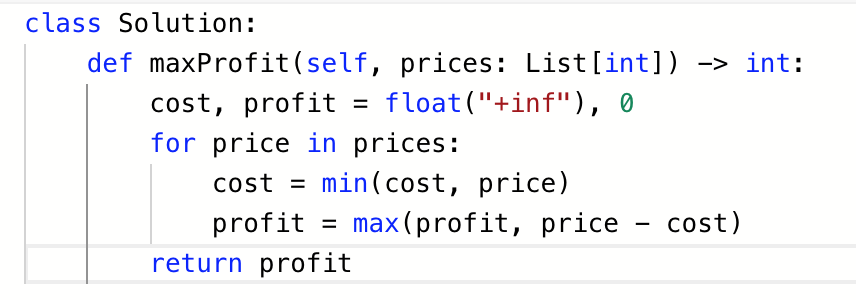




## 8. [股票的最大利润](https://leetcode-cn.com/problems/gu-piao-de-zui-da-li-run-lcof/)

假设把某股票的价格按照时间先后顺序存储在数组中，请问买卖该股票一次可能获得的最大利润是多少



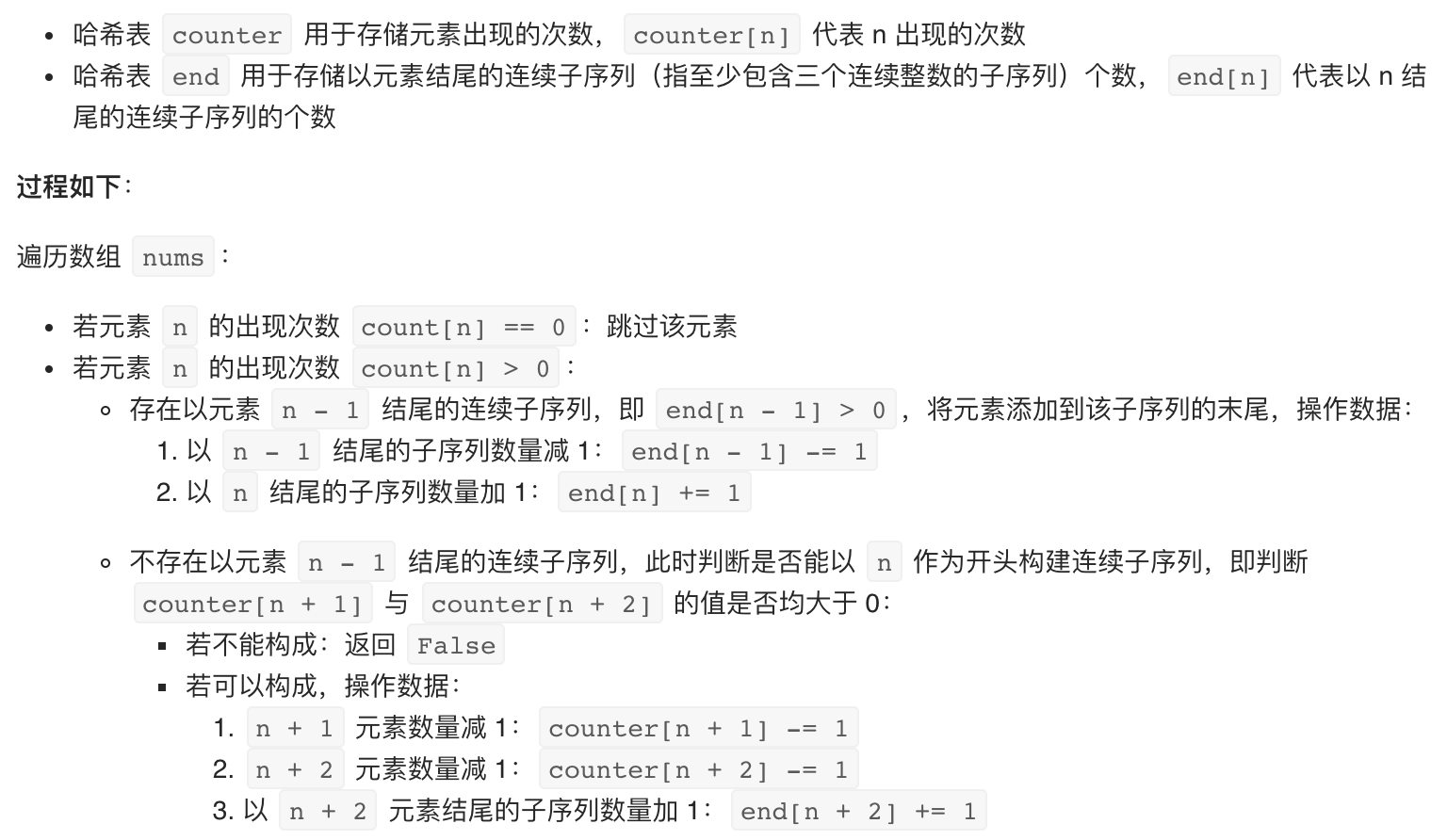


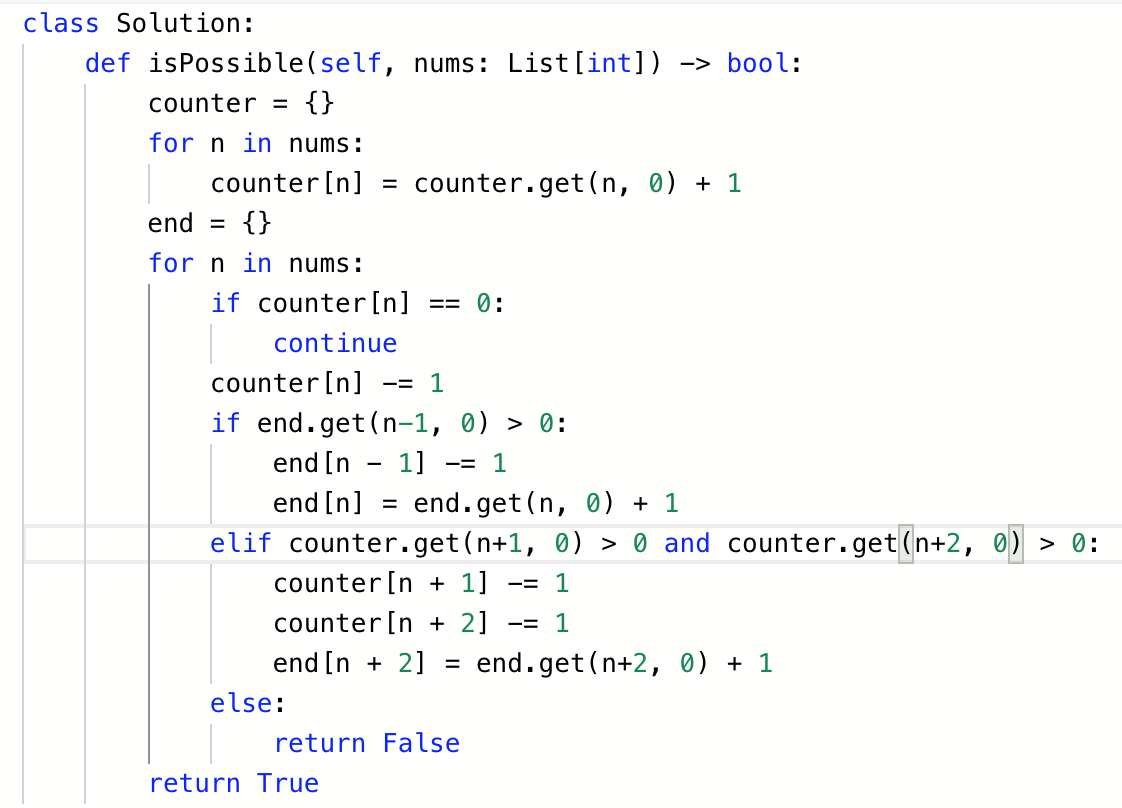
# 贪心算法

## [分割数组为连续子序列](https://leetcode-cn.com/problems/split-array-into-consecutive-subsequences/)

给你一个按升序排序的整数数组 num（可能包含重复数字），请你将它们分割成一个或多个子序列，其中每个子序列都由连续整数组成且长度至少为 3 。

