拉勾教育
—互联网人实战大学—

哈希表：无重复字符的最长子串

中等/双指针、哈希表

学习目标

拉勾教育
—互联网人实战大学—

了解算法题的解题思路

哈希表的概念

哈希表的应用

题目描述

给定一个字符串，请你找出其中不含有重复字符的最长子串的长度。

输入："abcabcbb"
输出：3
解释：因为无重复字符的最长子串是 "abc"，所以其长度为 3

输入："bbbbb"
输出：1
解释：因为无重复字符的最长子串是 "b"，所以其长度为 1

输入："pwwkew"
输出：3
解释：因为无重复字符的最长子串是 "wke"，所以其长度为 3

一. Comprehend 理解题意

拉勾教育
—互联网人实战大学—

1. 题目主干要求

返回最长子串的长度

子串中无重复字符

子串，而非子序列：

"wke"是子串

"pwke"是子序列

p	w	w	k	e	w

2. 其它细节

测试数据仅包含ASCII码表中的字符

字符串可能为空，或全部由空字符组成

一. Comprehend 理解题意

拉勾教育
—互联网人实战大学—

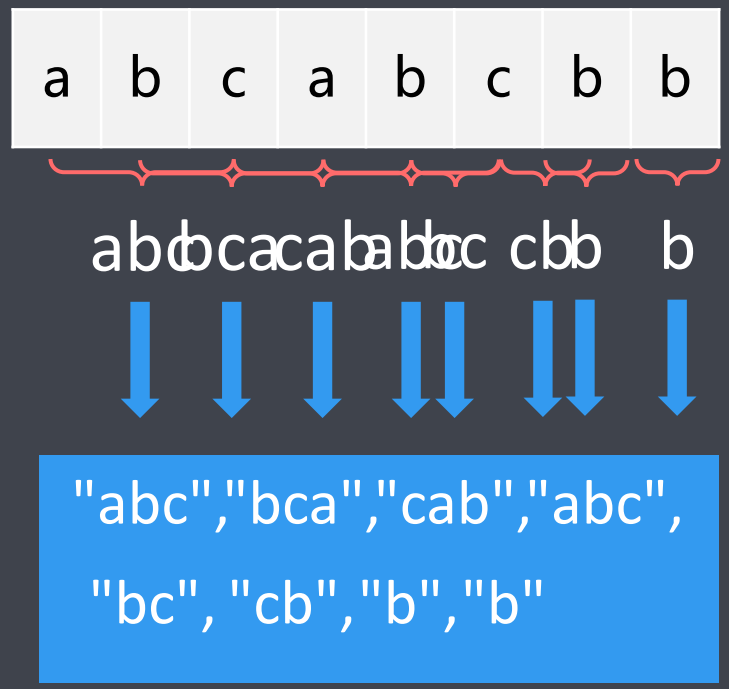
解法一：暴力解法

1. 先找到所有不重复子串，再统计最长子串的长度

   查找子串时，只保留不含重复字符的串

   需要将这些子串临时存储在一个容器中

   使用语言特性（Java）



一. Comprehend 理解题意

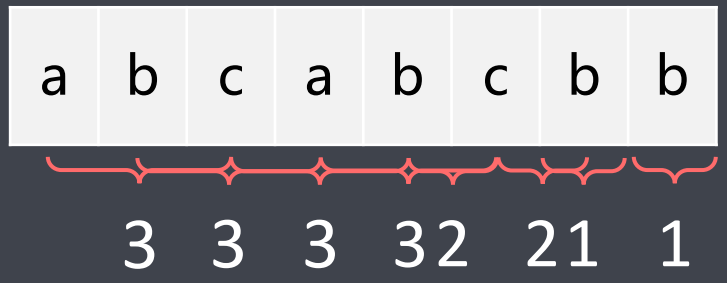
拉勾教育
—互联网人实战大学—

解法二：优化解法

2. 在原字符串上定位并计算子串长度，取最大值

   查找不含重复字符的子串，通过索引计算其长度

   每次计算与上次子串长度对比，只保留最大的数值



maxLength = 3

maxLength = 3

二. Choose 数据结构及算法思维选择

拉勾教育
—互联网人实战大学—

解法一：统计最长子串的长度

- 数据结构：数组/栈/链表/队列+字符串
- 算法思维：遍历+双指针（外层循环start，内层循环end）

a	b	c	a	b	c	b	b

解法二：计算并保留最大子串长度

- 数据结构：字符串（临时子串）
- 算法思维：遍历+双指针

三. Code 基本解法及编码实现

解法一：暴力解法思路分析

1. 生成所有不包含重复字符的子串

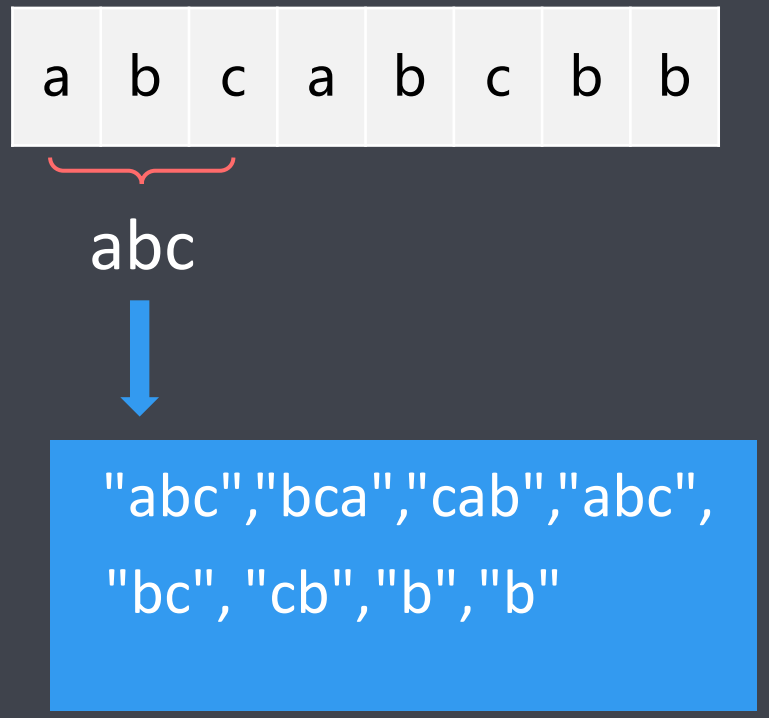
   将所有单字符子串添加到集合（ArrayList）中

   遍历字符串，外层循环为start，内层为end

   截取不含重复字符的子串，添加到集合中

2. 统计最长子串的长度

   遍历集合，统计最大子串长度并返回



三. Code 基本解法及编码实现

解法一：暴力解法边界和细节问题

