【剑指offer】

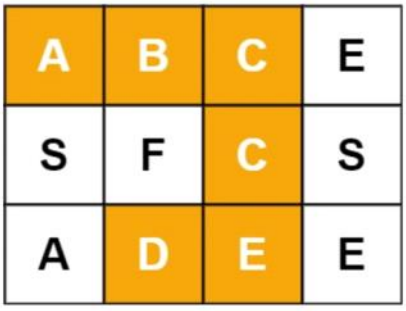
2021年12月2日 21:38

- 剑指 Offer 03. 数组中重复的数字
  - 原地哈希
  - ```
    class Solution(object):
        def findRepeatNumber(self, nums):
            """
            :type nums: List[int]
            :rtype: int
            """
            i = 0
            while i < len(nums):
                if nums[i] == i:
                    i += 1
                    continue
                if nums[i] == nums[nums[i]]:
                    return nums[i]
                else:
                    nums[nums[i]], nums[i] = nums[i], nums[nums[i]]
            return -1
    ```
- 剑指 Offer 06. 从尾到头打印链表
  - 递归
  - ```
    class Solution(object):
        def reversePrint(self, head):
            """
            :type head: ListNode
            :rtype: List[int]
            """
            if not head:
                return []
            return self.reversePrint(head.next) + [head.val]
    ```
- 剑指 Offer 09. 用两个栈实现队列
  - 模拟
  - ```
    class CQueue(object):

        def __init__(self):
            self.stack_in = []
            self.stack_out = []

        def appendTail(self, value):
            """
            :type value: int
            :rtype: None
            """
            self.stack_in.append(value)
            return
        def deleteHead(self):
            """
            :rtype: int
            """
            if self.stack_out:
                return self.stack_out.pop()
            else:
                if self.stack_in:
                    while self.stack_in:
                        self.stack_out.append(self.stack_in.pop())
                    return self.stack_out.pop()
                else:
                    return -1
    # Your CQueue object will be instantiated and called as such:
    # obj = CQueue()
    # obj.appendTail(value)
    ```
    ```

```
        # param_2 = obj.deleteHead()
```

- 剑指 Offer 11. 旋转数组的最小数字
  - 二分查找
  - ```
    class Solution(object):
        def minArray(self, numbers):
            """
            :type numbers: List[int]
            :rtype: int
            """
            left = 0
            right = len(numbers) - 1
            while left < right:
                mid = left + (right - left) // 2
                if numbers[mid] > numbers[right]:
                    left = mid + 1
                elif numbers[mid] < numbers[right]:
                    right = mid
                elif numbers[mid] == numbers[right]:
                    right -= 1
            return numbers[left]
    ```
- 剑指 Offer 12. 矩阵中的路径
  - 

    示例 1:

    ```
    输入: board = [["A","B","C","E"],["S","F","C","S"],
    ["A","D","E","E"]], word = "ABCCED"
    输出: true
    ```
  - 递归法
  - ```
    class Solution(object):
        def exist(self, board, word):
            """
            :type board: List[List[str]]
            :type word: str
            :rtype: bool
            """
            def recur(row, col, x):
                if x == len(word):
                    return True
                if row < 0 or col < 0 or row >= len(board) or col >= len(board[0]):
                    return False
                if board[row][col] != word[x]:
                    return False
                else:
                    board[row][col] = ' '
                    res = recur(row-1, col, x+1) or recur(row+1, col, x+1)
                          or recur(row, col-1, x+1) or recur(row, col+1, x+1)
                    board[row][col] = word[x]
                    return res

            for i in range(len(board)):
                for j in range(len(board[0])):
                    if recur(i, j, 0):
                        return True
            return False
    ```
- 剑指 Offer 13. 机器人的运动范围

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？

示例 1：

```
输入：m = 2, n = 3, k = 1
输出：3
```

- 递归法
- ```
  class Solution(object):
      def movingCount(self, m, n, k):
          """
          :type m: int
          :type n: int
          :type k: int
          :rtype: int
          """
          self.res = 0
          vis = [[0 for _ in range(n)] for _ in range(m)]
          def summ(x):
              rel = 0
              while x > 0:
                  rel += x % 10
                  x //= 10
              return rel
          def recur(row, col):
              if row < 0 or col < 0 or row >= m or col >= n:
                  return
              if summ(row) + summ(col) > k:
                  return
              if vis[row][col] == 1:
                  return
              else:
                  self.res += 1
                  vis[row][col] = 1
                  recur(row+1, col)
                  recur(row, col+1)
          recur(0, 0)
          return self.res
  ```

- 剑指 Offer 14- I. 剪绳子
  - 动态规划
  - ```
    class Solution(object):
        def cuttingRope(self, n):
            """
            :type n: int
            :rtype: int
            """
            if n <= 2:
                return 1
            dp = [0] * (n+1)
            dp[2] = 1
            for i in range(3, n+1):
                for j in range(1, i):
                    dp[i] = max(dp[i], j*(i-j), j*dp[i-j])
            return dp[-1]
    ```
  - 数学
  - ```
    class Solution(object):
        def cuttingRope(self, n):
            """
            :type n: int
            :rtype: int
            """
    ```
    ```

