76. Minimum Window Substring

Ex

S = "ADOBECODEBANC"

t = "ABC"

ans = "BANC"

S = "ADOBECODEBANC"
window1 = 6
window2 = 4

Approach-1

Try all possible substrings from string s of length - n

→ O(n²)

→ Now check if that sub-string is containing all characters from string 't'

→ O(m) (length of 't')

T.C. = O(n² * m)

Can be improvised by keeping track of 26 characters in hash map

comparision of 26 character with window O(26)

T.C. = O(n² * 26)

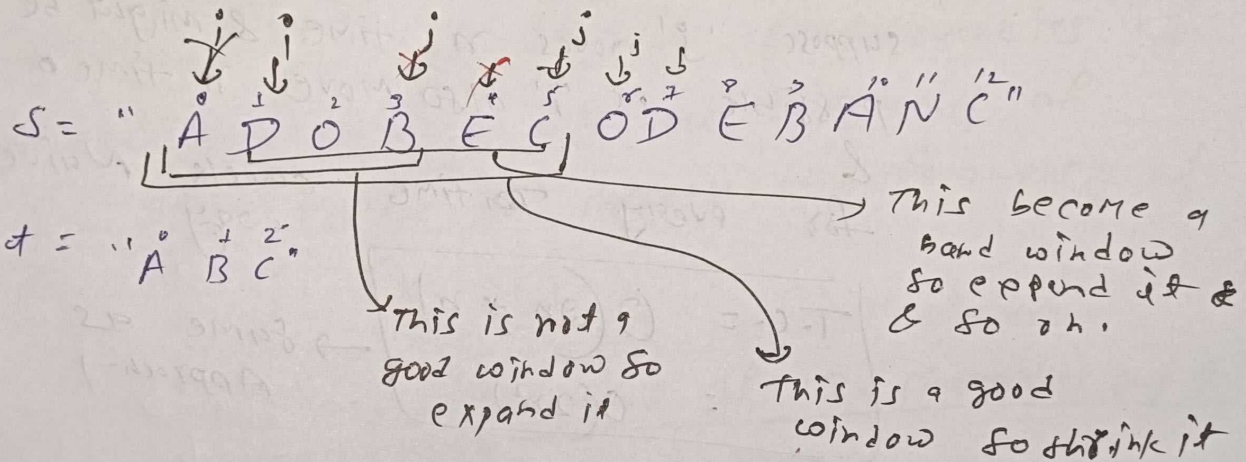
⇓

T.C. = O(n²)

Approch-2

if your window is not good then think for bigger window
↳ expand the window

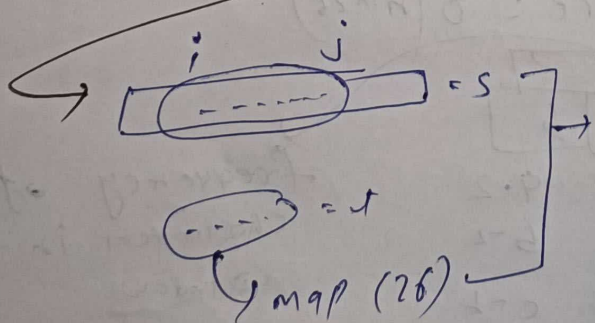
if your window is good then shrink the window
↳ shrink the window



There is standard template for sliding window approch—

(i) Keep expanding 'j' at every step.

(ii) when correct window matches with condition to t then keep on shrinking 'i'.



condition is all characters of 't' should be in the window (i to j) of S.

To implement this logic we have 3 methods which are as following

(i) Naive way :- Iterate the window i to j cont for (s) & (t)

String t is stored is has map

↳ T.C. = O(n) → for every window
S.C. = O(1) → for every pointer movement we take O(1) time

Overall Time Complexity

pointer movement = ~~2n~~ O(2n)

suppose 'j' moves n time & might be possible is 'i' also move n time

&

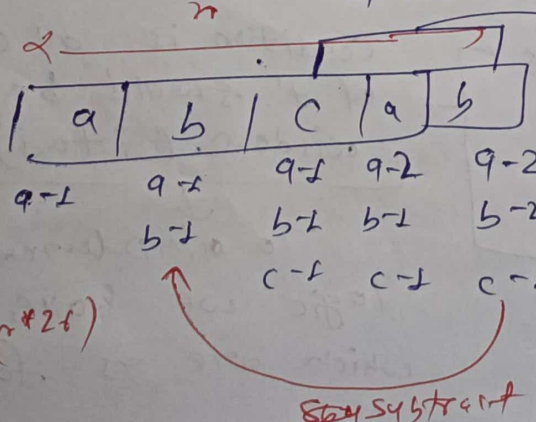
for every time check Naive way

T.C. = O(2n * n)
= O(2n²) → same as Approch-1

(ii) Smart way :- Keep track of prefix sum of (26) english character at every index

- O(26) time to get count of all characters in a window.
- Space = O(n*26)

T.C. = O(n*26)
S.C. = O(n*26)



(iii) Smartest way

In (ii) smart way The space is required bcz we are keeping track of the frequency of 's'

But our only concern is that if 't' is lies in s of a in a window of s or not

So rather that keep taking of frequencies of 's' we can simply use the frequency of 't' i.e. decrease & increase the frequency of 't'.

keep only count of 26 character of t

Time = O(1)
Space = 26

Dry Run (How Step 1 & 2 will work)

s = "A D O B E C O D E B A N C"
    0 1 2 3 4 5 6 7 8 9 10 11 12

t = "A B C"
    0 1 2

tMap

A → 1
B → 1
C → 1

(1)

i

S = A D O B E C O D E B A N C

0 1 2 3 4 5 6 7 8 9 10 11 12

j

t = 'ABC'

cnt = 3

gMap

A - 1
B - 1
C - 1
D - 1
O - 1
E - 1

if (cnt == 0)

↳ store your ans

ans = 6

Now decrease the size of window

cnt = 0

↓

count has become '0'

↓

this mean window has all element of 't'

Now, try to update your answer with min length

Then try to shrink the window

↳ i.e. increase i till window is good → i.e. increase 'i' till cnt = 0

if cnt > 0 → then increase the window i.e. j++;

i

j

A D O B E C O D E B A N C

cnt = 0

↳ increase i

A - 0
B - 0
C - 0
D - 1
O - 1
E - 1

→ cnt = 1

Code

```
string minWindow (string s, string t)
{
    unordered_map <char, int> tmp;
    for(auto ch : t) tmp[ch]++;
    int i=0, j=0, counter=0, size(), minStart=0,
        minLen = INT_MAX, n = s.size();

    while ( j < n)
    {   if (tmp[s[j]] > 0)  counter--;
        tmp[s[j]]--;
        j++
        whi
```

T.C. = O(n+m) → O(2n+m)

S.C. = O(26)