1. Two Sum
2. **Example 1:**
3. **Input:** nums = [2,7,11,15], target = 9
4. **Output:** [0,1]
5. **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].
6. class Solution:
7. def twoSum(self, nums, target):
9. # for i in range(len(nums)):
10. #     for j in range(1, (len(nums) - i)):
11. #         if nums[i] + nums[i + j] == target:
12. #             return i, i+j
13. # for i in range(len(nums)):
14. #     if i + 1 < len(nums):
15. #         if nums[i] + nums[i + 1] == target:
16. #             return i, i + 1
17. # won't work as we have to check all the numbers not just corresponding
18. test = {}
19. # for i in range(len(nums)):
20. #     if nums[i] in test:
21. #         return test[nums[i]], i
22. #     else:
23. #         test[target - nums[i]] = i
24. for i in range(len(nums)):
25. if nums[i] not in test:
26. test[target - nums[i]] = i
27. else:
28. return test[nums[i]], i

15. 3Sum (Medium)

**Input:** nums = [-1,0,1,2,-1,-4]

**Output:** [[-1,-1,2],[-1,0,1]]

**Explanation:**

nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.

nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.

nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.

The distinct triplets are [-1,0,1] and [-1,-1,2].

Notice that the order of the output and the order of the triplets does not matter.

class Solution:

    def threeSum(self, nums: List[int]) -> List[List[int]]:

        # res = []

        # nums.sort()

        # for i, a in enumerate(nums):

        #     if i > 0 and a == nums[i - 1]:

        #         continue

        #     l, r = i + 1, len(nums) - 1

        #     while l < r:

        #         s = a + nums[l] + nums[r]

        #         if s > 0:

        #             r -= 1

        #         elif s < 0:

        #             l += 0

        #         else:

        #             res.append([a, nums[l], nums[r]])

        #             l += 1

        #             while nums[l] == nums[l-1] and l < r:

        #                 l += 1

        # return res

        res = []

        nums.sort()

        for i, a in enumerate(nums):

            if i > 0 and a == nums[i-1]:

                continue

            l, r = i+1, len(nums)-1

            while l < r:

                s = a + nums[l] + nums[r]

                if s < 0:

                    l +=1

                elif s > 0:

                    r -= 1

                else:

                    res.append([a, nums[l], nums[r]])

                    l += 1

                    while l < r and nums[l] == nums[l-1]:

                        l += 1

        return res

26. Remove Duplicates from Sorted Array

**Example 1:**

**Input:** nums = [1,1,2]

**Output:** 2, nums = [1,2,\_]

**Explanation:** Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

**Example 2:**

**Input:** nums = [0,0,1,1,1,2,2,3,3,4]

**Output:** 5, nums = [0,1,2,3,4,\_,\_,\_,\_,\_]

**Explanation:** Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

class Solution:

    def removeDuplicates(self, nums: List[int]) -> int:

        x = 1

        for i in range(len(nums) - 1):

            if nums[i] != nums[i + 1]:

                nums[x] = nums[i + 1]

                x += 1

        return x

242. Valid Anagram

**Example 1:**

**Input:** s = "anagram", t = "nagaram"

**Output:** true

**Example 2:**

**Input:** s = "rat", t = "car"

**Output:** false

class Solution:

    def isAnagram(self, s: str, t: str) -> bool:

        #return sorted(s) == sorted(t)

        if len(s) != len(t):

            return False

        else:

            test = {}

            for char in s:

                if char not in test:

                    test[char] = 1

                else:

                    test[char] += 1

            for char in t:

                if char not in test:

                    return False

                else:

                    test[char] -= 1

            for v in test.values():

                if v != 0:

                    return False

            return True

        # print(stack)

        # a = len(s)

        # b = len(t)

        # mx = a

        # if b > a:

        #     mx = b

        # #j = -1

        # if a != b:

        #     return False

        # else:

        #     for i in range(a):

        #         found = False

        #         for j in range(b):

        #             if t[j] == stack[j]:

        #                 stack.pop()

        #                 found = True

        #                 print(char)

        #                 print(stack)

        #                 break

        #         if found == False:

        #             j -= 1

        #     # if len(stack) == 0:

        #     #     return True

        #     # else:

        #     #     return False

        #     return stack == []

66. Plus One

**Example 1:**

**Input:** digits = [1,2,3]

**Output:** [1,2,4]

**Explanation:** The array represents the integer 123.

Incrementing by one gives 123 + 1 = 124.

Thus, the result should be [1,2,4].

class Solution:

    def plusOne(self, digits: List[int]) -> List[int]:

        # if len(digits) == 1:

        #     if digits[0] == 9:

        #         digits[0] = 1

        #         #digits[1] = 0

        #         digits.append(0)

        #     else:

        #         digits[0] += 1

        #     return digits

        # else:

        #     sum = digits[0]

        #     for i in range(len(digits) - 1):

        #         sum = ((sum \* 10) + digits[i + 1])

        #     digits.clear()

        #     sum += 1

        #     while sum != 0:

        #         x = sum % 10

        #         digits.append(x)

        #         sum //= 10

        #     digits.reverse()

        #     return digits

        #-------------------

        # sum = 0

        # for i in range(len(digits)):

        #     sum += digits[i] \* pow(10, (len(digits) - 1 - i))

        #sum += 1

        #digits.clear()

        #x = str(sum)

        # for char in x:

        #     digits.append(int(char))

        # return digits

        # for i in range(len(x)):

        #     digits.append(int(x[i]))

        # return digits

        #----------------------

        #one liner

        # return [int(char) for char in str(sum + 1)]

        s = ""

        for char in digits:

            s += str(char)

        sum = int(s) + 1

        digits.clear()

        for char in str(sum):

            digits.append(int(char))

        return digits

202. Happy Number

**Input:** n = 19

**Output:** true

**Explanation:**

12 + 92 = 82

82 + 22 = 68

62 + 82 = 100

12 + 02 + 02 = 1

class Solution:

    def isHappy(self, n: int) -> bool:

        # if n== 1:

        #     return True

        # s = str(n)

        # store = {}

        # x = 0

        # for char in s:

        #     x += pow(int(char), 2)

        # store[s] = x

        # while s in store.keys():

        #     s = str(store[s])

        #     if s in store:

        #         return False

        #     x = 0

        #     for char in s:

        #         x += pow(int(char), 2)

        #     if x == 1:

        #         return True

        #     store[s] = x

        if n == 1:

            return True

        s = set()

        while n != 1:

            x = 0

            # num = str(n)

            # for i in range(len(num)):

            #     x += int(num[i]) \*\* 2

            # for i in range(len(str(n))):

            #     x += (int(str(n)[i])) \*\* 2

            for char in (str(n)):

                x += (int(char)) \*\* 2

            n = x

            if n == 1:

                return True

            if n in s:

                return False

            else:

                s.add(n)

118. Pascal's Triangle

class Solution:

    def generate(self, numRows: int) -> List[List[int]]:

        res = [[1]]

        for i in range(numRows - 1):

            temp = [0] + res[-1] + [0]

            row = []

            for j in range(len(res[-1]) + 1):

                row.append(temp[j] + temp[j + 1])

            res.append(row)

        return res

88. Merge Sorted Array

**Example 1:**

**Input:** nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3

**Output:** [1,2,2,3,5,6]

**Explanation:** The arrays we are merging are [1,2,3] and [2,5,6].

The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

class Solution:

    def merge(self, nums1: List[int], m: int, nums2: List[int], n: int) -> None:

        """

        Do not return anything, modify nums1 in-place instead.

        """

        while m > 0 and n > 0:

            if nums1[m - 1] >= nums2[n - 1]:

                nums1[m + n - 1] = nums1[m - 1]

                m -= 1

            else:

                nums1[m + n -1] = nums2[n - 1]

                n -= 1

        if n > 0:

            nums1[:n] = nums2[:n]

169. Majority Element

**Example 1:**

**Input:** nums = [3,2,3]

**Output:** 3

**Example 2:**

**Input:** nums = [2,2,1,1,1,2,2]

**Output:** 2

class Solution:

    def majorityElement(self, nums: List[int]) -> int:

        # test = {}

        # for char in nums:

        #     if char not in test:

        #         test[char] = 1

        #     else:

        #         test[char] += 1

        # mx = 0

        # for v in test.values():

        #     if v > (len(nums) / 2) and v > mx:

        #         mx = v

        # for key in test.keys():

        #     if test[key] == mx:

        #         return key

        # test = {}

        # mx, res = 0, 0

        # for char in nums:

        #     if char not in test:

        #         test[char] = 1

        #     else:

        #         test[char] += 1

        #     if test[char] > mx:

        #         mx = test[char]

        #         res = char

        # return res

        res, count = 0, 0

        for n in nums:

            if count == 0:

                res = n

            if n == res:

                count += 1

            else:

                count -= 1

        return res

        # count = {}

        # res, maxCount = 0, 0

        # for n in nums:

        #     count[n] = 1 + count.get(n, 0)

        #     res = n if count[n] > maxCount else res

        #     maxCount = max(count[n], maxCount)

        # return res

21. Merge Two Sorted Lists

**Example 1:**



# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) -> Optional[ListNode]:

        temp = current = ListNode(0)

        while list1 and list2:

            if list1.val < list2.val:

                current.next = list1

                list1 = list1.next

            else:

                current.next = list2

                list2 = list2.next

            current = current.next

        current.next = list1 or list2

        return temp.next

70. Climbing Stairs

**Example 1:**

**Input:** n = 2

**Output:** 2

**Explanation:** There are two ways to climb to the top.

1. 1 step + 1 step

2. 2 steps

**Example 2:**

**Input:** n = 3

**Output:** 3

**Explanation:** There are three ways to climb to the top.

1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step

class Solution:

    def climbStairs(self, n: int) -> int:

        prev1 = 1

        prev2 = 2

        current = 0

        if n <= 2:

            return n

        else:

            for i in range(2, n):

                current = prev1 + prev2

                prev1 = prev2

                prev2 = current

            return current

350. Intersection of Two Arrays II

**Example 1:**

**Input:** nums1 = [1,2,2,1], nums2 = [2,2]

**Output:** [2,2]

**Example 2:**

**Input:** nums1 = [4,9,5], nums2 = [9,4,9,8,4]

**Output:** [4,9]

**Explanation:** [9,4] is also accepted.

class Solution:

    def intersect(self, nums1: List[int], nums2: List[int]) -> List[int]:

        #space - O(min(len(nums1), len(nums2)), Time - O(n + m)

        intersect = []

        test = {}

        #find the length of the shortest array and use that for

        #dictionary for least memory.

        for char in nums1:

            if char not in test:

                test[char] = 1

            else:

                test[char] += 1

        for n in nums2:

            if n in test:

                if test[n] > 0:

                    intersect.append(n)

                    test[n] -= 1

        return intersect

        #Time - nlogn + n = nlogn from sorting, space - O(1)

        # intersect = []

        # nums1.sort()

        # nums2.sort()

        # i,j = 0, 0

        # while i < len(nums1) and j < len(nums2):

        #     if nums1[i] < nums2[j]:

        #         i += 1

        #     elif nums2[j < nums1[i]]:

        #         j += 1

        #     else:

        #         intersect.append(nums1[i])

        #         i += 1

        #         j += 1

        # return intersect

        # Since nums2 is too big, it's stored on disc. We can still use the first algo and build our hash map / dict and after that we can break nums2 into as small chunks as possible and check if that belongs in hash map and append to our intersect array/.

412. Fizz Buzz

Given an integer n, return *a string array*answer*(****1-indexed****) where*:

* answer[i] == "FizzBuzz" if i is divisible by 3 and 5.
* answer[i] == "Fizz" if i is divisible by 3.
* answer[i] == "Buzz" if i is divisible by 5.
* answer[i] == i (as a string) if none of the above conditions are true.

**Example 1:**

**Input:** n = 3

**Output:** ["1","2","Fizz"]

**Example 2:**

**Input:** n = 5

**Output:** ["1","2","Fizz","4","Buzz"]

class Solution:

    def fizzBuzz(self, n: int) -> List[str]:

        s = []

        for i in range(1, n + 1):

            if i % 3 == 0 and i % 5 == 0:

                s.append("FizzBuzz")

            elif i % 3 == 0:

                s.append("Fizz")

            elif i % 5 == 0:

                s.append("Buzz")

            else:

                s.append(str(i))

        return s

326. Power of Three

**Example 1:**

**Input:** n = 27

**Output:** true

**Explanation:** 27 = 33

**Example 2:**

**Input:** n = 0

**Output:** false

**Explanation:** There is no x where 3x = 0.

**Example 3:**

**Input:** n = -1

**Output:** false

**Explanation:** There is no x where 3x = (-1).

class Solution:

    def isPowerOfThree(self, n: int) -> bool:

        # if n == 0:

        #     return False

        # res = 0

        # x = 0

        # while res < n:

        #     res = pow(3, x)

        #     if n == res:

        #         return True

        #     x += 1

        # return False

        while n >= 3:

            if n % 3 != 0:

                return False

            n /= 3

        return n == 1

        # x = 0

        # if n > 0:

        #     x = math.log3(n)

        # x = int(x)

        # print(isinstance(x, int))

        # return isinstance(x, int) and n > 0

        #return isinstance(math.log10(n) / math.log10(3), int) and n > 0

        if n > 0:

            return (math.log10(n) / math.log10(3)) % 1 == 0

171. Excel Sheet Column Number

For example:

A -> 1

B -> 2

C -> 3

...