```
        if n <= 3:
            return n - 1
        p = n // 3
        q = n % 3
        if q == 0:
            return int(math.pow(3, p))
        elif q == 1:
            return int(math.pow(3, p-1) * 4)
        elif q == 2:
            return int(math.pow(3, p) * 2)
```

- 剑指 Offer 14- II. 剪绳子 II
  - 数学
  - ```
    class Solution(object):
        def cuttingRope(self, n):
            """
            :type n: int
            :rtype: int
            """
            mod = 10 ** 9 + 7
            if n <= 3:
                return n-1
            p = n // 3
            q = n % 3
            if q == 0:
                res = 3 ** p % mod
            elif q == 1:
                res = 3 ** (p-1) * 4 % mod
            elif q == 2:
                res = 3 ** p * q % mod
            return res
    ```
- 剑指 Offer 17. 打印从1到最大的n位数
  - 递归法
  - ```
    class Solution(object):
        def printNumbers(self, n):
            """
            :type n: int
            :rtype: List[int]
            """
            res = []
            cur = []
            def recur(x, l):
                if x == l:
                    res.append(int(''.join(cur)))
                    return
                idx = 0
                if x == 0:
                    idx = 1
                for i in range(idx, 10):
                    cur.append(str(i))
                    recur(x+1, l)
                    cur.pop()
            for j in range(1, n+1):
                recur(0, j)
            return res
    ```
- 数值的整数次方
  - ```
    class Solution(object):
        def myPow(self, x, n):
            """
            :type x: float
            :type n: int
            :rtype: float
            """
            if n == 0:
                return 1
            if n == 1:
                return x
            if n == -1:
    ```
    ```

```
            return 1/x
        half = self.myPow(x, n // 2)
        if n % 2 == 1:
                return half * half * x
        else:
                return half * half
```

- 剑指 Offer 19. 正则表达式匹配
  - 模拟
  - ```
    class Solution(object):
        def isMatch(self, s, p):
            """
            :type s: str
            :type p: str
            :rtype: bool
            """
            dp = [[False for _ in range(len(p)+1)] for _ in range(len(s)+1)]
            for i in range(len(s)+1):
                for j in range(len(p)+1):
                    if j == 0:
                        dp[i][j] = i == 0
                        continue
                    if p[j-1] != '*':
                        if i > 0 and (s[i-1] == p[j-1] or p[j-1] == '.'):
                            dp[i][j] = dp[i-1][j-1]
                    else:
                        if j > 1:
                            dp[i][j] = dp[i][j-2]
                        if j > 1 and i > 0 and (s[i-1] == p[j-2] or p[j-2] == '.'):
                            dp[i][j] |= dp[i-1][j]
            return dp[-1][-1]
    ```
- 剑指 Offer 20. 表示数值的字符串
  - 模拟
  - ```
    class Solution(object):
        def isNumber(self, s):
            """
            :type s: str
            :rtype: bool
            """
            s = s.strip()
            eflag = False
            dotflag = False
            numflag = False
            for i in range(len(s)):
                if s[i] == ' ':
                    return False
                if '0' <= s[i] <= '9':
                    numflag = True
                elif s[i] == '.':
                    if eflag or dotflag:
                        return False
                    else:
                        dotflag = True
                elif s[i] == 'e' or s[i] == 'E':
                    if i == 0 or not numflag or eflag:
                        return False
                    else:
                        eflag = True
                        numflag = False
                elif s[i] == '-' or s[i] == '+':
                    if i > 0 and s[i-1] != 'e' and s[i-1] != 'E':
                        return False
                else:
                    return False
            return numflag
    ```
- 剑指 Offer 21. 调整数组顺序使奇数位于偶数前面
  - 双指针

```
class Solution(object):
    def exchange(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        left = 0
        right = len(nums) - 1
        while left < right:
            while left < right and nums[left] % 2 == 1:
                left += 1
            while left < right and nums[right] % 2 == 0:
                right -= 1
            nums[left], nums[right] = nums[right], nums[left]
        return nums
```

