

✓  
TC  
Hint

Questions by Love Babbar:

Topic:	Problem:	Done [yes or no]
Array	<a href="#">Reverse the array</a>	<>
Array	<a href="#">Find the maximum and minimum element in an array</a>	<>
Array	<a href="#">Find the "Kth" max and min element of an array</a>	<>
Array	<a href="#">Given an array which consists of only 0, 1 and 2. Sort the array without using any sorting algo</a>	<>
Array	<a href="#">Move all the negative elements to one side of the array</a>	<>
Array	<a href="#">Find the Union and Intersection of the two sorted arrays.</a>	<>
Array	<a href="#">Write a program to cyclically rotate an array by one.</a>	<>
Array	<a href="#">find Largest sum contiguous Subarray [V. IMP]</a>	<>
Array	<a href="#">Minimise the maximum difference between heights [V.IMP]</a>	<>
Array	<a href="#">Minimum no. of Jumps to reach end of an array</a>	<>
Array	<a href="#">find duplicate in an array of N+1 Integers</a>	<>
Array	<a href="#">Merge 2 sorted arrays without using Extra space.</a>	<>
Array	<a href="#">Kadane's Algo [V.V.V.V.V IMP]</a>	<>
Array	<a href="#">Merge Intervals</a>	<>
Array	<a href="#">Next Permutation</a>	<>
Array	<a href="#">Count Inversion</a>	<>
Array	<a href="#">Best time to buy and Sell stock</a>	<>
Array	<a href="#">find all pairs on integer array whose sum is equal to given number</a>	<>
Array	<a href="#">find common elements In 3 sorted arrays</a>	<>
Array	<a href="#">Rearrange the array in alternating positive and negative items with O(1) extra space</a>	<>
Array	<a href="#">Find if there is any subarray with sum equal to 0</a>	<>
Array	<a href="#">Find factorial of a large number</a>	<>
Array	<a href="#">find maximum product subarray</a>	<>
Array	<a href="#">Find longest coinsecutive subsequence</a>	<>
Array	<a href="#">Given an array of size n and a number k, fin all elements that appear more than " n/k " times.</a>	<>
Array	<a href="#">Maximum profit by buying and selling a share atmost twice</a>	<>
Array	<a href="#">Find whether an array is a subset of another array</a>	<>
Array	<a href="#">Find the triplet that sum to a given value</a>	<>
Array	<a href="#">Trapping Rain water problem</a>	<>
Array	<a href="#">Chocolate Distribution problem</a>	<>
Array	<a href="#">Smallest Subarray with sum greater than a given value</a>	<>
Array	<a href="#">Three way partitioning of an array around a given value</a>	<>
Array	<a href="#">Minimum swaps required bring elements less equal K together</a>	<>
Array	<a href="#">Minimum no. of operations required to make an array palindrome</a>	<>
Array	<a href="#">Median of 2 sorted arrays of equal size</a>	<>
Array	<a href="#">Median of 2 sorted arrays of different size</a>	<>

Matrix	<a href="#">Spiral traversal on a Matrix</a>	<->
Matrix	<a href="#">Search an element in a matrix</a>	<->
Matrix	<a href="#">Find median in a row wise sorted matrix</a>	<->
Matrix	<a href="#">Find row with maximum no. of 1's</a>	<->
Matrix	<a href="#">Print elements in sorted order using row-column wise sorted matrix</a>	<->
Matrix	<a href="#">Maximum size rectangle</a>	<->
Matrix	<a href="#">Find a specific pair in matrix</a>	<->
Matrix	<a href="#">Rotate matrix by 90 degrees</a>	<->
Matrix	<a href="#">Kth smallest element in a row-column wise sorted matrix</a>	<->
Matrix	<a href="#">Common elements in all rows of a given matrix</a>	<->
String	<a href="#">Reverse a String</a>	<->
String	<a href="#">Check whether a String is Palindrome or not</a>	<->
String	<a href="#">Find Duplicate characters in a string</a>	<->
String	Why strings are immutable in Java?	<->
String	<a href="#">Write a Code to check whether one string is a rotation of another</a>	<->
String	<a href="#">Write a Program to check whether a string is a valid shuffle of two strings or not</a>	<->
String	<a href="#">Count and Say problem</a>	<->
String	<a href="#">Write a program to find the longest Palindrome in a string.[ Longest palindromic Substring]</a>	<->
String	<a href="#">Find Longest Recurring Subsequence in String</a>	<->
String	<a href="#">Print all Subsequences of a string.</a>	<->
String	<a href="#">Print all the permutations of the given string</a>	<->
String	<a href="#">Split the Binary string into two substring with equal 0's and 1's</a>	<->
String	<a href="#">Word Wrap Problem [VERY IMP].</a>	<->
String	<a href="#">EDIT Distance [Very Imp]</a>	<->
String	<a href="#">Find next greater number with same set of digits. [Very Very IMP]</a>	<->
String	<a href="#">Balanced Parenthesis problem.[Imp]</a>	<->
String	<a href="#">Word break Problem[ Very Imp]</a>	<->
String	<a href="#">Rabin Karp Algo</a>	<->
String	<a href="#">KMP Algo</a>	<->
String	<a href="#">Convert a Sentence into its equivalent mobile numeric keypad sequence.</a>	<->
String	<a href="#">Minimum number of bracket reversals needed to make an expression balanced.</a>	<->
String	<a href="#">Count All Palindromic Subsequence in a given String.</a>	<->
String	<a href="#">Count of number of given string in 2D character array</a>	<->
String	<a href="#">Search a Word in a 2D Grid of characters.</a>	<->
String	<a href="#">Boyer Moore Algorithm for Pattern Searching.</a>	<->
String	<a href="#">Converting Roman Numerals to Decimal</a>	<->
String	<a href="#">Longest Common Prefix</a>	<->

