

是的，你对滑动窗口的理解很准确。滑动窗口是一种在线性数据结构（如数组、字符串）中，用来优化计算的技术，特别适合于需要多次重复计算的情境。通过维护一个动态调整的窗口（即一部分数据），可以避免重复计算，从而提升效率。

滑动窗口通常分为两种：

1. **固定长度窗口**：窗口大小不变，用于处理固定范围的连续数据。例如，求一个固定长度子数组的最大或最小值。
2. **可变长度窗口**：窗口大小根据条件动态调整，用于满足特定条件的最大或最小长度子数组或子字符串。例如，最长不含重复字符的子串、允许最多  $k$  个 0 的最长子数组等。

每次滑动窗口右移时，通过更新窗口的边界（通常是 `left` 和 `right` 指针）来维护窗口内的状态，从而避免重复计算整个窗口内的数据。这种方法可以将复杂度从平方级别降到线性级别，是一种非常有效的优化方式。

### 滑动窗口题目：3. 无重复字符的最长子串

Handwritten diagram illustrating the sliding window algorithm for finding the longest substring without repeating characters. The diagram shows a sequence of characters `[d e a b c]` followed by `a b c a`. A `left` pointer points to the `a` in the first group, and a `right` pointer points to the first `a` in the second group. Below the diagram, text explains the process:

滑动窗口, `right` 找到重复元素(hash), `right` 停.  
`left++`, 并且 `erase[left]`直到无重复元素.

```
1 class Solution {
2 public:
3     int lengthOfLongestSubstring(string s) {
4         if(s == "") return 0;
5         int n = s.size();
6         unordered_set<char> hash;
7         int left = 0;
8         int right = 0;
9         int len = 1;
10        for(;right < n;right++)
11        {
12            while (hash.count(s[right])) {
13                hash.erase(s[left]);
14                left++;
15            }
16            hash.insert(s[right]);
17            len = max(len, right - left + 1);
18        }
19        return len;
20    }
21 };
```

## 1004. 最大连续1的个数 III

```
1 class Solution {
2 public:
3     int longestOnes(vector<int>& nums, int k) {
4         int n = nums.size();
5         int left = 0;
6         int right = 0;
7         int len = 0;
8         int zero = 0;
9         for(;right < n;right++) //right++ 进窗口
10        {
11            //判断是否 = 0, 等于0计数器zero++;
12            if(nums[right] == 0) zero++;
13            //判断条件
14            while(zero > k)
15            {
16                if(nums[left] == 0)
17                {
18                    zero--;
19                }
20                left++;
21            }
22            //更新结果
23            len = max(len,right-left+1);
24        }
25        return len;
26    }
27 }
28 };
```