3. Longest Substring Without Repeating Characters

Given a string, find the length of the **longest substring** without repeating characters. 给定一个字符串，找到最长的没有重复元素的子字符串，并返回长度。

Example:

```
Input: "pwwkew"
Output: 3
Explanation: 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.
```

解法一:

—— python

用变量max_len存储最大长度值。用max_tmp作比较。用了两个for循环列了所有的情况，用if判断有没有重复的字符。重复的情况就跳出单次循环。

```python
class Solution:
    def lengthOfLongestSubstring(self, s):
        if len(s) == 1:
            return 1
        max_all = 0
        for i in range(len(s)):
            sub_s = str(s[i])
            max_tmp = 1
            for j in range(i + 1,len(s)):
                if s[j] not in sub_s:
                    sub_s += str(s[j])
                    max_tmp = max_tmp + 1
                else:
                    #restart counting, variables needs to be initialized

                    break
            if max_all < max_tmp:
                max_all = max_tmp
        return max_all
```

好笨的方法，还踩了好多坑。

解法二:

—— python

优化1:判断是否在字符串，可以用数组存储，减少时间复杂度。

一旦重复字符出现，马上更新起点的位置。

```
class Solution {
    public int lengthOfLongestSubstring(String s) {
        int n = s.length(), max_len = 0;
        int[] index = new int[128];
        for (int i = 0, j = 0; j < n; j++){
            i = Math.max(index[s.charAt(j)],i); // charAt 的函数，返回指定 index 的
char 值
            max_len = Math.max(max_len, j - i + 1);
            index[s.charAt(j)] =  j + 1; //下标 + 1,  代表 i 将要移动下一个位置。自己肯
定想不到。。
        }
        return max_len;
    }
}
```

路漫漫。。。