边界问题

- 遍历字符串的字符，注意索引越界
- 生成子串时，注意子串的起止索引

细节问题

- 子串添加到ArrayList，它会动态扩容

三. Code 基本解法及编码实现

拉勾教育
—互联网人实战大学—

```java
public int lengthOfLongestSubstring(String s) {
    int length;
    if (s == null || (length = s.length()) == 0) return 0;
    // 1.生成所有不包含重复字符的子串
    List<String> list = new ArrayList<>();
    list.addAll(Arrays.asList(s.split(""))); // 单字符，直接添加到集合中
    for (int start = 0; start < length; start++) { // 遍历子串的起始字符
        for (int end = start + 1; end < length; end++) { // 遍历子串的终止字符
            String subStr = s.substring(start, end);
            // 当前字符在前面的子串中已出现，则跳过该字符
            if (subStr.indexOf(s.charAt(end)) != -1) {
                break;
            }
            list.add(s.substring(start, end + 1)); // 否则，添加到集合中
        }
    }
    // 2.统计最长子串的长度
    int maxLength = 1;
    for (String sub : list) {
        int subLen;
        if ((subLen = sub.length()) > maxLength)
            maxLength = subLen;
    }
    return maxLength;
}
```

时间复杂度：O(n^3)

- 将字符串切割成单字符数组：O(n)
- 遍历并截取子串：O(n^3)
- 统计最长子串长度：O(n^2)
- 实际时间消耗巨大

空间复杂度：O(n^2)

- 数组列表：O(n^2)，理论上最多有 n(n + 1) / 2 个子串
- 子串都是常量：O(n^2)
- 子串都是字符串常量，实际空间消耗巨大

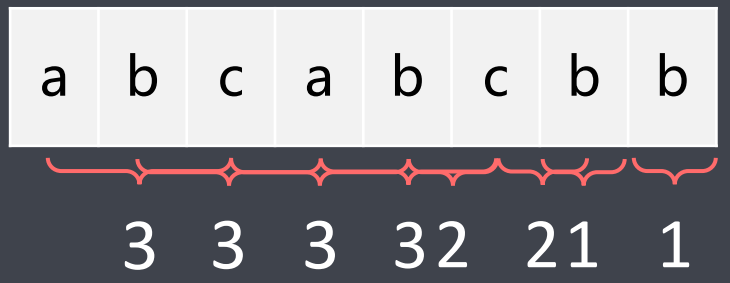
执行耗时:270 ms,击败了6.86% 的Java用户
内存消耗:39.8 MB,击败了5.04% 的Java用户

