Most frequently asked coding questions with solutions in Python

Part-1

Table of Contents

Merge K Sorted Linked Lists

Merge k sorted linked lists and return it as one sorted list. Analyze and describe its complexity.

URL: https://leetcode.com/problems/merge-k-sorted-lists/

```
import heapq
# Definition for singly-linked list.
class ListNode(object):
    def___init__(self, x):
        self.val = x
        self.next = None

class Solution(object):
    def mergeKLists(self, lists):
        """
        :type lists: List[ListNode]
        :rtype: ListNode
        """
        if lists == [] or lists == None:
            return None
        else:
            pq = []
            for i in range(len(lists)):
                if lists[i] != None:
                    item = (lists[i].val, i, lists[i])
                    heapq.heappush(pq, item)

            dummy = ListNode(0)
            p = dummy

            while pq != []:
                heap_item = heapq.heappop(pq)
                p.next = heap_item[2]
                p = p.next
                if heap_item[2].next != None:
                    item = (heap_item[2].next.val, heap_item[1],
 heap_item[2].next)
                    heapq.heappush(pq, item)

            return dummy.next
```

Kth Largest Element in an Array

Find the kth largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

For example, Given [3,2,1,5,6,4] and k = 2, return 5.

Note: You may assume k is always valid, 1 ≤ k ≤ array's length

URL: https://leetcode.com/problems/kth-largest-element-in-an-array/

```
class Solution(object):
    def findKthLargest(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: int
        """
        if nums == []:
            return nums
        else:
            heap = []
            for i in range(0, k):
                heapq.heappush(heap, (nums[i], i))

            for j in range(k, len(nums)):
                root_element = [entries for entries in heapq.nsm
allest(1, heap)][0]
                index_root_element = heap.index(root_element)
                if nums[j] > root_element[0]:
                    heap[index_root_element] = (nums[j], j)
                    heapq.heapify(heap)

            return heapq.heappop(heap)[0]
```

1 Sum II

Given an array of integers that is already sorted in ascending order, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2. Please note that your returned answers (both index1 and index2) are not zero-based.

You may assume that each input would have exactly one solution.

Input: numbers={2, 7, 11, 15}, target=9 Output: index1=1, index2=2

URL: https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/

```
class Solution(object):
    def twoSum(self, numbers, target):
        """
        :type numbers: List[int]
        :type target: int
        :rtype: List[int]
        """
        if len(numbers) == 0:
            return [-1]
        else:
            start = 0
            end = len(numbers) - 1
            while start < end:
                curr_sum = numbers[start] + numbers[end]
                if curr_sum == target:
                    return [start+1, end+1]
                elif curr_sum < target:
                    start += 1
                elif curr_sum > target:
                    end -= 1
            return [-1]
```

2 Sum III

Design and implement a TwoSum class. It should support the following operations: add and find.

add - Add the number to an internal data structure. find - Find if there exists any pair of numbers which sum is equal to the value.

For example, add(1); add(3); add(5); find(4) -> true find(7) -> false

URL: https://leetcode.com/problems/two-sum-iii-data-structure-design/

```
import collections

class TwoSum(object):

    def___init__(self):
        """
        initialize your data structure here
        """
        self.__num_list = collections.defaultdict(int)


    def add(self, number):
        """
        Add the number to an internal data structure.
        :rtype: nothing
        """
        self.__num_list[number] += 1


    def find(self, value):
        """
        Find if there exists any pair of numbers which sum is eq
ual to the value.
        :type value: int
        :rtype: bool
        """
```

```
        if len(self.__num_list) == 0:
            return False
        else:
            for entries in self.__num_list.keys():
                target = value - entries
                if (target in self.__num_list) and (entries != t
arget or self.__num_list[target] > 1):
                    return True
            return False

if___name__== "__main__":
    # Your TwoSum object will be instantiated and called as such
:
    twoSum = TwoSum()
    twoSum.add(0)
    twoSum.add(0)
    print(twoSum.find(0))
```

Contains Duplicate

Given an array of integers, find if the array contains any duplicates. Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

URL: https://leetcode.com/problems/contains-duplicate/

```
class Solution(object):
    def containsDuplicate(self, nums):
        """
        :type nums: List[int]
        :rtype: bool
        """
        if not nums:
            return False
        elif len(nums) == 1:
            return False
        else:
            dup_dict = {}

            for entries in nums:
                if entries in dup_dict:
                    return True
                else:
                    dup_dict[entries] = 1
            return False
```

Rotate Array

Rotate an array of n elements to the right by k steps.

For example, with n = 7 and k = 3, the array [1,2,3,4,5,6,7] is rotated to [5,6,7,1,2,3,4].

Note: Try to come up as many solutions as you can, there are at least 3 different ways to solve this problem.

URL: https://leetcode.com/problems/rotate-array/

```
class Solution(object):
    def rotate(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: void Do not return anything, modify nums in-plac
e instead.
        """
        n = len(nums)
        if n < 2 or k == 0:
            pass
        else:
            if k >= n:
                k = k % n
            a = n - k
            self.reverse(nums, 0, a-1)
            self.reverse(nums, a, n-1)
            self.reverse(nums, 0, n-1)


    def reverse(self, nums, start, end):
        i = start
        j = end
        while i < j:
            nums[i], nums[j] = nums[j], nums[i]
            i += 1
            j -= 1

if__name__== "__main__":
    soln  =  Solution()
    nums = [1,2,3,4,5,6,7]
    soln.rotate(nums, 3)
    print(nums)
```

3 Sum Smaller

Given an array of n integers nums and a target, find the number of index triplets i, j, k with 0 <= i < j < k < n that satisfy the condition nums[i] + nums[j] + nums[k] < target.

For example, given nums = [-2, 0, 1, 3], and target = 2.

