11. Container with most water

```
class Solution:
    def maxArea(self, height: List[int]) -> int:
        left = 0
        right = len(height) - 1
        max_value = 0

        while left < right:
            crr_water = min(height[left], height[right]) * (right - left)
            max_value = max(max_value, crr_water)

            if height[left] < height[right]:
                left += 1
            else:
                right -= 1

        return max_value
```

12. Integer to roman

```
class Solution:
    def intToRoman(self, num: int) -> str:
        values = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]
        numerals = {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 value in values:
            while num >= value:
                num -= value
                res += numerals[value]

        return res
```

13. Roman to integer

```
class Solution:
    def romanToInt(self, s: str) -> int:
        translations = {
            "I": 1,
            "V": 5,
            "X": 10,
            "L": 50,
```

```
            "C": 100,
            "D": 500,
            "M": 1000
        }
        number = 0
        s = s.replace("IV", "IIII").replace("IX", "VIIII")
        s = s.replace("XL", "XXXX").replace("XC", "LXXXX")
        s = s.replace("CD", "CCCC").replace("CM", "DCCCC")
        for char in s:
            number += translations[char]
        return number
```

14.Longest common prefix

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

        mn, mx = min(strs), max(strs)

        for i in range(len(mn)):
            if mn[i] != mx[i]: return mn[:i]

        return mn
```

15. 3sum

```
class Solution:
    def threeSum(self, nums: List[int]) -> List[List[int]]:
        n,ans = len(nums),set()
        nums.sort()
        for i in range(n):
            j,k = i+1,n-1
            while j<=k:
                s = nums[i] + nums[j] + nums[k]
                if i != j and i != k and j != k and  s== 0:
                    ans.add((nums[i],nums[j],nums[k]))
                    j+=1
                    k-=1
                elif s>0:
                    k-=1
                else:
                    j+=1
        return ans
```

16. 3sums closest

```
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        mindiff = float('inf')
        nums.sort()
        n = len(nums)
        ans = 0

        for i in range(n):
            j = i + 1
            k = n - 1

            while j < k:
                sum_val = nums[i] + nums[j] + nums[k]

                if sum_val == target:
                    return target
                else:
                    diff = abs(target - sum_val)

                    if diff < mindiff:
                        mindiff = diff
                        ans = sum_val

                if sum_val < target:
                    j += 1
                elif sum_val > target:
                    k -= 1

        return ans
```

17. Letter combination of phone number

```
class Solution:
    def letterCombinations(self, digits: str) -> List[str]:
        if not digits:
            return []

        d_letters = {
        '2': 'abc',
        '3': 'def',
        '4': 'ghi',
        '5': 'jkl',
```

```
        '6': 'mno',
        '7': 'pqrs',
        '8': 'tuv',
        '9': 'wxyz'
        }
        queue = deque(d_letters[digits[0]])


        for digit in digits[1:]:

            letters = d_letters[digit]
            current_length = len(queue)
            for _ in range(current_length):
                combination = queue.popleft()
                for letter in letters:
                    queue.append(combination + letter)

        return list(queue)
```

18. 4sum

```
class Solution:
    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        n = len(nums)
        ans = set()
        for i in range(n-3):
            for j in range(i+1, n-2):
                for k in range(j+1, n-1):
                    for l in range(k+1, n):
                        if nums[i] + nums[j] + nums[k] + nums[l] == target:
                            ans.add(tuple(sorted((nums[i], nums[j], nums[k], nums[l]))))

        res = []
        for i in ans:
            res += list(i),
        return res
```

19. Remove the nth node from end of the list

```
class Solution:
    def removeNthFromEnd(self, head: Optional[ListNode], n: int) -> Optional[ListNode]:
        temp = head
        c = 0
        while temp is not None:
```

```
            temp = temp.next
            c += 1

        if c==n:
            head = head.next
            return head

        temp = head
        for i in range(c-n-1):
            temp = temp.next

        temp.next = temp.next.next

        return head
```

20. Valid parentheses

```
class Solution:
    def isValid(self, s: str) -> bool:
        stack = list(s)
        valid_check = []

        for i in stack:
            if i == "(" or i == "[" or i == "{":
                valid_check.append(i)
            elif i == ")" and valid_check and valid_check[-1] == "(":
                valid_check.pop()
            elif i == "]" and valid_check and valid_check[-1] == "[":
                valid_check.pop()
            elif i == "}" and valid_check and valid_check[-1] == "{":
                valid_check.pop()
            else:
                return False

        return len(valid_check) == 0
```