Data Structures and Algorithms (DSA) Questions in Java

---

1. Arrays & Strings

1. Find the largest element in an array.
2. Reverse an array.
3. Find the second largest and second smallest element.
4. Check if an array is sorted.
5. Move all zeros to the end of an array.
6. Find the union and intersection of two sorted arrays.
7. Rotate an array by K positions (cyclically).
8. Find the maximum sum subarray (Kadane's Algorithm).
9. Find all pairs in an array that sum to a given number.
10. Implement the **KMP (Knuth-Morris-Pratt) Algorithm** for pattern matching in strings.

---

2. Recursion & Backtracking

1. Print numbers from N to 1 using recursion.
2. Find factorial of a number using recursion.
3. Compute nth Fibonacci number using recursion.
4. Check if a string is a palindrome using recursion.
5. Solve the Tower of Hanoi problem.
6. Generate all subsets of a given string.
7. Solve the N-Queens problem using backtracking.
8. Find all permutations of a given string.
9. Word Break Problem (Check if a word can be segmented using a given dictionary).
10. Rat in a Maze problem using backtracking.

---

3. Linked List

1. Implement a singly linked list (insert, delete, display).
2. Reverse a linked list.
3. Detect a loop in a linked list (Floyd's cycle detection algorithm).

4. Find the middle element of a linked list.
5. Merge two sorted linked lists.
6. Find the intersection point of two linked lists.
7. Remove duplicates from a sorted linked list.
8. Add two numbers represented as linked lists.
9. Flatten a linked list.
10. Clone a linked list with a random pointer.

---

4. Stack & Queue

1. Implement a stack using an array.
2. Implement a queue using an array.
3. Implement a stack using two queues.
4. Implement a queue using two stacks.
5. Find the next greater element for each element in an array.
6. Implement a min stack (stack with constant-time minimum retrieval).
7. Implement an LRU cache using a deque.
8. Evaluate a postfix expression.
9. Implement a circular queue.
10. Find the maximum element in every sliding window of size K.

---

5. Hashing

1. Implement a hashmap from scratch.
2. Find the first non-repeating character in a string.
3. Check if two arrays are disjoint.
4. Find the largest subarray with a sum of zero.
5. Find all pairs in an array with a given sum using a hashmap.
6. Find the longest consecutive sequence in an array.
7. Find the subarray with the given sum.
8. Implement a phone directory using hashing.
9. Find the longest substring without repeating characters.
10. Implement a simple spell checker using a dictionary (hashing).

6. Trees & Binary Trees

1. Implement a binary tree (insert, delete, traversal).
2. Find the height of a binary tree.
3. Perform level-order traversal of a binary tree.
4. Check if a binary tree is balanced.
5. Find the lowest common ancestor (LCA) of two nodes.
6. Convert a binary tree to a doubly linked list.
7. Check if two binary trees are identical.
8. Find the diameter of a binary tree.
9. Print top view and bottom view of a binary tree.
10. Construct a binary tree from inorder and preorder traversal.

---

7. Heap & Priority Queue

1. Implement a min heap and a max heap.
2. Find the Kth largest element in an array.
3. Find the median of a data stream.
4. Merge K sorted linked lists using a priority queue.
5. Find the top K frequent elements in an array.
6. Find the K closest points to the origin.
7. Implement a max heap using a priority queue.
8. Find the smallest range that includes at least one number from each of the given sorted lists.
9. Check if an array represents a max heap.
10. Implement a heap sort algorithm.

---

8. Graphs

1. Represent a graph using an adjacency list.
2. Implement BFS traversal of a graph.
3. Implement DFS traversal of a graph.
4. Detect a cycle in a graph (directed & undirected).

5. Find the shortest path using Dijkstra's algorithm.
6. Find the shortest path using the Bellman-Ford algorithm.
7. Implement topological sorting.
8. Find strongly connected components (Kosaraju's Algorithm).
9. Find the minimum spanning tree using Kruskal's algorithm.
10. Implement the Floyd-Warshall algorithm.

---

9. Dynamic Programming (DP)

1. Compute Fibonacci numbers using memoization.
2. Find the nth Catalan number.
3. Solve the 0/1 Knapsack problem.
4. Find the longest common subsequence (LCS).
5. Solve the coin change problem.
6. Find the minimum number of insertions to make a string palindrome.
7. Find the number of ways to climb stairs with variable jumps.
8. Find the maximum sum increasing subsequence.
9. Compute the edit distance between two strings.
10. Solve the matrix chain multiplication problem.

---

10. Tries & Segment Trees

1. Implement a Trie (Insert, Search, Delete).
2. Find words with a given prefix in a Trie.
3. Implement a simple autocomplete system using a Trie.
4. Count the number of words in a Trie.
5. Implement a basic segment tree for range queries.
6. Implement a segment tree with lazy propagation.
7. Find the longest prefix match using a Trie.
8. Implement a Fenwick Tree (Binary Indexed Tree).
9. Solve the range minimum query using a segment tree.
10. Implement a persistent segment tree.

How to Use These Questions?

1. **Start with easy questions** to get comfortable with concepts.
2. **Write code from scratch** instead of copying solutions.
3. **Use debugging** to understand errors.
4. **Optimize solutions** and understand their complexity.
5. **Apply learned concepts to solve competitive programming problems.**