1. LONGEST SUBSTRING WITHOUT REPEATING CHARACTERS

Ex: Input — abcabcbb
Output - 3.

Brute Force — Check each substring one by one to see if it has no repeating character.

$O(n^3)$ — $O(n) \times O(n) \times O(n)$

$O(n)$: scan element

$O(n)$: Find if all are unique.

Approach 2 → SLIDING WINDOW TECHNIQUE

In the Brute force step we keep checking if the the substring has duplicate character. This is unnecessary.

A sliding window is an abstract concept commonly used in array/string problems. A window is a range of elements in the array/string which is usually defined by start/end indices.

i: abcabcbb

i = 0, j = 0 & 1 & 2 & 3, table = { }, ans = 0

max(0, 3) = 3

Complexity:

In the worst case each element will be repeated twice

Hence $O(2n) \approx O(n)$

table

substring	size
abc	3
bca	3

pwwkew i = 0 1

j = 0 1 2

0 1 2 3 4 5 6

pwwwkeyw

i = 0 1 p, 0

j = 0 1 2 w, 1

APPROACH-3 - SLIDING WINDOW Optimized

"tmnzuxtz"

0 1 2 3 4 5 6 7

i = 0 2

j = 0 1 2 3 4 5 6 7

ans = 1 2 3 4 5

map

t	1
m	2
z	4
u	5
x	6
t	7

The above solution requires $O(2n)$ steps. In fact it can be optimized to require only n steps. Instead of telling if a character exists or not we can define a mapping of characters to its index. Then

we can skip the characters immediately when we find a repeating character.

0 1 2 3 4 5
p w w k e w

hash_set

~~p~~ w
~~w~~ k e

a_x = ~~0~~ + 2

b_x = ~~0~~ + 2 3 4

max = ~~0~~ + 2