- 剑指 Offer 26. 树的子结构
  - 递归法

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution(object):
    def isSubStructure(self, A, B):
        """
        :type A: TreeNode
        :type B: TreeNode
        :rtype: bool
        """
        if not A or not B:
            return False
        return self.recur(A, B) or self.isSubStructure(A.left,B) or
self.isSubStructure(A.right, B)
    def recur(self, root, B):
        if not B:
            return True
        if not root or root.val != B.val:
            return False
        return self.recur(root.left, B.left) and self.recur(root.right, B.right)
```

- 二叉树的镜像

```
class Solution:
    def mirrorTree(self, root: TreeNode) -> TreeNode:
        if not root:
            return
        l = self.mirrorTree(root.left)
        r = self.mirrorTree(root.right)
        root.left = r
        root.right = l
        return root
```

- 剑指 Offer 28. 对称的二叉树
  - 递归法

```
class Solution(object):
    def isSymmetric(self, root):
        """
        :type root: TreeNode
        :rtype: bool
        """
        if not root:
            return True
        return self.recur(root.left, root.right)
    def recur(self, left, right):
        if not left and not right:
            return True
        if not left or not right:
            return False
```

```
            if left.val == right.val:
                return self.recur(left.left, right.right) and self.recur(left.right,
right.left)
            else:
                return False
```

- 剑指 Offer 31. 栈的压入、弹出序列

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。

  - **示例 1:**

```
输入: pushed = [1,2,3,4,5], popped = [4,5,3,2,1]
输出: true
解释: 我们可以按以下顺序执行:
push(1), push(2), push(3), push(4), pop() -> 4,
push(5), pop() -> 5, pop() -> 3, pop() -> 2, pop() -> 1
```

  - 模拟+双重循换
  - ```
    class Solution(object):
        def validateStackSequences(self, pushed, popped):
            """
            :type pushed: List[int]
            :type popped: List[int]
            :rtype: bool
            """
            stack = collections.deque()
            for item in pushed:
                stack.append(item)
                while stack and popped and stack[-1] == popped[0]:
                    stack.pop()
                    popped.pop(0)
            return not stack and not popped
    ```

- 剑指 Offer 33. 二叉搜索树的后序遍历序列

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。

参考以下这颗二叉搜索树：

  - ```
        5
       / \
      2   6
     / \
    1   3
    ```

示例 1:

```
输入: [1,6,3,2,5]
输出: false
```

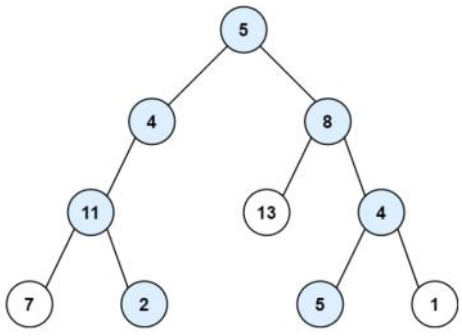
  - 递归法+左右中+递归判断子树结构是否满足BST条件+【i,j】区间是否满足BST
  - ```
    class Solution(object):
        def verifyPostorder(self, postorder):
            """
            :type postorder: List[int]
            :rtype: bool
            """
            def recur(i, j):
                if i >= j:
                    return True
                p = i
                while postorder[p] < postorder[j]:
                    p += 1
                m = p
                while postorder[p] > postorder[j]:
                    p += 1
                return p == j and recur(i, m-1) and recur(m, j-1)
            return recur(0, len(postorder)-1)
    ```

- 剑指 Offer 34. 二叉树中和为某一值的路径
    ```

给你二叉树的根节点 root 和一个整数目标和 targetSum ，找出所有 从根节点到叶子节点 路径总和等于给定目标和的路径。

叶子节点 是指没有子节点的节点。

示例 1：

- 

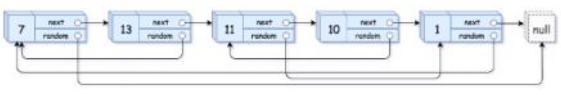

```
输入：root = [5,4,8,11,null,13,4,7,2,null,null,5,1], targetSum = 22
输出：[[5,4,11,2],[5,8,4,5]]
```

