[1. Two Sum](https://leetcode.com/problems/two-sum/)

class Solution:

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

dic = {}

for i,n in enumerate(nums):

diff = target-n

if diff in dic:

return [i, dic[diff]]

dic[n]=i

[167. Two Sum II - Input Array Is Sorted](https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/)

class Solution:

def twoSum(self, numbers: List[int], target: int) -> List[int]:

l = 0

r = len(numbers)-1

while l<r:

currentSum = numbers[l] + numbers[r]

if currentSum == target:

return [l+1, r+1]

elif currentSum <target:

l = l+1

else:

r = r-1

[2. Add Two Numbers](https://leetcode.com/problems/add-two-numbers/)

# Definition for singly-linked list.

# class ListNode:

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

# self.val = val

# self.next = next

class Solution:

def addTwoNumbers(self, l1: Optional[ListNode], l2: Optional[ListNode]) -> Optional[ListNode]:

dummy = ListNode(0)

current = dummy

carry = 0

while l1 or l2 :

val1 = l1.val if l1 else 0

val2 = l2.val if l2 else 0

addition = val1 + val2 + carry

rem = addition % 10

carry = addition//10

current.next = ListNode(rem)

current = current.next

l1 = l1.next if l1 else None

l2 = l2.next if l2 else None

if carry:

current.next = ListNode(carry)

return dummy.next

[3. Longest Substring Without Repeating Characters](https://leetcode.com/problems/longest-substring-without-repeating-characters/)

class Solution:

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

if len(s)==1:

return 1

count = 0

s\_result = ''

for ch in s:

if ch not in s\_result:

s\_result+=ch

else:

s\_result = s\_result[s\_result.index(ch)+1:]+ch

count = max(count, len(s\_result))

return count

[4. Median of Two Sorted Arrays](https://leetcode.com/problems/median-of-two-sorted-arrays/)

class Solution:

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

merged = sorted(nums1+nums2)

n = len(merged)

if n%2==0:

middle1 = merged[n//2]

middle2 = merged[n//2-1]

median =( middle1 + middle2) / 2

else:

median = merged[n//2]

return median

class Solution:

def findMedianSortedArrays(self, nums1, nums2) -> float:

total = len(nums1)+len(nums2)

half = total//2

if len(nums2)<len(nums1):

nums1,nums2=nums2,nums1

l=0

r=len(nums1)-1

while True:

m1 = (l+r)//2

m2=half-m1-2

nums1\_left =nums1[m1] if m1>=0 else float('-inf')

nums1\_right=nums1[m1+1] if m1+1<len(nums1) else float('inf')

nums2\_left = nums2[m2] if m2>=0 else float('-inf')

nums2\_right = nums2[m2+1] if m2+1<len(nums2) else float('inf')

if nums1\_left<=nums2\_right and nums2\_left<=nums1\_right:

if total%2:

return min(nums1\_right,nums2\_right)

return (min(nums1\_right,nums2\_right)+max(nums1\_left,nums2\_left))/2

elif nums1\_left>nums2\_right:

r=m1-1

else:

l=m1+1

[5. Longest Palindromic Substring](https://leetcode.com/problems/longest-palindromic-substring/)

class Solution:

def longestPalindrome(self, s: str) -> str:

s\_result = ''

count = 0

for i in range(len(s)):

l = i

r = i

while l>=0 and r<len(s) and s[l]==s[r]:

if (r-l+1)>count:

count = r-l+1

s\_result = s[l:r+1]

l = l-1

r = r+1

l = i

r = i+1

while l>=0 and r<len(s) and s[l] == s[r]:

if (r-l+1)>count:

count = r-l+1

s\_result = s[l:r+1]

l = l - 1

r = r + 1

return s\_result

[7. Reverse Integer](https://leetcode.com/problems/reverse-integer/)

class Solution:

def reverse(self, x: int) -> int:

negativeInteger = False

if x<0:

negativeInteger = True

x = -x

revStr = str(x)[::-1]

x= int(revStr)

if negativeInteger:

x = -x

if x<-2\*\*31 or x>2\*\*31-1:

return 0

return x

[8. String to Integer (atoi)](https://leetcode.com/problems/string-to-integer-atoi/)

class Solution:

