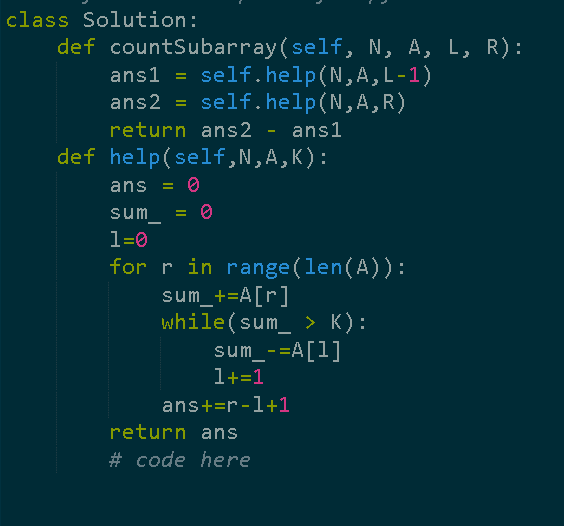
SLIDING WINDOW + TWO POINTERS

Problem link:

<https://practice.geeksforgeeks.org/problems/count-the-number-of-subarrays/1>



Roman to integer:

<https://leetcode.com/problems/roman-to-integer/description/>

just loop and add the value to the res except in one edge case

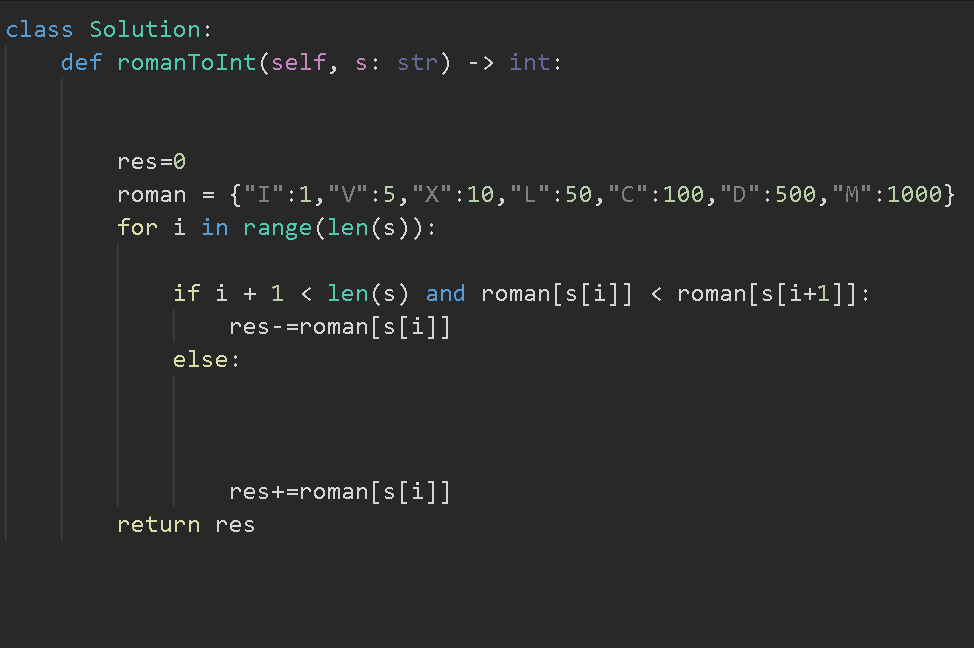
i.e

IV 🡪 in this case ‘I’ have to be substracted from ‘V’ to get 4

Algorithm

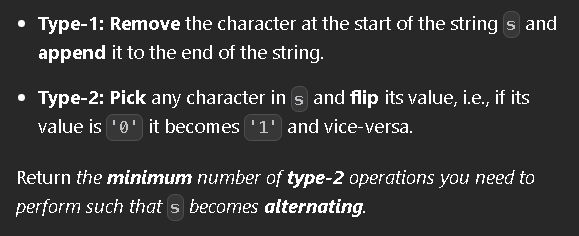
Just subtract the value from the res if the next character has a value greater than yours.

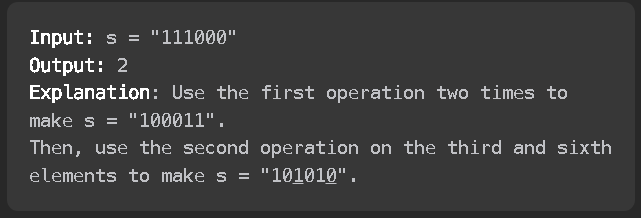
Or else add the value as usual



Problem link:

<https://leetcode.com/problems/minimum-number-of-flips-to-make-the-binary-string-alternating/description/>





S = 111000

Possible answers:

1) 101010

2) 010101

1) With S

101010

111000

0+ 1+0+0 +1+0 = 2 operations (flips)