  - 递归法
  - ```
    # Definition for a binary tree node.
    # class TreeNode(object):
    #     def __init__(self, val=0, left=None, right=None):
    #         self.val = val
    #         self.left = left
    #         self.right = right
    class Solution(object):
        def pathSum(self, root, target):
            """
            :type root: TreeNode
            :type target: int
            :rtype: List[List[int]]
            """
            res = []
            path = []
            def recur(cur, diff):
                if not cur:
                    return
                path.append(cur.val)
                if cur.val == diff and not cur.left and not cur.right:
                    res.append(list(path))
                recur(cur.left, diff-cur.val)
                recur(cur.right, diff-cur.val)
                path.pop()
            recur(root, target)
            return res
    ```

- 剑指 Offer 35. 复杂链表的复制

请实现 copyRandomList 函数，复制一个复杂链表。在复杂链表中，每个节点除了有一个 next 指针指向下一个节点，还有一个 random 指针指向链表中的任意节点或者 null 。

示例 1：

- 


```
输入：head = [[7,null],[13,0],[11,4],[10,2],[1,0]]
输出：[[7,null],[13,0],[11,4],[10,2],[1,0]]
```

  - 复制合并+构建关系+拆分
  - ```
    """
    # Definition for a Node.
    class Node:
        def __init__(self, x, next=None, random=None):
            self.val = int(x)
            self.next = next
            self.random = random
    """
    ```

```
class Solution(object):
    def copyRandomList(self, head):
        """
        :type head: Node
        :rtype: Node
        """
        if not head:
            return
        cur = head
        while cur:
            temp = Node(cur.val)
            temp.next = cur.next
            cur.next = temp
            cur = temp.next
        cur = head
        while cur and cur.next:
            if cur.random:
                cur.next.random = cur.random.next
            cur = cur.next.next
        cur = head
        new = dummy = head.next
        while new and new.next:
            cur.next = cur.next.next
            new.next = new.next.next
            cur = cur.next
            new = new.next
        cur.next = None
        return dummy
```

- 剑指 Offer 36. 二叉搜索树与双向链表
  - 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。
  - 递归法
  - """

```
# Definition for a Node.
class Node(object):
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
"""
class Solution(object):
    def treeToDoublyList(self, root):
        """
        :type root: Node
        :rtype: Node
        """
        if not root:
            return
        self.pre = None
        self.head = None
        def inorder(root):
            if not root:
                return
            inorder(root.left)
            if not self.pre:
                self.head = root
            else:
                root.left, self.pre.right = self.pre, root
            self.pre = root
            inorder(root.right)
        inorder(root)
        self.pre.right, self.head.left = self.head, self.pre
        return self.head
```

- 剑指 Offer 37. 序列化二叉树
  - BFS
  - class Codec:

```
        def serialize(self, root):
            """Encodes a tree to a single string.

            :type root: TreeNode
            :rtype: str
            """
            if not root:
                return []
            res = []
            queue = [root]
            while  queue:
                cur = queue.pop(0)
                if cur:
                    res.append(cur.val)
                    queue.append(cur.left)
                    queue.append(cur.right)
                else:
                    res.append(None)
            return res

        def deserialize(self, data):
            """Decodes your encoded data to tree.

            :type data: str
            :rtype: TreeNode
            """
            if not data:
                return
            root = TreeNode(data[0])
            queue = [root]
            i = 1
            while queue:
                cur = queue.pop(0)
                if data[i] != None:
                    cur.left = TreeNode(data[i])
                    queue.append(cur.left)
                i += 1
                if data[i] != None:
                    cur.right = TreeNode(data[i])
                    queue.append(cur.right)
                i += 1
            return root

    # Your Codec object will be instantiated and called as such:
    # codec = Codec()
    # codec.deserialize(codec.serialize(root))
```

- 剑指 Offer 38. 字符串的排列
  - 回溯法
  - ```
    class Solution(object):
        def permutation(self, s):
            """
            :type s: str
            :rtype: List[str]
            """
            if not s:
                return []
            res = []
            cur = []
            s = list(s)
            s.sort()
            vis = [0] * len(s)
            def recur(x, l):
                if x == l:
                    res.append(''.join(cur))
                    return
                for i in range(len(s)):
                    if vis[i] == 1:
                        continue
                    else:
    ```
    ```

