**LEET CODE DAILY SOLUTIONS**

**621. Task Scheduler**

class Solution:

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

        f=Counter(tasks)

        max\_count = max(f.values())

        max\_interval\_count = sum(1 for count in f.values() if max\_count == count)

        max\_interval = (max\_count-1)\*(n+1)+max\_interval\_count

        return max(max\_interval,len(tasks))

[**452. Minimum Number of Arrows to Burst Balloons**](https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/)

class Solution:

    def findMinArrowShots(self, points: List[List[int]]) -> int:

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

        end = points[0][1]

        arrow = 1

        for baloon in points[1:]:

            if baloon[0] > end:

                arrow+=1

                end = baloon[1]

            else:

                end = min(baloon[1],end)

        return arrow

**57. Insert Interval**

class Solution:

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

        res = []

        i = 0

        while i < len(intervals) and intervals[i][1] < newInterval[0]:

            res.append(intervals[i])

            i += 1

        while i < len(intervals) and intervals[i][0] <= newInterval[1]:

            newInterval = [min(intervals[i][0],newInterval[0]),max(intervals[i][1],newInterval[1])]

            i += 1

        res.append(newInterval)

        while i < len(intervals):

                    res.append(intervals[i])

                    i += 1

        return res

525. Contiguous Array

class Solution:

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

        count = 0

        max\_length = 0

        count\_map = {0:-1}

        for i in range(len(nums)):

            if nums[i]==0:

                count -=1

            else:

                count+=1

            if count in count\_map:

                max\_length = max(max\_length,i-count\_map[count])

            else:

                count\_map[count]=i

        return max\_length

238. Product of Array Except Self

class Solution:

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

        n=len(nums)

        prefix = [1] \* n

        suffix  = [1] \* n

        for i in range(1,n):

            prefix[i] = prefix[i-1] \* nums[i-1]

        for i in range(n-2,-1,-1):

            suffix[i] = suffix[i+1] \* nums[i+1]

        nums = [prefix[i]\* suffix[i] for i in range(n)]

        return nums

**1669. Merge In Between Linked Lists**

def mergeInBetween(self, list1: ListNode, a: int, b: int, list2: ListNode) -> ListNode:

        ptr = list1

        for \_ in range(a-1):

            ptr = ptr.next

        qtr = ptr.next

        for \_ in range(b-a+1):

            qtr = qtr.next

        ptr.next = list2

        while list2.next:

            list2 = list2.next

        list2.next = qtr

        return list1

442. Find All Duplicates in an Array

nums = [2,2]

n = len(nums)

dp = [0]\*(n+1)

for i in nums:

    dp[i]+=1

print(dp)

r =[]

r =[i for i in range(n+1) if dp[i]>1]

print(r)

Second Approach

nums = [4,3,2,7,8,2,3,1]

res = []

for num in nums:

    ind = abs(num)-1

    if nums[ind] < 0:

        res.append(ind+1)

    else:

        nums[ind] = -nums[ind]

print(nums)

print(res)

713. Subarray Product Less Than K

num = [10,5,2,6]

n=len(num)

right = 0

left = 0

count= 0

product = 1

k=100

while right < n:

    product=product\*num[right]

    while product>=k:

         product= product//num[left]

         left = left + 1

    count+= 1 + (right - left)

    right = right + 1

print(count)