2) with S

010101

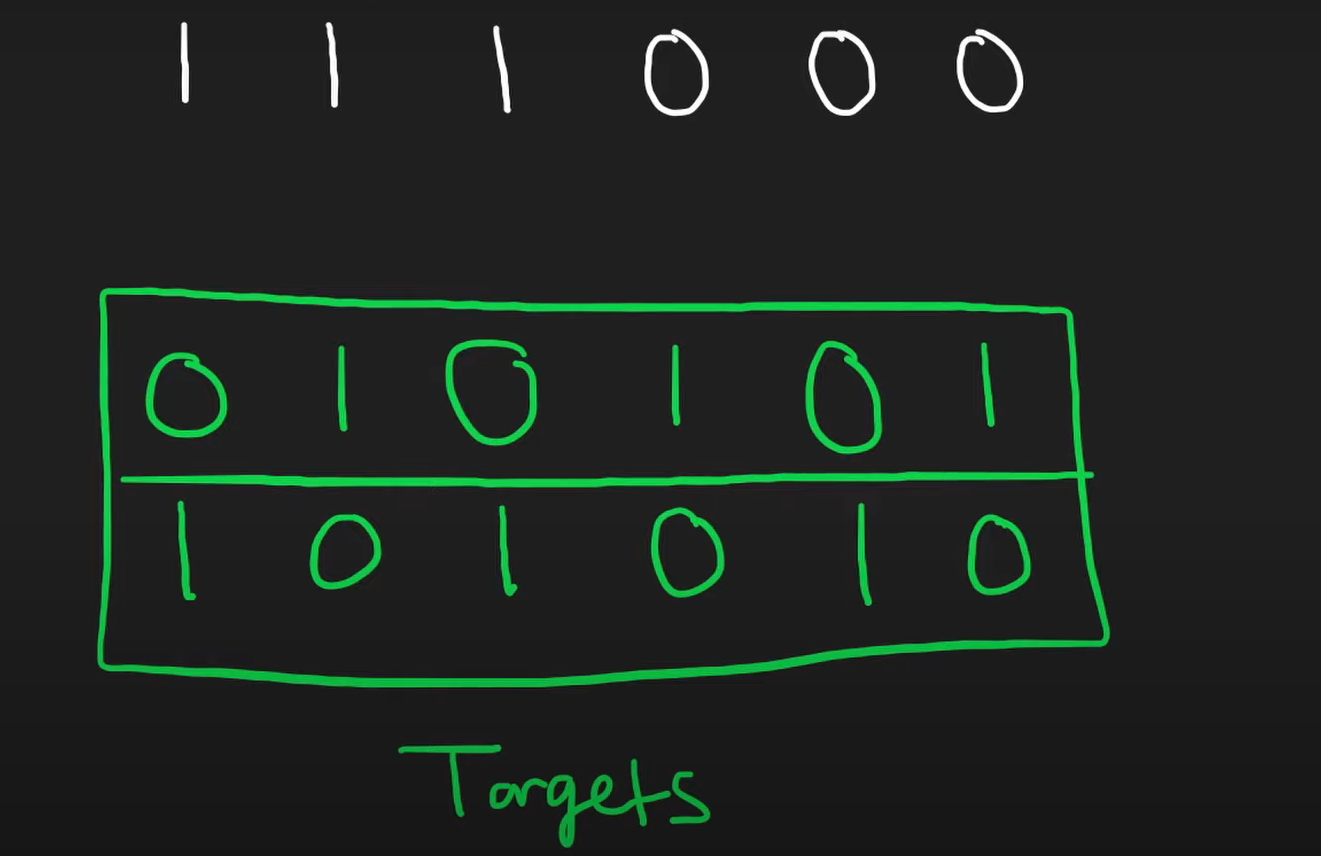
111000

1+0+1+1+0+1 = 4 operations (flips means difference)

So min(2,4) = 2

Now instead of moving element ,one by one to end and then computing the difference rather we would add the entire string to the end and we will use sliding window to compute the difference.

Possible s are = 111000 and 110001 and 100011 and 000111 and 001110 and 011100



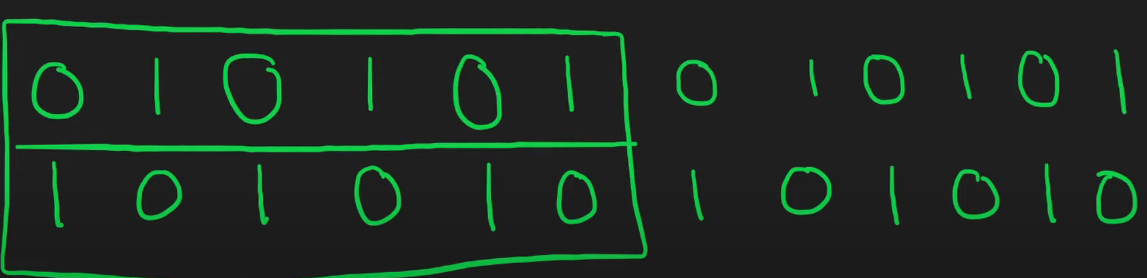
S= s+s(to avoid the repeat work for each s)