如果可以完成上述分割，则返回 true ；否则，返回 false

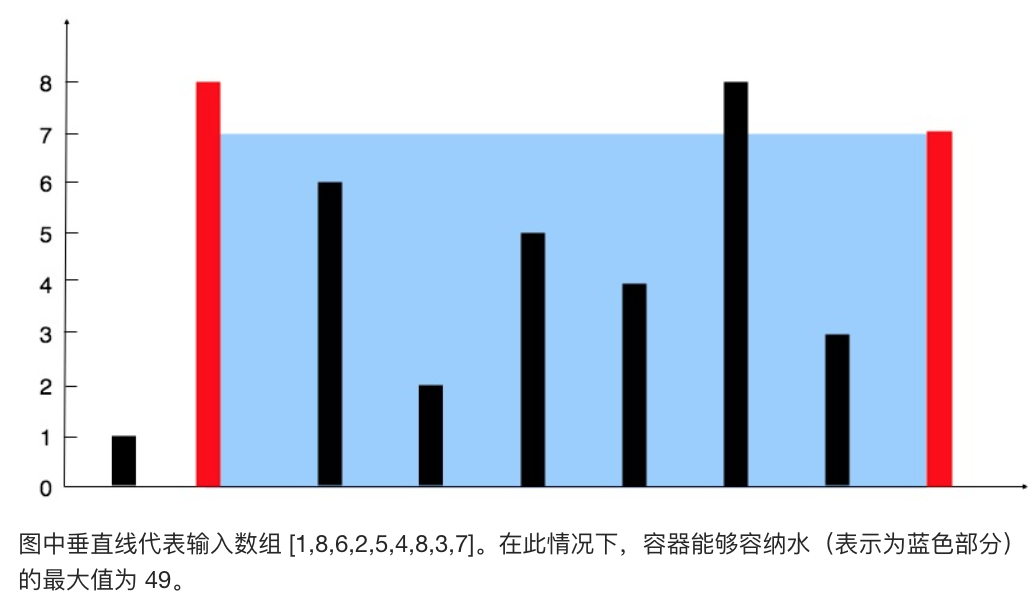


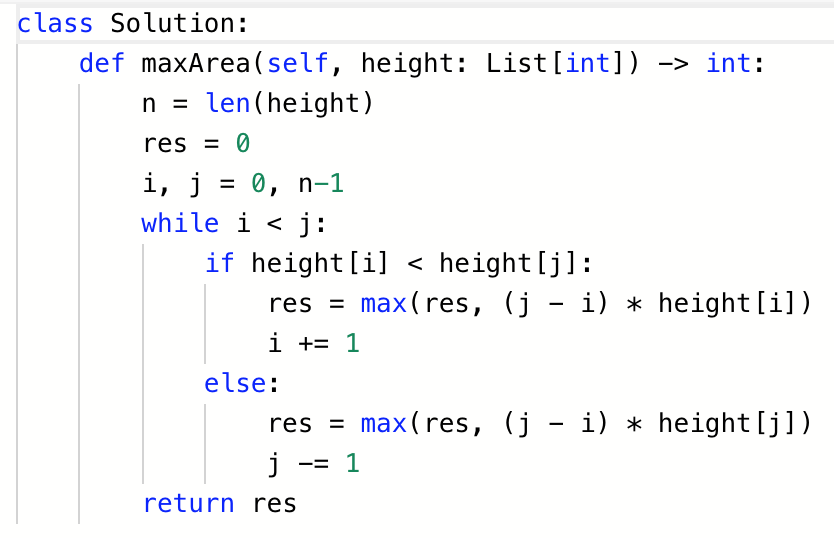


# 双指针

## [盛最多水的容器](https://leetcode-cn.com/problems/container-with-most-water/)

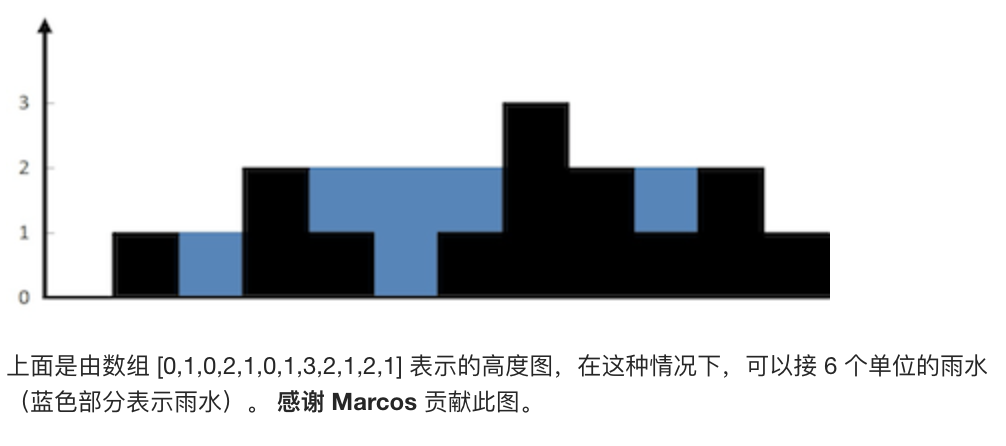
给你 n 个非负整数 a1，a2，...，an，每个数代表坐标中的一个点 (i, ai) 。在坐标内画 n 条垂直线，垂直线 i 的两个端点分别为 (i, ai) 和 (i, 0)。找出其中的两条线，使得它们与 x 轴共同构成的容器可以容纳最多的水

