**Stack:**

|  |  |  |
| --- | --- | --- |
|  | Problem | Status |
|  | Implement Queue using Stacks |  |
|  | Design and Implement Special Stack Data Structure | Added Space Optimized Version |  |
|  | Implement two stacks in an array |  |
|  | Implement Stack using Queues |  |
|  | Design a stack with operations on middle element |  |
|  | How to efficiently implement k stacks in a single array? |  |
|  | How to create mergable stack? |  |
|  | Design a stack that supports getMin() in O(1) time and O(1) extra space |  |
|  | Implement a stack using single queue |  |
|  | How to implement stack using priority queue or heap? |  |
|  | Create a customized data structure which evaluates functions in O(1) |  |
|  | Infix to Postfix Conversion using Stack |  |
|  | The Stock Span Problem |  |
|  | Check for balanced parentheses in an expression |  |
|  | Next Greater Element |  |
|  | The Celebrity Problem |  |
|  | Expression Evaluation |  |
|  | Evaluation of Postfix Expression |  |
|  | Iterative Tower of Hanoi |  |
|  | Reverse a stack using recursion |  |
|  | Sort a stack using recursion |  |
|  | Sort a stack using a temporary stack |  |
|  | Reverse a stack without using extra space in O(n) |  |
|  | Iterative Postorder Traversal | Set 1 (Using Two Stacks) |  |
|  | Iterative Postorder Traversal | Set 2 (Using One Stack) |  |
|  | Merge Overlapping Intervals |  |
|  | Largest Rectangular Area in a Histogram | Set 2 |  |
|  | Print ancestors of a given binary tree node without recursion |  |
|  | Reverse a string using stack |  |
|  | Program for Tower of Hanoi |  |
|  | Find maximum depth of nested parenthesis in a string |  |
|  | Find maximum of minimum for every window size in a given array |  |
|  | Length of the longest valid substring |  |
|  | Iterative Depth First Traversal of Graph |  |
|  | Minimum number of bracket reversals needed to make an expression balanced |  |
|  | Check if a given array can represent Preorder Traversal of Binary Search Tree |  |
|  | Form minimum number from given sequence |  |
|  | Find if an expression has duplicate parenthesis or not |  |
|  | Find maximum difference between nearest left and right smaller elements |  |
|  | Find next Smaller of next Greater in an array |  |
|  | Find maximum sum possible equal sum of three stacks |  |
|  | Count natural numbers whose all permutation are greater than that number |  |
|  | Delete consecutive same words in a sequence |  |
|  | Decode a string recursively encoded as count followed by substring |  |
|  | Bubble sort using two Stacks |  |
|  | Pattern Occurrences : Stack Implementation Java |  |
|  | Iterative method to find ancestors of a given binary tree |  |
|  | Stack Permutations (Check if an array is stack permutation of other) |  |
|  | Tracking current Maximum Element in a Stack |  |
|  | Check mirror in n-ary tree |  |
|  | Reverse a number using stack |  |
|  | Reversing the first K elements of a Queue |  |
|  | Reversing a Queue |  |
|  | Check if stack elements are pairwise consecutive |  |
|  | Spaghetti Stack |  |
|  | Interleave the first half of the queue with second half |  |
|  | Remove brackets from an algebraic string containing + and – operators |  |
|  | Remove repeated digits in a given number |  |
|  | Tom is a string freak. He has got sequences of n words to manipulate. If in a sequence, two same words come together then he’ll destroy each other. |  |
|  | Longest valid Parentheses |  |
|  | Next larger element |  |
|  | Parenthesis Checker |  |
|  | Maximum of minimum for every window size |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |