1.Implement Queue using Stacks

```
class MyQueue:

    def __init__(self):
        # Use two stacks, one for push and one for pop.
        self.push_stack = []
        self.pop_stack = []

    def push(self, x: int) -> None:
        # Push just consist of pushing the element into push_stack: This is
O(1) and the front of queue is at the bottom of push_stack
        self.push_stack.append(x)

    def pop(self) -> int:
        # if pop_stack is empty
        if not self.pop_stack:
            if not self.push_stack:
                return None
            # we move the elements from push_stack to pop_stack - the elements
are now in reverse order -
            while self.push_stack:
                self.pop_stack.append(self.push_stack.pop())
        # then, we just remove the top element of pop_stack
        return self.pop_stack.pop()

    def peek(self) -> int:
        if not self.empty():
            #  if pop_stack is not empty
            if self.pop_stack:
                # we get the top of pop_stack
                return self.pop_stack[-1]
            else:
                # else we get the bottom of push_stack
                return self.push_stack[0]
        return None

    def empty(self) -> bool:
        return not self.push_stack and not self.pop_stack
```

2. Design HashSet

class MyHashSet:

```
def __init__(self):

    self.h = {}
```

```
def add(self, key: int) -> None:
    self.h[key] = key

def remove(self, key: int) -> None:
    if key in self.h:
        del self.h[key]

def contains(self, key: int) -> bool:

    if key in self.h:
        return True
    else:
        return False
```

33. Search in Rotated Sorted Array

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        start, end = 0, len(nums) - 1
        while start <= end:
            mid = start + (end - start) // 2
            if nums[mid] == target:
                return mid
            elif nums[mid] >= nums[start]:
                if target >= nums[start] and target < nums[mid]:
                    end = mid - 1
                else:
                    start = mid + 1
            else:
                if target <= nums[end] and target > nums[mid]:
                    start = mid + 1
                else:
                    end = mid - 1
        return -1
```

4.Reverse words in a string

```
class Solution:
    def reverseWords(self, s: str) -> str:
        return " ".join(s.split()[::-1])
```

5.longest Prefix

```
class Solution(object):
    def longestCommonPrefix(self, strs):
        """
        :type strs: List[str]
        :rtype: str
        """
        # verify not empty
        if not strs:
            return ""

        # iterate through prefix in first string
        for i in range(0,len(strs[0])):
            chars=strs[0][i]

            # iterate through all strings
            for j in range(1,len(strs)):
                if i == len(strs[j]) or strs[j][i] != chars:
                    return strs[0][:i]

        # if empty string
        return strs[0]
```

6.Reverse of words in a sentense:

```
from collections import deque
class Solution:
    def reverseWords(self, s: str) -> str:
        left, right = 0, len(s) - 1
        # remove leading spaces
        while left <= right and s[left] == ' ':
            left += 1

        # remove trailing spaces
        while left <= right and s[right] == ' ':
            right -= 1

        d, word = deque(), []
        # push word by word in front of deque
        while left <= right:
            if s[left] == ' ' and word:
                d.appendleft(''.join(word))
                word = []
            elif s[left] != ' ':
                word.append(s[left])
            left += 1
        d.appendleft(''.join(word))

        return ' '.join(d)
```

7.Heaters

https://leetcode.com/problems/heaters/

```
def findRadius(houses, heaters):
        """
        :type houses: List[int]
        :type heaters: List[int]
        :rtype: int
        """
        #verify there are heaters and houses
        #If there are no houses we set the distance to 0
        If not houses:
                return 0
        #If there are no heaters we set the distance to \infty
        If not heaters:
                return float('inf')

        #sort both lists
        houses.sort()
        heaters.sort()

        heat_index = 0
        min_dist = 0
        #Now iterate through houses and find the distance to closest heater
        for house in houses:
                #Increase the heater_index as long as the distance to the previous heater
                #(left of house) is bigger than the distance to the next heater (right of house).
                #Also stop if there are no heaters left
                while heat_index<len(heaters)-1 and \
                        (heaters[heat_index+1]-house)<=(house-heaters[heat_index]):
                        heat_index+=1
                #update the min_distance
                        min_dist = max(min_dist,abs(heaters[heat_index]-house))
        return min_dist
```

8. Intersection of Two Arrays

```
class Solution:
    def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:
        d = set(nums1)
        d2 = set(nums2)
        d3 = d2.intersection(d)
        d4 = list(d3)
        return d4
```

9.Find the Difference

https://leetcode.com/problems/find-the-difference/

