双指针算法之同向双指针(下)

主讲人 令狐冲
课程版本 v7.0

滑动窗口求和

https://www.lintcode.com/problem/window-sum/description

nums = [1,2,7,8,5], k = 3

求出数组中所有连续 k 个数之和

返回 [10,17,20]

套用模板, 唯一不同是 j = 0 作为初始


```python
def winSum(self, nums, k):
    if not nums or len(nums) < k:
        return []

    result = []
    j, window_sum = 0, 0
    for i in range(len(nums)):
        while j < len(nums) and j - i < k:
            window_sum += nums[j]
            j += 1
        if j - i == k:
            result.append(window_sum)
        window_sum -= nums[i]

    return result
```

```java
public int[] winSum(int[] nums, int k) {
    if (nums == null || nums.length < k) {
        return new int[]{};
    }
    if (k == 0) {
        return new int[nums.length];
    }

    int[] results = new int[nums.length - k + 1];
    int j = 0, sum = 0;
    for (int i = 0; i < nums.length; i++) {
        while (j < nums.length && j - i < k) {
            sum += nums[j];
            j++;
        }
        if (j - i == k) {
            results[i] = sum;
        }
        sum -= nums[i];
    }

    return results;
}
```

k次替换后的最长重复字符

https://www.lintcode.com/problem/longest-repeating-character-replacement/

允许替换字符串中的字符 K 次

问替换之后，最长的重复字符有多长

s = ABABA, k = 2

替换 B 到 A 可以获得长度为 5 的全A子串

万能的模板


```python
def characterReplacement(self, s, k):
    counter = {}
    answer = 0
    j = 0
    max_freq = 0
    for i in range(len(s)):
        while j < len(s) and j - i - max_freq <= k:
            counter[s[j]] = counter.get(s[j], 0) + 1
            max_freq = max(max_freq, counter[s[j]])
            j += 1

        # update answer
        if j - i - max_freq > k:
            answer = max(answer, j - 1 - i)
        else:
            answer = max(answer, j - i)

        # update max_freq
        counter[s[i]] -= 1
        max_freq = max(counter.values())
    return answer
```

```java
public int characterReplacement(String s, int k) {
    if (s == null) {
        return 0;
    }

    int j = 0, answer = 0, maxFreq = 0, count;
    HashMap<Character, Integer> counter = new HashMap<>();

    for (int i = 0; i < s.length(); i++) {
        while (j < s.length() && j - i - maxFreq <= k) {
            count = counter.getOrDefault(s.charAt(j), 0) + 1;
            counter.put(s.charAt(j), count);
            maxFreq = Math.max(maxFreq, count);
            j++;
        }
        if (j - i - maxFreq > k) {
            answer = Math.max(answer, j - i - 1);
        } else {
            answer = Math.max(answer, j - i);
        }

        count = counter.get(s.charAt(i)) - 1;
        counter.put(s.charAt(i), count);
        maxFreq = getMaxFreq(counter);
    }

    return answer;
}
```

数组和字符串上的同向双指针总结

数组和字符串的问题有很多题都是和双指针，特别是同向双指针有关

通常问题让我们求是一段子数组或者子字符串

所以遇到“子数组 SubArray” 和 “子字符串 SubString” 就需要往同向双指针思考