Day-14

Python DSA

Largest Subarray with 0 Sum

https://www.geeksforgeeks.org/problems/largest-subarray-with-0-sum/1

```
def maxLen(arr):
    max_length =0
    n = len(arr)
    for i in range(n):
        Sum=0
        for j in range(i,n):
            Sum +=arr[j]
                if Sum==0:
                    max_length = max(max_length, j-i+1)
    return max_length
arr = [1, -1, 3, 2, -2, -3, 3]
maxLen(arr)
```

Output

6

TC- O(N^2)

SC-O(1)

Optimal

```
class Solution:
    def maxLength(self, arr):
        n= len(arr)
        prefix_sum={}
        maxi=0
        Sum=0
        for i in range(n):
            Sum +=arr[i]
            if Sum==0:
                maxi = i+1
            else:
                if Sum in prefix_sum:
                    maxi= max(maxi, i- prefix_sum[Sum])
                else:
                    if Sum in prefix_sum:
                        maxi= max(maxi, i- prefix_sum[Sum])
                    else:
                        prefix_sum[Sum] =i
        return maxi
```

TC- O(N)

SC- O(N)

Leetcode 56 Merge Intervals

https://leetcode.com/problems/merge-intervals/description/

Bruteforce

```
def merge(intervals):
    n = len(intervals)
    intervals.sort()
    ans=[]
    for i in range(n):
        start, end = intervals[i][0] , intervals[i][1]
        if ans and end<= ans[-1][1]:
            continue
        for j in range(i+1,n):
            if intervals[j][0]<= end:
                end= max(end, intervals[j][1])
            else:
                break
        ans.append([start,end])
    return ans
intervals = [[1, 3], [2, 6], [8, 10], [15, 18]]
merge(intervals)
```

Output `[[1, 6], [8, 10], [15, 18]]`

TC- O(n*logn) +O(2 * n)

SC-O(N)

Optimal

```python
class Solution:
    def merge(self, intervals: List[List[int]]) -> List[List[int]]:
        n= len(intervals)
        intervals.sort()
        ans=[]
        for i in range(n):
            if len(ans)==0 or intervals[i][0]> ans[-1][1]:
                ans.append(intervals[i])
            else:
                ans[-1][1]= max(ans[-1][1], intervals[i][1])
        return ans
```

TC- O(n*log n) + O(n)
SC-O(n)