```
class Solution:
    def findTheDifference(self, s: str, t: str) -> str:
        s = sorted(s)
        t = sorted(t)
        for i in range(len(s)):
            if s[i] != t[i]:
                return t[i]
        return t[-1]
```

10. Word Pattern

https://leetcode.com/problems/word-pattern/

```
class Solution:
    def wordPattern(self, pattern: str, str: str) -> bool:
        from collections import Counter
        p = Counter(pattern)
        st = Counter(str.split())
        patt = list(p)
        string = list(st)
        if len(patt) != len(string):
            return False
        for i in range(len(patt)):
            if p[patt[i]] != st[string[i]]:
                return False
        return True
```

11.Two Sum
https://leetcode.com/explore/interview/card/amazon/76/array-and-strings/508/

```
   kv = {}

        for i in range(0, len(nums)):
            if target - nums[i] in kv:
                return [kv[target - nums[i]], i]

            kv[nums[i]] = i

        raise Exception('No pairs found')
```

12.Most Common Words:
https://leetcode.com/explore/interview/card/amazon/76/array-and-strings/2973/

```
class Solution:
    def mostCommonWord(self, paragraph: str, banned: List[str]) -> str:
        banset = set(banned)
        for c in "!?',;.":
            paragraph = paragraph.replace(c, " ")
        count = collections.Counter(
            word for word in paragraph.lower().split())

        ans, best = '', 0
        for word in count:
            if count[word] > best and word not in banset:
                ans, best = word, count[word]

        return ans
```

13. Reorder Log Files

https://leetcode.com/explore/interview/card/amazon/76/array-and-strings/2974

```
class Solution:
    def reorderLogFiles(self, logs: List[str]) -> List[str]:
        def rank(s):
            s = s.split()
            return " ".join(s[1:]+[s[0]])
        let = [i for i in logs if i.split()[1].isalpha()]
        dig = [i for i in logs if i.split()[1].isnumeric()]
        return sorted(let, key=rank) + dig
```

14.Trapping Rain Water:

15.Copy List with Random Pointer
https://leetcode.com/explore/interview/card/amazon/77/linked-list/2978

```
def copyRandomList(self, head: 'Node') -> 'Node':
        if not head:
            return None

        node = head
        map = {}

        while node:
            map[hash(node)] = Node(node.val)
            node = node.next

        node = head

        while node:
            new_node = map[hash(node)]
            new_node.next = map[hash(node.next)] if node.next else None
            new_node.random = map[hash(node.random)] if node.random else None
            node = node.next

        return map[hash(head)]
```

16.Merge Two Sorted Lists
https://leetcode.com/explore/interview/card/amazon/77/linked-list/2976/

```
class Solution:
    def mergeTwoLists(self, l1: ListNode, l2: ListNode) -> ListNode:
        if l1 is None:
            return l2
        elif l2 is None:
            return l1
        elif l1.val < l2.val:
            l1.next = self.mergeTwoLists(l1.next,l2)
            return l1
        else:
            l2.next = self.mergeTwoLists(l1,l2.next)
            return l2
```

17.Reverse Nodes in k-Group
https://leetcode.com/explore/interview/card/amazon/77/linked-list/2977

```
class Solution:
    def reverseKGroup(self, head: ListNode, k: int) -> ListNode:
        currK = head
        for i in range(k):
            if currK == None:
                return head
            currK = currK.next
                    # after the first for loop we ensure that the first K nodes can be reversed

        prev = None
        curr = head
        while curr != currK:
            tmp = curr.next
            curr.next = prev
            prev = curr
            curr = tmp
        head.next = self.reverseKGroup(currK, k)
        return prev
```

18. Reverse of Linked List
https://leetcode.com/explore/interview/card/amazon/77/linked-list/2979/

```
class Solution:
    def reverseList(self, head: ListNode) -> ListNode:
        if head is None or head.next is None:
            return head
        reverse_list = self.reverseList(head.next)
        head.next.next = head
        head.next = None
        return reverse_list
```

Second 2:

```
class Solution:
    def reverseList(self, head: ListNode) -> ListNode:
        prev = None
        current = head
        while current is not None:
            next = current.next
            current.next = prev
            prev = current
            current = next

        return prev
```

19.Merge k Sorted Lists

https://leetcode.com/explore/interview/card/amazon/77/linked-list/512

```
class Solution:
    def mergeKLists(self, lists: List[ListNode]) -> ListNode:
        merge, head, pointer = [], None, None

        for l in lists:
            while l:
                heapq.heappush(merge, l.val)
                l = l.next

        while merge:
            if head == None:
                head = ListNode(heapq.heappop(merge))
                pointer = head
            else:
                pointer.next = ListNode(heapq.heappop(merge))
                pointer = pointer.next

        return head
```

20.Median of Two Sorted Arrays
https://leetcode.com/explore/interview/card/amazon/79/sorting-and-searching/2991/

