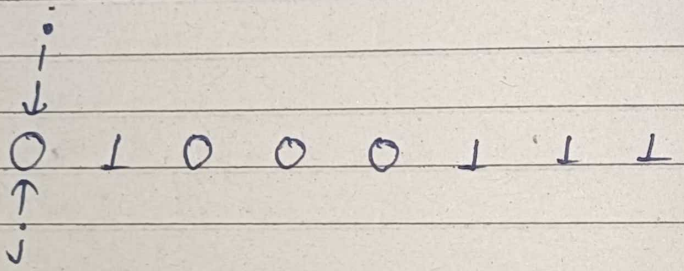
525. Contiguous Array

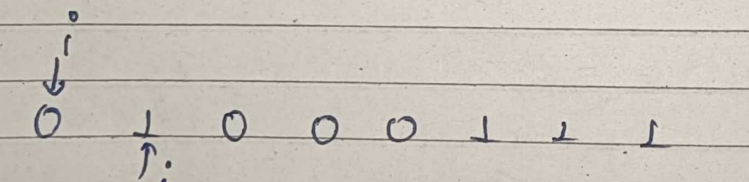
⇒ Similar Problems
- Leetcode - 560
- Leetcode - 930
- Leetcode - 1074

⇒ Why ~~not~~ Sliding window approch fails here?

⇒

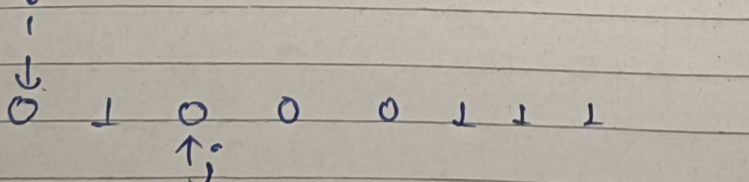


zero = 1
one = 0



zeroCnt = 1
oneCnt = 1

ans = max (ans, (j - i + 1))
ans = 1 - 0 + 1 = 2



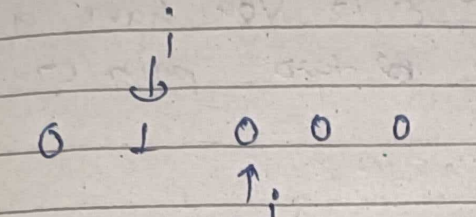
zeroCnt = 2
oneCnt = 1

⟶ Now zeroCnt > oneCnt

So in the Sliding window approch जैसे ही cnt बढ़ता था आप window को Shrink करते थे तब तक cnt equal न हो जाएं

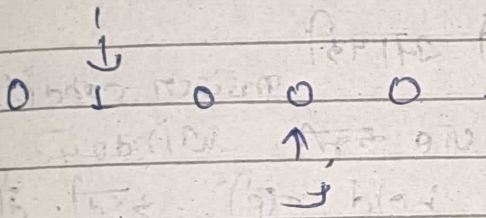
Teacher's Signature

i.e. increase the i



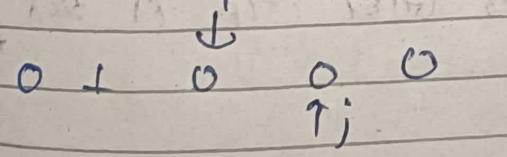
zeroCnt = 1
oneCnt = 1
} same → store the answer.

↳ Now increase the j



zeroCnt = 2
oneCnt = 1
→ Now zeroCnt ~~फिर~~ से 0 से ज्यादा है तो window को shrink करना पड़ेगा But Now 'i' is pointing to 1 if you increase the i then that will not affect the zeroCnt while previously i is pointing to zero.

if you increase the i

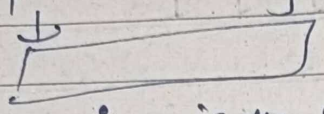


zeroCnt = 2
oneCnt = 0

So here it is clear that sliding window approach is not applicable.

