496. Next Greater Element I https://leetcode.com/problems/next-greater-element-i/description/ (https://leetcode.com/problems/next-greater-element-i/description/)

In [ ]:
```
class Solution:
    def nextGreaterElement(self, nums1: List[int], nums2: List[int]) -> List[int]:
        nums1idx= { n:i for i,n in enumerate(nums1)}
        res=[-1]*len(nums1)
        stack=[]
        for i in range(len(nums2)):
            cur=nums2[i]
            while stack and cur > stack[-1]:
                val= stack.pop()
                idx= nums1idx[val]
                res[idx]=cur
            if cur in nums1idx:
                stack.append(cur)
        return res
```

49.anagram https://leetcode.com/problems/valid-anagram/submissions/879678564/ (https://leetcode.com/problems/valid-anagram/submissions/879678564/)

In [ ]:
```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        if len(s)!= len(t):
            return False
        cs,ct={},{}
        for i in range(len(s)):
            cs[s[i]]= 1+ cs.get(s[i],0)
            ct[t[i]]= 1+ ct.get(t[i],0)
        for c in cs:
            if cs[c]!= ct.get(c):
                return False
        return True


##  2nd method
    return sorted(s)==sorted(t)


## 3rd method
    from collections import Counter
    return Counter(s)== Counter(t)
```

two sums https://leetcode.com/problems/two-sum/submissions/ (https://leetcode.com/problems/two-sum/submissions/)

In [ ]:
```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        prevmap={}
        for i,n in enumerate(nums):
            diff = target - n
            if diff in prevmap:
                return [prevmap[diff],i]
            prevmap[n]=i
        return
```

53. maximum subarray https://leetcode.com/problems/maximum-subarray/description/ (https://leetcode.com/problems/maximum-subarray/description/)

In [ ]:
```
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        maxsub = nums[0]
        cursum=0
        for n in nums:
            if cursum<0:
                cursum=0
            cursum+=n
            maxsub=max(cursum,maxsub)
        return maxsub
```

167. Two Sum II - Input Array Is Sorted https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/description/ (https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/description/)

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

198.house robber https://leetcode.com/problems/house-robber/description/ (https://leetcode.com/problems/house-robber/description/)

In [ ]:
```
class Solution:
    def rob(self, nums: List[int]) -> int:
        r1,r2=0,0
        for n in nums:
            temp = max(n+r1,r2)
            r1=r2
            r2=temp
        return r2
```

121.best-time-to-buy-and-sell-stock https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/ (https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/)

In [ ]:
```python
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        l,r = 0,1  ## l= buy r = sell
        maxprof = 0
        while r < len(prices):
            if prices[l]< prices[r]:
                profit = prices[r] - prices[l]
                maxprof = max(maxprof,profit)
            else:
                l=r
            r+=1
        return maxprof
```

70. climbing stairs https://leetcode.com/problems/climbing-stairs/description/ (https://leetcode.com/problems/climbing-stairs/description/)

In [ ]:
```python
class Solution:
    def climbStairs(self, n: int) -> int:
        one ,two =1,1
        for i in range(n-1):
            temp = one
            one =one +two
            two= temp
        return one
```

20. valid-parentheses https://leetcode.com/problems/valid-parentheses/description/ (https://leetcode.com/problems/valid-parentheses/description/)

In [ ]:
```python
class Solution:
    def isValid(self, s: str) -> bool:
        stack=[]
        map = { ")":"(","]":"[","}":"{"}
        for c in s:
            if c in map:

                if stack and stack[-1]==map[c]:
                    stack.pop()
                else:
                    return False
            else:
                stack.append(c)
        return True if not stack else False
```

1299.replace-elements-with-greatest-element-on-right-side https://leetcode.com/problems/replace-elements-with-greatest-element-on-right-side/description/ (https://leetcode.com/problems/replace-elements-with-greatest-element-on-right-side/description/)

In [ ]:
```python
class Solution:
    def replaceElements(self, arr: List[int]) -> List[int]:
        # max = -1
        # reverse order
        # newmax = max(oldmax,arr[i])

        rightmax= -1
        for i in range(len(arr)-1,-1,-1):
            newmax = max(rightmax,arr[i])
            arr[i]= rightmax
            rightmax = newmax
        return arr
```

202. happy-number https://leetcode.com/problems/happy-number/description/ (https://leetcode.com/problems/happy-number/description/)

In [ ]:
```python
class Solution:
    def isHappy(self, n: int) -> bool:
        visit = set()
        while n not in visit:
            visit.add(n)
            n = self.sumofsquares(n)
            if n==1:
                return True
        return False

    def sumofsquares(self, n: int) -> int:
        op= 0
        while n:
            digit = n%10
            digit = digit ** 2
            op+=digit
            n= n//10
        return op
```

35. search-insert-position https://leetcode.com/problems/search-insert-position/description/ (https://leetcode.com/problems/search-insert-position/description/)

In [ ]:
```python
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int:
        l,r = 0 ,len(nums)-1
        while l<=r:
            mid = (l+r)//2
            if target == nums[mid]:
                return mid
            if target> nums[mid]:
                l= mid+1
            else:
                r = mid-1
        return l
```

046. last-stone-weight https://leetcode.com/problems/last-stone-weight/description/ (https://leetcode.com/problems/last-stone-weight/description/)

In [ ]:
```python
import heapq
class Solution:
    def lastStoneWeight(self, stones: List[int]) -> int:
        stones=[-s for s in stones]
        heapq.heapify(stones)

        while len(stones)>1:
            first = heapq.heappop(stones)
            sec = heapq.heappop(stones)
            if (sec> first):
                heapq.heappush(stones,first-sec)
        stones.append(0)
        return abs(stones[0])
```

