BAT算法面试题(四)-无重复字符的最长子串(3)

> 文章出自 Hello Code 开发者学习平台 CC老师
> 获取更新文章/视频 关注公众号:
> **HelloCode开发者学习平台**
>
> 
>
>
> 上一次分享的是滑动窗口解决方法.执行的次数2N个步骤.但是是否还有办法优化了? 答案是肯定的!

一.算法题

- **题目**

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

- **Example**

  > - Given "abcabcbb", the answer is "abc", which the length is 3.
  > - Given "bbbbb", the answer is "b", with the length of 1.
  > - Given "pwwkew", the answer is "wke", with the length of
  > - Note that the answer must be a substring, "pwke" is a subsequence and not a substring.

二.算法题解读

- **题目大意**:给定一个字符串,找出不含有重复字符的最长子串的长度
- **解读Example**

  > - 给定"abcabcbb",没有重复字符的最长子串是"abc",那么长度就是3
  > - 给定"bbbbb",最长子串就是"b",长度就是1
  > - 给定pwwkew,最长子串就是"wke",长度为3,
  > - 注意,必须是一个子串."pwke",是子序列,而不是子串

三.优化"滑动窗口"解决思路

到底如何在滑动窗口方法上优化了? 实际上我们可以如果采用进一步优化,可以达到只需要N次即可计算成功.我们可以定义字符到索引映射.而不是使用集合来判断这个字符的存在与否.当遇到重复的字符时,我们即可跳过该滑动窗口.

也可以理解为,如果`s[j]`在`[i,j)`的范围内有与`j'`重复的字符.我们不需要逐渐增加`i`.而是直接跳过`[i,j']`范围内的所有元素.并将`i`变成为`j'+1`就可以做到.

四.代码实现

java code

```
public class Solution {
    public int lengthOfLongestSubstring(String s) {
        int n = s.length(), ans = 0;
       //获取当前字符索引
        Map<Character, Integer> map = new HashMap<>();
        //修改[i,j]的范围
         for (int j = 0, i = 0; j < n; j++) {
            if (map.containsKey(s.charAt(j))) {
                i = Math.max(map.get(s.charAt(j)), i);
            }
            ans = Math.max(ans, j - i + 1);
            map.put(s.charAt(j), j + 1);
        }
        return ans;
    }
}
```

五.使用ASCII 128码 思路

字符串,其实由字符构成.而字符则可以用ASC码来替代.如此,我们可以用整数数组作为直接访问表来替换Map.

常用表如下:

`int [26],用于表示字母 "a" - "z" 或 "A" - "Z"`;

`int [128],用于表示ASCII码`

`int [256],用于表示扩展ASCII码`

`A = 65, a = 97`

ASCII 码		字符	ASCII 码		字符	ASCII 码		字符	ASCII 码		字符
十进位	十六进位		十进位	十六进位		十进位	十六进位		十进位	十六进位	
032	20		056	38	8	080	50	P	104	68	h
033	21	!	057	39	9	081	51	Q	105	69	i
034	22	"	058	3A	:	082	52	R	106	6A	j
035	23	#	059	3B	;	083	53	S	107	6B	k
036	24	$	060	3C	<	084	54	T	108	6C	l
037	25	%	061	3D	=	085	55	U	109	6D	m
038	26	&	062	3E	>	086	56	V	110	6E	n
039	27	'	063	3F	?	087	57	W	111	6F	o
040	28	(	064	40	@	088	58	X	112	70	p
041	29	)	065	41	A	089	59	Y	113	71	q
042	2A	*	066	42	B	090	5A	Z	114	72	r
043	2B	+	067	43	C	091	5B	[	115	73	s
044	2C	,	068	44	D	092	5C	\	116	74	t
045	2D	-	069	45	E	093	5D	]	117	75	u
046	2E	.	070	46	F	094	5E	^	118	76	v
047	2F	/	071	47	G	095	5F	_	119	77	w
048	30	0	072	48	H	096	60	`	120	78	x
049	31	1	073	49	I	097	61	a	121	79	y
050	32	2	074	4A	J	098	62	b	122	7A	z
051	33	3	075	4B	K	099	63	c	123	7B	{
052	34	4	076	4C	L	100	64	d	124	7C	|
053	35	5	077	4D	M	101	65	e	125	7D	}
054	36	6	078	4E	N	102	66	f	126	7E	~
055	37	7	079	4F	O	103	67	g	127	7F	⌂

六.代码实现

java code

```
public class Solution {
```

```java
    public int lengthOfLongestSubstring(String s) {
        int n = s.length(), ans = 0;
        int[] index = new int[128];
        for (int j = 0, i = 0; j < n; j++) {
            i = Math.max(index[s.charAt(j)], i);
            ans = Math.max(ans, j - i + 1);
            index[s.charAt(j)] = j + 1;
        }
        return ans;
    }
}
```