```
                if i > 0 and s[i] == s[i-1] and vis[i-1] == 0:
                    continue
                else:
                    cur.append(s[i])
                    vis[i] = 1
                    recur(x+1, l)
                    vis[i] = 0
                    cur.pop()
        recur(0, len(s))
        return res
```

- 剑指 Offer 43. 1 ~ n 整数中 1 出现的次数

  输入一个整数 n ，求1～n这n个整数的十进制表示中1出现的次数。

  例如，输入12，1～12这些整数中包含1 的数字有1、10、11和12，1一共出现了5次。

  - **示例 1:**

    ```
    输入：n = 12
    输出：5
    ```

  - 数学+当前位+高位+低位+根据当前位判断1的个数由（高、低）位确定
  - ```
    class Solution(object):
        def countDigitOne(self, n):
            """
            :type n: int
            :rtype: int
            """
            res = 0
            digit = 1
            high = n // 10
            cur = n % 10
            low = 0
            while high != 0 or cur != 0:
                if cur == 0:
                    res += high * digit
                elif cur == 1:
                    res += high * digit + low + 1
                else:
                    res += (high + 1) * digit
                low = cur * digit + low
                cur = high % 10
                high = high // 10
                digit = digit * 10
            return res
    ```

- 剑指 Offer 44. 数字序列中某一位的数字
  - 数学
  - ```
    class Solution(object):
        def findNthDigit(self, n):
            """
            :type n: int
            :rtype: int
            """
            start = 1
            digit = 1
            count = 9
            while n > count:
                n = n - count
                start = start * 10
                digit += 1
                count = 9 * start * digit
            num = start + (n - 1) // digit
            idx = (n - 1) % digit
            return int(str(num)[idx])
    ```

- 剑指 Offer 45. 把数组排成最小的数
  - 模拟快排
  - ```
    class Solution(object):
    ```
    ```

```
    def minNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: str
        """
        def recur(nums):
            if len(nums) < 2:
                return nums
            left = []
            right = []
            pivot = nums[0]
            del nums[0]
            for item in nums:
                if str(item) + str(pivot) > str(pivot) + str(item):
                    right.append(item)
                else:
                    left.append(item)
            return recur(left) + [pivot] + recur(right)
        nums = recur(nums)
        nums = [str(item) for item in nums]
        return ''.join(nums)
```

- 剑指 Offer 46. 把数字翻译成字符串
  - 动态规划
  - ```
    class Solution(object):
        def translateNum(self, num):
            """
            :type num: int
            :rtype: int
            """
            num = str(num)
            dp = [1] * (len(num)+1)
            for i in range(2, len(num)+1):
                if '10' <= num[i-2:i] <= '25':
                    dp[i] = dp[i-1] + dp[i-2]
                else:
                    dp[i] = dp[i-1]
            return dp[-1]
    ```
  - 优化版
  - ```
    class Solution(object):
        def translateNum(self, num):
            """
            :type num: int
            :rtype: int
            """
            num = str(num)
            if len(num) == 1:
                return 1
            p = 1
            q = 1
            r = 0
            for i in range(1, len(num)):
                if '10' <= num[i-1:i+1] <= '25':
                    r = p + q
                    p = q
                    q = r
                else:
                    r = q
                    p = q
                    q = r
            return r
    ```
- 剑指 Offer 47. 礼物的最大价值
  - 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?
  - 动态规划

- class Solution(object):
    def maxValue(self, grid):
        """
        :type grid: List[List[int]]
        :rtype: int
        """
        if not grid:
            return 0
        for i in range(1, len(grid)):
            grid[i][0] = grid[i][0] + grid[i-1][0]
        for i in range(1, len(grid[0])):
            grid[0][i] = grid[0][i] + grid[0][i-1]
        for i in range(1, len(grid)):
            for j in range(1, len(grid[0])):
                grid[i][j] = grid[i][j] + max(grid[i-1][j], grid[i][j-1])
        return grid[-1][-1]

- 剑指 Offer 48. 最长不含重复字符的子字符串
  - 滑动窗口
  - class Solution(object):
    def lengthOfLongestSubstring(self, s):
        """
        :type s: str
        :rtype: int
        """
        if not s:
            return 0
        if len(s) == 1:
            return 1
        res = 0
        left = 0
        right = 1
        while right < len(s):
            if s[right] not in s[left:right]:
                cur = right - left + 1
                res = max(res, cur)
                right += 1
            else:
                while left < right and s[right] in s[left:right]:
                    left += 1
        return res

- 剑指 Offer 49. 丑数
  - 三指针
  - class Solution(object):
    def nthUglyNumber(self, n):
        """
        :type n: int
        :rtype: int
        """
        if n == 1:
            return 1
        dp = [1]
        a = 0
        b = 0
        c = 0
        for i in range(1, n):
            cur2 = dp[a] * 2
            cur3 = dp[b] * 3
            cur5 = dp[c] * 5
            dp.append(min(cur2, cur3, cur5))
            if dp[-1] == cur2:
                a += 1
            if dp[-1] == cur3:
                b += 1
            if dp[-1] == cur5:
                c += 1
        return dp[-1]