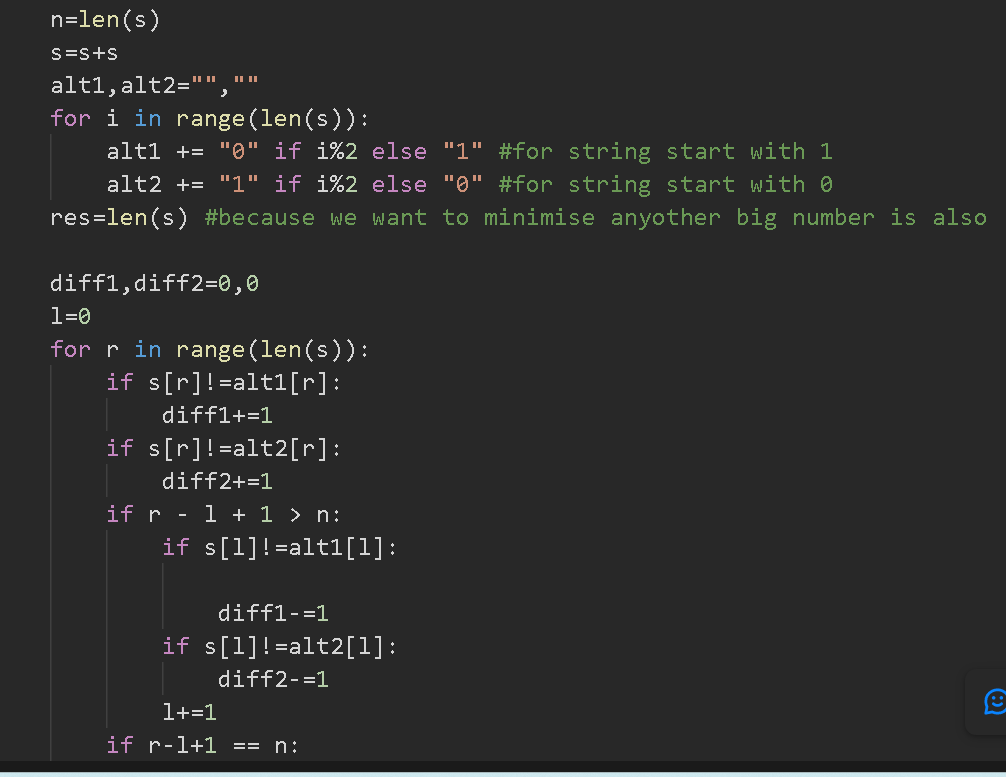
we will loop the given S and if we find that s[currentindex]!=possible\_answer[currentindex]

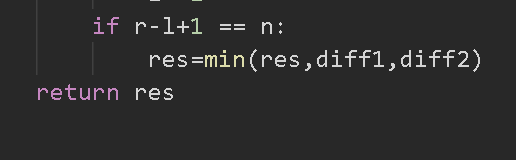
then we will increase the difference by 1

and do not forget to decrease the difference once the window size exceeds

alt1 and alt2 looks like



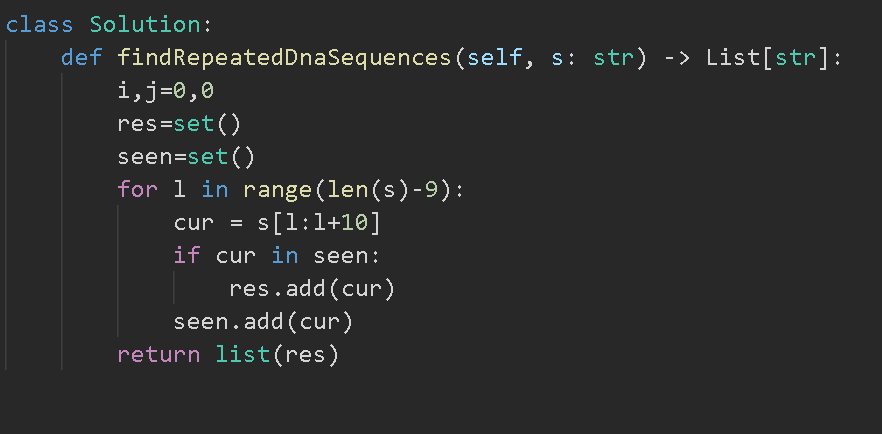




problem link:

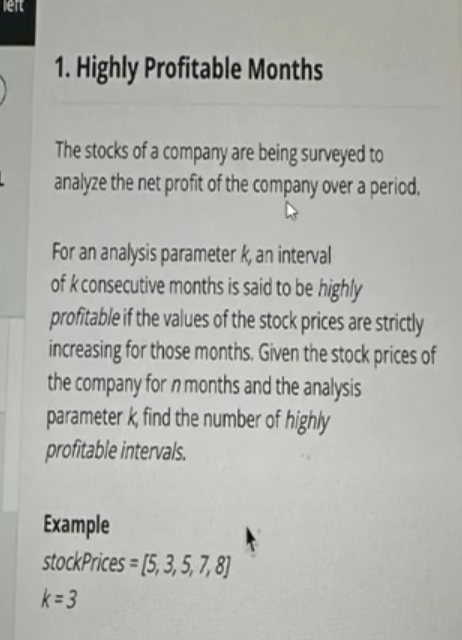
<https://leetcode.com/problems/repeated-dna-sequences/>

intuition is to add all the DNA sequence to set and add the DNA sequence to the res which is already present in the set



Link to problem : <https://www.desiqna.in/12812/morgan-sde1-coding-questions-and-solutions-india-april-2023>

-> Q : Given an array, find the count of subarrays of size “K” such that they are strictly increasing.



