```
class Solution {
    public String minWindow(String s, String t) {
        int[] cs = new int[256];
        for (char c : t.toCharArray()) {
            cs[c]--;
        }
        int l = 0, r = 0, left = -1, right = -1, diff = t.length();
        while (r < s.length()) {
            char cr = s.charAt(r++);
            cs[cr]++;
            if (cs[cr] <= 0) {
                diff--;
            }
            while (diff == 0 && l < s.length()) {
                if (right == -1 || right - left > r - l) {
                    left = l;
                    right = r;
                }
                char cl = s.charAt(l++);
                cs[cl]--;
                if (cs[cl] < 0) {
                    diff++;
                }
            }
        }
        return right == -1 ? "" : s.substring(left, right);
    }
}
```

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

Example:

```
Input: S = "ADOBECODEBANC", T = "ABC"
Output: "BANC"
```

Note:

- If there is no such window in S that covers all characters in T, return the empty string `""`.
- If there is such window, you are guaranteed that there will always be only one unique minimum window in S.