Sliding Window Technique in Programming

Key Points of Sliding Window:

1. Fixed Size Window: The window size is constant and does not change as it slides across the array.

2. Variable Size Window: The window size can change dynamically based on conditions.

Fixed Size Sliding Window

In this type, the window size is predetermined, and you slide it from the start to the end of the array.

Example: Maximum Sum of a Subarray of Size k

Given an array of integers and a number k, find the maximum sum of a subarray of size k.

Steps:

1. Calculate the sum of the first k elements.

2. Slide the window one element to the right, adding the new element of the window and subtracting the element that is no longer in the window.

3. Repeat until the window reaches the end of the array.

```
function maxSumSubarray(arr, k) {
    let n = arr.length;
    if (n < k) {
        return null;
    }

    // Calculate the sum of the first window
    let windowSum = 0;
    for (let i = 0; i < k; i++) {
        windowSum += arr[i];
```

```
    }
    let maxSum = windowSum;

    // Slide the window from start to end of the array
    for (let i = 0; i < n - k; i++) {
        windowSum = windowSum - arr[i] + arr[i + k];
        maxSum = Math.max(maxSum, windowSum);
    }

    return maxSum;
}

// Example usage:
let arr = [1, 3, 2, 6, -1, 4, 1, 8, 2];
let k = 3;
console.log(maxSumSubarray(arr, k));  // Output: 13
```

Variable Size Sliding Window

In this type, the window size can change dynamically. This is useful for problems where you need to find a subarray that meets certain conditions and the subarray length is not fixed.

Example: Smallest Subarray with Sum Greater Than or Equal to S

Given an array of integers and an integer S, find the length of the smallest contiguous subarray whose sum is greater than or equal to S. Return 0 if no such subarray exists.

Steps:

1. Initialize the window's start and end to the beginning of the array.

2. Expand the end to increase the window size until the sum is greater than or equal to S.

3. Once the sum is greater than or equal to S, shrink the window from the start to see if a smaller subarray can still satisfy the condition.

4. Repeat until the end of the array is reached.

```
function smallestSubarrayWithSum(arr, S) {
    let n = arr.length;
    let minLength = Infinity;
    let windowSum = 0;
    let start = 0;

    for (let end = 0; end < n; end++) {
        windowSum += arr[end];

        while (windowSum >= S) {
            minLength = Math.min(minLength, end - start + 1);
            windowSum -= arr[start];
            start += 1;
        }
    }

    return minLength === Infinity ? 0 : minLength;
}

// Example usage:
let arr = [2, 1, 5, 2, 3, 2];
let S = 7;
console.log(smallestSubarrayWithSum(arr, S));  // Output: 2
```

Types of Sliding Window Problems

1. Fixed Size Window:

- Maximum sum of a subarray of size k.
- Minimum sum of a subarray of size k.
- Average of each subarray of size k.

2. Variable Size Window:

- Smallest subarray with a sum greater than or equal to S.
- Longest subarray with sum less than or equal to S.
- Subarrays with a given sum.

Advantages of Sliding Window Technique

- Efficiency: Reduces the time complexity compared to brute force methods by avoiding unnecessary repeated calculations.

- Simplicity: Easy to implement and understand, especially for problems involving contiguous subarrays.

Conclusion

The sliding window technique is a powerful tool for solving various array and string problems efficiently. Understanding the difference between fixed and variable size windows is crucial for applying the right approach to different problem scenarios.