-> [5 3 5 7 8]

K = 3

Answer : 2 ( [3,5,7] | [5,7,8])

Brute - Force : - Just check all subarrays of size K ; check if they are increasing ; if yes increase the counter and print the final answer.

C++ : <https://ideone.com/QKxMQ3>

Python : <https://ideone.com/CQ7o7J>

Java : <https://ideone.com/F4y080>

—-----------------

→ Observe the different increasing islands….

[ 2 5 1 1 5 8 10 4 3 5 6 7 8]

[1 2 3 …5…………………]

K = 3

-> [2 5]---> 0

-> [1 5 8 10] →  2 → [1 5 8] + [5 8 10]

-> [3 5 6 7 8] → 3    —->[3 5 6] + [5 6 7] + [6 7 8]

= 5

—> these strictly increasing island will never overlap each other

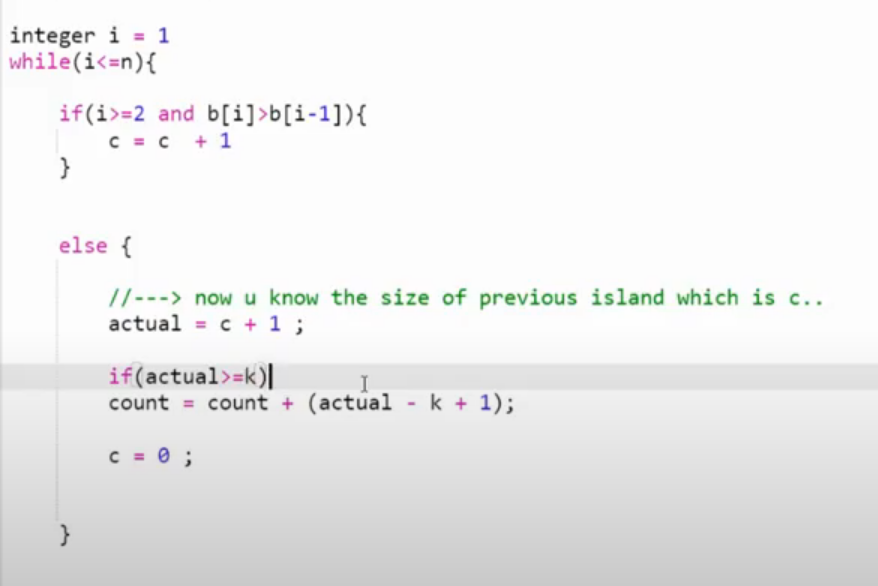
—--> solution is to first find all the island ;

—--> for each island ; number of valid subarrays is (size of island - k + 1)

—--> whenever the size of island is >=3 it can satisfy the condition

—--> travel the array in 1 go and in O(N) find all the islands and calculate the answer for each island and add it to the sum : TC : O(N) = O(2\*10^5) < 1S ===EFFICIENT.

If k == 1; return n (common sense) because there is no need to check for adjacent elements



If condition is for the get the size of the island and i >= 2 because our array indexing starts from 1 and there is no element in 0th index so b[2] is should be compared with b[1]

And b[1] should not be compared with b[0] as there is no element in b[0] .

Actual represents the size of the island

Actual = c + 1 because

For example in [3,5,8]

C is incremented when 3 is compared with 5 🡪 so c = 1

C is incremented when 5 is compared with 8 🡪 so c = 1 + 1 = 2

But the actual size is 3 so we did c+1

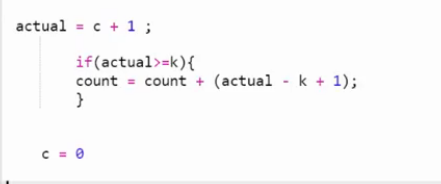
Count 🡪 represents the answer we need i.e the count of subarrays.

and the length of the actual should be >= k as we already discussed.

We have to take care of the last subarray

As when we reach the last element in the array , we won’t be going inside the else condition to find the count [but it is mandatory to go inside the else condition to find the count]

So we will take care of the last subarray after the while loop.



