Sliding windows

(Ref: https://medium.com/outco/how-to-solve-sliding-window-problems-28d67601a66)

Min. window substring

Given string S, and a set T of characters, find min window in S which will contain all characters in T

Naive: Check $S(i)\ S(i+1)\ ..\ S(j)$ $\forall\ i \leq j$

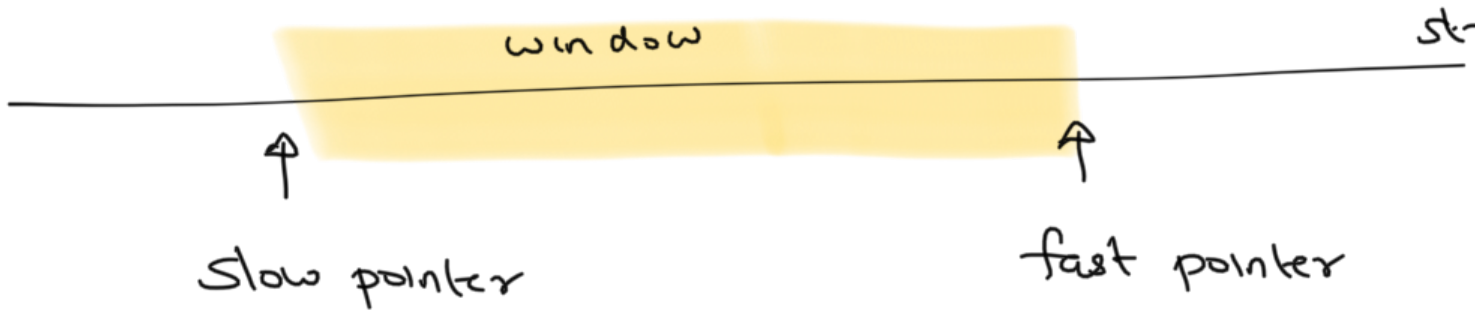
If it works. Let $n = |S|$

$O(n^2)$ pairs (i, j) to check, each pair takes $O(n)$ to check. So $O(n^3)$ algorithm

↑

[with memoization, can become $O(1)$]

Sliding window solution: $O(n)$



→ move fast pointer to grow your window till you see all characters

→ once you find a valid window, start sliding slow pointer up shrinking window till no longer valid

→ Take min / all valid windows

(eg) s = 'adobe codebanc' t = {a, b, c}

a d o be c o d e b a n c — [First valid window length = 6]

a d o be c o d e b a n c — [shrink to get first invalid window]

a d o be c o d e b a n c — [second valid window length = 10]

a d o be c o d e b a n c — [shrunk valid, length=9]

a d o be c o d e b a n c — [shrunk valid, length 8]

a d o be c o d e b a n c — [shrunk valid, length 7]

a d o be c o d e b a n c — [shrunk valid, length 6]

a d o be c o d e b a n c — [shrink to get second invalid window]

a d o be c o d e b a n c — [valid window, length 7]

a d o be c o d e b a n c — [valid window, length 6]

a d o be c o d e b a n c — [valid window, length 5]

a d o be c o d e b a n c — [valid window, length 4]

a d o be c o d e b a n c — [Terminate. can't move fast pointer up]

min window length = 4

Implementation

→ Keep a count dictionary

$$\{ \text{'a'}: 0,\ \text{'b'}: 0,\ \text{'c'}: 0 \}$$

→ Update it so that it reflects the count of characters in current window.

→ This will help determine validity of window

Complexity : O(n) because both pointers moving forward [at most n times]

correctness :

→ then any window ⊇ [s,f] will be valid but longer

→ so don't care about it for computing min valid window length