**ARRAY**

**26. Remove Duplicates From Sorted Array:**

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

Soln-1:

Using variable for index:

For eg:

nums = [0,0,1,1,1,2,2,3,3,4]

Code:

idx = 1

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

if nums[i] != nums[i-1]:

nums[idx] = nums[i]

idx += 1

return idx

Soln-2:

Using Set:

For eg:

nums = [0,0,1,1,1,2,2,3,3,4]

* First, we get the set of nums which is:

[0,1,2,3,4]

* Sort the set
* Then, replace the nums with this sorted set

Code:

res=list(set(nums))

nums[:len(res)]=sorted(res)

return len(res)

**1. Two Sum:**

[**https://leetcode.com/problems/two-sum/**](https://leetcode.com/problems/two-sum/)

Soln-1:

for i in range(len(nums)):

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

if nums[j] == target - nums[i]:

return [i, j]

Soln-2:

Using HashTable:

hash\_table = {}

for i in range(len(nums)):

if nums[i] in hash\_table:

return [hash\_table[nums[i]], i]

hash\_table[target - nums[i]] = i

53. Maximum Subarray:

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

KADANE’S ALGO:

Initialize:  
    max\_so\_far = INT\_MIN  
    max\_ending\_here = 0

Loop for each element of the array

  (a) max\_ending\_here = max\_ending\_here + a[i]  
  (b) if(max\_so\_far < max\_ending\_here)  
            max\_so\_far = max\_ending\_here  
  (c) if(max\_ending\_here < 0)  
            max\_ending\_here = 0  
return max\_so\_far

Soln:

maxsum, s = nums[0], 0

for n in nums:

s+=n

maxsum = max(s, maxsum)

if s<0:

s=0

return maxsum

**ROTATE ARRAY:**

<https://leetcode.com/problems/rotate-array/>

We are dividing the array into two parts: i.e. split from n-k th index.

% since we don’t want index to get out of position.

**Soln:**

n=len(nums)

nums[:] = nums[(n-k)%n:] + nums[:(n-k)%n]

**Check if array is sorted and rotated:**

<https://leetcode.com/problems/check-if-array-is-sorted-and-rotated/description/>

**Soln:**

count=0

for i in range(len(nums)):

if(nums[i]>nums[(i+1)%len(nums)]):

count+=1

return (count<=1)

**Subarray Sum equals to K:**

<https://leetcode.com/problems/maximum-subarray/description/>

Soln:

count, s = 0,0

hmap = {0:1}

for i in (nums):

s+= i

if (s - k) in hmap:

count+= hmap[s - k]

if s not in hmap:

hmap[s] = 1

else:

hmap[s]+=1

return count

**Best time to buy and sell stock:**

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/>

**Soln:**

profit = 0

buy = prices[0]

for sell in prices[1:]:

if sell > buy:

profit = max(profit, sell - buy)

else:

buy = sell

return profit

**Best time to buy and sell stock II:**

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

**Soln:**

ans=0

buy=prices[0]

for sell in prices[1:]:

if(sell < buy):

buy=sell

else:

ans+=sell-buy

buy = sell

return ans

**Rotate Image:**

[**https://leetcode.com/problems/rotate-image/**](https://leetcode.com/problems/rotate-image/)

**Soln:**

l,r=0,len(matrix)-1

while l<r:

for i in range(r-l):

t,b=l,r

# storing the top left number in temp

temp = matrix[t][l+i]

# storing bottom left into top left

matrix[t][l+i] = matrix[b-i][l]

# storing bottom right into bottom left

matrix[b-i][l] = matrix[b][r-i]

# storing top right into bottom right

matrix[b][r-i] = matrix[t+i][r]

# storing temp into top right

matrix[t+i][r] = temp

r-=1

l+=1

**Spiral matrix:**

<https://leetcode.com/problems/spiral-matrix/>

**Soln:**

res=[]

left,right=0,len(matrix[0])

top,bottom=0,len(matrix)

while left < right and top < bottom:

# to get the top row

for i in range(left, right):

res.append(matrix[top][i])

top+=1

# to get right col

for i in range(top, bottom):

res.append(matrix[i][right-1])

right-=1

if not(left < right and top < bottom):

break

# to get bottom row

for i in range(right-1, left-1,-1):

res.append(matrix[bottom-1][i])

bottom-=1

# to get left col

for i in range(bottom-1, top-1,-1):

res.append(matrix[i][left])

left+=1

return res