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

s = s.strip()

if not s:

return 0

isNegative = False

if s[0] in ['-','+']:

if s[0]=='-':

isNegative = True

s= s[1:]

num = 0

for char in s:

if not char.isdigit():

break

num = num \* 10 + int(char)

if isNegative:

num = -num

nums = max(min(num,2\*\*31-1), -2\*\*31)

return nums

[9. Palindrome Number](https://leetcode.com/problems/palindrome-number/)

class Solution:

def isPalindrome(self, x: int) -> bool:

a = str(x)

l = 0

r = len(a)-1

while l<r:

if a[l]!=a[r]:

return False

l = l + 1

r = r - 1

return True

[11. Container With Most Water](https://leetcode.com/problems/container-with-most-water/)

class Solution:

def maxArea(self, height: List[int]) -> int:

l = 0

r = len(height)-1

res = 0

while l<r:

width = r-l

res = max(res, min(height[l], height[r])\*width)

if height[l]<=height[r]:

l = l+1

else:

r = r-1

return res

[12. Integer to Roman](https://leetcode.com/problems/integer-to-roman/)

class Solution:

def intToRoman(self, num: int) -> str:

dic = {

1000:"M",

900:"CM",

500:"D",

400:"CD",

100 : "C",

90 : "XC",

50: "L",

40:"XL",

10:"X",

9:"IX",

5:"V",

4:"IV",

1:"I"

}

res = ''

for k, v in dic.items():

while num>=k:

res = res+v

num = num-k

return res

[13. Roman to Integer](https://leetcode.com/problems/roman-to-integer/)

class Solution:

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

dic = {

"I":1,

"V":5,

"X":10,

"L":50,

"C":100,

"D":500,

"M":1000

}

preValue = 0

total = 0

for ch in s[::-1]:

curValue = dic[ch]

if curValue<preValue:

total-=curValue

else:

total +=curValue

preValue = curValue

return total

[14. Longest Common Prefix](https://leetcode.com/problems/longest-common-prefix/)

class Solution:

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

strs = sorted(strs)

first = strs[0]

last = strs[-1]

ans = ''

for i in range(min(len(first), len(last))):

if first[i]!=last[i]:

return ans

ans += first[i]

return ans

[15. 3Sum](https://leetcode.com/problems/3sum/)

class Solution:

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

nums.sort()

ans = []

for i in range(len(nums)-2):

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

continue

l = i+1

r = len(nums)-1

while l<r:

total = nums[i]+nums[l]+nums[r]

if total < 0:

l = l+1

elif total>0:

r = r-1

else:

ans.append([nums[i], nums[l], nums[r]])

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

l = l+1

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

r = r-1

l = l+1

r = r-1

return ans

[16. 3Sum Closest](https://leetcode.com/problems/3sum-closest/)

class Solution:

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

nums.sort()

closest = float('inf')

for i in range(len(nums)-2):

l = i+1

r = len(nums)-1

while l<r:

total = nums[i]+nums[l]+nums[r]

if abs(total-target)<abs(closest-target):

closest = total

if total<target:

l = l+1

else:

r = r-1

return closest

[17. Letter Combinations of a Phone Number](https://leetcode.com/problems/letter-combinations-of-a-phone-number/)

class Solution:

def letterCombinations(self, digits: str) -> List[str]:

phone = {

"2":"abc",

"3":"def",

"4":"ghi",

"5":"jkl",

"6":"mno",

"7":"pqrs",

"8":"tuv",

"9":"wxyz"

}

ans = []

def backtrack(i,curStr):

if len(curStr)==len(digits):

ans.append(curStr)

return

for c in phone[digits[i]]:

backtrack(i+1, curStr+c)

if digits:

backtrack(0,"")

return ans

[18. 4Sum](https://leetcode.com/problems/4sum/)

class Solution:

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

nums.sort()

ans = []

for i in range(len(nums)-3):

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

continue

for j in range(i+1, len(nums)-2):

if j>i+1 and nums[j]==nums[j-1]:

continue

l = j+1

r = len(nums)-1

while l<r:

total = nums[i]+nums[j]+nums[l]+nums[r]

if total<target:

l = l+1

elif total>target:

r = r-1

else:

ans.append([nums[i], nums[j], nums[l], nums[r]])

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

l = l+1

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

r = r-1

l = l+1

r = r-1

return ans

[19. Remove Nth Node From End of List](https://leetcode.com/problems/remove-nth-node-from-end-of-list/)

# Definition for singly-linked list.

# class ListNode:

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

# self.val = val

# self.next = next

class Solution:

def removeNthFromEnd(self, head: Optional[ListNode], n: int) -> Optional[ListNode]:

dummy = ListNode(0)

dummy.next = head

slow = fast=dummy

for \_ in range(n+1):

fast = fast.next

while fast:

fast = fast.next

slow = slow.next

slow.next = slow.next.next

return dummy.next

[20. Valid Parentheses](https://leetcode.com/problems/valid-parentheses/)

class Solution:

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

dic = {

")":"(",

"}":"{",

"]":"["

}

stack = []

for char in s:

if char in dic.values():

stack.append(char)

elif char in dic.keys():

if not stack or stack.pop()!=dic[char]:

return False

return len(stack)==0