

## Minimum Window Substring (/category/84/minimum-window-substring)

/ Here is a 10-line template that can solve most 'substring' problems 📄 (/topic/30941.rss)

New users please **read the instructions** (<https://discuss.leetcode.com/topic/22/welcome-new-users-please-read-this-before-posting>) to *format your code* properly. Discuss is a place to **post interview questions** (/category/5/interview-questions) or share solutions / ask questions related to OJ problems (<https://leetcode.com/problems>).

▲  
410  
▼

Z

● **zjh08177** (/user/zjh08177)  
(/user/zjh08177) Reputation: ★ 547

I will first give the solution then show you the magic template.

**The code of solving this problem is below. It might be the shortest among all solutions provided in Discuss.**

```
string minWindow(string s, string t) {
    vector<int> map(128,0);
    for(auto c: t) map[c]++;
    int counter=t.size(), begin=0, end=0, d=INT_MAX, head=0;
    while(end<s.size()){
        if(map[s[end++]]-->0) counter--; //in t
        while(counter==0){ //valid
            if(end-begin<d) d=end-(head=begin);
            if(map[s[begin++]]++==0) counter++; //make it invalid
        }
    }
    return d==INT_MAX? "":s.substr(head, d);
}
```

**Here comes the template.**

For most substring problem, we are given a string and need to find a substring of it which satisfy some restrictions. A general way is to use a hashmap assisted with two pointers. The template is given below.

```

int findSubstring(string s){
    vector<int> map(128,0);
    int counter; // check whether the substring is valid
    int begin=0, end=0; //two pointers, one point to tail and one head
    int d; //the length of substring

    for() { /* initialize the hash map here */ }

    while(end<s.size()){

        if(map[s[end++]]-- ?){ /* modify counter here */ }

        while(/* counter condition */){

            /* update d here if finding minimum*/

            //increase begin to make it invalid/valid again

            if(map[s[begin++]]++ ?){ /*modify counter here*/ }
        }

        /* update d here if finding maximum*/
    }
    return d;
}

```

One thing needs to be mentioned is that when asked to find maximum substring, we should update maximum after the inner while loop to guarantee that the substring is valid. On the other hand, when asked to find minimum substring, we should update minimum inside the inner while loop.

The code of solving **Longest Substring with At Most Two Distinct Characters** is below:

```

int lengthOfLongestSubstringTwoDistinct(string s) {
    vector<int> map(128, 0);
    int counter=0, begin=0, end=0, d=0;
    while(end<s.size()){
        if(map[s[end++]]++==0) counter++;
        while(counter>2) if(map[s[begin++]]--==1) counter--;
        d=max(d, end-begin);
    }
    return d;
}

```

The code of solving **Longest Substring Without Repeating Characters** is below:

**Update 01.04.2016, thanks @weiyi3 (<https://discuss.leetcode.com/uid/19426>) for advise.**

```

int lengthOfLongestSubstring(string s) {
    vector<int> map(128,0);
    int counter=0, begin=0, end=0, d=0;
    while(end<s.size()){
        if(map[s[end++]]++>0) counter++;
        while(counter>0) if(map[s[begin++]]-->1) counter--;
        d=max(d, end-begin); //while valid, update d
    }
    return d;
}

```

I think this post deserves some upvotes! : )

10 months ago (/post/32626)

**SOLUTION-SHARING** 15.7k (/tags/solution-sharing) **SUBSTRING** 12 (/tags/substring)

[Log in to reply \(/login\)](/login)

<https://leetcode.com/problems/minimum-window-substring>

▲  
0  
▼

W ● **weiyi3 (/user/weiyi3)**  
(/user/weiyi3) Reputation: ★ 5

This post is deleted!

10 months ago (/post/91551)

▲  
1  
▼

W ● **weiyi3 (/user/weiyi3)**  
(/user/weiyi3) Reputation: ★ 5

make the template more applicable for Longest Substring Without Repeating Character

```

public int lengthOfLongestSubstring(String s) {
    if (s == null || s.length() == 0) {
        return 0;
    }
    int len = 0;
    int head = 0, i = 0;
    int[] sTable = new int[256];
    int repeat = 0;
    while (i < s.length()) {
        if (sTable[s.charAt(i++)]++ > 0) repeat++; //total number of repeat
        while(repeat > 0) {
            if (sTable[s.charAt(head++)]-- > 1) repeat--;
        }
        len = Math.max(len, i - head);
    }
    return len;
}

```

9 months ago (/post/92636)

⏏ ⏴ 1 out of 32 ⏵ ⏏

▲

0

▼

Z

● zjh08177 (/user/zjh08177)

(/user/zjh08177) Reputation: ★ 547


This post is deleted!

9 months ago (/post/93234)

▲

0

▼



● clubmaster (/user/clubmaster)

(/user/clubmaster) Reputation: ★ 100

Hi, I think your post is the most excellent solution for this problem! Learned a lot from you. Thank you so much!

9 months ago (/post/93913)

▲

0

▼

Z

● zjh08177 (/user/zjh08177)

(/user/zjh08177) Reputation: ★ 547

Thank you very much. Actually I'm inspired by the solutions here and then find out a general one.

9 months ago (/post/94068)

▲

0

▼

Z

● ZhuEason (/user/zhueason)

(/user/zhueason) Reputation: ★ 53

cool, I love you. Help me a lot, I hope I can get more template from you, haha!

8 months ago (/post/95030)

▲

20

▼

S

● singku (/user/singku)

(/user/singku) Reputation: ★ 32

Though your code is short, but code like

" if(map[s[end++]]++>0) counter++;

while(counter>0) if(map[s[begin++]]-->1) counter--"

is barely readable. I will not recommend this kind of code in reality

8 months ago (/post/95794)

▲

0

▼

S

● stellari (/user/stellari)

(/user/stellari) Reputation: ★ 1,430

Nice post. I don't think 'map' is a good choice for variable names though.

8 months ago (/post/95982)

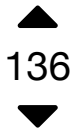


**G**  **gschen (/user/gschen)**  
(/user/gschen) Reputation: ★ 5

Yes, it is hard to read.

Could any one tell me which is one calculated first in "if(map[s[end++]]++>0) counter++;"? How to rewrite it one line by one line? I am very confused the order. Thanks.

8 months ago (/post/96194)



**V**  **vinceyuan (/user/vinceyuan)**  
(/user/vinceyuan) Reputation: ★ 189

Thanks zjh08177 for this great idea. Let me explain this algorithm.

1. **Use** two pointers: **start** and **end** to represent a window.
2. **Move end** to find a valid window.
3. **When** a valid window **is found**, **move start** to find a smaller window.

To check if a window is valid, we use a map to store (char, count) for chars in t. And use counter for the number of chars of t to be found in s. The key part is `m[s[end]]--;`. We decrease count for each char in s. If it does not exist in t, the count will be negative.

To really understand this algorithm, please see my code which is much clearer, because there is no code like `if(map[s[end++]]++>0) counter++;`.

```

string minWindow(string s, string t) {
    unordered_map<char, int> m;
    // Statistic for count of char in t
    for (auto c : t) m[c]++;
    // counter represents the number of chars of t to be found in s.
    size_t start = 0, end = 0, counter = t.size(), minStart = 0, minLen = INT_MAX;
    size_t size = s.size();

    // Move end to find a valid window.
    while (end < size) {
        // If char in s exists in t, decrease counter
        if (m[s[end]] > 0)
            counter--;
        // Decrease m[s[end]]. If char does not exist in t, m[s[end]] will be ne
        m[s[end]]--;
        end++;
        // When we found a valid window, move start to find smaller window.
        while (counter == 0) {
            if (end - start < minLen) {
                minStart = start;
                minLen = end - start;
            }
            m[s[start]]++;
            // When char exists in t, increase counter.
            if (m[s[start]] > 0)
                counter++;
            start++;
        }
    }
    if (minLen != INT_MAX)
        return s.substr(minStart, minLen);
    return "";
}

```

The code above costs 76ms. If we change the first line `unordered_map<char, int> m;` to `vector<int> m(128, 0);`, it costs 12ms.

8 months ago (/post/96557)



1

A



**angrybirdR** (/user/angrybirdR)

(/user/angrybirdR) Reputation: ★ 3

Based on the above code, and save more space, here is what I got:



1 out of 32



```

class Solution {
public:
    string minWindow(string s, string t) {
        unordered_map<char, int> map;
        int count = t.size();
        int begin = 0, end = 0;
        int d = INT_MAX;
        int head = 0;
        for (auto& i: t) map[i]++;
        while (end < s.size()){
            auto foundToPush = map.find(s[end]);
            if (foundToPush != map.end()) if (map[s[end]]-- > 0) count--;
            end++;

            while (count == 0){
                if (end-begin < d) {d = end - begin; head = begin;}
                auto foundToPop = map.find(s[begin]);
                if (foundToPop != map.end()) if (map[s[begin]]++ == 0) count++;
                begin++;
            }
        }
        return d == INT_MAX? "": s.substr(head, d);
    }
};

```

8 months ago (/post/96589)



0



**vinceyuan (/user/vinceyuan)**

(/user/vinceyuan) Reputation: ★ 189



vector does not use too much space, but it is much faster. If I use unordered\_map, it costs 76ms. But if I use vector, it costs only 12ms.

8 months ago (/post/96687)



-9



**9415 (/user/9415)**

(/user/9415) Reputation: ★ -29



<http://www.rudy-yuan.net/archives/185/> (<http://www.rudy-yuan.net/archives/185/>)

7 months ago (/post/100566)



0



**9415 (/user/9415)**

(/user/9415) Reputation: ★ -29



<http://www.rudy-yuan.net/archives/185/> (<http://www.rudy-yuan.net/archives/185/>)

7 months ago (/post/100567)



1 out of 32





9

● **9415 (/user/9415)**  
 (/user/9415) Reputation: ★ -29

<http://www.rudy-yuan.net/archives/185/> (<http://www.rudy-yuan.net/archives/185/>)

7 months ago (/post/100568)



M

● **Matttteo (/user/matttteo)**  
 (/user/matttteo) Reputation: ★ 1

I think it's better to explain the "start" pointer in this way:

We want to find the minimum legal substr which begin at "start" pointer, when the substr is smaller than current minimum ,update the minimum.

I think it is a greedy solution.

6 months ago (/post/102059)



1

M

● **mcaine0 (/user/mcaine0)**  
 (/user/mcaine0) Reputation: ★ 1

Well I learn and understand this solution. However, doesn't this create some suspicion of memorising solutions in interview ? I clearly understand it and probably solve anything similar in extreme speed because of this. Doesn't that makes me someone memorising solutions from the perspective of interviewer ?

6 months ago (/post/102583)



0

Z

● **zhongqixiang (/user/zhongqixiang)**  
 (/user/zhongqixiang) Reputation: ★ 0

I want to point out that although the code "if(map[s[end++]]++>0) counter++" is not as easy to read as writing line by line, the efficiency of this writing style is better than writing line by line. Because writing line by line will cause extra access for hashmap(here vector or array). Actually, if you understand the difference between i++ and ++i, it's not difficult to read the code. Just understand it and get used to it.

6 months ago (/post/102826)



0

B

● **binz (/user/binz)**  
 (/user/binz) Reputation: ★ 1

By far the most impressive explanation I saw in LC.

5 months ago (/post/103607)



0

W

● **wrrwrlyf (/user/wrrwrlyf)**  
 (/user/wrrwrlyf) Reputation: ★ 1

The time complexity is O(n) actually right?

Did I take crazy pills? why does question ask for linear time?

1 out of 32



5 months ago (/post/104650)

---

**SOLUTION-SHARING** 15.7k (/tags/solution-sharing) **SUBSTRING** 12 (/tags/substring)

Log in to reply (/login)

(<https://leetcode.com/problems/minimum-window-substring>)

---