三. Code 基本解法及编码实现

拉勾教育
—互联网人实战大学—

解法二：优化解法解法思路分析

1. 定义变量maxLength表示最大长度
2. 使用双指针截取不含重复字符的子串
3. 计算子串长度，保留较大值到maxLength



$2 \leq 3$

$maxLength = 3$

三. Code 基本解法及编码实现

拉勾教育
—互联网人实战大学—

```java
public int lengthOfLongestSubstring(String s) {
    int len;
    if (s == null || (len = s.length()) == 0) {
        return 0;
    }

    int maxLength = 1; // 最长子串的长度。默认值1：原字符串有数据，至少是1

    // 1.遍历字符串，生成所有的不含重复字符的子串
    for (int start = 0; start < len; start++) { // 遍历子串的起始字符
        for (int end = start + 1; end < len; end++) { // 遍历子串的终止字符
            String subStr = s.substring(start, end); // 截取当前字符的前置子串
            // 当前字符在前面的子串中已出现，则跳过该字符
            if (subStr.indexOf(s.charAt(end)) != -1) {
                break;
            }
            // 2.统计最长子串的长度
            int subLen = end + 1 - start; // 子串长度
            if (subLen > maxLength)
                maxLength = subLen;
        }
    }
    return maxLength;
}
```

时间复杂度：O(n^3)

- 遍历并截取子串：O(n^3)
- 实际时间消耗巨大

空间复杂度：O(n^2)

- ~~数组列表：O(n^2)，理论上最多有 n(n + 1) / 2 个子串~~
- 子串都是常量：O(n^2)
- 子串都是字符串常量，实际空间消耗巨大

执行耗时:260 ms,击败了7.06% 的Java用户
内存消耗:39.4 MB,击败了6.54% 的Java用户

四. Consider 思考更优解

拉勾教育
—互联网人实战大学—

1. 剔除无效代码或优化空间消耗
   - 能否不存储子串？
   - 能否避免生成字符串常量？
2. 寻找更好的算法思维
   - 能否只扫描一遍字符串？
   - 定位子串并检查重复字符的过程比较耗时，能否优化？
   - 参考其它算法

五. Code 最优解思路及编码实现

关键知识点：哈希表与哈希算法

- Hash table：哈希表，也叫散列表

  把关键码值映射到表中的一个位置，以加快查找速度

- Hash算法

  散列值：把任意长度的输入通过算法变成固定长度的输出

  是一种压缩映射，直接取余操作

  哈希冲突的解决：开放寻址；再散列；链地址法；

- 位运算

  & | ~ ^ << >> >>>

  取模操作： a % (Math.pow(2,n))　　33 % 16 = 1

  等价于：a&(Math.pow(2,n)-1)　　33 & 15 = 1

```
// 1.定义哈希表，仅支持ASCII码表字符
char[] chs = new char[128];

int hash(char key) {
    return key; // 不会冲突/碰撞
}
```