Z -> 26

AA -> 27

AB -> 28

...

**Example 1:**

**Input:** columnTitle = "A"

**Output:** 1

**Example 2:**

**Input:** columnTitle = "AB"

**Output:** 28

class Solution:

    def titleToNumber(self, columnTitle: str) -> int:

        column = 0

        for i in range(len(columnTitle)):

            value = ord(columnTitle[i]) - ord('A') + 1

            column += value \* pow(26, len(columnTitle) - 1 - i)

        return column

        # column = 0

        # test = {'A' : 1, 'B' : 2, 'C' : 3,

        #         'D' : 4,

        #         'E' : 5,

        #         'F' : 6,

        #         'G' : 7,

        #         'H' : 8,

        #         'I' : 9,

        #         'J' : 10,

        #         'K' : 11,

        #         'L' : 12,

        #         'M' : 13,

        #         'N' : 14,

        #         'O' : 15,

        #         'P' : 16,

        #         'Q' : 17,

        #         'R' : 18,

        #         'S' : 19,

        #         'T' : 20,

        #         'U' : 21,

        #         'V' : 22,

        #         'W' : 23,

        #         'X' : 24,

        #         'Y' : 25,

        #         'Z' : 26

        #         }

        # for i in range(len(columnTitle)):

        #     column += test[columnTitle[i]] \* pow(26, len(columnTitle) - i -1)

        # return column

344. Reverse String

**Example 1:**

**Input:** s = ["h","e","l","l","o"]

**Output:** ["o","l","l","e","h"]

class Solution:

    def reverseString(self, s: List[str]) -> None:

        """

        Do not return anything, modify s in-place instead.

        """

        l, r = 0, len(s) - 1

        while l < r:

            s[l], s[r] = s[r], s[l]

            l += 1

            r -= 1

        #recursion requires extra memory though cause of the call stack

        # def reverse(l, r):

        #     if l < r:

        #         s[l], s[r] = s[r], s[l]

        #         reverse(l + 1, r - 1)

        # reverse(0, len(s) - 1)

        #using stack, requires extra memory

        # stack = []

        # for char in s:

        #     stack.append(char)

        # for i in range(len(stack)):

        #     s[i] = stack.pop()

        # nums = []

        # for i in s[::-1]:

        #     nums.append(i)

        # s.clear()

        # #s = nums

        # for i in range(len(nums)):

        #     s.append(nums[i])

283. Move Zeroes

**Example 1:**

**Input:** nums = [0,1,0,3,12]

**Output:** [1,3,12,0,0]

**Example 2:**

**Input:** nums = [0]

**Output:** [0]

class Solution:

    def moveZeroes(self, nums: List[int]) -> None:

        """

        Do not return anything, modify nums in-place instead.

        """

        l = 0

        for i in range(len(nums)):

            if nums[i] != 0:

                # temp = nums[i]

                # nums[i] = nums[l]

                # nums[l] = temp

                nums[l], nums[i] = nums[i], nums[l]

                l += 1

387. First Unique Character in a String

**Example 1:**

**Input:** s = "leetcode"

**Output:** 0

**Example 2:**

**Input:** s = "loveleetcode"

**Output:** 2

**Example 3:**

**Input:** s = "aabb"

**Output:** -1

class Solution:

    def firstUniqChar(self, s: str) -> int:

        test = {}

        for char in s:

            if char not in test:

                test[char] = 1

            else:

                test[char] += 1

        for i in range(len(s)):

            if test[s[i]] == 1:

                return i

        return -1

        # test = {}

        # for char in s:

        #     if char not in test: #{'a' = 2, 'b' = 2}

        #         test[char] = 1

        #     else:

        #         test[char] += 1

        # found = False

        # for x, y in test.items():

        #     if y == 1:

        #         found = True

        #         z = x

        #         break

        # if found == False:

        #     return -1

        # for i in range(len(s)):

        #     if s[i] == z:

        #         return i

13. Roman to Integer

* I can be placed before V (5) and X (10) to make 4 and 9.
* X can be placed before L (50) and C (100) to make 40 and 90.
* C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

