Two Pointers + Sliding Window Handbook (Arrays + Strings)

Arrays Q1: Maximum Sum Subarray of Size K

Problem: Given an array and an integer k, find the maximum sum of any contiguous subarray of size k. Input: arr = [2,1,5,1,3,2], k=3 Output: 9 (subarray [5,1,3])

```
function maxSumSubarray(arr, k) {
    let windowStart = 0, sum = 0, maxSum = -Infinity;
    for (let windowEnd = 0; windowEnd < arr.length; windowEnd++) {
        sum += arr[windowEnd];
        if (windowEnd - windowStart + 1 === k) {
            maxSum = Math.max(maxSum, sum);
            sum -= arr[windowStart];
            windowStart++;
        }
    }
    return maxSum;
}
```

Arrays Q2: Longest Subarray with Sum ≤ K

Problem: Find the longest subarray length such that its sum ≤ K. Input: arr = [1,2,1,0,1,1,0], k=4 Output: 5

```
function longestSubarraySumK(arr, k) {
    let windowStart = 0, sum = 0, maxLen = 0;
    for (let windowEnd = 0; windowEnd < arr.length; windowEnd++) {
        sum += arr[windowEnd];
        while (sum > k) {
            sum -= arr[windowStart];
            windowStart++;
        }
        maxLen = Math.max(maxLen, windowEnd - windowStart + 1);
    }
    return maxLen;
}
```

Arrays Q3: Longest Subarray Without Forbidden Value

Problem: Find the longest continuous segment without the forbidden value. Input: arr=[1,3,5,3,6,7,3,5,6], forbidden=3 Output: 2

```
function longestSafeSegment(arr, forbidden) {
    let windowStart = 0, maxLen = 0;
    for (let windowEnd = 0; windowEnd < arr.length; windowEnd++) {
        if (arr[windowEnd] === forbidden) {
            windowStart = windowEnd + 1;
        }
        maxLen = Math.max(maxLen, windowEnd - windowStart + 1);
    }
    return maxLen;
}
```

Arrays Q4: Sliding Window Maximum

Problem: Return the maximum for every window of size k. Input: arr=[1,3,-1,-3,5,3,6,7], k=3 Output: [3,3,5,5,6,7]

```
function maxSlidingWindow(nums, k) {
    let deque = [], result = [];
    for (let windowEnd = 0; windowEnd < nums.length; windowEnd++) {
        while (deque.length && deque[0] <= windowEnd - k) deque.shift();
        while (deque.length && nums[deque[deque.length - 1]] < nums[windowEnd]) deque.pop();
```

```
            deque.push(windowEnd);
            if (windowEnd >= k - 1) result.push(nums[deque[0]]);
        }
        return result;
    }
```

Arrays Q5: Smallest Subarray with Sum ≥ K

Problem: Find the length of the smallest contiguous subarray with sum ≥ K. Input: arr=[2,3,1,2,4,3], target=7 Output: 2 ([4,3])

```
    function minSubarrayLen(target, arr) {
        let windowStart = 0, sum = 0, minLen = Infinity;
        for (let windowEnd = 0; windowEnd < arr.length; windowEnd++) {
            sum += arr[windowEnd];
            while (sum >= target) {
                minLen = Math.min(minLen, windowEnd - windowStart + 1);
                sum -= arr[windowStart];
                windowStart++;
            }
        }
        return minLen === Infinity ? 0 : minLen;
    }
```

Strings Q6: Longest Substring Without Repeating Characters

Problem: Find the length of the longest substring without repeating characters. Input: s='abcabcbb' Output: 3 ('abc')

```
    function lengthOfLongestSubstring(s) {
        let windowStart = 0, set = new Set(), maxLen = 0;
        for (let windowEnd = 0; windowEnd < s.length; windowEnd++) {
            while (set.has(s[windowEnd])) {
                set.delete(s[windowStart]);
                windowStart++;
            }
            set.add(s[windowEnd]);
            maxLen = Math.max(maxLen, windowEnd - windowStart + 1);
        }
        return maxLen;
    }
```

Strings Q7: Longest Substring with At Most K Distinct Characters

Problem: Find the longest substring with at most K distinct characters. Input: s='eceba', k=2 Output: 3 ('ece')

```
    function longestSubstringKDistinct(s, k) {
        let windowStart = 0, map = new Map(), maxLen = 0;
        for (let windowEnd = 0; windowEnd < s.length; windowEnd++) {
            map.set(s[windowEnd], (map.get(s[windowEnd]) || 0) + 1);
            while (map.size > k) {
                map.set(s[windowStart], map.get(s[windowStart]) - 1);
                if (map.get(s[windowStart]) === 0) map.delete(s[windowStart]);
                windowStart++;
            }
            maxLen = Math.max(maxLen, windowEnd - windowStart + 1);
        }
        return maxLen;
    }
```

Strings Q8: Minimum Window Substring

Problem: Find the smallest substring in s that contains all characters of t. Input: s='ADOBECODEBANC', t='ABC' Output: 'BANC'

```
    function minWindow(s, t) {
        let need = new Map();
        for (let c of t) need.set(c, (need.get(c) || 0) + 1);
```

```
    let have = new Map(), windowStart = 0;
    let minLen = Infinity, result = "";
    let formed = 0, required = need.size;
    for (let windowEnd = 0; windowEnd < s.length; windowEnd++) {
        let c = s[windowEnd];
        have.set(c, (have.get(c) || 0) + 1);
        if (need.has(c) && have.get(c) === need.get(c)) formed++;
        while (formed === required) {
            if (windowEnd - windowStart + 1 < minLen) {
                minLen = windowEnd - windowStart + 1;
                result = s.substring(windowStart, windowEnd + 1);
            }
            let leftChar = s[windowStart];
            have.set(leftChar, have.get(leftChar) - 1);
            if (need.has(leftChar) && have.get(leftChar) < need.get(leftChar)) formed--;
            windowStart++;
        }
    }
    return result;
}
```

Strings Q9: Longest Repeating Character Replacement

Problem: Replace at most k characters to make the longest repeating substring. Input: s='AABABBA', k=1 Output: 4

```
function characterReplacement(s, k) {
    let windowStart = 0, freq = {}, maxCount = 0, maxLen = 0;
    for (let windowEnd = 0; windowEnd < s.length; windowEnd++) {
        let char = s[windowEnd];
        freq[char] = (freq[char] || 0) + 1;
        maxCount = Math.max(maxCount, freq[char]);
        while ((windowEnd - windowStart + 1) - maxCount > k) {
            freq[s[windowStart]]--;
            windowStart++;
        }
        maxLen = Math.max(maxLen, windowEnd - windowStart + 1);
    }
    return maxLen;
}
```

Strings Q10: Find All Anagrams in a String

Problem: Find all start indices of substrings in s that are anagrams of p. Input: s='cbaebabacd', p='abc' Output: [0,6]

```
function findAnagrams(s, p) {
    let need = {}, window = {};
    for (let c of p) need[c] = (need[c] || 0) + 1;
    let res = [], windowStart = 0, match = 0;
    for (let windowEnd = 0; windowEnd < s.length; windowEnd++) {
        let c = s[windowEnd];
        window[c] = (window[c] || 0) + 1;
        if (need[c] && window[c] === need[c]) match++;
        while (windowEnd - windowStart + 1 >= p.length) {
            if (match === Object.keys(need).length) res.push(windowStart);
            let leftChar = s[windowStart];
            if (need[leftChar] && window[leftChar] === need[leftChar]) match--;
            window[leftChar]--;
            windowStart++;
        }
    }
    return res;
}
```