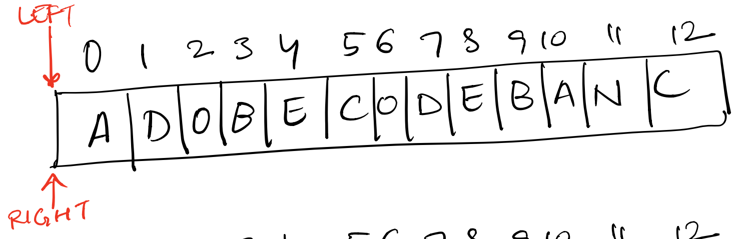
MINIMUM WINDOW SUBSTRING

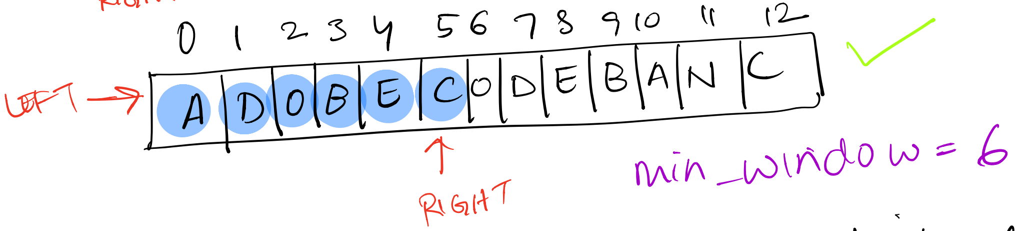
Given a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

Input S = "ADOBECODEBANC".
(0 1 2 3 4 5 6 7 8 9 10 11 12)

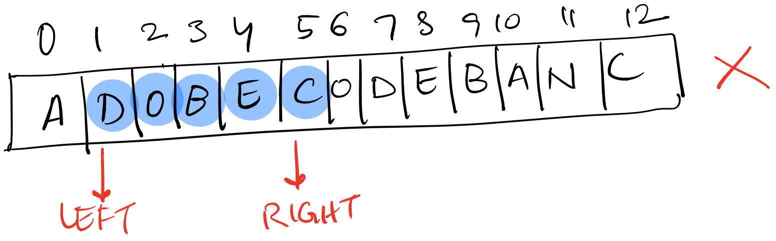
T = "ABC"