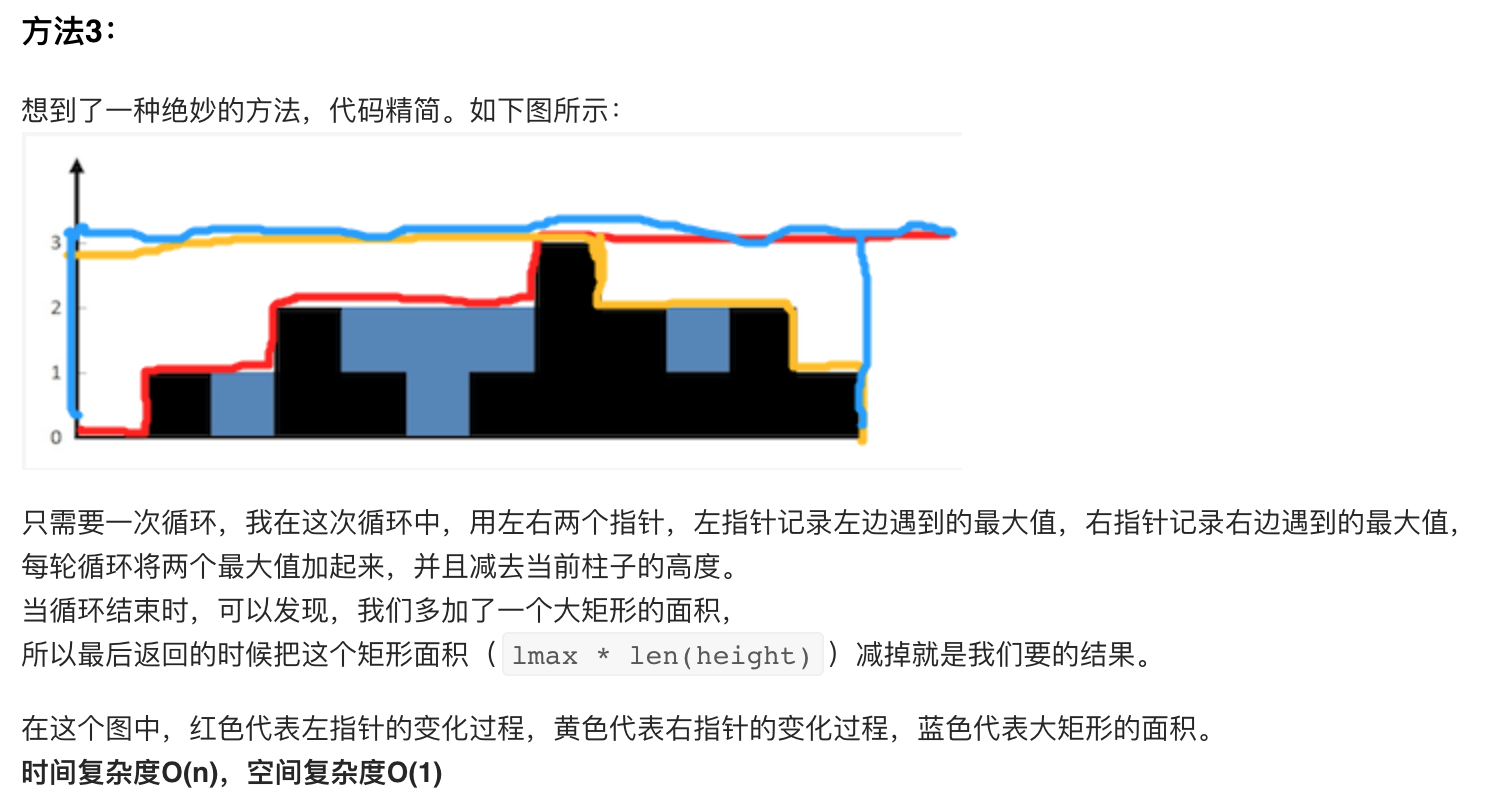


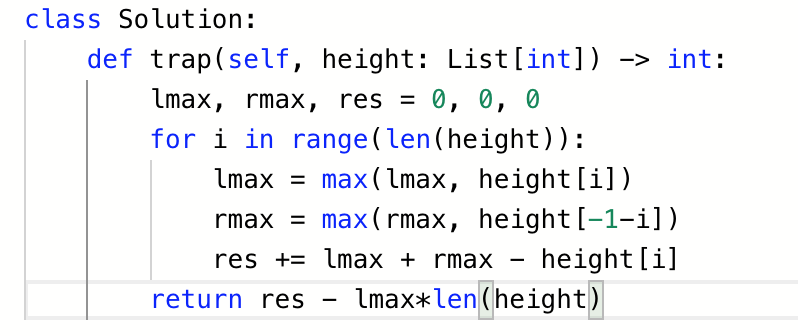


## 2. [接雨水](https://leetcode-cn.com/problems/trapping-rain-water/)

给定 n 个非负整数表示每个宽度为 1 的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水。



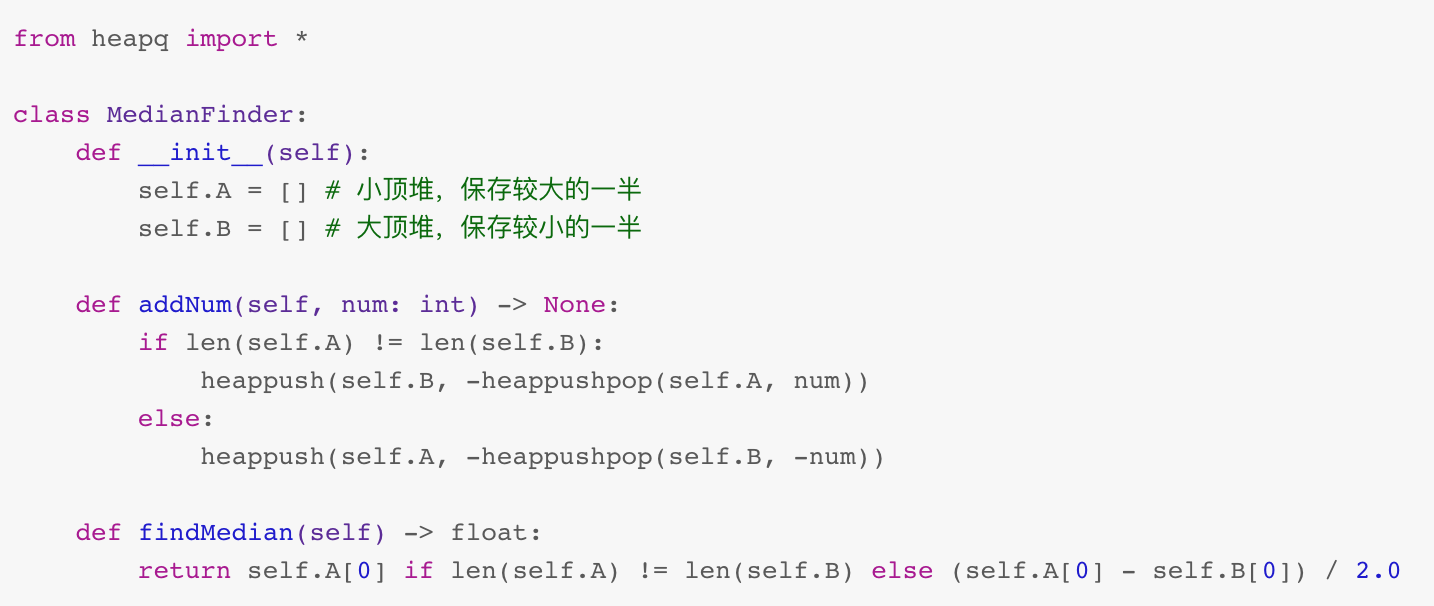




# 数组

## 1.[数据流中的中位数](https://leetcode-cn.com/problems/shu-ju-liu-zhong-de-zhong-wei-shu-lcof/)

如何得到一个数据流中的中位数？如果从数据流中读出奇数个数值，那么中位数就是所有数值排序之后位于中间的数值。如果从数据流中读出偶数个数值，那么中位数就是所有数值排序之后中间两个数的平均值。



## 2.[数字序列中某一位的数字](https://leetcode-cn.com/problems/shu-zi-xu-lie-zhong-mou-yi-wei-de-shu-zi-lcof/)

数字以0123456789101112131415…的格式序列化到一个字符序列中。在这个序列中，第5位（从下标0开始计数）是5，第13位是1，第19位是4，等等。请写一个函数，求任意第n位对应的数字。

