Java DSA Cheatsheet: Arrays and Strings

When & Where to Use Arrays and Strings in Java DSA

ARRAYS IN JAVA DSA

WHEN TO USE ARRAYS:
- When you need a fixed-size, contiguous block of memory.
- When you require fast index-based access to elements (O(1) access).
- When elements are of the same data type (e.g., all integers).
- When you want to perform frequent read/write/update operations at specific positions.

WHERE TO USE ARRAYS (PROBLEM TYPES):
1. Subarray Problems
   - Max Sum Subarray (Kadanes Algorithm)
   - Minimum Length Subarray with Given Sum
   - Count Subarrays with Given XOR

2. Prefix Sum & Difference Arrays
   - For range sum queries
   - Efficient frequency counting

3. Sorting and Searching
   - Binary Search on Sorted Array
   - In-place Sorting Algorithms (QuickSort, MergeSort)

4. Sliding Window Problems
   - Fixed-size or variable size window problems
   - Maximum sum in window of size k

5. Two Pointer Technique
   - Pair with given sum in sorted array
   - Remove duplicates from sorted array

6. Frequency Counting (via index mapping)
   - Count character/number frequency using int[26] or int[1000001]

7. Matrix Problems (2D Arrays)
   - Search a key in 2D sorted matrix
   - Rotate matrix, Transpose matrix

8. Dynamic Programming
   - Store subproblem results in arrays for memoization/tabulation

STRINGS IN JAVA DSA

WHEN TO USE STRINGS:
- When you work with textual data, words, characters.
- When your data needs to be manipulated as sequences of characters.
- When you need to validate, transform, or search for patterns.

Java DSA Cheatsheet: Arrays and Strings

WHERE TO USE STRINGS (PROBLEM TYPES):

1. Palindrome Problems
   - Check if string is palindrome
   - Longest Palindromic Substring

2. Anagram Problems
   - Check if two strings are anagrams
   - Group Anagrams (use hashmap + char count)

3. Pattern Matching
   - Naive Search, KMP Algorithm, Rabin-Karp
   - strStr(), find substring

4. String Transformation/Parsing
   - StringBuilder usage for efficient editing
   - Tokenizing string using split(), substring(), indexOf()

5. Sliding Window on String
   - Longest Substring Without Repeating Characters
   - Minimum Window Substring

6. Frequency Maps on Strings
   - Count frequency of each character using int[26] or HashMap

7. Validity Checking
   - Valid Parentheses using Stack
   - Balanced Brackets

8. Reversals and Rotations
   - Reverse string using two-pointer method
   - Rotate string left/right by k positions

9. Dynamic Programming on Strings
   - Longest Common Subsequence (LCS)
   - Edit Distance

10. Trie-Based Problems
    - Word Dictionary, Prefix Search

RULES OF THUMB:

- Use arrays when dealing with numbers, indexes, or positions.
- Use strings when working with characters, words, or textual rules.
- Convert String to char[] when you need character-level manipulation.
- Use StringBuilder for mutable string operations inside loops.

BEST PRACTICES:

- For large mutations, always prefer StringBuilder.
- Use Arrays.sort() for quick sorting of arrays.
- Prefer HashMap for frequency-based problems in both arrays and strings.

Java DSA Cheatsheet: Arrays and Strings

- Know utility methods like Arrays.copyOf(), toCharArray(), and Collections.reverse().