There is no use of c = 0 so ignore it

C++ : <https://ideone.com/FAkHNZ>

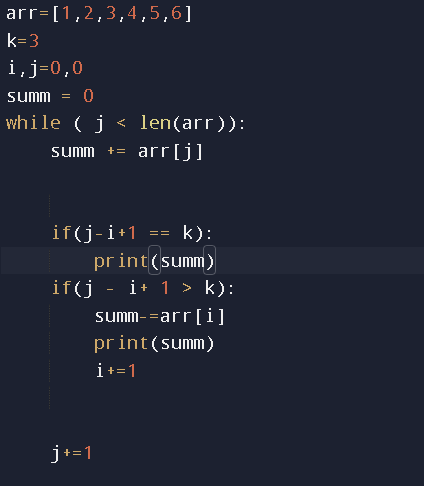
Py : <https://ideone.com/fVdX7n>

Jv : <https://ideone.com/lnJ5ny>

Practice questions

Link:

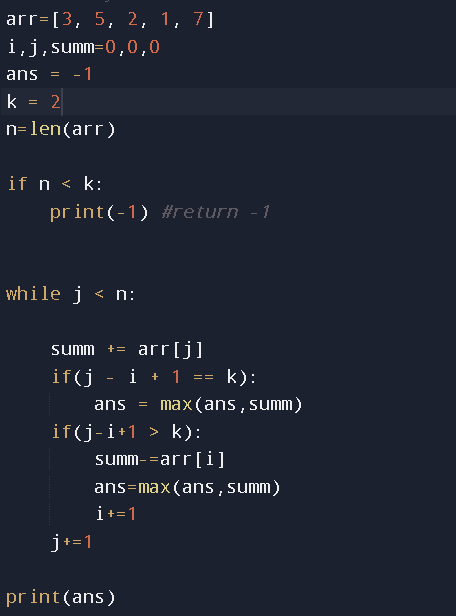
<https://www.geeksforgeeks.org/sum-of-all-subarrays-of-size-k/>



Link:

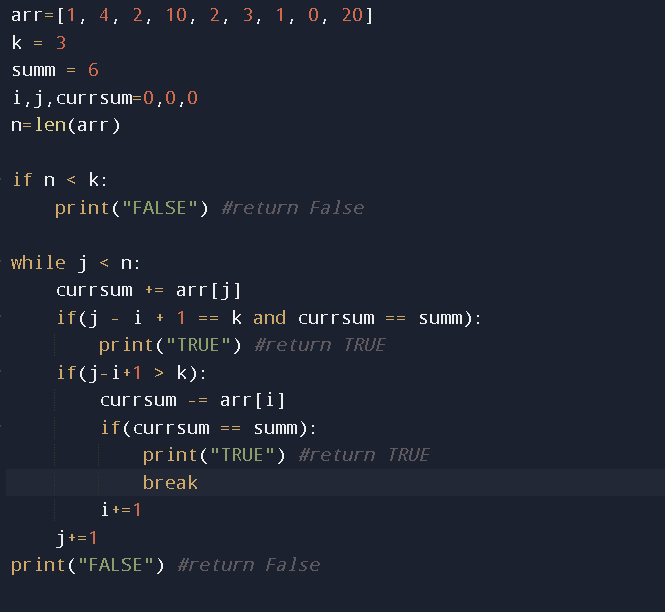
<https://www.geeksforgeeks.org/window-sliding-technique/>

(maximum sum subarray of size k)



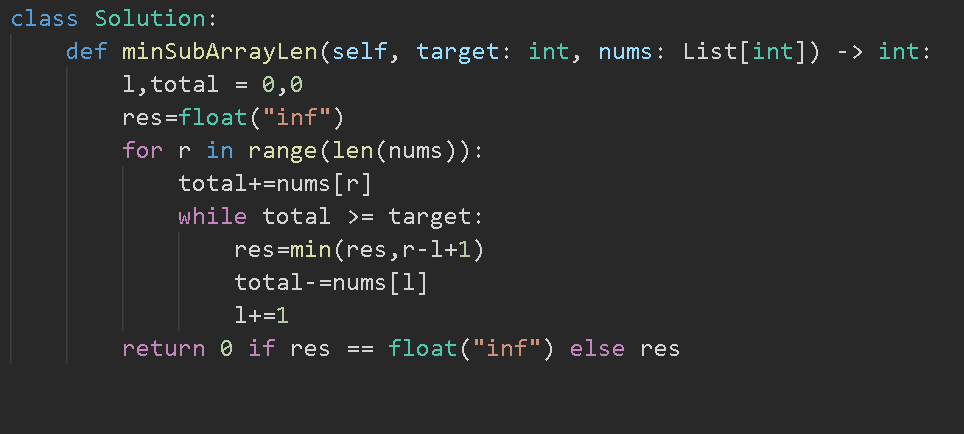
Link:

<https://www.geeksforgeeks.org/subarray-of-size-k-with-given-sum/>



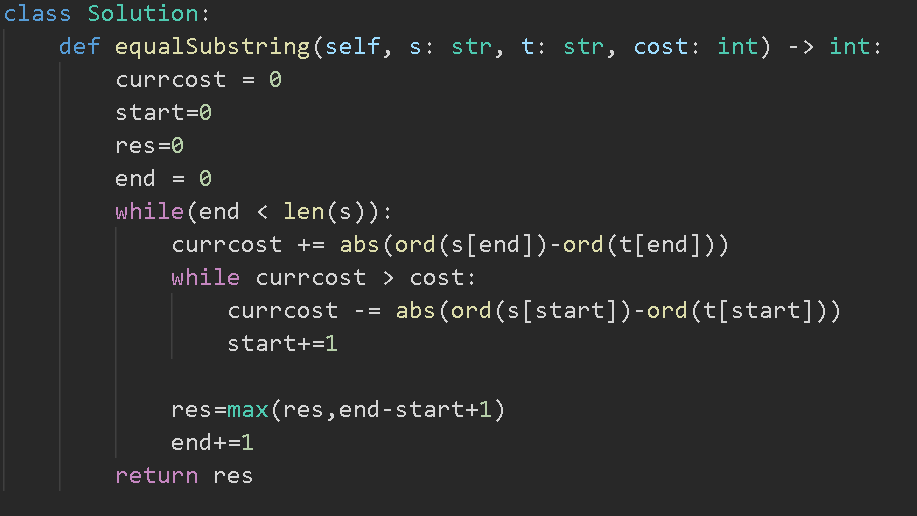
Problem link:

<https://leetcode.com/problems/minimum-size-subarray-sum/>

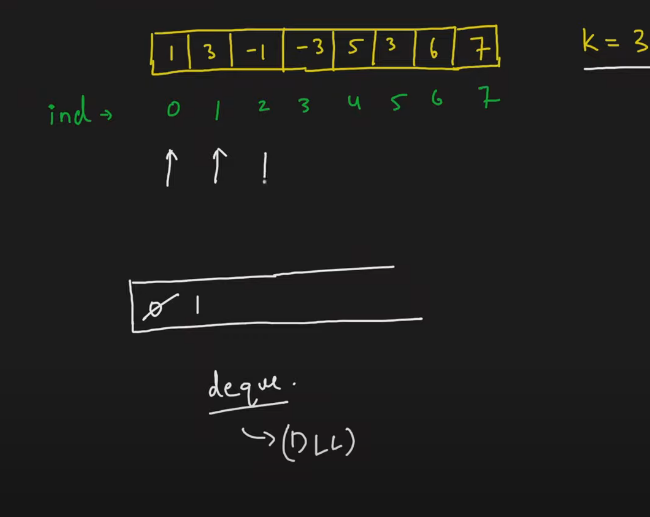


Problem link:

<https://leetcode.com/problems/get-equal-substrings-within-budget/description/>



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



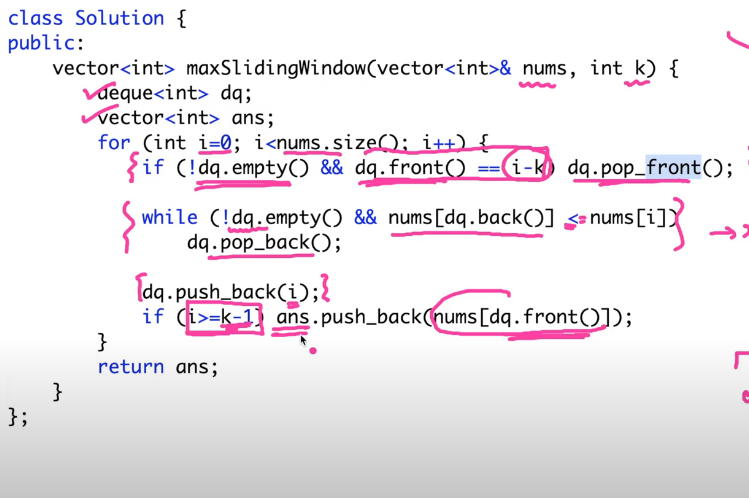
INTUITION:

To somehow put all the index of max elements in the front of the deque by maintaining a high value elements to low value element trend in the deque

We are maintaining a deque

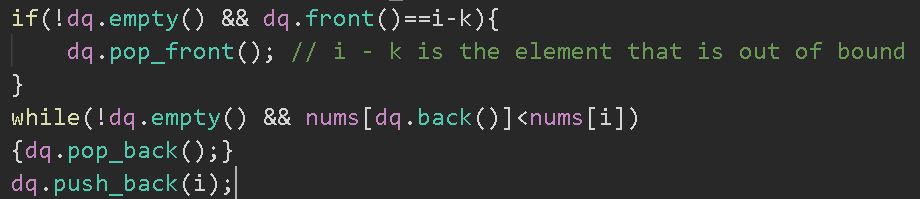
If we see a element that is greater than the element at the back in the deque we will be keep on poping the element from the back

But if the incoming element is lesser than the element in deque then the element is appended



i>=k-1 because

i should atleast be k-1 to reach the window size of k as i starts from 0.



