# Dry Run

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Window (s[L...R])** | **Left** | **Right** | **Window Chars** | **Valid Count** | **Action** |
| 1 | A | 0 | 0 | {A:1} | 1 | Found A ✅ |
| 2 | AD | 0 | 1 | {A:1, D:1} | 1 | D not in t → ignore |
| 3 | ADO | 0 | 2 | {A:1, D:1, O:1} | 1 | O not in t → ignore |
| 4 | ADOB | 0 | 3 | {A:1, B:1} | 2 | Found B ✅ |
| 5 | ADOBE | 0 | 4 | {..., E:1} | 2 | E not in t → ignore |
| 6 | ADOBEC | 0 | 5 | {A:1, B:1, C:1} | 3 | All matched ✅✅✅ — window = 6 |
| 7 | DOBEC | 1 | 5 | {B:1, C:1} | 2 | Removed A ❌ → no longer valid |
| 8-10 | → Expand: CODE | 5 | 8 | +O, +D, +E | 2 | Still not valid |
| 11 | DOBECODEB | 1 | 9 | B:2 | 2 | Extra B → still not valid |
| 12 | DOBECODEBA | 1 | 10 | A:1 | 3 | Valid again ✅ — window = 10 |
| 13-15 | Shrink to BECODEBA | 2 | 10 | Removed D, O | 3 | Still valid |
| 16 | ECODEBA | 4 | 10 | Removed B:1 | 3 | Still valid |
| 17 | CODEBA | 5 | 10 | Removed E | 3 | Still valid |
| 18 | ODEBA | 6 | 10 | Removed C:0 | 2 | Lost C ❌ — stop shrinking |
| 19 | ODEBAN | 6 | 11 | Add N | 2 | Not in t |
| 20 | ODEBANC | 6 | 12 | Add C:1 | 3 | Valid again ✅ → window = 4 ("BANC") ✅ |

# Freq Array:

public class Solution {

    public String minWindow(String s, String t) {

        int[] need = new int[128];

        int[] window = new int[128];

        // Fill the frequency of t

        for (char c : t.toCharArray()) {

            need[c]++;

        }

        int left = 0, right = 0;

        int validCount = 0;

        int minLen = Integer.MAX\_VALUE;

        int start = 0;

        while (right < s.length()) {

            char c = s.charAt(right);

            window[c]++;

            // Count only if char is needed and frequency not exceeded

            if (need[c] > 0 && window[c] <= need[c]) {

                validCount++;

            }

            right++;

            // Try to shrink the window if all t's chars are in the current window

            while (validCount == t.length()) {

                if (right - left < minLen) {

                    start = left;

                    minLen = right - left;

                }

                char d = s.charAt(left);

                if (need[d] > 0 && window[d] <= need[d]) {

                    validCount--;

                }

                window[d]--;

                left++;

            }

        }

        return minLen == Integer.MAX\_VALUE ? "" : s.substring(start, start + minLen);

    }

}

# HashMap

import java.util.HashMap;

public class Solution {

    public String minWindow(String s, String t) {

        if (s.length() < t.length()) return "";

        HashMap<Character, Integer> need = new HashMap<>();

        HashMap<Character, Integer> window = new HashMap<>();

        // Fill need map with characters in t

        for (char ch : t.toCharArray()) {

            need.put(ch, need.getOrDefault(ch, 0) + 1);

        }

        int left = 0, right = 0;

        int valid = 0; // how many chars meet the need

        int start = 0, len = Integer.MAX\_VALUE;

        while (right < s.length()) {

            char ch = s.charAt(right);

            right++;

            if (need.containsKey(ch)) {

                window.put(ch, window.getOrDefault(ch, 0) + 1);

                if (window.get(ch).equals(need.get(ch))) {

                    valid++;

                }

            }

            // Try to shrink window from left

            while (valid == need.size()) {

                // Update minimum length

                if (right - left < len) {

                    start = left;

                    len = right - left;

                }

                char d = s.charAt(left);

                left++;

                if (need.containsKey(d)) {

                    if (window.get(d).equals(need.get(d))) {

                        valid--;

                    }

                    window.put(d, window.get(d) - 1);

                }

            }

        }

        return len == Integer.MAX\_VALUE ? "" : s.substring(start, start + len);

    }

}