- 剑指 Offer 50. 第一个只出现一次的字符

- 哈希
- ```
  class Solution(object):
      def firstUniqChar(self, s):
          """
          :type s: str
          :rtype: str
          """
          hash = collections.OrderedDict()
          for item in s:
              if item in hash:
                  hash[item] += 1
              else:
                  hash[item] = 1
          for item in hash:
              if hash[item] == 1:
                  return item
          return ' '
  ```

- 剑指 Offer 51. 数组中的逆序对
  - 归并排序
  - ```
    class Solution(object):
        def reversePairs(self, nums):
            """
            :type nums: List[int]
            :rtype: int
            """
            if not nums:
                return 0
            self.res = 0
            self.Mergesort(nums, 0, len(nums)-1)
            return self.res
        def Mergesort(self, num, left, right):
            if left >= right:
                return
            mid = left + (right - left) // 2
            self.Mergesort(num, left, mid)
            self.Mergesort(num, mid+1, right)
            self.Merge(num, left, mid, right)
        def Merge(self, num, left, mid, right):
            total = []
            q1 = left
            q2 = mid+1
            while q1 <= mid and q2 <= right:
                if num[q1] <= num[q2]:
                    total.append(num[q1])
                    q1 += 1
                else:
                    total.append(num[q2])
                    self.res += mid - q1 + 1
                    q2 += 1
            if q1 > mid:
                total.extend(num[q2:right+1])
            else:
                total.extend(num[q1:mid+1])

            for i in range(len(total)):
                num[left+i] = total[i]
    ```
  - 快速排序
  - ```
    # 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可
    # @param data int整型一维数组
    # @return int整型
    #
    class Solution:
        def InversePairs(self , data: List[int]) -> int:
            # write code here
            if not data:
                return 0
    ```
    ```

```
        self.res = 0
        def recur(data):
            if len(data) < 2:
                return data
            pivot = data[0]
            small = []
            big = []
            for item in data[1:]:
                if item >= pivot:
                    big.append(item)
                else:
                    small.append(item)
                    self.res += len(big) + 1
            return recur(small) + [pivot] + recur(big)
        recur(data)
        return self.res % (10 ** 9 + 7)
```

- 剑指 Offer 53 - II. 0～n-1中缺失的数字
  - 二分法+有序数组
  - ```
    class Solution(object):
        def missingNumber(self, nums):
            """
            :type nums: List[int]
            :rtype: int
            """
            # 找到右子数组的首位元素
            left = 0
            right = len(nums)-1
            while left <= right:
                mid = left + (right-left) // 2
                if nums[mid] == mid:
                    left += 1
                else:
                    right -= 1
            return left
    ```
- 剑指 Offer 56 - I. 数组中数字出现的次数
  - 位运算：异或
  - ```
    class Solution(object):
        def singleNumbers(self, nums):
            """
            :type nums: List[int]
            :rtype: List[int]
            """
            x = y = 0
            temp = 0
            for item in nums:
                temp = temp ^ item
            m = 1
            while temp & m == 0:
                m = m << 1
            for item in nums:
                if item & m:
                    x = x ^ item
                else:
                    y = y ^ item
            return [x, y]
    ```
- 剑指 Offer 56 - II. 数组中数字出现的次数 II
  - 位运算+取余
  - ```
    class Solution(object):
        def singleNumber(self, nums):
            """
            :type nums: List[int]
            :rtype: int
            """
            temp = [0] * 32
            for item in nums:
                for i in range(len(temp)):
    ```
    ```

```
                temp[i] += item & 1
                item >>= 1
        for j in range(len(temp)):
            temp[j] = str(temp[j] % 3)
        return int(''.join(temp[::-1]), 2)
```

- 剑指 Offer 57 - II. 和为s的连续正数序列

  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。

  - 双指针+数列求和
  - ```
    class Solution(object):
        def findContinuousSequence(self, target):
            """
            :type target: int
            :rtype: List[List[int]]
            """
            left = 1
            right = 2
            res = []
            while left < right:
                if (right+left) * (right - left + 1) // 2 == target:
                    cur = [item for item in range(left, right+1)]
                    res.append(cur[::1])
                    left += 1
                    right += 1
                elif (right+left) * (right - left + 1) // 2 < target:
                    right += 1
                else:
                    left += 1
            return res
    ```
  - ```
    class Solution:
        def findContinuousSequence(self, target: int) -> List[List[int]]:
            res = []
            def recur(num, diff):
                cur.append(num)
                if num > diff:
                    return
                if num == diff:
                    res.append(cur[:])
                    return
                recur(num+1, diff-num)
                cur.pop()
            for i in range(1, target):
                cur = []
                recur(i, target)
            return res
    ```

- 剑指 Offer 58 - I. 翻转单词顺序

  输入一个英文句子，翻转句子中单词的顺序，但单词内字符的顺序不变。为简单起见，标点符号和普通字母一样处理。例如输入字符串"I am a student. "，则输出"student. a am I"。

  - **示例 1：**

    ```
    输入: "the sky is blue"
    输出: "blue is sky the"
    ```

  - 双指针
  - ```
    class Solution(object):
        def reverseWords(self, s):
            """
            :type s: str
            :rtype: str
            """
            s = s.strip()
            if not s:
                return ''
    ```
    ```

```
        left = right = len(s) - 1
        res = []
        while left >= 0:
            while left >= 0 and s[left] != ' ':
                left -= 1
            res.append(s[left+1:right+1])
            while left >= 0 and s[left] == ' ':
                left -= 1
            right = left
        return ' '.join(res)
```

