DAY 5 – SLIDING WINDOW & STRING CHARACTER COUNT NOTES

1️⃣WHEN TO USE FIXED-SIZE SLIDING WINDOW

Use FIXED sliding window when:

- Window size is **given and constant (k)**
- You are asked about:
- Sum of subarray of size k
- Maximum / minimum of subarray of size k
- Count of subarrays of size k

Keywords in problem statement:

- "subarray of size k"
- "window size k"
- "exactly k elements"

Fixed-Size Sliding Window Template

```
int windowSum = 0;

// Step 1: calculate first window
for (int i = 0; i < k; i++) {
    windowSum += arr[i];
}

int answer = windowSum;

// Step 2: slide the window
for (int i = k; i < arr.length; i++) {
    windowSum = windowSum + arr[i] - arr[i - k];
    answer = Math.max(answer, windowSum); // or Math.min / count
}
```

Examples of Fixed Window Problems

- Maximum sum subarray of size k
- Minimum sum subarray of size k
- Count subarrays of size k with sum ≥ X

2️⃣WHEN TO USE VARIABLE-SIZE SLIDING WINDOW

Use VARIABLE sliding window when:

- Window size is **not fixed**
- You are asked about:
- Longest or shortest substring / subarray
- At most / at least conditions
- Without repeating characters

Keywords in problem statement:

- "longest"
- "shortest"
- "at most"
- "without repeating"
- "substring"

Variable-Size Sliding Window Template

```
int left = 0;
int answer = 0;

for (int right = 0; right < n; right++) {

    // add element at right

    while (condition breaks) {
        // remove element at left
        left++;
    }

    answer = Math.max(answer, right - left + 1);
}
```

Examples of Variable Window Problems

- Longest substring without repeating characters
- Longest subarray with sum $\leq$ K
- Longest substring with at most K distinct characters

3️⃣KEY DIFFERENCE: FIXED vs VARIABLE WINDOW

Fixed Sliding Window	Variable Sliding Window
Window size fixed	Window size changes
One loop after init	One loop + while loop
Add & remove once	Expand + shrink window
Sum / count problems	Longest / shortest

4️⃣HOW TO COUNT CHARACTERS IN A STRING

Use character counting when:

- Anagram problems
- Frequency problems
- First non-repeating character
- Sliding window on strings

Frequency Array (Lowercase a–z)

```
int[] freq = new int[26];
```

Count Characters in a String

```
for (int i = 0; i < s.length(); i++) {
    char ch = s.charAt(i);
    freq[ch - 'a']++;
}
```

Character to Index Mapping

```
ch - 'a'
```

Examples: - 'a' → 0 - 'b' → 1 - 'c' → 2

Index to Character Mapping

```
(char)(i + 'a')
```

Check Character Frequency

```
if (freq[ch - 'a'] == 1)
```

5️⃣CHARACTER COUNT IN SLIDING WINDOW (STRINGS)

Add character (expand window)

```
freq[s.charAt(right) - 'a']++;
```

Remove character (shrink window)

```
freq[s.charAt(left) - 'a']--;
left++;
```

6️⃣ONE-PAGE MEMORY RULES (VERY IMPORTANT)

1. Fixed window → window size is given
2. Variable window → longest / shortest problems
3. Always expand using `right`
4. Shrink only when condition breaks
5. `right - left + 1` gives window length
6. Use frequency array for string problems
7. `ch - 'a'` maps character to index

FINAL NOTE

These notes cover: - Core sliding window patterns - String character counting logic - Most common interview use-cases

Use this as a **quick revision sheet before interviews**.