…	…	a	b	c	…
…	…	97	98	99	…

五. Code 最优解思路及编码实现

最优解：哈希表 + 双指针解法

1. 定义哈希表，临时存储子串字符和查重

   定义哈希函数，对任意字符生成唯一整数值

2. 遍历字符串，通过双指针循环定位子串

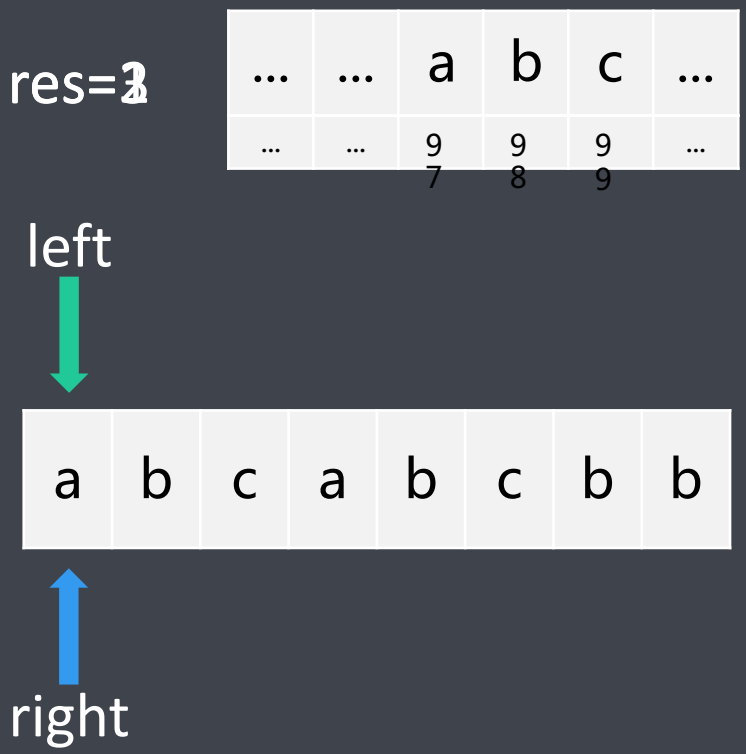
   右指针在哈希表中是否存在：

   否，记录到哈希表，移动右指针，计算长度；

   是，删除哈希表中左指针元素，移动左指针，

   重复检查右指针元素是否还存在；

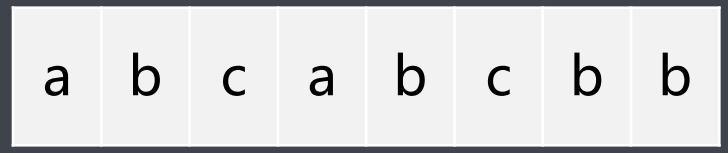
3. 每次计算子串长度，比较并保留最大值



五. Code 最优解思路及编码实现

最优解：边界和细节问题

边界问题



- 遍历字符串的字符，注意索引越界
- 计算子串长度时，注意子串的起止索引
- 根据测试用例，子串长度不会超过哈希表容量：new char[128]

细节问题

- 子串长度是：end + 1 - start
- 出现重复元素后，左指针逐个移动，直到与当前重复的字符索引+1

五. Code 最优解思路及编码实现


```java
public int lengthOfLongestSubstring(String s) {
    int res = 0, left = 0, right = 0, len = s.length();
    // 1.定义哈希表，支持ASCII码表的全部字符
    char[] chs = new char[128];
    // 2.遍历字符串的所有字符
    while (right < len) { // 右指针后移，不超过源字符串长度
        char rightChar = s.charAt(right); // 右指针字符
        char c = chs[(chs.length - 1) & hash(rightChar)]; // hash算法计算索引
        if (rightChar != c) { // 未重复出现
            // 2.1.记录到哈希表，移动右指针，计算长度
            char v = s.charAt(right++);
            // 将不重复字符记录到哈希表中
            chs[(chs.length - 1) & hash(v)] = v;
            // 3.每次记录子串长度，并计算最大值
            int size = right - left; // 每个不重复子串的长度
            res = res > size ? res : size; // 取较大值
        } else { // 重复出现
            // 2.2.删除左指针元素，移动左指针。重复检查右指针元素是否还存在
            char leftChar = s.charAt(left++);
            chs[(chs.length - 1) & hash(leftChar)] = '\u0000';
        }
    }
    return res;
}
```

按位与：取模

时间复杂度：O(n)

- 遍历字符串：O(n)
- 定位重复字符：O(1)

空间复杂度：O(1)

- 哈希表占用固定空间：O(1)
- 双指针：O(1)