- 剑指 Offer 58 - II. 左旋转字符串
  - 字符串的左旋转操作是把字符串前面的若干个字符转移到字符串的尾部。请定义一个函数实现字符串左旋转操作的功能。比如，输入字符串"abcdefg"和数字2，该函数将返回左旋转两位得到的结果"cdefgab"。
  - 双指针+分部分旋转
  - ```
    class Solution(object):
        def reverseLeftWords(self, s, n):
            """
            :type s: str
            :type n: int
            :rtype: str
            """
            s = list(s)
            s1 = self.reverses(s[:n])
            s2 = self.reverses(s[n:])
            res = self.reverses(s1+s2)
            return ''.join(res)
        def reverses(self, num):
            left = 0
            right = len(num) - 1
            while left <= right:
                num[left], num[right] = num[right], num[left]
                left += 1
                right -= 1
            return num
    ```
  - 坐标取余
  - ```
    class Solution(object):
        def reverseLeftWords(self, s, n):
            """
            :type s: str
            :type n: int
            :rtype: str
            """
            res = [''] * len(s)
            for i in range(len(s)):
                res[(i-n)%len(s)] = s[i]
            return ''.join(res)
    ```
- 剑指 Offer 59 - I. 滑动窗口的最大值
  - 滑动窗口+单调栈(降序排列)
  - ```
    class Solution(object):
        def maxSlidingWindow(self, nums, k):
            """
            :type nums: List[int]
            :type k: int
            :rtype: List[int]
            """
            if not nums or len(nums) == 0:
                return []
            res = []
            max_que = collections.deque()
            for i in range(k):
                while max_que and max_que[-1] < nums[i]:
                    max_que.pop()
                max_que.append(nums[i])
            res.append(max_que[0])
    ```
    ```

```
        for i in range(k, len(nums)):
            if max_que[0] == nums[i-k]:
                max_que.popleft()
            while max_que and max_que[-1] < nums[i]:
                max_que.pop()
            max_que.append(nums[i])
            res.append(max_que[0])
        return res
```

- 剑指 Offer 59 - II. 队列的最大值
  - 请定义一个队列并实现函数 `max_value` 得到队列里的最大值，要求函数`max_value`、`push_back` 和 `pop_front` 的**均摊**时间复杂度都是O(1)。

    若队列为空，`pop_front` 和 `max_value` 需要返回 -1
  - 栈+单调栈（降序排列）
  - ```
    class MaxQueue(object):

        def __init__(self):
            self.queue_in = collections.deque()
            self.queue_max = collections.deque()

        def max_value(self):
            """
            :rtype: int
            """
            if not self.queue_max:
                return -1
            else:
                return self.queue_max[0]

        def push_back(self, value):
            """
            :type value: int
            :rtype: None
            """
            self.queue_in.append(value)
            while self.queue_max and self.queue_max[-1] <= value:
                self.queue_max.pop()
            self.queue_max.append(value)