Teacher's Signature

Sliding Window :- Sliding window approach का मतलब है कि you have a window with two pointer i & j



mainly i → is the starting point of the window

j → ending point of the window.

Mainly में जो window है तो आपकी answer condition को hold करती है. but जब कभी window answer condition को hold नहीं करती तो आप pointer i & j को ऐसे move करते हैं कि window answer condition को hold करे।

→ if you increment the j i.e. window is expanding.

→ if you increment the i i.e. window is shrinking.

हर time पे window या तो expand होगी या तो shrink होगी दोनो एक साथ नहीं हो सकते।

→ Using same pattern as LeetCode 560 ~~→~~ cnt of subarrays whose sum equal to k.

assume given array → { 1, 0, 1 } is this array have equal no. of zeros & ones ⟶ ans = NO. zeroCnt = 1, oneCnt = 2.

if array → { 1, 0, 1, 0 }
↳ zeroCnt = 2, oneCnt = 2 } equal no. of zeros & ones.

So if you replace the zeros with -1 then
{ 1, -1, 1, -1 } ⟶ sum = 0

whenever sum = 0 with replacing 0 with -1 ~~to~~ i.e. equal no. of zeros & ones.

e.g. { 0, 0, 1, 0, 0, 0, 1, 1 }

So the above same approach you can apply here also.

So if you replace the zeros with -1

Teacher's Signature

then your question is like you have to find the largest subarray whose target sum equal to zero.

{0, 0, 1, 0, 0, 0, 1, 1} targetSum=0 → largest Subarr

Use Map

cummulative sum	idx
0	-1
-1	0
-2	1
-3	4
-4	5

→ This is bcz जब हमने array पर iterate करना start ही नहीं किया था तब भी तो current sum = 0 है।

→ Map में बस वो sum ही डालोगे जो past में नहीं मिला हो।

currSum = 0

{0, 0, 1, 0, 0, 0, 1, 1}
 0 1 2 3 4 5 6 7

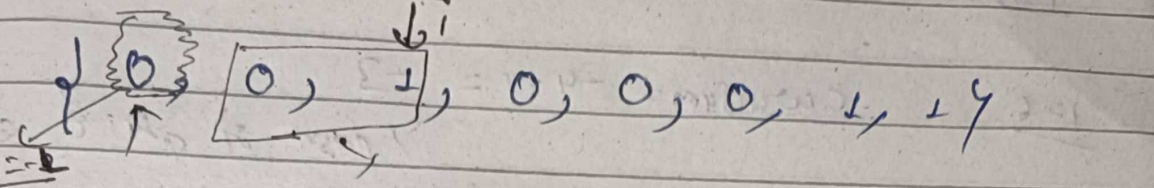
if(nus(i)==0)
  currSum += -1 ⇒ -1

i=1
currSum = -1 + -1 = -2

i=2 currSum = -1 + 1 = -1

Teacher's Signature

at i=2 currSum = -1



अब हम check करते हैं कि क्या हमने past में कभी currSum = -1 देखा है

answer → yes at index = 0

that means हमने idx = 0 पर currSum -1 देखा और उसके बाद में 0 हमने उसमें जुड़ा add किया और currentSum वापस -1 हो गया। i.e. जो add किया है उसका sum zero होगा।

और if sum zero होगा तो वह subarray equal no. of zeros & ones को contain करेगा।

length = i - 0 = 2

i = 3 currSum = -1 - 1 = -2

-2 को past में देखा → yes at idx = 1

i.e. from idx 2 to i sum = 0

length = i - 1 = 3 - 1 = 2

i = 4 currSum = -2 - 1 = -3

↳ push into map

Teacher's Signature

i=5 currSum = -3 - 1 = -4

i=6 currSum = -4 + 1 = -3
↳ past में देखा है कि
idx 4
length = i - 4 = 6 - 4 = 2

i=7 currentSum = -3 + 1 = -2
↳ past में देखा है
कि idx 1
length = i - 1 = 7 - 1 = 6

T.C. = O(n)
S.C. = O(n)