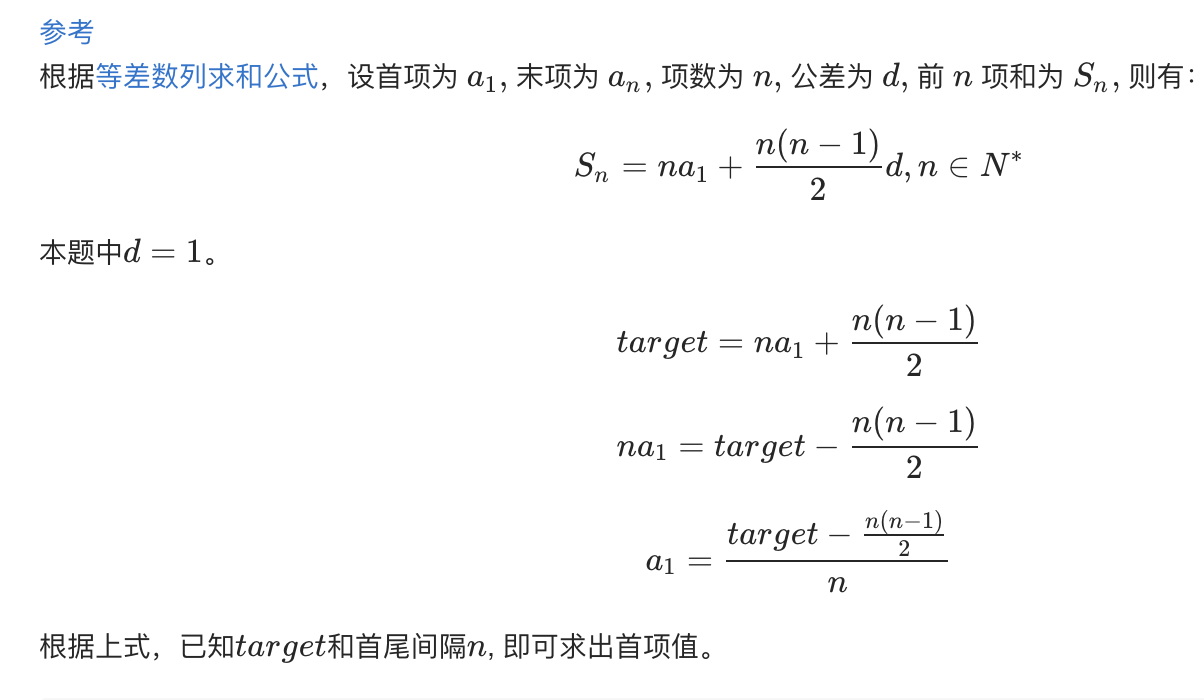


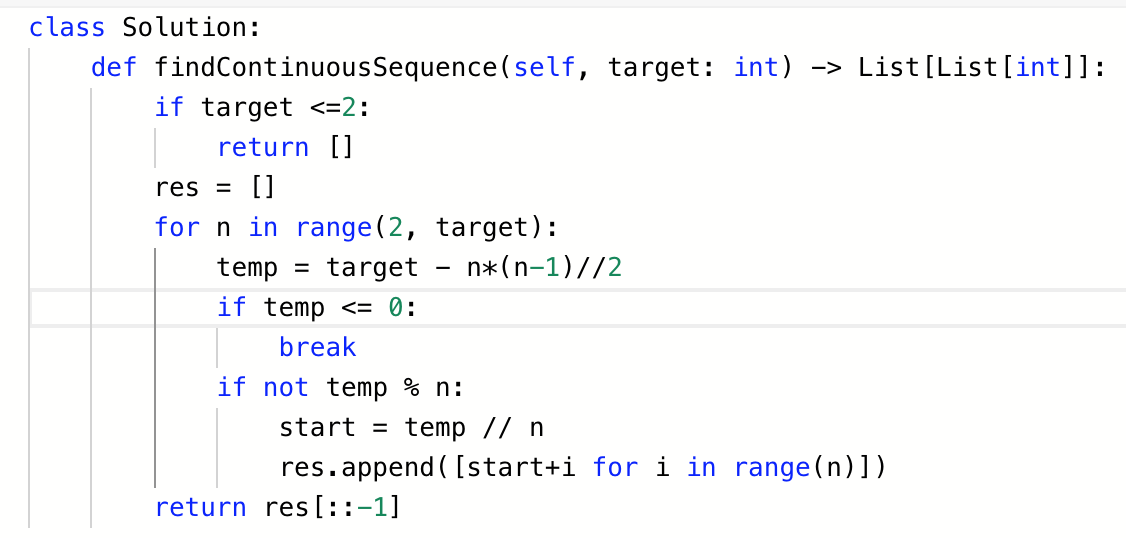


## 3.[和为s的连续正数序列](https://leetcode-cn.com/problems/he-wei-sde-lian-xu-zheng-shu-xu-lie-lcof/)

输入一个正整数 target，输出所有和为 target的连续正整数序列（至少含有两个数）

序列内的数字由小到大排列，不同序列按照首个数字从小到大排列





# 排序算法

## 堆排序(nlogn)，快速排序(nlogn)，归并排序[O(nlogn)]，冒泡排序(n方)，动态规划(n方)

堆排序与快速排序，归并排序一样都是时间复杂度为O(n\*logn)

## 1. 归并排序[O(nlogn)]

归并排序（Merge sort）是建立在归并操作上的一种有效的排序算法。该算法是采用分治法（Divide and Conquer）的一个非常典型的应用。

作为一种典型的分而治之思想的算法应用，归并排序的实现由两种方法：

* 自上而下的递归（所有递归的方法都可以用迭代重写，所以就有了第 2 种方法）；
* 自下而上的迭代；

2. 算法步骤

1. 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列；
2. 设定两个指针，最初位置分别为两个已经排序序列的起始位置；