Return 2. Because there are two triplets which sums are less than 2:

[-2, 0, 1] [-2, 0, 3]

URL: https://leetcode.com/problems/3sum-smaller/

```
class Solution(object):
    def threeSumSmaller(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: int
        """
        if len(nums) == 0 or len(nums) == 2 or len(nums) == 1:
            return len([])
        else:
            triplet_list = []
            sorted_nums = sorted(nums)
            for i in range(0, len(nums) - 2):
                start = i + 1
                end = len(nums) - 1
                while start < end:
                    curr_sum = sorted_nums[i] + sorted_nums[star
t] + sorted_nums[end]
                    if curr_sum == target:
                        end -= 1
                    elif curr_sum < target:
                        triplet = (sorted_nums[i], sorted_nums[s
tart], sorted_nums[end])
                        triplet_list.append(triplet)
                        start += 1
                    elif curr_sum > target:
                        end -= 1
            print(triplet_list)
            #return len([list(entries) for entries in set(triple
t_list)])
            return len(triplet_list)

if__name__== "__main__":
    soln = Solution()
    print(soln.threeSumSmaller([3,1,0,-2], 4))
```

3 Sum Closest

Given an array S of n integers, find three integers in S such that the sum is closest to a given number, target. Return the sum of the three integers. You may assume that each input would have exactly one solution.

```
For example, given array S = {-1 2 1 -4}, and target = 1.

The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

URL: https://leetcode.com/problems/3sum-closest/

```
import sys
class Solution(object):
    def threeSumClosest(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: int
        """
        if len(nums) in [0,1,2]:
            return 0
        else:
            min_diff = sys.maxsize
            result = 0
            sorted_nums = sorted(nums)
            for i in range(len(nums)):
                start = i + 1
                end = len(nums) - 1
                while start < end:
                    curr_sum = sorted_nums[i] + sorted_nums[star
t] + sorted_nums[end]
                    diff = abs(curr_sum - target)
                    if diff == 0:
                        return curr_sum
                    if diff < min_diff:
                        min_diff = diff
                        result = curr_sum
                    if curr_sum <= target:
                        start += 1
                    else:
                        end -= 1
            return result

if___name__== "__main__":

    soln = Solution()
    print(soln.threeSumClosest([-1, 2, 1, -4], 1))
    print(soln.threeSumClosest([-1, 2, 1, -4], 3))
```

3 Sum

Given an array S of n integers, are there elements a, b, c in S such that a + b + c = 0? Find all unique triplets in the array which gives the sum of zero.

Note: The solution set must not contain duplicate triplets.

For example, given array S = [-1, 0, 1, 2, -1, -4],

A solution set is: [[-1, 0, 1], [-1, -1, 2]]

URL: https://leetcode.com/problems/3sum/

```
class Solution(object):
    def threeSum(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        if len(nums) == 0 or len(nums) == 2 or len(nums) == 1:
            return []
        else:
            sum_zero_list = []
            sorted_nums = sorted(nums)
            for i in range(0, len(nums) - 2):
                start = i + 1
                end = len(nums) - 1
                while start < end:
                    curr_sum = sorted_nums[i] + sorted_nums[star
t] + sorted_nums[end]
                    if curr_sum == 0:
                        zero_triplet = (sorted_nums[i], sorted_n
ums[start], sorted_nums[end])
                        sum_zero_list.append(zero_triplet)
                        start += 1
                        end -= 1
                    elif curr_sum < 0:
                        start += 1
                    elif curr_sum > 0:
                        end -= 1

            return [list(entries) for entries in set(sum_zero_li
st)]

if__name__== "__main__":
    soln = Solution()
    print(soln.threeSum([-1, 0, 1, 2, -1, -4]))
```

Two Sum

Given an array of integers, return indices of the two numbers such that they add up to a specific target.

You may assume that each input would have exactly one solution.

Example: Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9, return [0, 1].

URL: https://leetcode.com/problems/two-sum/

```
class Solution(object):
    def twoSum(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[int]
        """
        dict = {}
        for i in range(len(nums)):
            x = nums[i]
            if target-x in dict:
                return (dict[target-x], i)
            dict[x] = i
```

Plus One

Given a non-negative number represented as an array of digits, plus one to the number.

The digits are stored such that the most significant digit is at the head of the list.

URL: https://leetcode.com/problems/plus-one/

```
class Solution(object):
    def plusOne(self, digits):
        """
        :type digits: List[int]
        :rtype: List[int]
        """
        if len(digits) <= 0:
            return [0]
        else:
            carry = 1
            i = len(digits)-1
            running_sum = 0
            new_digits = []
            while i >= 0:
                running_sum = digits[i] + carry
                if running_sum >= 10:
                    carry = 1
                else:
                    carry = 0
                new_digits.append(running_sum % 10)
                i -= 1

            if carry == 1:
                new_digits.append(1)
                return new_digits[::-1]
            else:
                return new_digits[::-1]
```

Best Time to Buy and Sell Stock

Say you have an array for which the ith element is the price of a given stock on day i.

If you were only permitted to complete at most one transaction (ie, buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Example 1: Input: [7, 1, 5, 3, 6, 4] Output: 5

max. difference = 6-1 = 5 (not 7-1 = 6, as selling price needs to be larger than buying price) Example 2: Input: [7, 6, 4, 3, 1] Output: 0

In this case, no transaction is done, i.e. max profit = 0.

URL: https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

```
class Solution(object):
    def maxProfit(self, prices):
        """
        :type prices: List[int]
        :rtype: int
        """
        if len(prices) == 0:
            return 0
        else:
            max_profit = 0
            min_price = prices[0]
            for i in range(len(prices)):
                profit = prices[i] - min_price
                max_profit = max(profit, max_profit)
                min_price = min(min_price, prices[i])

            return max_profit
```

Shortest Word Distance

Given a list of words and two words word1 and word2, return the shortest distance between these two words in the list.

For example, Assume that words = ["practice", "makes", "perfect", "coding", "makes"].

Given word1 = “coding”, word2 = “practice”, return 3. Given word1 = "makes", word2 = "coding", return 1.

Note: You may assume that word1 does not equal to word2, and word1 and word2 are both in the list.

URL: https://leetcode.com/problems/shortest-word-distance/

```
import sys
class Solution(object):
    def shortestDistance(self, words, word1, word2):
        """
        :type words: List[str]
        :type word1: str
        :type word2: str
        :rtype: int
        """

        word2_positions = []
        word1_positions = []

        for i in range(len(words)):
            if word1 == words[i]:
                word1_positions.append(i)
            if word2 == words[i]:
                word2_positions.append(i)

        min_dist = sys.maxint

        for pos1  in  word1_positions:
            for pos2 in word2_positions:
                if abs(pos1 - pos2) < min_dist:
                    min_dist = abs(pos1 - pos2)

        return min_dist
```

Move Zeroes

Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements.

For example, given nums = [0, 1, 0, 3, 12], after calling your function, nums should be [1, 3, 12, 0, 0].

Note: You must do this in-place without making a copy of the array. Minimize the total number of operations.

URL:https://leetcode.com/problems/move-zeroes/

```
class Solution(object):
    def moveZeroes(self, nums):
        """
        :type nums: List[int]
        :rtype: void Do not return anything, modify nums in-plac
e instead.
        """

        i = 0
        j = 0
        while j < len(nums):
            if nums[j] == 0:
                j += 1
            else:
                nums[i] = nums[j]
                i += 1
                j += 1

        while i < len(nums):
            nums[i] = 0
            i += 1
```

Contains Duplicate II

Given an array of integers and an integer k, find out whether there are two distinct indices i and j in the array such that nums[i] = nums[j] and the difference between i and j is at most k.

URL: https://leetcode.com/problems/contains-duplicate-ii/

