Date ____________ Page ____________

Day-195

Sliding Window-3

* Min. size Subarray Sum:

2	3	1	2	4	3

target = 7

⇒ We have ~~a~~ to find a min size subarray that sum is greater than or equal to target sum.

Ex:

1	2	4

7 >= 7

size = 3

,

4	3

7 >= 7

size = 2

⇒ Basic approach will be finding all the subarrays and then select the min size array from all the subarrays.

⇒ In the next approach, we will select the first element then increase the window size until ~~the~~ the sum ~~[illegible]~~ >= target.

⇒ If our sum >= target, then we will start finding smallest array by decreasing the size of subarray from the left.

0	1	2	3	4	5
2	3	1	2	4	3

start → ↗ end

→ nums

sum = 0

Date Page

Code

```
while(end<n){
        sum += nums[end];
        while (sum >= target){
                total = min(total, end-start+1);
                sum -= nums[start++];
        }
        end++;
}
```

* Min. window Substring

S = ADOBECODEBANC → n

t = ABC → m

⇒ We have to find min. length window that have 't' string all the chars.

⇒ In brute force approach, find all substring of S → match it with string t → if matches then it will be one of the answer.

⇒ Select the min. from all the answers.

T.C. → $O(n^2 * (n+m))$

Date Page

⇒ In more optimized approach, we will use previous question sliding window approach.

⇒ So, we will increase the window size until all chars of 't' are present & then we will decrease the window size ~~to~~ from starting to find min. size substring.

⇒ We will take a map in that, we will write target chars & their occurance i.e. 1 shows doesn't occur yet & 0 shows occurred in the substring.

⇒ Also, take a total variable so that we don't have to check again & again in the map.

⇒ Also, we will insert other chars into map i.e. D → -1

⇒ We will decrease total only if we do any change in the +ve no. (decreasing +ve no.) & vice versa.

⇒ Code

```
unordered_map <char, int> m;
for( i=0; i< t.size(); i++){
        m[t[i]]++;
}
```

Date ____________ Page ________

```
int start=0 , end= 0, ans = INT_MAX ,
index = -1; n=  s.size(); total = t.size()
                                          ;
while( end <n){
    m[s [end]] --;
    if(m[s[end]] >=0)
        total --;
    while( !total && start <=end){
        if (ans > end-start +1){
              ans =  end-start +1;
              index = start;
        }
        m[s[start]] ++;
        if(m[s [start] ] >0)
              total ++;
        start++;
    }
    end++;
}
```