3. 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置；
4. 重复步骤 3 直到某一指针达到序列尾；
5. 将另一序列剩下的所有元素直接复制到合并序列尾。
6. O(nlogn)



**def** mergeSort(arr):

**import** math

**if**(len(arr)<2):

**return** arr

    middle = math.floor(len(arr)/2)

    left, right = arr[0:middle], arr[middle:]

**return** merge(mergeSort(left), mergeSort(right))

**def** merge(left,right):

    result = []

**while** left **and** right:

**if** left[0] <= right[0]:

            result.append(left.pop(0))

**else**:

            result.append(right.pop(0));

**while** left:

        result.append(left.pop(0))

**while** right:

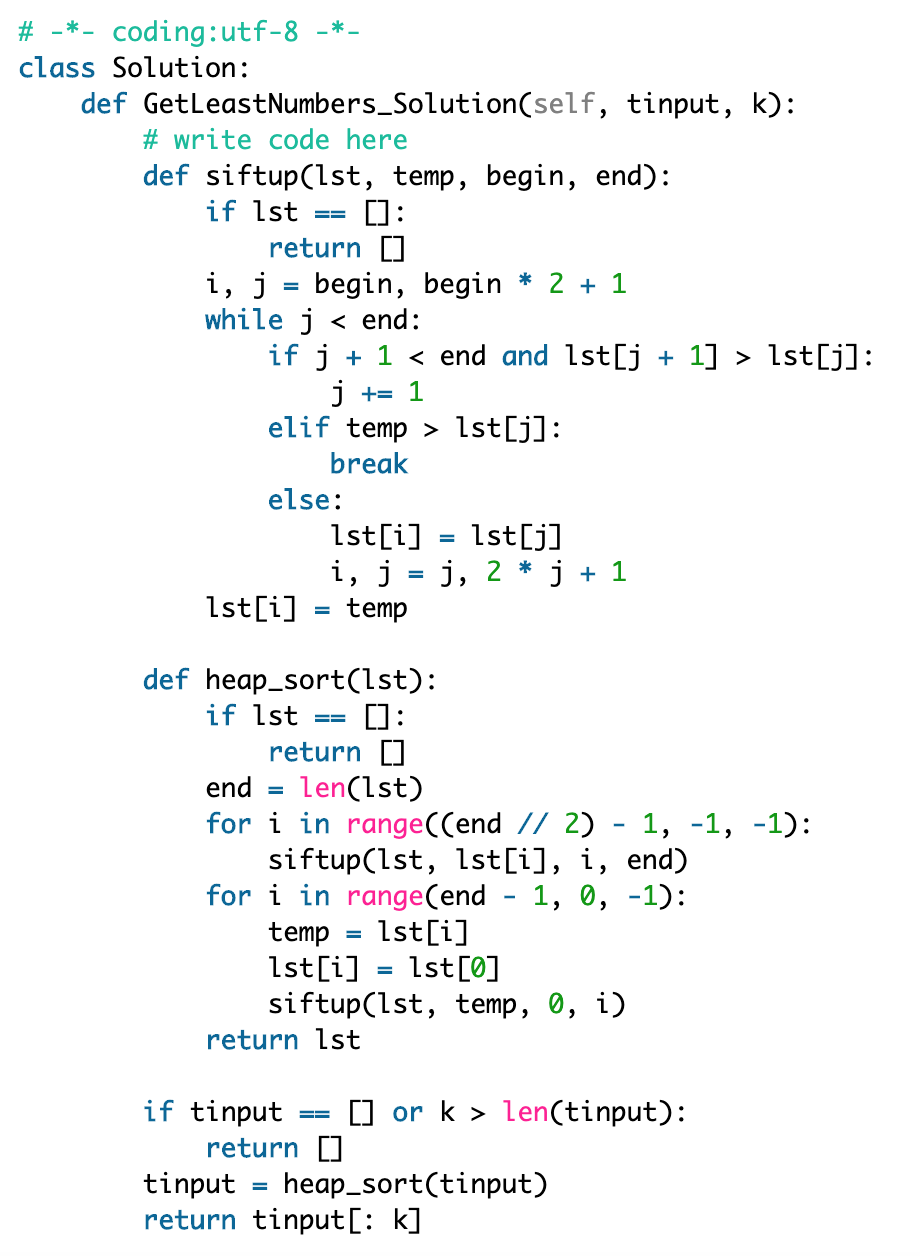
        result.append(right.pop(0));