```
class Solution(object):
    def containsNearbyDuplicate(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: bool
        """
        if not nums:
            return False
        elif len(nums) == 1:
            return False
        elif len(nums) == 2:
            if nums[0] != nums[1]:
                return False
            else:
                if nums[0] == nums[1] and k >= 1:
                    return True
                else:
                    return False
        else:
            index_dict = {}
            for i  in  range(len(nums)):
                if nums[i] in index_dict:
                    prev_index = index_dict[nums[i]]
                    if i - prev_index <= k:
                        return True
                index_dict[nums[i]] = i
            return False
```

Majority Element

Given an array of size n, find the majority element. The majority element is the element that appears more than ⌊ n/2 ⌋ times.

You may assume that the array is non-empty and the majority element always exist in the array.

URL: https://leetcode.com/problems/majority-element/

```
class Solution(object):
    def majorityElement(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        candidate = self.get_candidate(nums)
        candidate_count = 0
        if candidate != None:
            for entries in nums:
                if entries == candidate:
                    candidate_count += 1
            if candidate_count >= len(nums)//2:
                return candidate
            else:
                return None
        else:
            return None


    def get_candidate(self, nums):
        count = 0
        candidate  =  None
        for entries in nums:
            if count == 0:
                candidate = entries
                count = 1
            else:
                if candidate == entries:
                    count += 1
                else:
                    count -= 1
        if count > 0:
            return candidate
        else:
            return None
```

Remove Duplicates from Sorted Array

Given a sorted array, remove the duplicates in place such that each element appear only once and return the new length.

Do not allocate extra space for another array, you must do this in place with constant memory.

For example, Given input array nums = [1,1,2],

Your function should return length = 2, with the first two elements of nums being 1 and 2 respectively. It doesn't matter what you leave beyond the new length.

URL: https://leetcode.com/problems/remove-duplicates-from-sorted-array/

```
class Solution(object):
    def removeDuplicates(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        if len(nums) < 2:
            return len(nums)
        else:
            j = 0
            i = 1
            while i < len(nums):
                if nums[j] == nums[i]:
                    i += 1
                else:
                    j += 1
                    nums[j] = nums[i]
                    i += 1
            return j+1
```

Nested List Weight Sum

Given a nested list of integers, return the sum of all integers in the list weighted by their depth.

Each element is either an integer, or a list -- whose elements may also be integers or other lists.

Example 1: Given the list [[1,1],2,[1,1]], return 10. (four 1's at depth 2, one 2 at depth 1)

Example 2: Given the list [1,[4,[6]]], return 27. (one 1 at depth 1, one 4 at depth 2, and one 6 at depth 3; 1 + 4*2* + 6*3* = 27)

URL: https://leetcode.com/problems/nested-list-weight-sum/

```
# """
# This is the interface that allows for creating nested lists.
# You should not implement it, or speculate about its implementa
tion
# """
#class NestedInteger(object):
#    def isInteger(self):
#        """
#        @return True if this NestedInteger holds a single integ
er, rather than a nested list.
#        :rtype bool
#        """
#
#    def getInteger(self):
#        """
#        @return the single integer that this NestedInteger hold
s, if it holds a single integer
#        Return None if this NestedInteger holds a nested list
#        :rtype int
#        """
#
#    def getList(self):
```

```
#         """
#         @return the nested list that this NestedInteger holds,
if it holds a nested list
#         Return None if this NestedInteger holds a single intege
r
#         :rtype List[NestedInteger]
#         """

class Solution(object):
    def depthSum(self, nestedList):
        """
        :type nestedList: List[NestedInteger]
        :rtype: int
        """
        return self.depthSum_helper(nestedList, 1)

    def depthSum_helper(self, nested_list, depth):
        if len(nested_list) == 0 or nested_list == None:
            return 0
        else:
            sum = 0
            for entries in nested_list:
                if entries.isInteger():
                    sum += entries.getInteger()*depth
                else:
                    sum += self.depthSum_helper(entries.getList(
), depth + 1)

            return sum
```

Nested List Weighted Sum II

Given a nested list of integers, return the sum of all integers in the list weighted by their depth.

Each element is either an integer, or a list -- whose elements may also be integers or other lists.

Different from the previous question where weight is increasing from root to leaf, now the weight is defined from bottom up. i.e., the leaf level integers have weight 1, and the root level integers have the largest weight.

Example 1: Given the list [[1,1],2,[1,1]], return 8. (four 1's at depth 1, one 2 at depth 2)

Example 2: Given the list [1,[4,[6]]], return 17. (one 1 at depth 3, one 4 at depth 2, and one 6 at depth 1; 1*3* + *4*2 + 6*1 = 17)

URL: https://leetcode.com/problems/nested-list-weight-sum-ii/

Remove Element

Given an array and a value, remove all instances of that value in place and return the new length.

Do not allocate extra space for another array, you must do this in place with constant memory.

The order of elements can be changed. It doesn't matter what you leave beyond the new length.

Example: Given input array nums = [3,2,2,3], val = 3

Your function should return length = 2, with the first two elements of nums being 2.

URL: https://leetcode.com/problems/remove-element/

```
class Solution(object):
    def removeElement(self, nums, val):
        """
        :type nums: List[int]
        :type val: int
        :rtype: int
        """
        if val == []:
            return 0
        else:
            i = 0
            j = 0
            while j < len(nums):
                if nums[j] == val:
                    j += 1
                else:
                    nums[i] = nums[j]
                    i += 1
                    j += 1

            return len(nums[0:i])
```

Intersection of Two Arrays II

Given two arrays, write a function to compute their intersection.

Example: Given nums1 = [1, 2, 2, 1], nums2 = [2, 2], return [2, 2].

Note: Each element in the result should appear as many times as it shows in both arrays. The result can be in any order. Follow up: What if the given array is already sorted? How would you optimize your algorithm? What if nums1's size is small compared to nums2's size? Which algorithm is better? What if elements of nums2 are stored on disk, and the memory is limited such that you cannot load all elements into the memory at once?

URL: https://leetcode.com/problems/intersection-of-two-arrays-ii/