```
def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:
        full = nums1 + nums2
        full.sort()
        if len(full) % 2 != 0:
            return full[len(full)//2]
        elif len(full) % 2 == 0:
            return (full[len(full)//2 - 1] + full[len(full)//2])/2
```

21. Search in Rotated Sorted Array

https://leetcode.com/explore/interview/card/amazon/79/sorting-and-searching/2992/

```
 def search(self, nums: List[int], target: int) -> int:
        start,end = 0,len(nums) -1
        while start <= end:
            mid = start + (end -start) //2
            if nums[mid] == target:
                return mid
            elif target >= nums[start] and target < nums[mid]:
                end = mid-1
                else:
                    start = mid+1
            else:
                if target > nums[mid] and target <= nums[end]:
                    start = mid+1
                else:
                    end = mid-1
        return -1
```

22.Search in Rotated Sorted Array

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        start, end = 0, len(nums) - 1
        while start <= end:
            mid = start + (end - start) // 2
            if nums[mid] == target:
```

```
            return mid
        elif nums[mid] >= nums[start]:
            if target >= nums[start] and target < nums[mid]:
                end = mid - 1
            else:
                start = mid + 1
        else:
            if target <= nums[end] and target > nums[mid]:
                start = mid + 1
            else:
                end = mid - 1
    return -1
```

23.   Two Sum II - Input array is sorted
https://leetcode.com/explore/interview/card/amazon/79/sorting-and-searching/2994

```
class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        left = 0
        right = len(numbers) -1
        while left < right:
            sum = numbers[left] + numbers[right]
            if sum == target:
                return(left+1,right+1)
            elif sum < target:
                left+=1
            else:
                right-=1

        return []
```

24.Lowest Common Ancestor of a Binary Search Tree
https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/

```
 def lowestCommonAncestor(self, root: 'TreeNode', p: 'TreeNode', q: 'TreeNode') ->
'TreeNode':
        # Value of current node or parent node.
        parent_val = root.val

        # Value of p
        p_val = p.val

        # Value of q
        q_val = q.val

        # If both p and q are greater than parent
        if p_val > parent_val and q_val > parent_val:
            return self.lowestCommonAncestor(root.right, p, q)
        # If both p and q are lesser than parent
        elif p_val < parent_val and q_val < parent_val:
            return self.lowestCommonAncestor(root.left, p, q)
        # We have found the split point, i.e. the LCA node.
        else:
            return root
```

25. Kth Smallest Element in a BST
https://leetcode.com/problems/kth-smallest-element-in-a-bst/

```
class Solution:
    def kthSmallest(self, root: TreeNode, k: int) -> int:
        stack = [ ]

        while True:

            while root:
                stack.append(root)
                root = root.left

            root = stack.pop()
            k-=1

            if k == 0:
                return root.val
            root = root.right
```

26.Kth Largest Element:
https://leetcode.com/explore/interview/card/amazon/79/sorting-and-searching/482

```
def findKthLargest(self, nums: List[int], k: int) -> int:
        p = sorted(nums,reverse = True)
        count = 1
        if len(p) == 1:
            return p[0]

        for i in range(len(p)):
            if (count == k):
                return p[count-1]
            count+=1
        return 0
```

Sol:2

```
class Solution:
    def findKthLargest(self, nums: List[int], k: int) -> int:

        return heapq.nlargest(k,nums)[-1]
```

27.Top K Frequent Elements
https://leetcode.com/explore/interview/card/amazon/79/sorting-and-searching/2995/

Solution

```
def topKFrequent(self, nums: List[int], k: int) -> List[int]:

        from collections import Counter

        num_dic = Counter(nums)

        return heapq.nlargest(k,num_dic.keys(),key=num_dic.get)
```

28. K Closest Points to Origin

https://leetcode.com/explore/interview/card/amazon/79/sorting-and-searching/2996

```
import math
import heapq


import math
import heapq
class Solution:
    def kClosest(self, points: List[List[int]], K: int) -> List[List[int]]:





        c = list()
        ans = list()

        for point in points:
            d = self.find_distance(point)
            heapq.heappush(c,(d,point))

        for _ in range(K):
            ans.append(heapq.heappop(c)[1])
            K-=1

        return ans
    def find_distance(self,point):
            return math.sqrt(point[0]**2 + point[1]**2)
```