**return** result

## 2. 快速排序 时间复杂度为O(nlogn)

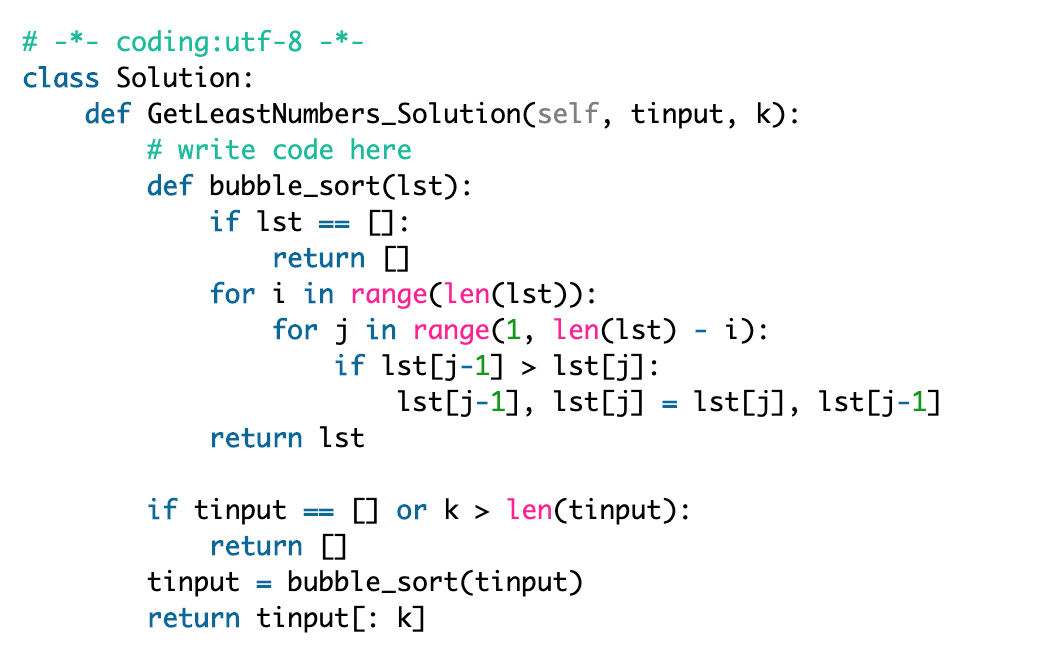


## 3.堆排序(nlogn)





## 4.冒泡排序(n^2)



# DFS和BFS

## 1. DFS深度优先

### 适用场景：于快速发现底部节点

def dfs(graph, start):

visited, stack = set(), [start]

while stack:

vertex = stack.pop()

if vertex not in visited:

visited.add(vertex)

stack.extend(graph[vertex] - visited)

return visited

dfs(graph, 'A') # {'E', 'D', 'F', 'A', 'C', 'B'}

def dfs\_paths(graph, start, goal):

stack = [(start, [start])]

while stack:

(vertex, path) = stack.pop()

for it in graph[vertex] - set(path):

if it == goal:

yield path + [it]

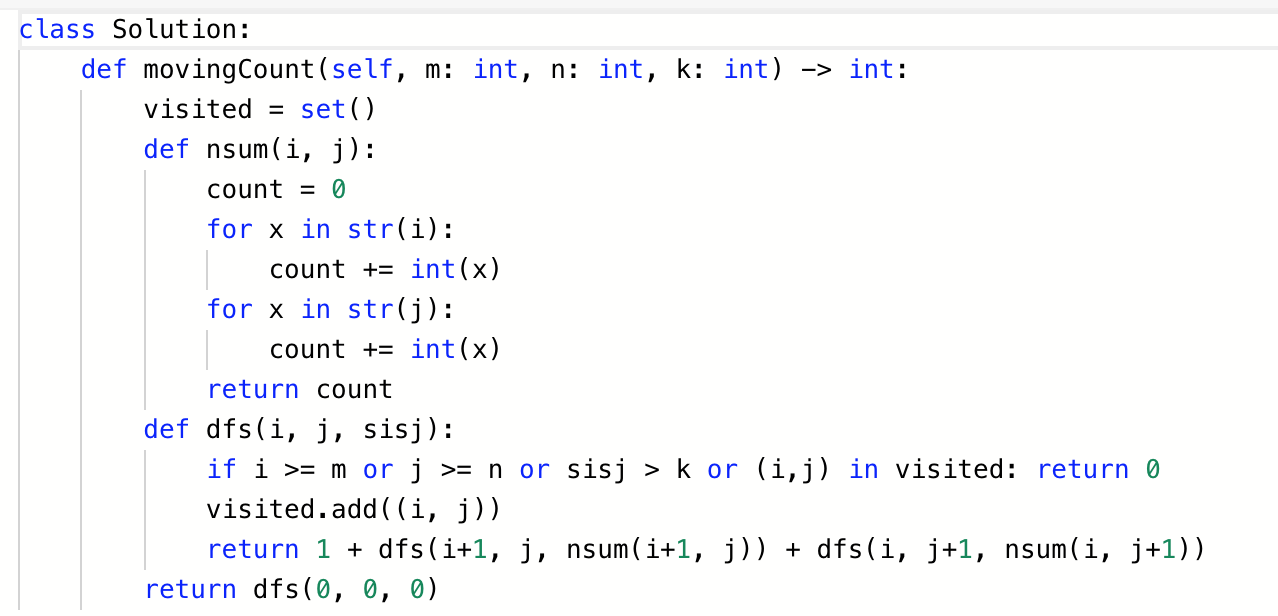
else:

stack.append((it, path + [it]))

list(dfs\_paths(graph, 'A', 'F')) # [['A', 'C', 'F'], ['A', 'B', 'E', 'F']]