```
class Solution(object):
    def intersect(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        sorted_nums1 = sorted(nums1)
        sorted_nums2 = sorted(nums2)

        i = 0
        j = 0
        intersect_list = []

        while i < len(sorted_nums1) and j < len(sorted_nums2):
            if sorted_nums1[i] < sorted_nums2[j]:
                i += 1
            elif sorted_nums2[j] < sorted_nums1[i]:
                j += 1
            else:
                intersect_list.append(sorted_nums1[i])
                i += 1
                j += 1

        return intersect_list
```

Merge Sorted Arrays

Given two sorted integer arrays nums1 and nums2, merge nums2 into nums1 as one sorted array.

Note: You may assume that nums1 has enough space (size that is greater or equal to m + n) to hold additional elements from nums2. The number of elements initialized in nums1 and nums2 are m and n respectively.

URL: https://leetcode.com/problems/merge-sorted-array/

```
class Solution(object):
    def merge(self, nums1, m, nums2, n):
        """
        :type nums1: List[int]
        :type m: int
        :type nums2: List[int]
        :type n: int
        :rtype: void Do not return anything, modify nums1 in-pla
ce instead.
        """
        last1 = m - 1
        last2 = n - 1
        last = m + n - 1

        while last1 >= 0 and last2 >= 0:
            if nums1[last1] > nums2[last2]:
                nums1[last] = nums1[last1]
                last1 -= 1
                last -= 1
            else:
                nums1[last] = nums2[last2]
                last2 -= 1
                last -= 1

        while last2 >= 0:
            nums1[last] = nums2[last2]
            last -= 1
            last2 -= 1
```

Reverse Vowels of a String

Write a function that takes a string as input and reverse only the vowels of a string.

Example 1: Given s = "hello", return "holle".

Example 2: Given s = "leetcode", return "leotcede".

Note: The vowels does not include the letter "y".

URL: https://leetcode.com/problems/reverse-vowels-of-a-string/

```
class Solution(object):
    def___init__(self):
        self.__vowels = {"a" : True, "e" : True, "i" : True, "o"
 : True, "u" : True, "A" : True, "E" : True, "I" : True, "O" : T
rue, "U" :             True,}

    def reverseVowels(self, s):
        """
        :type s: str
        :rtype: str
        """
        if s == None or s == "" or len(s) == 1:
            return s
        else:
            i=0
            j = len(s) - 1
            s = list(s)

            while i < j:
                if s[i] not in self.__vowels:
                    i += 1
                    continue
                if s[j] not in self.__vowels:
                    j -= 1
                    continue
                s[i], s[j] = s[j], s[i]
                i += 1
                j -= 1
            return "".join(s)
```

Intersection of Two Arrays

Given two arrays, write a function to compute their intersection.

Example: Given nums1 = [1, 2, 2, 1], nums2 = [2, 2], return [2].

Note: Each element in the result must be unique. The result can be in any order.

URL: https://leetcode.com/problems/intersection-of-two-arrays/

```
class Solution(object):
    def intersection(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        nums1 = sorted(nums1)
        nums2 = sorted(nums2)
        intersection = {}
        i = 0
        j = 0
        while i < len(nums1) and j < len(nums2):
            if nums1[i] < nums2[j]:
                i += 1
            elif nums2[j] < nums1[i]:
                j += 1
            else:
                intersection[nums1[i]] = nums1[i]
                i += 1
                j += 1

        return intersection.keys()
```

Container With Most Water

Givennnon-negative integersa1,a2, ...,an, where each represents a point at coordinate (i,ai).nvertical lines are drawn such that the two endpoints of lineiis at (i,ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container and n is at least 2.

URL: https://leetcode.com/problems/container-with-most-water/

```
class Solution(object):
    def maxArea(self, height):
        """
        :type height: List[int]
        :rtype: int
        """
        max_area = 0
        i = 0
        j = len(height) - 1
        while i<j:
            max_area = max(max_area, min(height[i], height[j])*(
j-i))
            if height[i] < height[j]:
                i += 1
            else:
                j -= 1

        return max_area
```

Given an array ofnintegers wheren> 1, `nums` , return an array `output` such that `output[i]` is equal to the product of all the elements of `nums` except `nums[i]` .

Solve it**without division**and in O(n).

For example, given `[1,2,3,4]` , return `[24,12,8,6]` .

Follow up:
Could you solve it with constant space complexity? (Note: The output array**does not**count as extra space for the purpose of space complexity analysis.)

URL: https://leetcode.com/problems/product-of-array-except-self/

```
class Solution(object):
    def productExceptSelf(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        before = [1]*len(nums)
        after = [1]*len(nums)
        product = [0]*len(nums)

        for i in range(1, len(nums)):
            before[i] = before[i-1]*nums[i-1]

        for i in range(len(nums)-2, -1, -1):
            after[i] = after[i+1]*nums[i+1]

        for i in range(0, len(nums)):
            product[i] = before[i]*after[i]

        return product
```

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

For example,
Given `[0,1,0,2,1,0,1,3,2,1,2,1]`, return `6`.

The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.**Thanks Marcos** for contributing this image!

URL: https://leetcode.com/problems/trapping-rain-water/

```
class  Solution(object):
    def trap(self, height):
        """
        :type height: List[int]
        :rtype: int
        """
        maxseenright = 0
        maxseenright_arr = [0]*len(height)
        maxseenleft = 0
        rainwater = 0

        for i in range(len(height) - 1, -1, -1):
            if height[i] > maxseenright:
                maxseenright_arr[i] = height[i]
                maxseenright = height[i]
            else:
                maxseenright_arr[i] = maxseenright


        for i in range(0, len(height)):
            rainwater = rainwater + max(min(maxseenright_arr[i],
 maxseenleft) - height[i],0)
            if height[i] > maxseenleft:
                maxseenleft = height[i]

        return rainwater
```

Find the contiguous subarray within an array (containing at least one number) which has the largest sum.

For example, given the array `[-2,1,-3,4,-1,2,1,-5,4]`,
the contiguous subarray `[4,-1,2,1]` has the largest sum = `6`.

URL: https://leetcode.com/problems/maximum-subarray/

```
import sys
class Solution(object):
    def maxSubArray(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        if nums == []:
            return 0
        elif len(nums) == 1:
            return nums[0]
        elif len(nums) == 2:
            return max(nums[0], nums[1], nums[0]+nums[1])
        else:
            all_neg = True
            for entries in nums:
                if entries >= 0:
                    all_neg = False
            if all_neg == False:
                curr_sum = 0
                max_sum = - sys.maxsize - 1
                for i in range(len(nums)):
                    curr_sum += nums[i]
                    if curr_sum < 0:
                        curr_sum = 0
                    if curr_sum > max_sum:
                        max_sum = curr_sum
                return max_sum
            else:
                return max(nums)
```

Say you have an array for which theithelement is the price of a given stock on dayi.

Design an algorithm to find the maximum profit. You may complete as many transactions as you like (ie, buy one and sell one share of the stock multiple times). However, you may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

URL: https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/