29.Validate Binary Search Tree

https://leetcode.com/explore/interview/card/amazon/78/trees-and-graphs/507/

```
class Solution:
    def isValidBST(self, root):

        def helper(node, minValue, maxValue):

            if not node:
                return True

            if node.val <= minValue or node.val >= maxValue:
                return False

            return helper(node.left,minValue,node.val) and
helper(node.right,node.val, maxValue)

        return helper(root, float("-inf"), float("inf"))
```

30.Lowest Common Ancestor of a Binary Tree

https://leetcode.com/explore/interview/card/amazon/78/trees-and-graphs/2984/discuss/499118/Python-O(-n-)-sol.-by-DFS-recursion.-With-explanation

```
class Solution:
    def lowestCommonAncestor(self, root: 'TreeNode', p: 'TreeNode', q:
'TreeNode') -> 'TreeNode':

        if root and ( root is p or root is q ):
                  # hit
                  # root is either node p or node q
            return root

        if root is None:
                  # empty tree or empty node
            return None

        else:
                  # common ancestor of p, q exists in left sub-tree
            left_ancestor = self.lowestCommonAncestor( root.left, p ,q)

                  # common ancestor of p, q exists in right sub-tree
            right_ancestor = self.lowestCommonAncestor( root.right, p ,q)

            if left_ancestor and right_ancestor:
                        # p, q reside in two sides, one in left sub-tree, the
other in right sub-tree
                return root

            elif left_ancestor:
                        # both p, q reside in left sub-tree
                return left_ancestor

            elif right_ancestor:
```

```
                              # both p, q reside in right sub-tree
                    return right_ancestor

               else:
                              # both p, q do not exist in current binary tree
                    return None
```

31 Longest Palindromic Substring
https://leetcode.com/problems/longest-palindromic-substring/

```
     class Solution:
    def longestPalindrome(self, s: str) -> str:
        m=0
        res=''
        for i in range(len(s)):
            for j in range(i+1,len(s)+1):
                a=s[i:j]
                if(a==a[::-1] and len(s[i:j])>m):
                    res=s[i:j]
                    m=len(s[i:j])
        return res
```

32. House Robber
https://leetcode.com/problems/house-robber/

```
class Solution:
    def rob(self, nums: List[int]) -> int:
        n = len(nums)
        if n == 0:
            return 0
        if n == 1:
            return nums[0]
        if n == 2:
            return max(nums[0], nums[1])
        dp = [0]*n
        dp[0] = nums[0]
        dp[1] = max(nums[0], nums[1])

        for i in range(2, n):
            dp[i] = max(nums[i]+dp[i-2], dp[i-1])
        return dp[-1]
```

33.Symmetric Tree

https://leetcode.com/explore/interview/card/amazon/78/trees-and-graphs/507/

```
class Solution:
    def isSymmetric(self, root: TreeNode) -> bool:
        return self.isMirror(root,root)

    def isMirror(self, t1: TreeNode, t2: TreeNode ) -> bool:
        if (t1 is None and t2 is None):
            return True
        if (t1 is None or t2 is None):
            return False
        else:
            return (t1.val == t2.val) and self.isMirror(t1.left,t2.right) and
self.isMirror(t1.right,t2.left)
```

34.Number of Islands
https://leetcode.com/problems/number-of-islands/

```
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        island_num = 0
        if grid == []:
            return island_num
        x_length = len(grid)
        y_length = len(grid[0])
        def resetCurIsland(grid, i, j):
            if grid[i][j] == "0":
                return grid
            else:
              grid[i][j] = "0"

            if i + 1 < x_length:
                grid = resetCurIsland(grid, i+1, j)
            if j + 1 < y_length:
                grid = resetCurIsland(grid, i, j+1)
            if i - 1 > -1:
                grid = resetCurIsland(grid, i-1, j)
            if j - 1 > -1:
                grid = resetCurIsland(grid, i, j-1)
            return grid

        for i in range(x_length):
            for j in range(y_length):
                if grid[i][j] == '1':
                    island_num += 1
                    grid = resetCurIsland(grid, i, j)

        return island_num
```

35.Zombie Matrix:
https://leetcode.com/discuss/interview-question/411357/

```
lass Solution:
    def minHour(self, rows, columns, grid):
        if not rows or not columns:
            return 0

        q = [[i,j] for i in range(rows) for j in range(columns) if
grid[i][j]==1]
        directions = [[1,0],[-1,0],[0,1],[0,-1]]
        time = 0

        while True:
            new = []
            for [i,j] in q:
                for d in directions:
                    ni, nj = i + d[0], j + d[1]
                    if 0 <= ni < rows and 0 <= nj < columns and grid[ni][nj] ==
0:
                        grid[ni][nj] = 1
                        new.append([ni,nj])
            q = new
            if not q:
                break
            time += 1

        return time
```