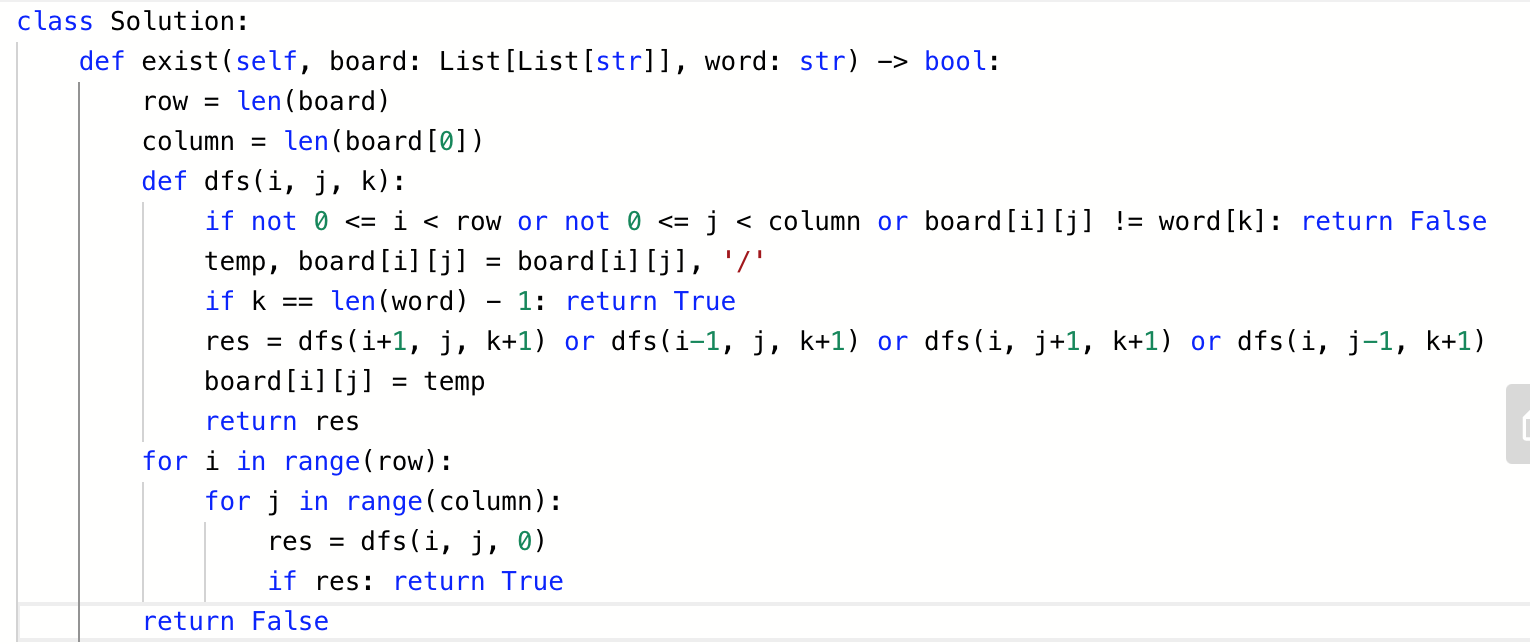
### 1.[机器人的运动范围](https://leetcode-cn.com/problems/ji-qi-ren-de-yun-dong-fan-wei-lcof/)

地上有一个m行n列的方格，从坐标 [0,0] 到坐标 [m-1,n-1] 。一个机器人从坐标 [0, 0] 的格子开始移动，它每次可以向左、右、上、下移动一格（不能移动到方格外），也不能进入行坐标和列坐标的数位之和大于k的格子。例如，当k为18时，机器人能够进入方格 [35, 37] ，因为3+5+3+7=18。但它不能进入方格 [35, 38]，因为3+5+3+8=19。请问该机器人能够到达多少个格子？



### 2. [矩阵中的路径](https://leetcode-cn.com/problems/ju-zhen-zhong-de-lu-jing-lcof/)

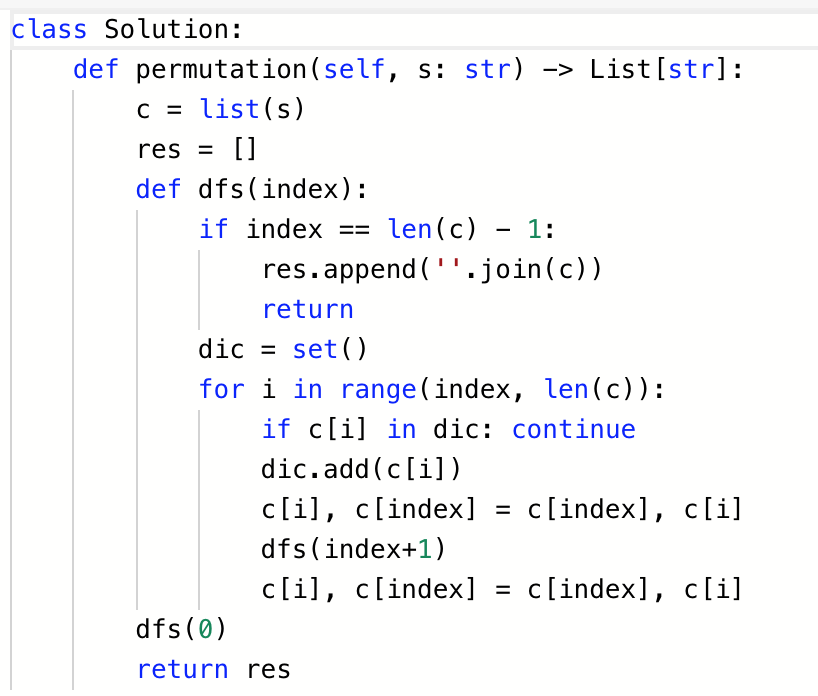
请设计一个函数，用来判断在一个矩阵中是否存在一条包含某字符串所有字符的路径。路径可以从矩阵中的任意一格开始，每一步可以在矩阵中向左、右、上、下移动一格。如果一条路径经过了矩阵的某一格，那么该路径不能再次进入该格子。例如，在下面的3×4的矩阵中包含一条字符串“bfce”的路径（路径中的字母用加粗标出）。