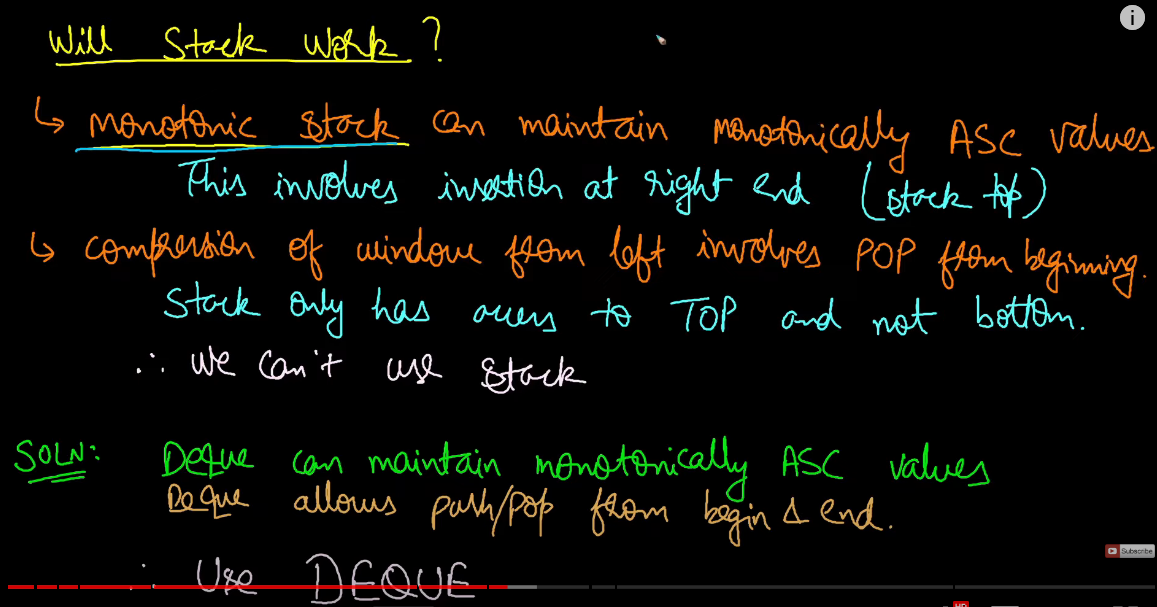
The if condition comes after the while loop because

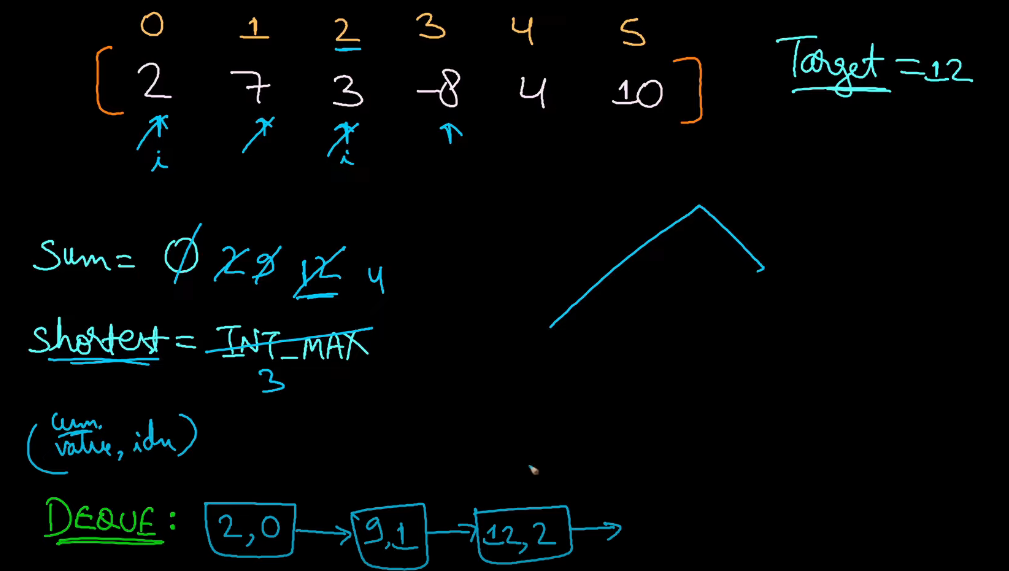
Firstly we have to remove the elements that is out of bound before appending the new elements.

<https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/description/>

the difficulty with this question is it has negative numbers.

The sliding window window approach will working only for monotonically increasing graph i.e only positive number and not in this case





In the above dry run

i = 0 🡪 sum = 2

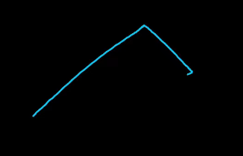
i= 1 -🡪 sum = 2+7=9

i=2 🡪 sum = 2+7+3 = 12

now 12 is >= 12 so

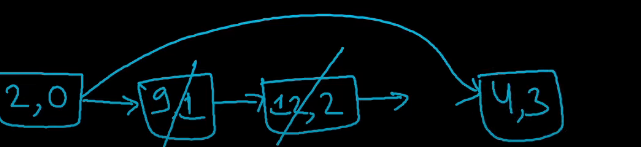
shortest = min(shortest, i+1) = 3

i = 3 🡪 sum = 12-8 = 4 this make the graph the come down and not monotonically increase

 that is only represented in that image.

So to maintain a montonically increase the function we will remove 12 and 9 from the deque because 12 and 9 are greater than 4.

