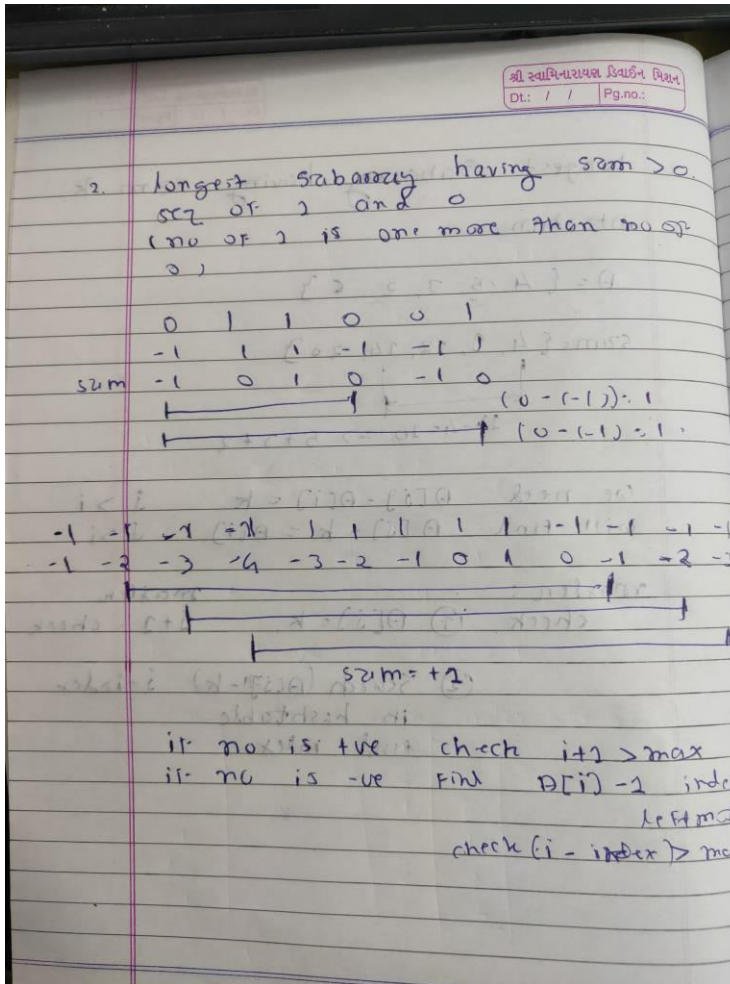


1. Longest subarray having sum > 0 in sequence of 0 and 1

<https://leetcode.com/problems/longest-well-performing-interval/submissions/>



2. Longest Subarray having sum K

<https://www.interviewbit.com/problems/longest-subarray-sum-b/>

2. longest Subarray having sum k.

intuition:

$A = \{4, 5, 3, 2, 6\}$

$sum = \{4, 9, 12, 14, 20\}$

$14 - 4 = 10 \Rightarrow 5 + 3 + 2$

We need $A[j] - A[i] = k$ $j > i$
will find $A[j] + k = A[i]$ $j > i$

maxlen: $i+1$ check

check: ① $A[j] = k$ $i+1$ check

② search $(A[j] - k)$ i -index
in hashtable
find index

3. Longest Subarray sum divides k

<https://www.geeksforgeeks.org/longest-subarray-sum-divisible-k/>

3. Longest subarray with sum divisible by k.

$A[i] = A[j] \quad \forall k \neq 0$

$\therefore (A[i] \cdot k) = (A[j] \cdot k)$

$k = 5$

2	2	3	6	1	4	5
sum	2	4	15	16	20	25
(1.5)	2	4	0	1	0	0

↑ ↑ ↑

i+1 i+1 i+1

(1.3) 2 0 0 0 2 3

sum = 8 div by 3

sum = 15 div by 3

sum = 5 div by 3

sum = 15 div by 3

$$X = ((X \cdot k) + k) \cdot k$$

4. Longest Subarray sum having sum greater than k

<https://www.geeksforgeeks.org/longest-subarray-sum-divisible-k/>

श्री स्वामिनारायण डिवाईन विद्या
Dt: / / Pg.no:

4. longest Subarray sum having sum greater than k

if $A[i] > k$ check $it > maxlen$
else
 $A[i] - A[j] > k$
 $A[i] - k > A[j]$
 $A[j] \geq A[i] - k$

From left search first no having value $\geq A[i] - k$

no -2 2 3 -1 +6 -3
 sum -2 0 3 +4 +2 -1
 index 0 1 2 3 4 5

$K=0$ $i+1$ \rightarrow $(-2) A[j] > -1$
 (-2)

sum	index	sum	index	minval
-2	0	-4	3	3
0	1	-2	0	0
3	2	-4	5	0
-4	3	0	1	0
+2	4	2	4	0
-1	5	3	2	0

For -1 $\Rightarrow (5-6) = 5$

5. Sliding Window

a. Longest Subarray sum at most k

<https://www.geeksforgeeks.org/longest-subarray-sum-elements-atmost-k/>

श्री स्वामिनारायण विद्यापीठ
Dt: / / Pg.no.:

6. longest subarray of size k elements
atmost k
all elements are positive.

