入选理由

1. 难度变大了哦，来看看滑动窗口的 hard 题到底 hard 在哪。

标签

- 滑动窗口

难度

- 困难

题目地址(76. 最小覆盖子串)

https://leetcode-cn.com/problems/minimum-window-substring

题目描述

给你一个字符串 S、一个字符串 T 。请你设计一种算法，可以在 O(n) 的时间复杂度内，从字符串 S 里面找出：包含 T 所有字符的最小子串。

示例：

输入：S = "ADOBECODEBANC", T = "ABC"
输出："BANC"

提示：

如果 S 中不存这样的子串，则返回空字符串 ""。
如果 S 中存在这样的子串，我们保证它是唯一的答案。

前置知识

- Sliding Window
- 哈希表

思路

读完该题，是否发现和前一天的题目有些类似呢，前一天的那个说法叫异位词，今天这个直接说包含 T 的所有字符，意思其实是一样的，那不一样的在哪呢?

- 这次的窗口长度并不固定为 T 的长度，实际窗口大小是 ≥T.length 的
- 这次输出的是最小子串，也就是长度最小的子串，因此我们要维护一个 min，代表当前符合要求的子串长度，遇到更短的，则进行更新。

针对上面的，我们开始分析讲义中所讲的滑窗流程的核心三步，在分析之前，再贴一遍简单的伪代码：

```
while 右边界 < 合法条件:
  # 右边界扩张
  window右边界+1
  更新状态信息
  # 左边界收缩
  while 符合收缩条件:
    window左边界+1
    更新状态信息
```

按上边的模版一步步分析：

- 右边界<合法条件：条件自然是 right 不能超过字符串长度
  - 右端 add：将当前字符加入窗口
  - 更新 update：当前加入窗口的字符是否 match 了 T 的字符集，match 了则更新状态（这里需要注意的是，如果当前 match 的字符已经够了，则只更新哈希表中的状态而不更新 match 计数器）
  - 循环左端 delete，目的是尽量缩短窗口大小达到题目最小的要求：符合收缩条件的前提自然是当前已经 match 了 T 的所有字符，然后不断缩小窗口，移除左端字符。
    - 更新 update：当前移除的字符若在 T 字符集中则要更新状态，方式同上。
- 记得记录一下最短子串对应窗口的左右指针方便后续返回结果。

按照上述大致分析流程得到如下代码：

代码

代码支持：Java，CPP

```java
public String minWindow(String s, String t) {

    Map<Character, Integer> map = new HashMap<>();

    int num = t.length();

    for (int i = 0; i < num; i++)
        map.put(t.charAt(i), map.getOrDefault(t.charAt(i), 0) - 1);

    int len = Integer.MAX_VALUE, match = 0, resLeft = 0, resRight = 0;

    int left = 0, right = 0;

    while (right < s.length()) {

        char ch = s.charAt(right);

        if (map.containsKey(ch)) {

            int val = map.get(ch) + 1;
            if (val <= 0)
                match++;
            map.put(ch, val);
        }

        while (match == num) {

            if (right - left + 1 < len) {

                len = right - left + 1;
                resLeft = left;
                resRight = right;
            }

            char c = s.charAt(left);
            if (map.containsKey(c)) {

                int val = map.get(c) - 1;
                if (val < 0)
                    match--;
                map.put(c, val);
            }

            left++;
        }

        right++;
    }

    return len == Integer.MAX_VALUE ? "" : s.substring(resLeft, resRight + 1);
}
```

CPP Code:

```
class Solution {
public:
    string minWindow(string s, string t) {

        vector<int> trecord(256,0);
        vector<int> srecord(256,0);

        for(int i =0; i< t.size(); i++)
        {
            trecord[t[i]]++;
        }

        int left =0;
        int right =0;
        int count =0;
        string ret;
        int retIndex = INT_MAX;
        while(right<s.size())
        {
            int index = s[right] ;
            srecord[index]++;
            if(trecord[index]> 0 && srecord[index]<=trecord[index])
            {
                count++;
            }

            // shink window
            while(count == t.size())
            {
                if((right-left+1)<retIndex)
                {
                    retIndex = right - left+1;
                    ret = s.substr(left, retIndex);
                }
                int leftIndex = s[left] ;
                srecord[leftIndex]--;
                if(srecord[leftIndex]<trecord[leftIndex])
                {
                    count--;
                }
                left++;
            }
            right++;
        }

        return ret;
    }
};
```

Python Code:

```python
class Solution:
    def minWindow(self, s: str, t: str) -> str:
        l, counter, N, ct = 0, Counter(), len(s), Counter(t)
        k = 0
        ret, ans = inf, ""
        for r in range(N):
            counter[s[r]] += 1
            if s[r] in t and counter[s[r]] == ct[s[r]]:
                k += 1
            while k == len(ct):
                if r - l + 1 < ret:
                    ans = s[l:r+1]
                ret = min(r - l + 1, ret)
                counter[s[l]] -= 1
                if s[l] in t and counter[s[l]] == ct[s[l]]-1:
                    k -= 1
                l += 1
        return ans
```

复杂度分析

- 时间复杂度：$O(N + K)$，N 为 S 串长度，K 为 T 串长度

- 空间复杂度：$O(S)$，其中 S 为 T 字符集元素个数

上一页 下一页