So now the deque looks like



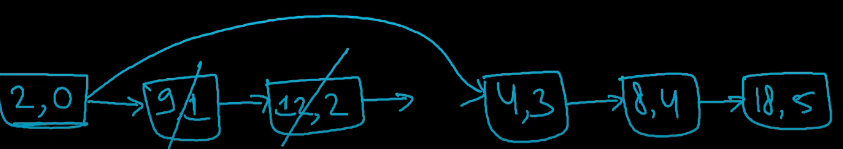
i = 4 🡪 sum = 4+4 = 8

i = 5 🡪 sum = 8+10 = 18

now again 18 >= 12 (target) so

shortest = min(3,i+1)=min(3,6)=3

but we know the answer is 2.(sum of 14 by last two elements)



Still the algorithm is not finished

Now we have to compress from the left side

We will check if we subtract 2 from 18 will it still be greater than the target. If yes then we will pop 2 from the deque

Similary we will check for 4

18-4=14 >=12 so we will pop 4 also

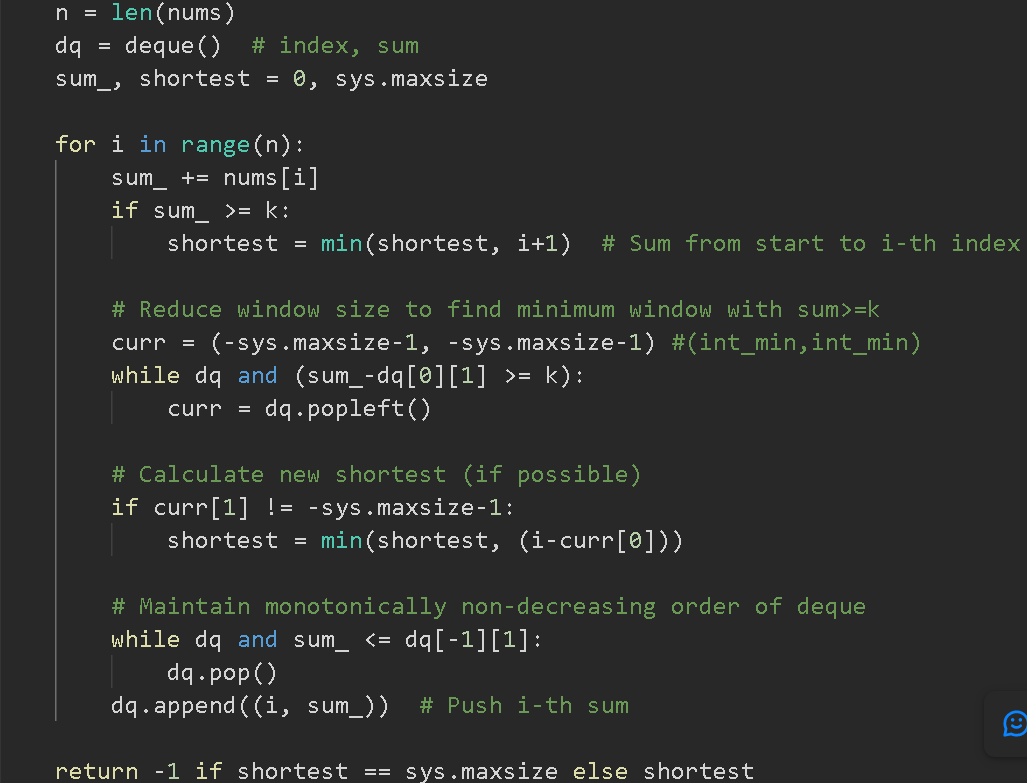
But now

For 8

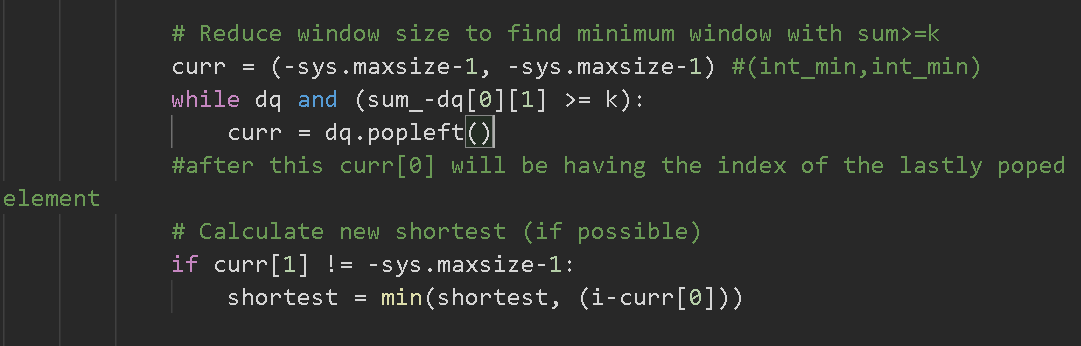
18-8 = 10 is not >= 12 so we cannot remove it

So the index of 8 would the start of our shortest subarray but the length would be start of 4 which is popped a step before. Because index of 8 (in the deque and not in array) and index of 18 gives a difference of 1

So extra 1 should be added to get the length as 2.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



This above piece of code comes before creating the deque() [endha order la venalum code panalam]

