Q1

class Solution:

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

sortedInt = sorted(intervals, key = lambda i: i[0])

res = list()

for start, end in sortedInt:

if res and res[-1][1] >= start:

res[-1][1] = max(res[-1][1], end)

else:

res.append([start, end])

return res

Q2

class Solution:

def sortColors(self, nums: List[int]) -> None:

left = count = 0

right = len(nums)-1

while count <= right:

if nums[count] == 2:

nums[count], nums[right] = nums[right], nums[count]

right -= 1

elif nums[count] == 0:

nums[count], nums[left] = nums[left], nums[count]

left += 1

count += 1

else:

count += 1

Q3

class Solution:

def firstBadVersion(self, n: int) -> int:

if n==1:

return n if isBadVersion(n) else 0

left,right=0,n

while left<right:

mid=(right+left)//2

if isBadVersion(mid):

if isBadVersion(mid-1):

right=mid-1

else:

return mid

else:

if isBadVersion(mid+1):

return mid+1

else:

left=mid+1

Q4

class Solution:

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

if len(nums) < 2:

return 0

maxNum = max(nums)

digit = 1

base = 16

while maxNum >= digit:

buckets = [[] for \_ in range(base)]

for num in nums:

buckets[num // digit % base].append(num)

nums = []

for bucket in buckets:

nums.extend(bucket)

digit \*= base

return max(nums[i] - nums[i - 1] for i in range(1, len(nums)))

Q5

class Solution:

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

hset = set()

for n in nums:

if n in hset:

return True

hset.add(n)

return False

Q6

class Solution:

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

points.sort(key = lambda x: x[1])

end = points[0][1]

count = 1

for s,e in points[1:]:

if s > end:

count += 1

end = e

return count

Q7

class Solution:

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

dp=[0]\*(len(nums))

dp[0]=1

for i in range(1,len(nums)):

maxi=0

for j in range(0,i):

if(nums[i]>nums[j]):

maxi=max(maxi,dp[j])

dp[i]=1+maxi

return max(dp)