26.remove-duplicates-from-sorted-array https://leetcode.com/problems/remove-duplicates-from-sorted-array/description/ (https://leetcode.com/problems/remove-duplicates-from-sorted-array/description/)

In [ ]:
```python
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        l=1
        for r in range(1,len(nums)):
            if nums[r]!=nums[r-1]:
                nums[l]=nums[r]
                l+=1
        ## return nums[:l] when arr needed
        return l
```

263.ugly-number https://leetcode.com/problems/ugly-number/description/ (https://leetcode.com/problems/ugly-number/description/)

In [ ]:
```python
class Solution:
    def isUgly(self, n: int) -> bool:
        if n<=0:
            return False
        for p in [2,3,5]:
            while n%p==0:
                n= n//p
        return n==1
```

746. min-cost-climbing-stairs https://leetcode.com/problems/min-cost-climbing-stairs/description/ (https://leetcode.com/problems/min-cost-climbing-stairs/description/)

In [ ]:
```python
class Solution:
    def minCostClimbingStairs(self, cost: List[int]) -> int:
        cost.append(0)
        for i in range(len(cost)-3,-1,-1):
            cost[i]+= min(cost[i+1],cost[i+2])
        return min(cost[0],cost[1])
```

125.valid-palindrome https://leetcode.com/problems/valid-palindrome/description/ (https://leetcode.com/problems/valid-palindrome/description/)

In [ ]:
```
class Solution:
    def isPalindrome(self, s: str) -> bool:
        newstr= ""
        for c in s:
            if c.isalnum():
                newstr+= c.lower()
        return newstr == newstr[::-1]
```

205. isomorphic-strings https://leetcode.com/problems/isomorphic-strings/description/ (https://leetcode.com/problems/isomorphic-strings/description/)

In [ ]:
```
class Solution:
    def isIsomorphic(self, s: str, t: str) -> bool:
        maps ,mapt={},{}
        for i in range(len(s)):
            c1,c2 = s[i],t[i]
            if ((c1 in maps and maps[c1]!= c2) or (c2 in mapt and mapt[c2]!= c1)):
                return False
            maps[c1]=c2
            mapt[c2]=c1
        return True
```

191.number-of-1-bits https://leetcode.com/problems/number-of-1-bits/description/ (https://leetcode.com/problems/number-of-1-bits/description/)

In [ ]:
```
class Solution:
    def hammingWeight(self, n: int) -> int:
        res = 0
        while n:
            res += n%2
            n= n>>1
        return res

## 2nd method
class Solution:
    def hammingWeight(self, n: int) -> int:
        res = 0
        while n:
            n &= (n-1)
            res+=1
        return res
```

217.contains-duplicate https://leetcode.com/problems/contains-duplicate/description/ (https://leetcode.com/problems/contains-duplicate/description/)

In [ ]:
```
class Solution:
    def containsDuplicate(self, nums: List[int]) -> bool:
        map = set()
        for n in nums:
            if n in map:
                return True
            map.add(n)
        return False
```

605. can-place-flowers https://leetcode.com/problems/can-place-flowers/description/ (https://leetcode.com/problems/can-place-flowers/description/)

In [ ]:
```python
class Solution:
    def canPlaceFlowers(self, flowerbed: List[int], n: int) -> bool:
        f = [0]+ flowerbed+[0]
        for i in range(1,len(f)-1):
            if f[i-1]==0 and f[i]==0 and f[i+1]==0:
                f[i]=1
                n-=1
        return n<=0
```

28. find-the-index-of-the-first-occurrence-in-a-string https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/description/ (https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/description/)

In [ ]:
```python
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        if needle=="":
            return 0
        for i in range(len(haystack)+1 -len(needle)):
            if haystack[i: i+len(needle)]== needle:
                return i
        return -1
```

977. squares-of-a-sorted-array https://leetcode.com/problems/squares-of-a-sorted-array/description/ (https://leetcode.com/problems/squares-of-a-sorted-array/description/)

In [ ]:
```python
class Solution:
    def sortedSquares(self, nums: List[int]) -> List[int]:
        res =[]
        l,r = 0, len(nums)-1
        while l<=r :
            if nums[l] **2 > nums[r] **2:
                res.append(nums[l] **2)
                l+=1
            else:
                res.append(nums[r] **2)
                r-=1
        return res[::-1]
```

283. move-zeroes https://leetcode.com/problems/move-zeroes/description/ (https://leetcode.com/problems/move-zeroes/description/)

In [ ]:
```
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        l=0
        for r in range(len(nums)):
            if nums[r]:
                nums[l],nums[r]= nums[r],nums[l]
                l+=1
        return nums
```

136. single number https://leetcode.com/problems/single-number/description/ (https://leetcode.com/problems/single-number/description/)

In [ ]:
```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        res = 0
        for n in nums:
            res = res^ n
        return res
```

213. house-robber-ii https://leetcode.com/problems/house-robber-ii/description/ (https://leetcode.com/problems/house-robber-ii/description/)

In [ ]:
```
class Solution:
    def rob(self, nums: List[int]) -> int:
        return max(nums[0],self.help(nums[1:]),self.help(nums[:-1]))

    def help(self,num):
        r1,r2 = 0,0
        for n in num:
            temp = max(n+r1,r2)
            r1=r2
            r2=temp
        return r2
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: