76. Minimum Window Substring

Hard 8720 501 Add to List Share

Given two strings `s` and `t` of lengths `m` and `n` respectively, return *the* ***minimum window substring*** *of* `s` *such that every character in* `t` *(****including duplicates****) is included in the window. If there is no such substring, return the empty string* `""`.

The testcases will be generated such that the answer is **unique**.

A **substring** is a contiguous sequence of characters within the string.

s = a f g m a c a g b r

t = a a b

window

└ aquire & release

si = 4

ei = 8

mc = ~~0 1 2 3~~ 2

j

a	j	g	m	a	c	a	q	b	r
0	1	2	3	4	5	6	7	8	9

i

t = aab

a - 2

b - 1

a - 1 q - 1

r - 1 b - 1

c - 1

Current substring :

j+1 to i

len = i - (j+1) + 1

= i - j.

① aquire until the substring become valid.

② release to be a smaller valid ans, stop this process when substring become invalid.

```
while(i < s.length() - 1) {
    //aquire
    while(i < s.length()-1  && mc < t.length()) {
        i++;

        //aquire ith character
        char ch = s.charAt(i);
        int of = map.getOrDefault(ch,0);
        map.put(ch,of + 1);

        //impact of aquired character on match count
        if(map.getOrDefault(ch,0) <= t_map.getOrDefault(ch,0)) {
            mc++;
        }
    }
    //release
    while(j < i && mc == t.length()) {
        int len = i - j;

        if(len < olen) {
            olen = len;
            si = j + 1;
            ei = i;
        }

        j++;

        //release jth character
        char ch = s.charAt(j);
        if(map.get(ch) == 1) {
            map.remove(ch);
        }
        else {
            int of = map.get(ch);
            map.put(ch,of - 1);
        }

        //impact of released character on match count
        if(map.getOrDefault(ch,0) < t_map.getOrDefault(ch,0)) {
            mc--;
        }
    }
```

t = aab

a-2

b-1

mc = 0 1 2 3 2 1 2 3 2

a f g m a c a q b r a q b a

0 1 2 3 4 5 6 7 8 9 10 11 12 13

i

a-1

q-1

b-1

olen = ~~5~~ 4

si = ~~4~~ 10

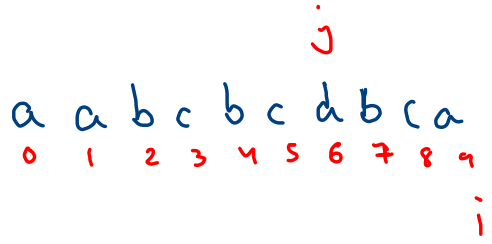
ei = ~~8~~ 13

Smallest distinct window

Medium Accuracy: 50.29% Submissions: 16340 Points: 4

Given a string '**s**'. The task is to find the **smallest** window length that contains all the characters of the given string at least one time.
For eg. A = "**aabcbcdbca**", then the result would be 4 as of the smallest window will be "**dbca**".

```
Input : "AABBBCBBAC"
Output : 3
Explanation : Sub-string -> "BAC"
```



a, b
c, d

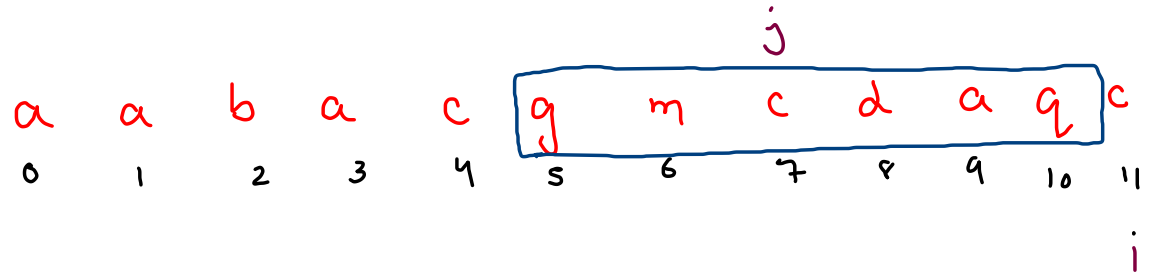
a - 1
b - 1
c - 1

olen = 4

si = 6

ei = 9

3. Longest Substring Without Repeating Characters



answer

① . aquire untill invalid.

② . release to be valid again.