```
class Solution(object):
    def maxProfit(self, prices):
        """
        :type prices: List[int]
        :rtype: int
        """
        if prices == []:
            return 0
        else:
            profit = 0
            for i in range(1, len(prices)):
                curr_profit = prices[i] - prices[i-1]
                if curr_profit > 0:
                    profit += curr_profit
            return profit
```

Suppose a sorted array is rotated at some pivot unknown to you beforehand.

(i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).

Find the minimum element.

You may assume no duplicate exists in the array.

URL: https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/

```
class Solution(object):
    def findMin(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        start = 0
        end = len(nums) - 1

        while start < end:
            mid = start + (end - start)// 2
            if nums[mid] >= nums[end]:
                start = mid + 1
            else:
                end = mid
        return nums[start]
```

Given num Rows, generate the firstnum Rows of Pascal's triangle.

For example, givennumRows= 5,
Return

```
[
      [1],
     [1,1],
    [1,2,1],
   [1,3,3,1],
  [1,4,6,4,1]
]
```

URL: https://leetcode.com/problems/pascals-triangle/

```
class Solution(object):
    def generate(self, numRows):
        """
        :type numRows: int
        :rtype: List[List[int]]
        """
        if numRows <= 0:
            return []

        result = []
        pre = []

        pre.append(1)
        result.append(pre)

        for i in range(0, numRows-1):
            curr = []
            curr.append(1)
            for j in range(0, len(pre)-1):
                curr.append(pre[j]+pre[j+1])
            curr.append(1)
            result.append(curr)
            pre = curr

        return result
```

Given an index k, return the kth row of the Pascal's triangle.

For example, givenk= 3,
Return `[1,3,3,1]` .

Note:
Could you optimize your algorithm to use only O(k) extra space?

URL: https://leetcode.com/problems/pascals-triangle-ii/

```
class Solution(object):
    def getRow(self, rowIndex):
        """
        :type rowIndex: int
        :rtype: List[int]
        """
        if rowIndex < 0:
            return []
        elif rowIndex == 0:
            return [1]
        else:
            pre = []
            pre.append(1)
            for i in range(1, rowIndex+1):
                curr = []
                curr.append(1)
                for j in range(0, len(pre) - 1):
                    curr.append(pre[j]+pre[j+1])
                curr.append(1)
                pre = curr

            return pre
```

Given a sorted integer array without duplicates, return the summary of its ranges.

For example, given `[0,1,2,4,5,7]` , return `["0->2","4->5","7"].`

URL: https://leetcode.com/problems/summary-ranges/

```
class Solution(object):
    def summaryRanges(self, nums):
        """
        :type nums: List[int]
        :rtype: List[str]
        """
        if nums == []:
            return nums
        elif len(nums) == 1:
            return [str(nums[0])]
        else:
            start = nums[0]
            end = nums[0]
            res = []
            for i in range(1, len(nums)):
                if nums[i] - nums[i-1] == 1:
                    end = nums[i]
                else:
                    res.append(self.to_str(start, end))
                    start = end = nums[i]

            res.append(self.to_str(start, end))

            return res

    def to_str(self, start, end):
        if start == end:
            return str(start)
        else:
            return str(start)+"->"+str(end)
```

Given an array containingndistinct numbers taken from `0, 1, 2, ..., n` , find the one that is missing from the array.

For example,
Givennums= `[0, 1, 3]` return `2` .

Note:
Your algorithm should run in linear runtime complexity. Could you implement it using only constant extra space complexity?

URL: https://leetcode.com/problems/missing-number/

```
class Solution(object):
    def missingNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        if not nums:
            return None
        else:
            xor_prod = 0
            xor_prod_index = 0

            for i in range(len(nums)+1):
                xor_prod_index ^= i

            for i in range(len(nums)):
                xor_prod ^= nums[i]

            return xor_prod ^ xor_prod_index
```

Valid Anagram

Given two strings s and t, write a function to determine if t is an anagram of s.

For example, s = "anagram", t = "nagaram", return true. s = "rat", t = "car", return false.

Note: You may assume the string contains only lowercase alphabets.

URL: https://leetcode.com/problems/valid-anagram/

```
class Solution(object):
    def isAnagram(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        if len(s) != len(t):
            return False
        elif sorted(s) == sorted(t):
            return True
        else:
            return False
```

Valid Palindrome

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

For example, "A man, a plan, a canal: Panama" is a palindrome. "race a car" is not a palindrome.

Note: Have you consider that the string might be empty? This is a good question to ask during an interview.

For the purpose of this problem, we define empty string as valid palindrome.

URL: https://leetcode.com/problems/valid-palindrome/

```
import  re
class Solution:
    # @param {string} s
    # @return {boolean}
    def isPalindrome(self, s):
        if len(s) == 0:
            return True
        else:
            start = 0
            s = s.lower()
            newS = re.sub(r"[^a-zA-Z0-9]","",s)
            end = len(newS)-1
            while start < end:
                if newS[start] == newS[end]:
                    start = start + 1
                    end = end - 1
                else:
                    return False
            return True
```

Word Pattern

Given a pattern and a string str, find if str follows the same pattern.

Here follow means a full match, such that there is a bijection between a letter in pattern and a non-empty word in str.

Examples: pattern = "abba", str = "dog cat cat dog" should return true. pattern = "abba", str = "dog cat cat fish" should return false. pattern = "aaaa", str = "dog cat cat dog" should return false. pattern = "abba", str = "dog dog dog dog" should return false. Notes: You may assume pattern contains only lowercase letters, and str contains lowercase letters separated by a single space.

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

```
class Solution(object):
    def wordPattern(self, pattern, str):
        """
        :type pattern: str
        :type str: str
        :rtype: bool
        """
        if pattern == None or str == None:
            return False
        else:
            len_str = len(str.split(" "))
            len_pattern  =  len(pattern)
            if len_str != len_pattern:
                return  False
            str = str.split(" ")
            lookup = {}
            for i in range(0, len(pattern)):
                s = str[i]
                p = pattern[i]
                if p in lookup:
                    if lookup[p] != s:
                        return False
                else:
                    if s in lookup.values():
                        return False
                    lookup[p] = s
            return True

if__name__== "__main__":

    pattern = "abba"
    str = "dog cat cat dog"
    soln = Solution()
    print(soln.wordPattern(pattern, str))
```

Valid Parentheses

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

The brackets must close in the correct order, "()" and "()[]{}" are all valid but "(]" and "([)]" are not.

URL: https://leetcode.com/problems/valid-parentheses/