String	<a href="#">Number of flips to make binary string alternate</a>	<->
String	<a href="#">Find the first repeated word in string.</a>	<->
String	<a href="#">Minimum number of swaps for bracket balancing.</a>	<->
String	<a href="#">Find the longest common subsequence between two strings.</a>	<->
String	<a href="#">Program to generate all possible valid IP addresses from given string.</a>	<->
String	<a href="#">Write a program to find the smallest window that contains all characters of string itself.</a>	<->
String	<a href="#">Rearrange characters in a string such that no two adjacent are same</a>	<->
String	<a href="#">Minimum characters to be added at front to make string palindrome</a>	<->
String	<a href="#">Given a sequence of words, print all anagrams together</a>	<->
String	<a href="#">Find the smallest window in a string containing all characters of another string</a>	<->
String	<a href="#">Recursively remove all adjacent duplicates</a>	<->
String	<a href="#">String matching where one string contains wildcard characters</a>	<->
String	<a href="#">Function to find Number of customers who could not get a computer</a>	<->
String	<a href="#">Transform One String to Another using Minimum Number of Given Operation</a>	<->
String	<a href="#">Check if two given strings are isomorphic to each other</a>	<->
String	<a href="#">Recursively print all sentences that can be formed from list of word lists</a>	<->
Searching & Sorting	<a href="#">Find first and last positions of an element in a sorted array</a>	<->
Searching & Sorting	<a href="#">Find a Fixed Point (Value equal to index) in a given array</a>	<->
Searching & Sorting	<a href="#">Search in a rotated sorted array</a>	<->
Searching & Sorting	<a href="#">square root of an integer</a>	<->
Searching & Sorting	<a href="#">Maximum and minimum of an array using minimum number of comparisons</a>	<->
Searching & Sorting	<a href="#">Optimum location of point to minimize total distance</a>	<->
Searching & Sorting	<a href="#">Find the repeating and the missing</a>	<->
Searching & Sorting	<a href="#">find majority element</a>	<->
Searching & Sorting	<a href="#">Searching in an array where adjacent differ by at most k</a>	<->
Searching & Sorting	<a href="#">find a pair with a given difference</a>	<->
Searching & Sorting	<a href="#">find four elements that sum to a given value</a>	<->
Searching & Sorting	<a href="#">maximum sum such that no 2 elements are adjacent</a>	<->
Searching & Sorting	<a href="#">Count triplet with sum smaller than a given value</a>	<->
Searching & Sorting	<a href="#">merge 2 sorted arrays</a>	<->
Searching & Sorting	<a href="#">print all subarrays with 0 sum</a>	<->
Searching & Sorting	<a href="#">Product array Puzzle</a>	<->
Searching & Sorting	<a href="#">Sort array according to count of set bits</a>	<->
Searching & Sorting	<a href="#">minimum no. of swaps required to sort the array</a>	<->
Searching & Sorting	<a href="#">Bishu and Soldiers</a>	<->
Searching & Sorting	<a href="#">Rasta and Kheshtak</a>	<->
Searching & Sorting	<a href="#">Kth smallest number again</a>	<->
Searching & Sorting	<a href="#">Find pivot element in a sorted array</a>	<->
Searching & Sorting	<a href="#">K-th Element of Two Sorted Arrays</a>	<->