a-1

d-1

c-1

q-1

si = 5

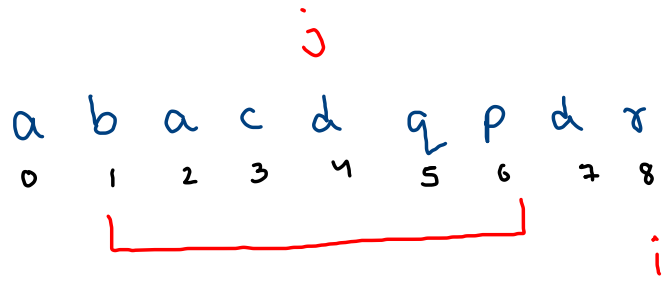
ei = 10

aquire (untill invalid)

release (untill valid)

to be a longer valid

```
while(i <  s.length() - 1) {
    //aquire
    while(i <  s.length() - 1) {
        i++;

        char ch = s.charAt(i);
        int of = map.getOrDefault(ch,0);
        map.put(ch,of + 1);

        if(map.get(ch) > 1) {
            break;
        }
        else {
            int len = i - j;
            olen = Math.max(len,olen);
        }
    }

    //release
    while(j < i) {
        j++;

        char ch = s.charAt(j);

        if(map.get(ch) == 1) {
            map.remove(ch);
        }
        else {
            int of = map.get(ch);
            map.put(ch,of - 1);
            break;
        }
    }
}
```



olen = 0 1 2 3 4 5 6

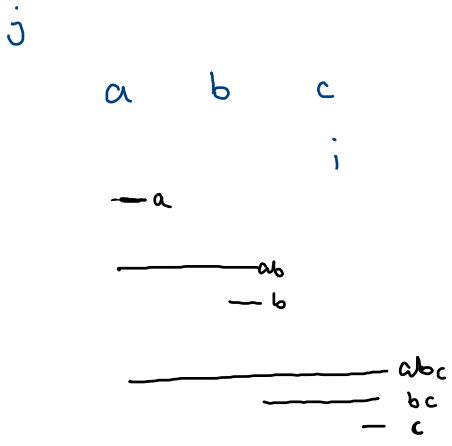
r-1

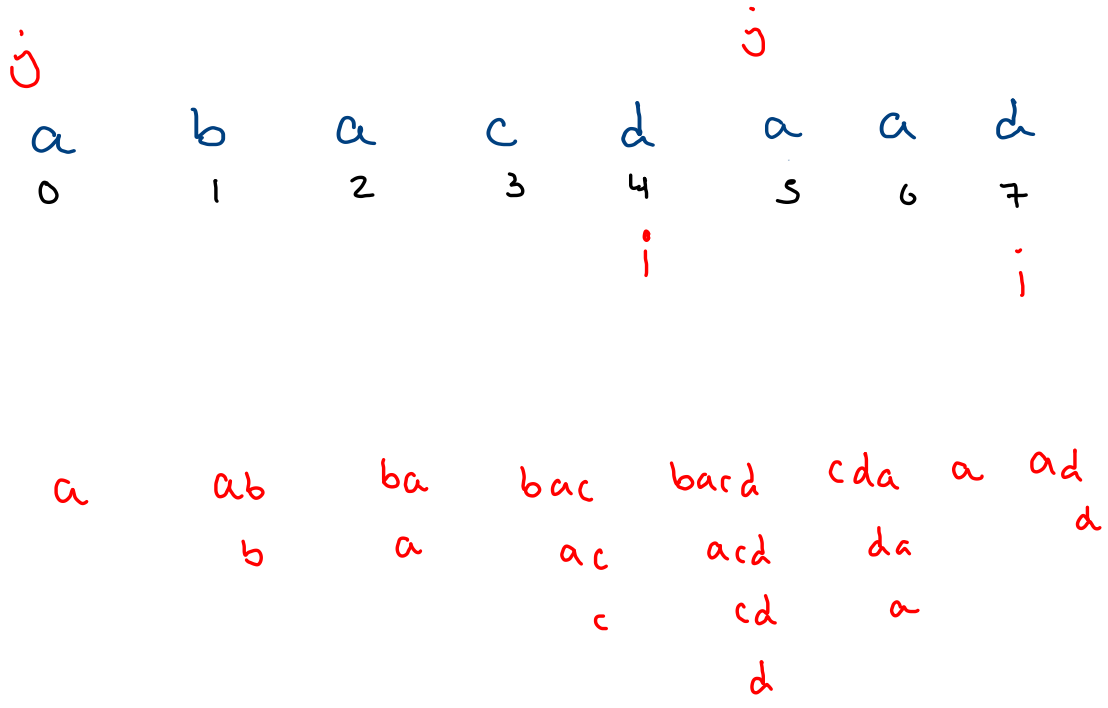
d-1

q-1

p-1

Count Of Substrings Having All Unique Characters





```
while(i < str.length() - 1) {
    //aquire
    while(i < str.length() - 1) {
        i++;

        char ch = str.charAt(i);
        int of = map.getOrDefault(ch,0);
        map.put(ch,of + 1);

        if(map.get(ch) > 1) {
            break;
        }
        else {
            count += (i - j);
        }
    }

    //release
    while(j < i) {
        j++;

        char ch = str.charAt(j);

        if(map.get(ch) == 1) {
            map.remove(ch);
        }
        else {
            int of = map.get(ch);
            map.put(ch,of - 1);
            count += (i - j);
            break;
        }
    }
}
```

count = 1 + 2 + 2 + 3 + 4 + 3 + 1 + 2

a	b	a	c	d	a	a	d
0	1	2	3	4	5	6	7

j at 1; i at 7

a-1

d-1

a | ab | ba | bac | bacd | cda | a | ad

b | a | ac | acd | da | d

c | cd | a

d