Output : BANC

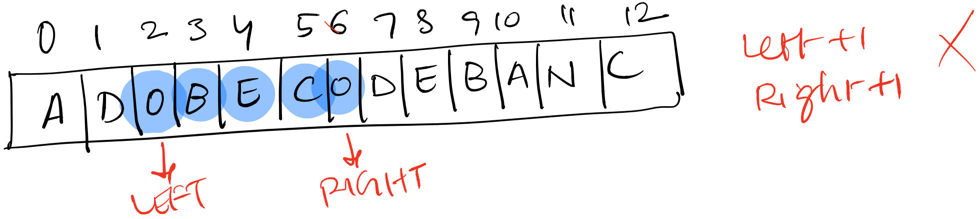


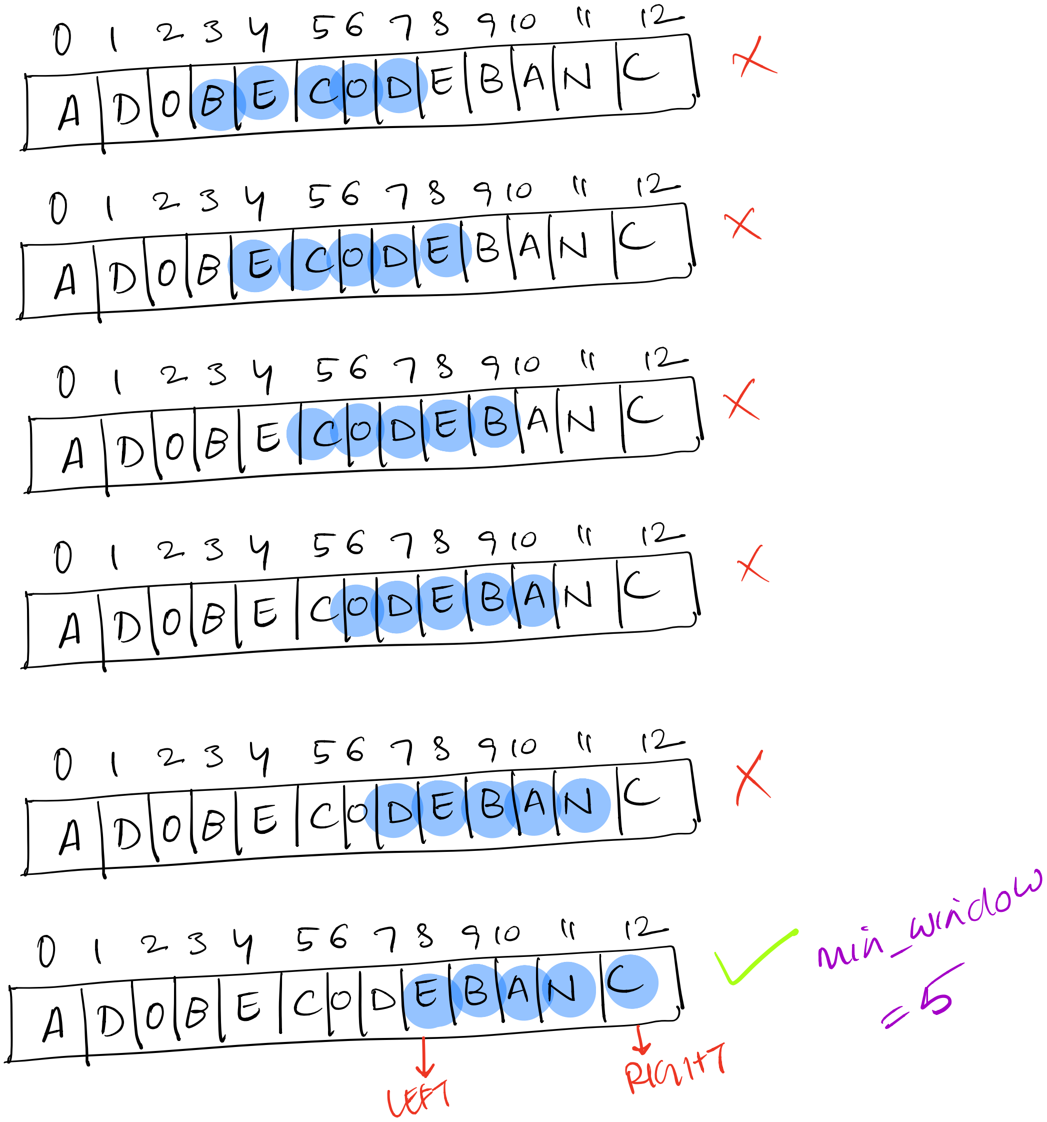


- keep moving Right, until you find all the letters in the substring.
- Then move left pointer more towards right and see if you can have all the characters in the new substring window.

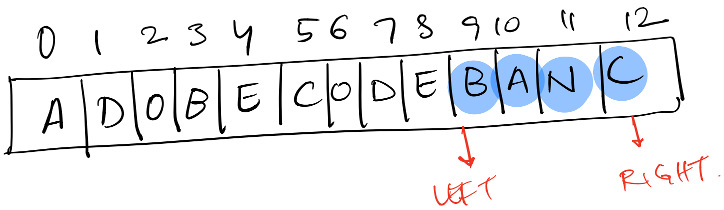


- Again keep moving window, till you find next substring window with all characters

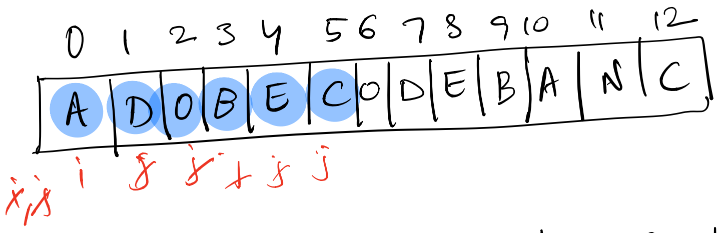




Now move left towards right to reduce the window and see if all the ~~us~~ characters still exists in the new reduced window of substring.



min_window = 4.



t = "ABC"

- First create a t_map with count of characters in t., such that;

  t_map = Counter(t) → { A:1, B:1, C:1 }

- Then filter out characters in s that are in t, so that we don't have to iterate the whole string. And store them as list of tuples as [(ch, idx), (ch, idx) .....] (characters and their index position in s).

filtered_s = [ ]

```
for id, ch   in enumerate(s):
        if ch in t_map:
              filtered_s.append((ch, i))

              left,right
// filtered_s = [(A,0),(B,3),(C,5),(B,9),(A,10),
                                         (C,12)]
   left, right = 0, 0
   formed = 0
   window_counts = { }
   ans = MAX_INT
```

• Look at the characters only in filtered_s
This helps to reduce our search.

```
  while right < len(filtered_s):
        character = filtered_s[right][0]
       # Add character to window with its count
        window_counts[character] = window_counts
                                     .get(character)
                                          + 1

       # check if the char in window count has
       # reached limit
       IF window_counts[character] == t_map[character]:
              formed += 1
```

```
    WHILE left <= right AND Formed == required:
        character = filtered_s[left][0]

        // save the smallest window until now
        start = filtered_s[left][1]
        end = filtered_s[right][1]

        IF end - start < ans :
            ans = end - start

        window_counts[character] -= 1
        IF window_counts[character] < t_map[char]:
            formed -= 1
        left += 1
    right += 1

return ans
```