Q1

class Solution:

def romanToInt(self, s: str) -> int:

roman={"I":1,"V":5,"X":10,"L":50,"C":100,"D":500,"M":1000}

number=0

for i in range(len(s)-1):

if roman[s[i]] < roman[s[(i+1)]]:

number-=roman[s[i]]

else:

number+=roman[s[i]]

return number+roman[s[-1]]

Q2

class Solution:

def lengthOfLongestSubstring(self, s: str) -> int:

charSet = set()

l = 0

res = 0

for r in range(len(s)):

while s[r] in charSet:

charSet.remove(s[l])

l += 1

charSet.add(s[r])

res = max(res,r-l+1)

return res

Q3

class Solution:

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

ele=nums[0]

count=0

for i in range(len(nums)):

if count==0:

ele=nums[i]

if nums[i]!=ele:

count=count-1

else:

count=count+1

return ele

Q4

class Solution:

def groupAnagrams(self, strs: List[str]) -> List[List[str]]:

dic={}

for word in strs:

sorted\_word="".join(sorted(word))

if sorted\_word not in dic:

dic[sorted\_word]=[word]

else:

dic[sorted\_word].append(word)

return dic.values()

Q5

class Solution:

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

list1=[0]\*n

list1[0]=1

a=b=c=0

for i in range(1,n):

list1[i]=min(list1[a]\*2,list1[b]\*3,list1[c]\*5)

if list1[a]\*2==list1[i]:a+=1

if list1[b]\*3==list1[i]:b+=1

if list1[c]\*5==list1[i]:c+=1

return list1[n-1]

Q6

from heapq import \*

class Solution:

def topKFrequent(self, words: List[str], k: int) -> List[str]:

ump={}

res = []

for i in words:

ump[i] = ump.get(i , 0) + 1

maxheap = []

for key , val in ump.items():

heappush(maxheap , [-val , key])

for \_ in range(k):

val , key = heappop(maxheap)

res.append(key)

return res

Q7

class Solution:

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

i=0

j=0

d=deque()

res=[]

while j<len(nums):

while len(d)>0 and d[-1]<nums[j]:

d.pop()

d.append(nums[j])

if j-i+1<k:

j+=1

elif j-i+1==k:

res.append(d[0])

if d[0]==nums[i]:

d.popleft()

i+=1

j+=1

return res

Q8

class Solution:

def findClosestElements(self, arr: List[int], k: int, x: int) -> List[int]:

left, right = 0, len(arr) - k

while left < right:

mid = (left + right) // 2

if abs(arr[mid] - x) > abs(arr[mid + k] - x):

left = mid + 1

elif arr[mid] == arr[mid + k] and arr[mid] < x:

left = mid + 1

else:

right = mid

return arr[left: left + k]