        def pop_front(self):
            """
            :rtype: int
            """
            if not self.queue_in:
                return -1
            cur = self.queue_in.popleft()
            if self.queue_max and self.queue_max[0] == cur:
                self.queue_max.popleft()
            return cur
    # Your MaxQueue object will be instantiated and called as such:
    # obj = MaxQueue()
    # param_1 = obj.max_value()
    # obj.push_back(value)
    # param_3 = obj.pop_front()
    ```
- 剑指 Offer 60. n个骰子的点数
  - 动态规划
  - ```
    class Solution:
        def dicesProbability(self, n: int) -> List[float]:
            dp = [1/6] * 6
            for i in range(2, n+1):
                temp = [0] * (5 * i + 1)
                for j in range(len(dp)):
                    for k in range(6):
                        temp[j+k] += dp[j] / 6
                dp = temp
            return dp
    ```

- 剑指 Offer 61. 扑克牌中的顺子

  从**若干副扑克牌**中随机抽 5 张牌，判断是不是一个顺子，即这5张牌是不是连续的。2～10为数字本身，A为1，J为11，Q为12，K为13，而大、小王为 0 ，可以看成任意数字。A 不能视为 14。

  - **示例 1:**

    ```
    输入: [1,2,3,4,5]
    输出: True
    ```

  - 排序
  - ```
    class Solution(object):
        def isStraight(self, nums):
            """
            :type nums: List[int]
            :rtype: bool
            """
            nums = sorted(nums)
            joker = 0
            for i in range(len(nums)-1):
                if nums[i] == 0:
                    joker += 1
                elif nums[i] == nums[i+1]:
                    return False
            return nums[-1]-nums[joker]<5
    ```
- 剑指 Offer 62. 圆圈中最后剩下的数字
  - 模拟+取余
  - ```
    class Solution(object):
        def lastRemaining(self, n, m):
            """
            :type n: int
            :type m: int
            :rtype: int
            """
            idx = 0
            l = 1
            while l < n:
                l += 1
                idx = (idx + m) % l
            return idx
    ```
- 剑指 Offer 64. 求1+2+...+n
  - ```
    class Solution:
        def __init__(self):
            self.res = 0
        def sumNums(self, n: int) -> int:
            n > 0 and self.sumNums(n-1)
            self.res += n
            return self.res
    ```
- 剑指 Offer 65. 不用加减乘除做加法
  - 位运算
  - ```
    class Solution(object):
        def add(self, a, b):
            """
            :type a: int
            :type b: int
            :rtype: int
            """
            x = 0xffffffff
            a = a & x
            b = b & x
            while b != 0:
                a, b = a^b, (a&b)<<1&x
            if a <= 0x7fffffff:
                return a
            else:
    ```

```
                return ~(a ^ x)
```

- 剑指 Offer 66. 构建乘积数组
  - 动态规划
  - ```
    class Solution(object):
        def constructArr(self, a):
            """
            :type a: List[int]
            :rtype: List[int]
            """
            if not a:
                return []
            left = [1] * len(a)
            right = [1] * len(a)
            res = []
            for i in range(1, len(a)):
                left[i] = left[i-1] * a[i-1]
            for j in range(len(a)-2, -1, -1):
                right[j] = right[j+1] * a[j+1]
            for i in range(len(a)):
                res.append(left[i] * right[i])
            return res
    ```
- 剑指 Offer 67. 把字符串转换成整数
  - 遍历
  - ```
    class Solution(object):
        def strToInt(self, str):
            """
            :type str: str
            :rtype: int
            """
            s = str.strip()
            if not s:
                return 0
            INT_MAX = 2 ** 31 -1
            INT_MIN = -1 * 2 ** 31
            bound = INT_MAX // 10
            sign = 1
            start = 0
            res = 0
            if s[0] == '-':
                sign = -1
            if s[0] == '+' or s[0] == '-':
                start = 1
            for item in s[start:]:
                if not '0' <= item <= '9':
                    break
                if res > bound or res == bound and item > '7':
                    if sign == 1:
                        return INT_MAX
                    else:
                        return INT_MIN
                res = res * 10 + int(item)
            return sign * res
    ```
- 剑指 Offer 68 - II. 二叉树的最近公共祖先
  - 回溯法+找路径
  - ```
    # Definition for a binary tree node.
    # class TreeNode(object):
    #     def __init__(self, x):
    #         self.val = x
    #         self.left = None
    #         self.right = None

    class Solution(object):
        def lowestCommonAncestor(self, root, p, q):
            """
            :type root: TreeNode
            :type p: TreeNode
    ```
    ```

```
        :type q: TreeNode
        :rtype: TreeNode
        """
        if not root:
            return
        path_p = []
        path_q = []
        res = 0
        self.find_path(root, p, path_p)
        self.find_path(root, q, path_q)
        if not path_q or not path_p:
            return
        min_l = min(len(path_p), len(path_q))
        for i in range(min_l):
            if path_p[i] == path_q[i]:
                res = path_q[i]
        return TreeNode(res)
    def find_path(self, root, p, path):
        if not root:
            return
        path.append(root.val)
        if root.val == p.val:
            return True
        if self.find_path(root.left, p, path):
            return True
        if self.find_path(root.right, p, path):
            return True
        else:
            path.pop()
            return False
```

-