$\{1, 2, 1, 0, 1, 1, 0\}$

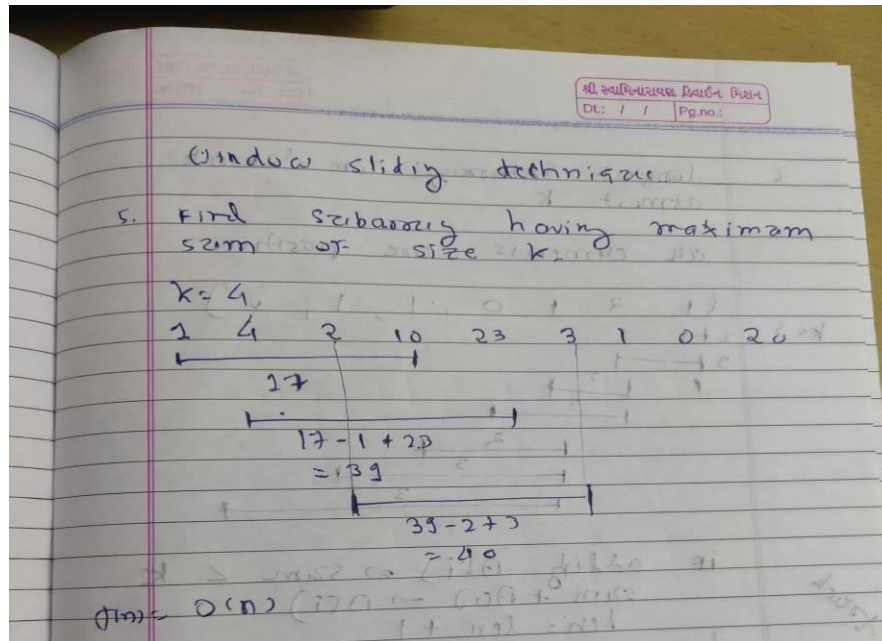
$k=3$

if adding $A[i]$ \Rightarrow sum $< k$
 sum $+ A[i] \rightarrow A[i]$
 len = len + 1

else
 max window right side
 will not decrease the window size
 becaz we want longest subarray.

b. Find subarray of size k having maximum sum

i. <https://www.geeksforgeeks.org/window-sliding-technique/>



6. Longest Substring without repeating char

- <https://leetcode.com/problems/longest-substring-without-repeating-characters/submissions/>
- Use Sliding Window
 - abcdec
 - abcdec
 - abcdec

Maintain a unordered map which maintains char to index mapping

Starting index j and iterate through i

When char is repeated make $j = \max(\text{first occur index} + 1, j)$

7. Longest Subarray not having more than k distinct elements

- <https://www.geeksforgeeks.org/longest-subarray-not-k-distinct-elements/>
 - 1 2 3 4 2 3 5 3 5
 - 1 2 3
 - 2 3 4 2 3
 - 2 3 5 3
 - 2 3 5 3 5

b. Idea is use unordered map whose size should be always $\leq k$. If more than k inc i and remove ith element from map. Keep track of max len.

c. i = starting window index , j = ending window index.

8. Maximum in each window of size k

- <https://leetcode.com/problems/sliding-window-maximum/submissions/>
- <https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k-using-stack-in-on-time/>

9. Maximum window substring

a. <https://leetcode.com/problems/minimum-window-substring/submissions/>

b. Use two pointers i = increment in map, j = decrement in map

Initially size = t.length()

If i increment from 0 then size++

If j decrement to 0 then size--

Check for min len while size = 0

When size is 0 try to reduce window size from front

10. Maximum product subarray

a. <https://leetcode.com/problems/maximum-product-subarray/submissions/>

b. int a = maxp;

int b = minp;

maxp = max(nums[i], max(a*nums[i], b*nums[i]));

minp = min(nums[i], min(a*nums[i], b*nums[i]));

ans = max(ans, maxp);

11. <https://leetcode.com/problems/longest-repeating-character-replacement/>