Longest Substring without repeating characters

Question:

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

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

Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Example 3:

```
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a
subsequence and not a substring.
```

Approach 1:

I initially tried Kadane's algorithm to find the unique substring and it worked well but failed at some test cases.

Ex: ***dvdf***

Kadane gave the output as ***2(df)*** whereas the correct output is ***3(vdf)***.

The problem here is that The previous max was ***dv*** and then ***dvd*** had repetitions so the if condition became **True** and the set as well count get reset.

After resetting, the new count starts from *d* while according to the correct output, it should start from ***v***.

Solution 1:

https://gist.github.com/vermaayush680/f8c5ecd2155f016675451a66e55fea7c

```
w=set()
c=0
m=0
for i in s:
        if i in w:
                m=max(m,c)
                c=0
                w=set(i)
        c+=1
        w.add(i)
m=max(m,c)
return m
```

Time Complexity: O(n)

Space Complexity: O(n)

Approach 2:

Tried sliding window to overcome the previous problem.

Used a stack to store the elements. Popped the first element and checked if it is repeating. If so, we reset everything.

Ex: In the previous approach, it failed at ***dvdf***.

In **sliding window**, After the ***dv*** step, we pop *d* and compare it with the current letter which is ***d*** at index **2**.

Since both are the same, we repeat steps again with *v* still present in the queue.

This ensures that the previous unique characters are still present even after reset and we get the optimal answer.

Solution 2:

https://gist.github.com/vermaayush680/839822a4c7212f005b06630b00fd5bab

```
w=set()
q=collections.deque([])
m=0
for i in s:
        if i in w:
                while q:
                        prev = q.popleft()
                        w.remove(prev)
                        if prev==i:
                                break
        q.append(i)
        w.add(i)
        m=max(m,len(w))
return m
```

Time Complexity: O(n)

Space Complexity: O(n)