### 3. [字符串的排列](https://leetcode-cn.com/problems/zi-fu-chuan-de-pai-lie-lcof/)

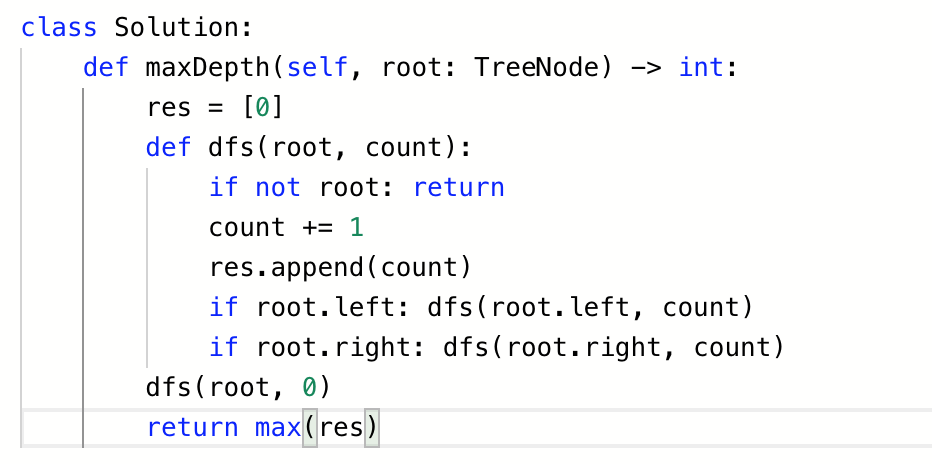
输入一个字符串，打印出该字符串中字符的所有排列

你可以以任意顺序返回这个字符串数组，但里面不能有重复元素



### 4. [二叉树的深度](https://leetcode-cn.com/problems/er-cha-shu-de-shen-du-lcof/)

输入一棵二叉树的根节点，求该树的深度。从根节点到叶节点依次经过的节点（含根、叶节点）形成树的一条路径，最长路径的长度为树的深度。





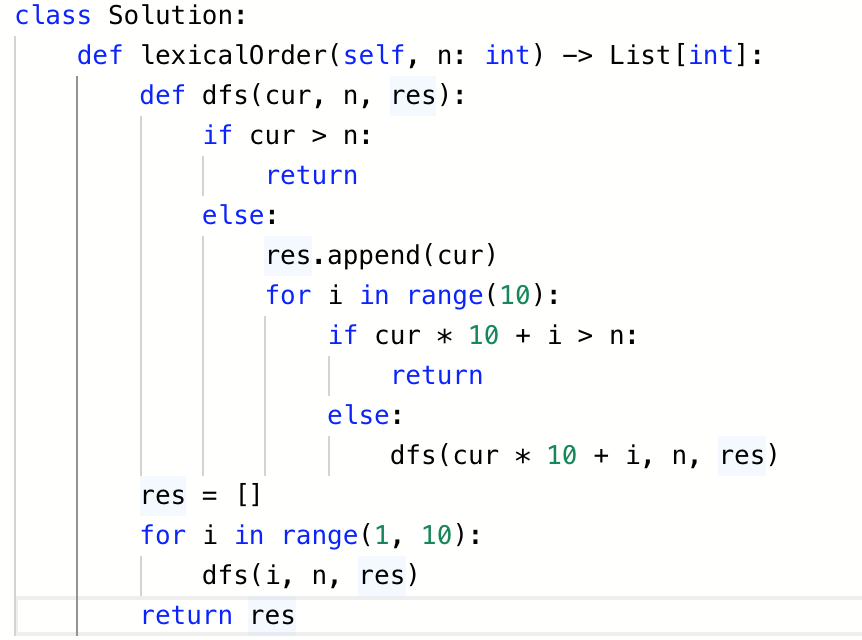
### 5.[平衡二叉树](https://leetcode-cn.com/problems/ping-heng-er-cha-shu-lcof/)

输入一棵二叉树的根节点，判断该树是不是平衡二叉树。如果某二叉树中任意节点的左右子树的深度相差不超过1，那么它就是一棵平衡二叉树



### 6. [字典序排数](https://leetcode-cn.com/problems/lexicographical-numbers/)

给定一个整数 n, 返回从 1 到 n 的字典顺序。



## 2. BFS宽度优先

### 适用场景：寻找最短路径的问题

与迭代DFS实现类似，唯一需要的更改是从列表结构的开头而不是最后一个堆栈中删除下一个项目。

<https://github.com/ShaoQiBNU/mazes_BFS>

### 迷宫

def bfs(map):

#### 上下左右四个方向的增量 #####

dx=[1,-1,0,0]

dy=[0,0,1,-1]

#### 节点列表 #####

nodes=[]

nodes.append((0,1,0))

#### 节点访问表——记录节点是否被访问 #####

node\_visit=np.array([[0]\*10]\*10)

node\_visit[0][1]=1

#### bfs过程 #####

while len(nodes)!=0:

#### 从节点列表输出一个节点 #####

node=nodes[0]

#### 上下左右四个方向遍历 #####

for i in range(0,4):

#### 上下左右四个方向遍历 #####

x=node[0]+dx[i]

y=node[1]+dy[i]

#### 步数 #####

step=node[2]

#### 判断是否到达终点 #####

if x==9 and y==8:

return step+1

#### 判断节点是否符合条件 #####

if x>=1 and x<=9 and y>=1 and y<=9 and node\_visit[x][y]==0 and map[x][y]==1:

#### 将节点压入节点列表nodes，说明进入下一层，step+1#####

nodes.append((x,y,step+1))

#### 该节点标记为已经访问过 #####

node\_visit[x][y]=1

#### 从节点列表移除上一层的节点 #####

del(nodes[0])

#### 如果没有路径，则表明迷宫无法走出 #####

return 0



def bfs(graph, start):

visited, queue = set(), [start]

while queue:

vertex = queue.pop(0)

if vertex not in visited:

visited.add(vertex)

queue.extend(graph[vertex] - visited)

return visited

bfs(graph, 'A') # {'B', 'C', 'A', 'F', 'D', 'E'}

### path

def bfs\_paths(graph, start, goal):

queue = [(start, [start])]

while queue:

(vertex, path) = queue.pop(0)

for next in graph[vertex] - set(path):

if next == goal:

yield path + [next]

else:

queue.append((next, path + [next]))

list(bfs\_paths(graph, 'A', 'F')) # [['A', 'C', 'F'], ['A', 'B', 'E', 'F']]