```
class Solution:
    # @param {string} s
    # @return {boolean}
    def isValid(self, s):
        if s == []:
            return False
        else:
            stack = []
            balanced = True
            index = 0
            while index < len(s) and balanced:
                symbol = s[index]
                if symbol in "({[":
                    stack.append(symbol)
                else:
                    if stack == []:
                        balanced = False
                    else:
                        top = stack.pop()
                        if not self.matches(top,symbol):
                            balanced = False
                index = index + 1

            if balanced and stack == []:
                return True
            else:
                return False

    def matches(self,open,close):
        openings = "({["
        closings = ")}]"

        return openings.index(open) == closings.index(close)
```

Isomorphic Strings

Given two strings s and t, determine if they are isomorphic.

Two strings are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character but a character may map to itself.

For example, Given "egg", "add", return true.

Given "foo", "bar", return false.

Given "paper", "title", return true.

Note: You may assume both s and t have the same length.

URL: https://leetcode.com/problems/isomorphic-strings/

```
class Solution(object):
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        if s == None or t == None:
            return False
        elif s == "" and t == "":
            return True
        else:
            if len(s) != len(t):
                return False

            lookup = {}
            for i in range(0, len(s)):
                c1 = s[i]
                c2 = t[i]

                if c1 in lookup:
                    if lookup[c1] != c2:
                        return False
                else:
                    if c2 in lookup.values():
                        return False
                    lookup[c1] = c2

            return True
```

Reverse String

Write a function that takes a string as input and returns the string reversed.

Example: Given s = "hello", return "olleh".

URL: https://leetcode.com/problems/reverse-string/

```
class Solution(object):
    def reverseString(self, s):
        """
        :type s: str
        :rtype: str
        """

        current_str = [char for char in s]

        i = 0
        j = len(s) - 1

        while i < j:
            temp = current_str[i]
            current_str[i] = current_str[j]
            current_str[j] = temp
            j -= 1
            i += 1

        return "".join(current_str)
```

Sum of Two Integers

Calculate the sum of two integers a and b, but you are not allowed to use the operator + and -.

Example: Given a = 1 and b = 2, return 3.

URL: https://leetcode.com/problems/sum-of-two-integers/

```
class  Solution(object):
    def getSum(self, a, b):
        """
        :type a: int
        :type b: int
        :rtype: int
        """
        if b == 0:
            return a
        sum = a ^ b
        carry = (a & b) << 1
        return self.getSum(sum, carry)
```

Single Number

Given an array of integers, every element appears twice except for one. Find that single one.

Note: Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

URL: https://leetcode.com/problems/single-number/

```
class Solution(object):
    def singleNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        if len(nums) == 0:
            return None
        elif len(nums) == 1:
            return nums[0]
        else:
            xor_prod = 0
            for entries in nums:
                xor_prod ^= entries

            return xor_prod
```

Reverse Integer

Reverse digits of an integer.

Example1: x = 123, return 321 Example2: x = -123, return -321

click to show spoilers.

Have you thought about this? Here are some good questions to ask before coding. Bonus points for you if you have already thought through this!

If the integer's last digit is 0, what should the output be? ie, cases such as 10, 100.

Did you notice that the reversed integer might overflow? Assume the input is a 32-bit integer, then the reverse of 1000000003 overflows. How should you handle such cases?

For the purpose of this problem, assume that your function returns 0 when the reversed integer overflows.

URL: https://leetcode.com/problems/reverse-integer/

```
import sys
class Solution(object):
    def reverse(self, x):
        """
        :type x: int
        :rtype: int
        """
        if x < 0:
            return -self.reverse(-x)

        result = 0
        while x:
            result = result * 10 + x % 10
            x /= 10
        return result if result <= 0x7fffffff else 0
```

Palindrome Number

Determine whether an integer is a palindrome. Do this without extra space.

click to show spoilers.

Some hints: Could negative integers be palindromes? (ie, -1)

If you are thinking of converting the integer to string, note the restriction of using extra space.

You could also try reversing an integer. However, if you have solved the problem "Reverse Integer", you know that the reversed integer might overflow. How would you handle such case?

There is a more generic way of solving this problem.

URL: https://leetcode.com/problems/palindrome-number/

```
class Solution(object):
    def isPalindrome(self, x):
        """
        :type x: int
        :rtype: bool
        """
        if x < 0:
            return False

        rev = 0
        copy = x
        while copy != 0:
            rev = rev*10 + copy%10
            copy = copy/10
        if rev == x:
            return True
        else:
            return False
```

Pow(x,n)

Implement pow(x, n).

```
class Solution(object):
    def myPow(self, x, n):
        """
        :type x: float
        :type n: int
        :rtype: float
        """
        if n < 0:
            return 1/self.power(x, -n)
        else:
            return self.power(x, n)

    def power(self, x, n):
        if n == 0:
            return 1

        v = self.power(x, n//2)

        if n % 2 == 0:
            return v * v
        else:
            return v * v * x
```

Subsets

Given a set of distinct integers, nums, return all possible subsets.

Note: The solution set must not contain duplicate subsets.

For example, If nums = [1,2,3], a solution is:

[[3], [1], [2], [1,2,3], [1,3], [2,3], [1,2], []]

URL: https://leetcode.com/problems/subsets/

Solution1:

```
class Solution(object):
    def subsets(self, S):
        def dfs(depth, start, valuelist):
            res.append(valuelist)
            if depth == len(S): return
            for i in range(start, len(S)):
                dfs(depth+1, i+1, valuelist+[S[i]])
        S.sort()
        res = []
        dfs(0, 0, [])
        return res
```

Solution2:

```
class Solution(object):
    def subsets(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        n = 1 << len(nums)
        result = []
        for i in range(0, n):
            subset = self.convert(i, nums)
            result.append(subset)

        return result

    def convert(self, i, nums):
        k = i
        index = 0
        res = []
        while k > 0:
            if (k & 1) == 1:
                res.append(nums[index])

            k >>= 1
            index += 1
        return res
```

Subsets II

Given a collection of integers that might contain duplicates, nums, return all possible subsets.

Note: The solution set must not contain duplicate subsets.

For example, If nums = [1,2,2], a solution is:

[[2], [1], [1,2,2], [2,2], [1,2], []]

URL: https://leetcode.com/problems/subsets-ii/

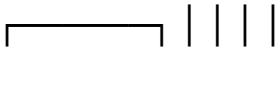
```
class Solution(object):
    def subsetsWithDup(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        n = 1 << len(nums)
        result = []
        for i in range(0, n):
            subset = self.convert(i, nums)
            result.append(tuple(sorted(subset)))

        result = set(result)
        return [list(entries) for entries in result]

    def convert(self, i, nums):
        k = i
        index = 0
        res = []
        while k > 0:
            if (k & 1) == 1:
                res.append(nums[index])

            k >>= 1
            index += 1
        return res
```

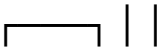
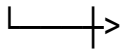
Self Crossing

You are given an array x of n positive numbers. You start at point (0,0) and moves x[0] metres to the north, then x[1] metres to the west, x[2] metres to the south, x[3] metres to the east and so on. In other words, after each move your direction changes counter-clockwise.

Write a one-pass algorithm with O(1) extra space to determine, if your path crosses itself, or not.

Example 1: Given x = [2, 1, 1, 2], 

Return true (self crossing) Example 2: Given x = [1, 2, 3, 4], 

Return false (not self crossing) Example 3: Given x = [1, 1, 1, 1], 


Return true (self crossing)

URL: https://leetcode.com/problems/self-crossing/

```
class Solution(object):
    def isSelfCrossing(self, x):
        """
        :type x: List[int]
        :rtype: bool
        """
        if x == None or len(x) <= 3:
            return False
        else:
            for i in range(3, len(x)):
                if (x[i-3] >= x[i-1]) and (x[i-2] <= x[i]):
                    return True
                if (i >= 4) and (x[i-4] + x[i] >= x[i-2]) and (x
[i-3] == x[i-1]):
                    return True
                if (i>=5) and (x[i-5] <= x[i-3]) and (x[i-4] <=
x[i-2]) and (x[i-1] <= x[i-3]) and (x[i-1] >= x[i-3] - x[i-5]) a
nd (x[i] >= x[i-2] - x[i-4]) and (x[i] <= x[i-2]):
                    return True

            return False
```

Paint Fence

There is a fence with n posts, each post can be painted with one of the k colors.

You have to paint all the posts such that no more than two adjacent fence posts have the same color.

Return the total number of ways you can paint the fence.

Note: n and k are non-negative integers.

URL: https://leetcode.com/problems/paint-fence/

```
class Solution(object):
    def numWays(self, n, k):
        """
        :type n: int
        :type k: int
        :rtype: int
        """
        dp = [0, k, k*k, 0]
        if n <= 2:
            return dp[n]
        for i in range(2, n):
            dp[3] = (k-1)*(dp[1] + dp[2])
            dp[1] = dp[2]
            dp[2] = dp[3]

        return dp[3]
```

Bulb Switcher

There are n bulbs that are initially off. You first turn on all the bulbs. Then, you turn off every second bulb. On the third round, you toggle every third bulb (turning on if it's off or turning off if it's on). For the ith round, you toggle every i bulb. For the nth round, you only toggle the last bulb. Find how many bulbs are on after n rounds.

Example:

Given n = 3.

At first, the three bulbs are [off, off, off]. After first round, the three bulbs are [on, on, on]. After second round, the three bulbs are [on, off, on]. After third round, the three bulbs are [on, off, off].

So you should return 1, because there is only one bulb is on.

URL: https://leetcode.com/problems/bulb-switcher/

```
import math
class Solution(object):
    def bulbSwitch(self, n):
        """
        :type n: int
        :rtype: int
        """
        return int(math.sqrt(n))
```

Nim Game

You are playing the following Nim Game with your friend: There is a heap of stones on the table, each time one of you take turns to remove 1 to 3 stones. The one who removes the last stone will be the winner. You will take the first turn to remove the stones.

Both of you are very clever and have optimal strategies for the game. Write a function to determine whether you can win the game given the number of stones in the heap.

For example, if there are 4 stones in the heap, then you will never win the game: no matter 1, 2, or 3 stones you remove, the last stone will always be removed by your friend.

Hint:

If there are 5 stones in the heap, could you figure out a way to remove the stones such that you will always be the winner?

URL: https://leetcode.com/problems/nim-game/

```
class  Solution(object):
    def canWinNim(self, n):
        """
        :type n: int
        :rtype: bool
        """
        return n%4 != 0
```

Rotate Image

You are given an n x n 2D matrix representing an image.

Rotate the image by 90 degrees (clockwise).

Follow up: Could you do this in-place?

URL: https://leetcode.com/problems/rotate-image/

```
class Solution(object):
    def rotate(self, matrix):
        """
        :type matrix: List[List[int]]
        :rtype: void Do not return anything, modify matrix in-pl
ace instead.
        """
        if matrix == None or matrix == []:
            pass
        else:
            n = len(matrix)
            for layer in range(0, n//2):
                first = layer
                last = n - 1 - layer
                for i in range(first, last):
                    offset = i - first
                    top = matrix[first][i]
                    matrix[first][i] = matrix[last - offset][fir
st]
                    matrix[last - offset][first] = matrix[last][
last - offset]
                    matrix[last][last - offset] = matrix[i][last
]
                    matrix[i][last] = top
```

Set Matrix Zeroes

Given a m x n matrix, if an element is 0, set its entire row and column to 0. Do it in place.

click to show follow up.

Follow up: Did you use extra space? A straight forward solution using O(mn) space is probably a bad idea. A simple improvement uses O(m + n) space, but still not the best solution. Could you devise a constant space solution?

URL: https://leetcode.com/problems/set-matrix-zeroes/

```
class Solution(object):
    def setZeroes(self, matrix):
        """
        :type matrix: List[List[int]]
        :rtype: void Do not return anything, modify matrix in-pl
ace instead.
        """
        if matrix == None or len(matrix) == 0:
            pass
        elif len(matrix) == 1 and len(matrix[0]) == 1:
            pass
        else:
            rows_with_0 = [False]*len(matrix)
            cols_with_0 = [False]*len(matrix[0])
            for i in range(len(matrix)):
                for j in range(len(matrix[0])):
                    if matrix[i][j] == 0:
                        rows_with_0[i] = True
                        cols_with_0[j] = True

            for i in range(len(matrix)):
                for j in range(len(matrix[0])):
                    if rows_with_0[i] or cols_with_0[j]:
                        matrix[i][j] = 0
```

Constant Space Solution:

```
class Solution(object):
    def setZeroes(self, matrix):
        """
        :type matrix: List[List[int]]
        :rtype: void Do not return anything, modify matrix in-pl
ace instead.
        """
        first_row = False
        first_col = False

        for j in range(len(matrix[0])):
            if matrix[0][j] == 0:
                first_row = True

        for i in range(len(matrix)):
            if matrix[i][0] == 0:
                first_col = True


        for i in range(1, len(matrix)):
            for j in range(1, len(matrix[0])):
                if matrix[i][j] == 0:
                    matrix[i][0] = 0
                    matrix[0][j] = 0

        for i in range(1, len(matrix)):
            for j in range(1, len(matrix[0])):
                if matrix[i][0] == 0 or matrix[0][j] == 0:
                    matrix[i][j] = 0

        if first_col:
            for i in range(len(matrix)):
                matrix[i][0] = 0

        if first_row:
            for i in range(len(matrix[0])):
                matrix[0][i] = 0
```

Search a 2D Matrix

Write an efficient algorithm that searches for a value in an m x n matrix. This matrix has the following properties:

Integers in each row are sorted from left to right. The first integer of each row is greater than the last integer of the previous row. For example,

Consider the following matrix:

[[1, 3, 5, 7], [10, 11, 16, 20], [23, 30, 34, 50]] Given target = 3, return true.