执行耗时:4 ms,击败了90.12% 的Java用户
内存消耗:38.8 MB,击败了84.17% 的Java用户

五. Code 最优解思路及编码实现

最优解：哈希表 + 双指针优化解法

1. 哈希表作用变形

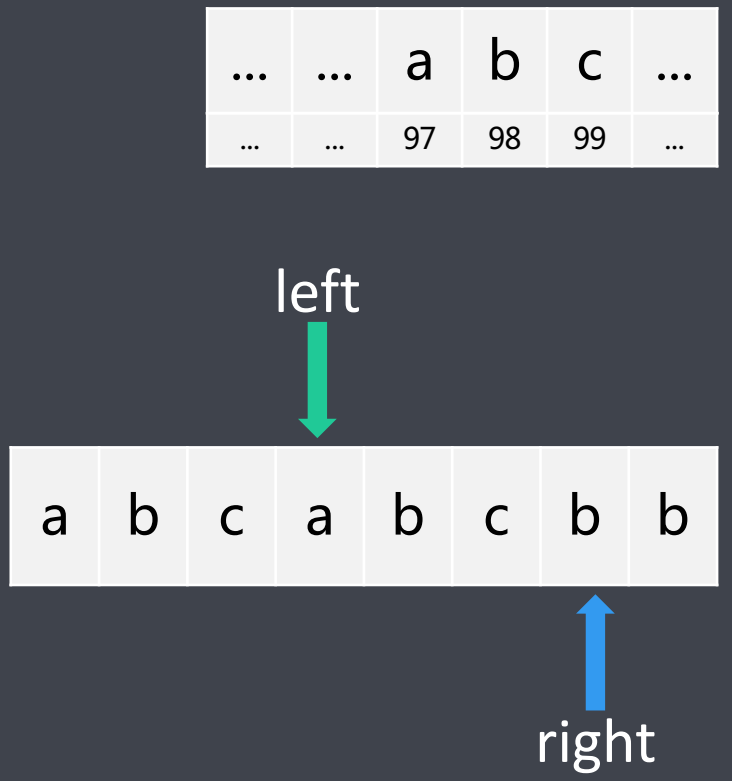
   字符ASCII码值 --> 字符

   字符ASCII码值 --> 字符最后出现索引

2. 遇到重复元素后，左指针移动优化

   逐个移动到前一个相同字符出现后的位置 -->

   一次性定位到前一个相同字符出现后的位置



五. Code 最优解思路及编码实现

最优解：哈希表 + 双指针优化解法

1. 初始化哈希表，存入非ASCII码值作为默认值
2. 遍历字符串，使用双指针定位子串索引

   字符已出现：取出哈希表中记录，左指针到记录+1

   无论是否出现，将右指针记录到哈希表
3. 每次移动都记录子串长度，保留最大值

```java
// 1.哈希表中填充 -1 作为默认值:
int[] arr = new int[128];
for (int i = 0; i < arr.length; i++)
    arr[i] = -1;
```

五. Code 最优解思路及编码实现

拉勾教育
—互联网人实战大学—

```java
public int lengthOfLongestSubstring(String s) {
    int res = 0,            // 最长子串的计算结果
        left = 0,           // 子串起始索引
        right = 0,          // 子串结束索引
        len = s.length();   // 字符串长度
    // 1.哈希表中填充ASCII码表不包含的数值作为默认值：-1
    int[] arr = new int[128];
    for (int i = 0; i < arr.length; i++)
        arr[i] = -1;
    // 2.遍历字符串的所有字符
    while (right < len) {
        int c = s.charAt(right);
        if (arr[c] != -1) { // 检测该字符是否已出现：已出现
            // 出现，则移动左指针，直接定位到上次出现的下一个索引
            int start0 = arr[c] + 1;
            // 2.1.使用双指针定位子串索引：左指针直接定位
            left = left >= start0 ? left : start0; // 只往右不往左
        }
        arr[c] = right; // 无论是否重复，记录该字符最后一次出现的索引
        // 3.计算子串长度，记录最大值：右索引+1 - 左索引
        int size = right + 1 - left;
        res = res > size ? res : size;
        // 2.2.使用双指针定位子串索引：右指针始终自增
        right++;
    }
    return res;
}
```

时间复杂度：O(n)

- 遍历字符串：O(n)
- 定位重复字符：O(1)

空间复杂度：O(1)

- 哈希表占用固定空间：O(1)
- 双指针：O(1)

执行耗时:2 ms,击败了100% 的Java用户
内存消耗:38.7 MB,击败了96.69% 的Java用户

六. Change 变形延伸

拉勾教育
—互联网人实战大学—

题目变形

- （练习）使用Set集合改进暴力解法
- （练习）使用Map集合实现哈希表解法

延伸扩展

- 合理的使用双指针能将时间复杂度从O(n^2)降低到O(n)级别
- 哈希表应用广泛，是非常重要的数据结构（比如HashMap）

本题来源

- Leetcode 3 https://leetcode-cn.com/problems/longest-substring-without-repeating-characters/

总结

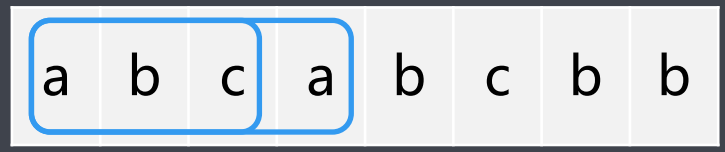
拉勾教育
—互联网人实战大学—

6C解题法

哈希表与哈希算法

- 把关键码值映射到表中的一个位置，以加快查找速度
- 散列值：把任意长度的输入通过算法变成固定长度的输出。是一种压缩映射

滑动窗口



- 用来解决查找满足一定条件的连续空间问题
- “请找到满足xx的最x的区间（子串、子数组）的xx”这类问题

课后练习

1. K 个不同整数的子数组(Leetcode 992/困难)
2. 至多包含两个不同字符的最长子串(Leetcode 159/中等)
3. 至多包含 K 个不同字符的最长子串(Leetcode 340/困难)
4. 字母异位词分组（Leetcode 49 /中等）

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容