# LeetCode

## Description

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.

## Solution

```
class Solution{
public String minWindow(String s, String t) {
    if(s == null || s.length() < t.length() || s.length() == 0){
        return "";
    }
    HashMap<Character,Integer> map = new HashMap<Character,Integer>();
    for(char c : t.toCharArray()){
        if(map.containsKey(c)){
            map.put(c,map.get(c)+1);
        }else{
            map.put(c,1);
        }
    }
    int left = 0;
    int minLeft = 0;
    int minLen = s.length()+1;
    int count = 0;
    for(int right = 0; right < s.length(); right++){
        if(map.containsKey(s.charAt(right))){
            map.put(s.charAt(right),map.get(s.charAt(right))-1);
            if(map.get(s.charAt(right)) >= 0){
                count ++;
            }
            while(count == t.length()){
                if(right-left+1 < minLen){
                    minLeft = left;
                    minLen = right-left+1;
```

```
                }
                if(map.containsKey(s.charAt(left))){
                    map.put(s.charAt(left),map.get(s.charAt(left))+1);
                    if(map.get(s.charAt(left)) > 0){
                        count --;
                    }
                }
                left ++ ;
            }
        }
    }
    if(minLen>s.length())
    {
        return "";
    }

    return s.substring(minLeft,minLeft+minLen);
}
```