```
public class Solution {
    public int lengthOfLongestSubstring(String s) {
        if (s.length()==0) return 0;
        HashMap<Character, Integer> map = new HashMap<Character, Integer>();
        int len = 0;

        for (int i = 0, j = 0; i < s.length(); i++){
            if (map.containsKey(s.charAt(i))){
                j = Math.max(j, map.get(s.charAt(i))+1);
            }
            len = Math.max(len, i - j + 1);
            map.put(s.charAt(i), i);
        }
        return len;
    }
}

/*
the basic idea is, keep a hashmap which stores the characters in string as keys and their
positions as values,
and keep two pointers which define the max substring. move the right pointer to scan through
the string ,
and meanwhile update the hashmap. If the character is already in the hashmap,
then move the left pointer to the right of the same character last found.
Note that the two pointers can only move forward.
*/

/*
public class Solution {
    public int lengthOfLongestSubstring(String s) {
        int len = s.length(), left = 0, right = 0, max = 0;
        int[] chars = new int[256];

        while (right <= len){
            if (noRepeat(chars)) {
                max = Math.max(max, right-left);
                if (right == len) break;
                chars[s.charAt(right++)]++;
            } else {
                chars[s.charAt(left++)]--;
            }
        }

        return max;
    }

    public boolean noRepeat(int[] chars){
        for (int i : chars){
            if (i != 0 && i != 1) return false;
        }
        return true;
    }
}
*/
```