36.Binary Tree Level Order Traversal

https://leetcode.com/explore/interview/card/amazon/78/trees-and-graphs/506

```
class Solution:

    def levelOrder(self, root: TreeNode) -> List[List[int]]:



        levels = [ ]



        if not root:

            return levels



        def helper(node,level):

            if len(levels) == level:

                levels.append([])
            levels[level].append(node.val)


            if node.left:
                helper(node.left,level+1)
            if node.right:
                helper(node.right,level+1)


        helper(root,0)


        return levels
```

37.Binary Tree Zigzag Level Order Traversal

https://leetcode.com/explore/interview/card/amazon/78/trees-and-graphs/2980

```
class Solution:
    def zigzagLevelOrder(self, root: TreeNode) ->
List[List[int]]:
```

```
def traversal(root, level, res):

    if root is None:

        return

    if level % 2 == 0:

        res[level].append(root.val)

    else:

        res[level].appendleft(root.val)



    traversal(root.left, level + 1, res)

    traversal(root.right, level + 1, res)




res = collections.defaultdict(collections.deque)
```

```
        traversal(root, 0, res)


        return res.values()
```

38.Binary Tree Maximum Path Sum

https://leetcode.com/explore/interview/card/amazon/78/trees-and-graphs/2981

```
class Solution:
    def maxPathSum(self, root: TreeNode) -> int:


        max_sum = float('-inf')
        def gain(node):
            nonlocal max_sum
            if not node:
                return 0
            cur = node.val
            left = gain(node.left)
            right = gain(node.right)
            cur_gain = cur + left + right
            max_sum = max(max_sum, cur_gain)
            return max(0, cur + max(left, right))


        gain(root)
        return max_sum
```

39.Diameter of Binary Tree

https://leetcode.com/explore/interview/card/amazon/78/trees-and-graphs/2985/

```
class Solution:
    def diameterOfBinaryTree(self, root: TreeNode) -> int:
        self.ans = 1


        def helper(node):
            if not node:
                return 0
            L = helper(node.left)
            R = helper(node.right)
            self.ans = max(self.ans, L+R+1)
            return max(L,R) + 1


        helper(root)


        return self.ans -1
```

40. Distant Barcodes

https://leetcode.com/problems/distant-barcodes/

```
class Solution:

    def rearrangeBarcodes(self, barcodes: List[int]) -> List[int]:

        n = len(barcodes)

        if n <= 2:

            return barcodes

        count = collections.Counter(barcodes)

        sort_k = [ ]



        for k,cnt in count.most_common():

            sort_k.extend([k]*cnt)



        j = 0

        new_sorted = [0]*n



        for i in range(0,n,2):

            new_sorted[i] = sort_k[j]

            j+=1
        for i in range(1,n,2):

            new_sorted[i] = sort_k[j]

            j+=1

        return new_sorted
```

41.Merge Intervals:

https://leetcode.com/problems/merge-intervals/

```
class Solution:

    def merge(self, intervals: List[List[int]]) -> List[List[int]]:

        merged = []

  intervals.sort(key = lambda x:x[0])

        for interval in intervals:

            if not merged or merged[-1][1] < interval[0]:

                merged.append(interval)

            else:

                # If overlaps

             merged[-1][1] = max(merged[-1][1],interval[1])
        return merged
```

42.Maximum Average Subtree

https://leetcode.com/problems/maximum-average-subtree/

```
class Solution:
    res = 0
    def maximumAverageSubtree(self, root: TreeNode) -> float:
        def dfs(root):
            left_sum,ln = dfs(root.left) if root.left else (0,0)
            right_sum,rn = dfs(root.right) if root.right else (0,0)
            self.res = max(self.res,(left_sum + right_sum + root.val)/(ln+rn+1))
            return left_sum + right_sum + root.val,ln+rn+1
        dfs(root)
        return self.res
```

43. Prison Cells After N Days

https://leetcode.com/discuss/interview-question/344650/Amazon-Online-Assessment-Questions

```
class Solution:

    def prisonAfterNDays(self, cells: List[int], N: int) -> List[int]:

        seen ,count = 0,0



        record = []

        while seen == 0:

            temp = []

            temp.append(0)

            for i in range(1,7):

                if cells[i-1] == cells[i+1]:

                    val = 1



                else:

                    val = 0

                temp.append(val)

            temp.append(0)



            cells = tuple(temp)
            if cells in record:

                seen = 1

            else:

                record.append(cells)

                count+=1


        return record[(N-1)%count]
```