**Example 1:**

**Input:** s = "III"

**Output:** 3

**Explanation:** III = 3.

**Example 2:**

**Input:** s = "LVIII"

**Output:** 58

**Explanation:** L = 50, V= 5, III = 3.

**Example 3:**

**Input:** s = "MCMXCIV"

**Output:** 1994

**Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

class Solution:

    def romanToInt(self, s: str) -> int:

        romans = {

            'I' : 1,

            'V' : 5,

            'X' : 10,

            'L' : 50,

            'C' : 100,

            'D' : 500,

            'M' : 1000

        }

        numerals = 0

        # s = s.replace('IV', 'IIII')

        # s = s.replace('IX', 'VIIII')

        # s = s.replace('XL', 'XXXX')

        # s = s.replace('XC', 'LXXXX').replace('CD', 'CCCC').replace("CM", 'DCCCC')

        # for characters in s:

        #     numerals += romans[characters]

        # return numerals

        for i in range(len(s)):

            numerals += romans[s[i]]

        #l = len(s)

        for i in range(len(s) - 1):

            if romans[s[i]] < romans[s[i + 1]]:

                numerals -= romans[s[i]]

            else:

                numerals += romans[s[i]]

        return numerals + romans[s[-1]]

14. Longest Common Prefix

**Example 1:**

**Input:** strs = ["flower","flow","flight"]

**Output:** "fl"

**Example 2:**

**Input:** strs = ["dog","racecar","car"]

**Output:** ""

**Explanation:** There is no common prefix among the input strings.

class Solution:

    def longestCommonPrefix(self, strs: List[str]) -> str:

        # smax, smin = max(strs), min(strs)

        # i, match = 0, 0

        # while i < len(smin) and smax[i] == smin[i]:

        #     i += 1

        #     match += 1

        # return smin[0:match]

        # I think this is only checking the max and min strings, not optimum solution

        smin = min(strs, key = len)

        if smin == "":

            return ""

        for i in range(len(smin)):

            for others in strs:

                if others[i] != smin[i]:

                    return smin[:i]

        return smin

20. Valid Parentheses

**Example 1:**

**Input:** s = "()"

**Output:** true

**Example 2:**

**Input:** s = "()[]{}"

**Output:** true

**Example 3:**

**Input:** s = "(]"

**Output:** false

class Solution:

    def isValid(self, s: str) -> bool:

        stack = []

        for char in s:

            if char == '(':

                stack.append(')')

            elif char == '{':

                stack.append('}')

            elif char == '[':

                stack.append(']')

            elif char == ')' or char == '}' or char == ']':

                if stack == [] or stack.pop() != char:

                    return False

            # elif stack == [] or stack.pop() != char:

            #     return False

        return stack == []

        # stack = []

        # characters = {')' : '(', '}' : '{', ']' : '['}

        # for char in s:

        #     if char in characters.values():

        #         stack.append(char)

        #     elif char in characters.keys():

        #         if stack == [] or stack.pop() != characters[char]:

        #             return False

        # return stack == []

        # x1, y1, z1, x2, y2, z2 = 0, 0, 0, 0, 0, 0

        # for char in s:

        #     if char == '(':

        #         x1 += 1

        #     elif char == '{':

        #         y1 += 1

        #     elif char == '[':

        #         z1 += 1

        #     elif char == ')':

        #         x2 += 1

        #     elif char == '}':

        #         y2 += 1

        #     else:

        #         z2 += 1

        # if x1 == x2:

        #     if y1 == y2:

        #         if z1 == z2:

        #             return True

        # else:

        #     return False

22. Generate Parentheses (Medium)

Given n pairs of parentheses, write a function to *generate all combinations of well-formed parentheses*.

**Example 1:**

**Input:** n = 3

**Output:** ["((()))","(()())","(())()","()(())","()()()"]

**Example 2:**

**Input:** n = 1

**Output:** ["()"]

class Solution:

    def generateParenthesis(self, n: int) -> List[str]:

        stack = []

        res = []

        def backtrack(openN, closedN):

            if openN == closedN == n:

                res.append("".join(stack))

                return None

            if openN < n:

                stack.append("(")

                print("After appending", stack)

                backtrack(openN + 1, closedN)

                stack.pop()

                print("After pop", stack)

            if closedN < openN:

                stack.append(")")

                print("After appending", stack)

                backtrack(openN, closedN + 1)

                stack.pop()

                print("After pop", stack)

        backtrack(0, 0)

        return res