

# 003. Longest Substring Without Repeating Characters

## 003 Longest Substring Without Repeating Characters

- Array+Hash Table+String

### Description

Given a string, find the length of the **longest substring** without repeating characters.

#### Examples:

Given "abcabcbb", the answer is "abc", which the length is 3.

Given "bbbb", the answer is "b", with the length of 1.

Given "pwwkew", the answer is "wke", with the length of 3. Note that the answer must be a **substring**, "pwke" is a *subsequence* and not a substring.

### 1. Thought Line

(1) Bit manipulation for duplicate elements.

### 2. Hash Table + String

```
1 class Solution {
2 public:
3     int lengthOfLongestSubstring(string s) {
4         int res = 0;
5         set<char> setHash;
6         int loopStart = 0;
7
8         for (int i=0; !s.empty()&&i<=s.size()-1; ++i){
9             char chr = s[i];
10            if (setHash.find(chr) == setHash.end())
11                setHash.insert(chr);
12            else{
13                // calculate the none duplicate number in this term.
14                res = (setHash.size()-1)>res?setHash.size()-1:res;
15
16                // find out the index of which one led to this
17                // duplication issue.
18                for (int duplicateSpt = loopStart; duplicateSpt<=i-1;
19                    ++duplicateSpt)
20                {
21                    setHash.erase(s[duplicateSpt]);
22                    if (s[duplicateSpt]==chr){
23                        i = i-1;
24                        loopStart = duplicateSpt+1;
25                        break;
26                    }
27                }
28            }
29        }
30        res = (s.size()-loopStart)>res?s.size()-loopStart:res;
31        return res;
32    }
33 }
34 };
```