```
class Solution(object):
    def searchMatrix(self, matrix, target):
        """
        :type matrix: List[List[int]]
        :type target: int
        :rtype: bool
        """
        if matrix == []:
            return False
        else:
            no_rows = len(matrix)
            no_cols = len(matrix[0])

            #get the first element and the last element of the m
atrix
            #compare it with the target
            if target < matrix[0][0] or target > matrix[no_rows-
1][no_cols-1]:
                return False

            r = 0
            c = no_cols - 1
            while r < no_rows and c >= 0:
                if target == matrix[r][c]:
                    return True
                elif target > matrix[r][c]:
                    r += 1
                elif target < matrix[r][c]:
                    c -= 1
            return False
```

Search a 2D Matrix II

Write an efficient algorithm that searches for a value in an m x n matrix. This matrix has the following properties:

Integers in each row are sorted in ascending from left to right. Integers in each column are sorted in ascending from top to bottom. For example,

Consider the following matrix:

[[1, 4, 7, 11, 15], [2, 5, 8, 12, 19], [3, 6, 9, 16, 22], [10, 13, 14, 17, 24], [18, 21, 23, 26, 30]] Given target = 5, return true.

Given target = 20, return false.

URL: https://leetcode.com/problems/search-a-2d-matrix-ii/

```
class Solution(object):
    def searchMatrix(self, matrix, target):
        """
        :type matrix: List[List[int]]
        :type target: int
        :rtype: bool
        """
        if matrix == []:
            return False
        else:
            no_rows = len(matrix)
            no_cols = len(matrix[0])

            if target < matrix[0][0] or target > matrix[no_rows-
1][no_cols-1]:
                return False
            else:
                r = 0
                c = no_cols-1

                while r < no_rows and c >=0:
                    if matrix[r][c] == target:
                        return True
                    elif target > matrix[r][c]:
                        r += 1
                    elif target < matrix[r][c]:
                        c -= 1
                return False
```

Spiral Matrix

Given a matrix of m x n elements (m rows, n columns), return all elements of the matrix in spiral order.

For example, Given the following matrix:

[[1, 2, 3], [4, 5, 6], [7, 8, 9]] You should return [1,2,3,6,9,8,7,4,5].

URL: https://leetcode.com/problems/spiral-matrix/

```
class Solution(object):
    def spiralOrder(self, matrix):
        """
        :type matrix: List[List[int]]
        :rtype: List[int]
        """
        if matrix == None or matrix == []:
            return matrix
        else:
            #no of rows
            m = len(matrix)
            #no of columns
            n = len(matrix[0])
            #starting row
            k = 0
            #starting column
            l = 0

            #spiral order matrix
            spiral = []

            while k < m and l < n:
                #print the first row from the remaining rows
                for i in range(l, n):
                    spiral.append(matrix[k][i])
                k += 1

                #print the last column from the remaining column
```

```
s
                for i in range(k, m):
                    spiral.append(matrix[i][n-1])
                n-= 1

                #print the last row from the remaining rows
                if k < m:
                    for i in range(n-1, l-1, -1):
                        spiral.append(matrix[m-1][i])

                    m -= 1

                #print the first column from the remaining colum
ns
                if l < n:
                    for i in range(m-1, k-1, -1):
                        spiral.append(matrix[i][l])

                    l += 1

            return spiral
```

Spiral Matrix II

Given an integer n, generate a square matrix filled with elements from 1 to n2 in spiral order.

For example, Given n = 3,

You should return the following matrix: [[1, 2, 3], [8, 9, 4], [7, 6, 5]]

URL: https://leetcode.com/problems/spiral-matrix-ii/

```
class Solution(object):
    def generateMatrix(self, n):
        """
        :type n: int
        :rtype: List[List[int]]
        """
        if n == 0:
            return []
        elif n == 1:
            return [[1]]
        else:
            #no of rows
            r = n
            #no of columns
            c = n
            #start of row
            k = 0
            #start of column
            l = 0

            #allocate a square matrix with all zeros
            matrix = [[0 for j in range(c)] for i in range(r)]
            #counter for the elements
            count = 1

            while k < r and l < c:
                #fill the first row
                for i in range(l, c):
```

```
                matrix[k][i] = count
                count += 1

            k += 1

            #fill the last column
            for i in range(k, r):
                matrix[i][c-1] = count
                count += 1

            c -= 1

            #fill the last row
            if k < r:
                for i in range(c-1, l-1, -1):
                    matrix[r-1][i] = count
                    count += 1
                r -= 1

            #fill the first column
            if l < c:
                for i in range(r-1, k-1, -1):
                    matrix[i][l] = count
                    count += 1
                l += 1

        return matrix
```

LRU Cache

Design and implement a data structure for Least Recently Used (LRU) cache. It should support the following operations: `get` and `set` .

`get(key)` - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.
`set(key, value)` - Set or insert the value if the key is not already present. When the cache reached its capacity, it should invalidate the least recently used item before inserting a new item.

```
class LRUCache(object):

    def___init__(self, capacity):
        """
        :type capacity: int
        """
        self.capacity = capacity
        self.cache = OrderedDict()


    def get(self, key):
        """
        :rtype: int
        """
        if key in self.cache:
            value = self.cache.pop(key)
            self.cache[key] = value
            return value
        else:
            return -1


    def set(self, key, value):
        """
        :type key: int
        :type value: int
        :rtype: nothing
        """
        if len(self.cache) >= self.capacity and key not in self.
cache:
            self.cache.popitem(last=False)
            self.cache[key] = value
        else:
            if key in self.cache:
                self.cache.pop(key)
                self.cache[key] = value
            else:
                self.cache[key] = value
```