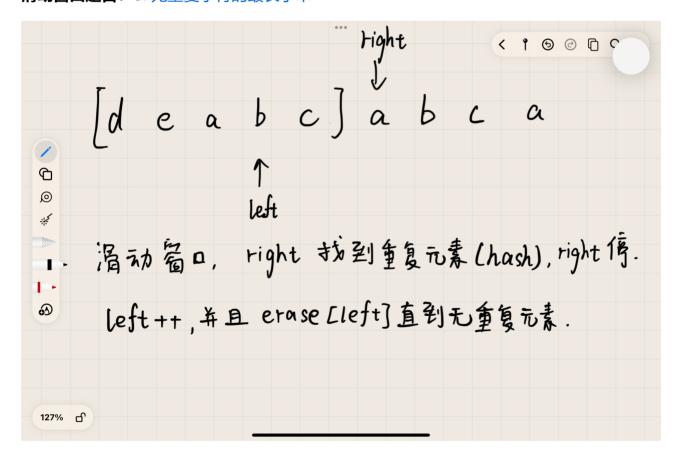
是的,你对滑动窗口的理解很准确。滑动窗口是一种在**线性数据结构**(如数组、字符串)中,用来优化计算的技术,特别适合于需要多次重复计算的情境。通过维护一个动态调整的窗口(即一部分数据),可以避免重复计算,从而提升效率。

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



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

1004. 最大连续1的个数 Ⅲ

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