Searching & Sorting	<a href="#">Aggressive cows</a>	<->
Searching & Sorting	<a href="#">Book Allocation Problem</a>	<->
Searching & Sorting	<a href="#">EKOSPOJ:</a>	<->
Searching & Sorting	<a href="#">Job Scheduling Algo</a>	<->
Searching & Sorting	<a href="#">Missing Number in AP</a>	<->
Searching & Sorting	<a href="#">Smallest number with atleastn trailing zeroes infactorial</a>	<->
Searching & Sorting	<a href="#">Painters Partition Problem:</a>	<->
Searching & Sorting	<a href="#">ROTI-Prata SPOJ</a>	<->
Searching & Sorting	<a href="#">DoubleHelix SPOJ</a>	<->
Searching & Sorting	<a href="#">Subset Sums</a>	<->
Searching & Sorting	<a href="#">Findthe inversion count</a>	<->
Searching & Sorting	<a href="#">Implement Merge-sort in-place</a>	<->
Searching & Sorting	<a href="#">Partitioning and Sorting Arrays with Many Repeated Entries</a>	<->
LinkedList	<a href="#">Write a Program to reverse the Linked List. (Both Iterative and recursive)</a>	<->
LinkedList	<a href="#">Reverse a Linked List in group of Given Size. [Very Imp]</a>	<->
LinkedList	<a href="#">Write a program to Detect loop in a linked list.</a>	<->
LinkedList	<a href="#">Write a program to Delete loop in a linked list.</a>	<->
LinkedList	<a href="#">Find the starting point of the loop.</a>	<->
LinkedList	<a href="#">Remove Duplicates in a sorted Linked List.</a>	<->
LinkedList	<a href="#">Remove Duplicates in a Un-sorted Linked List.</a>	<->
LinkedList	<a href="#">Write a Program to Move the last element to Front in a Linked List.</a>	<->
LinkedList	<a href="#">Add “1” to a number represented as a Linked List.</a>	<->
LinkedList	<a href="#">Add two numbers represented by linked lists.</a>	<->
LinkedList	<a href="#">Intersection of two Sorted Linked List.</a>	<->
LinkedList	<a href="#">Intersection Point of two Linked Lists.</a>	<->
LinkedList	<a href="#">Merge Sort For Linked lists.[Very Important]</a>	<->
LinkedList	<a href="#">Quicksort for Linked Lists.[Very Important]</a>	<->
LinkedList	<a href="#">Find the middle Element of a linked list.</a>	<->
LinkedList	<a href="#">Check if a linked list is a circular linked list.</a>	<->
LinkedList	<a href="#">Split a Circular linked list into two halves.</a>	<->
LinkedList	<a href="#">Write a Program to check whether the Singly Linked list is a palindrome or not.</a>	<->
LinkedList	<a href="#">Deletion from a Circular Linked List.</a>	<->
LinkedList	<a href="#">Reverse a Doubly Linked list.</a>	<->
LinkedList	<a href="#">Find pairs with a given sum in a DLL.</a>	<->
LinkedList	<a href="#">Count triplets in a sorted DLL whose sum is equal to given value “X”.</a>	<->
LinkedList	<a href="#">Sort a “k”sorted Doubly Linked list.[Very IMP]</a>	<->
LinkedList	<a href="#">Rotate DoublyLinked list by N nodes.</a>	<->
LinkedList	<a href="#">Rotate a Doubly Linked list in group of Given Size.[Very IMP]</a>	<->
LinkedList	Can we reverse a linked list in less than O(n) ?	<->

LinkedList	Why Quicksort is preferred for. Arrays and Merge Sort for LinkedLists ?	<->
LinkedList	<a href="#">Flatten a Linked List</a>	<->
LinkedList	<a href="#">Sort a LL of 0's, 1's and 2's</a>	<->
LinkedList	<a href="#">Clone a linked list with next and random pointer</a>	<->
LinkedList	<a href="#">Merge K sorted Linked list</a>	<->
LinkedList	<a href="#">Multiply 2 no. represented by LL</a>	<->
LinkedList	<a href="#">Delete nodes which have a greater value on right side</a>	<->
LinkedList	<a href="#">Segregate even and odd nodes in a Linked List</a>	<->
LinkedList	<a href="#">Program for n'th node from the end of a Linked List</a>	<->
LinkedList	<a href="#">Find the first non-repeating character from a stream of characters</a>	<->
Binary Trees	<a href="#">level order traversal</a>	<->
Binary Trees	<a href="#">Reverse Level Order traversal</a>	<->
Binary Trees	<a href="#">Height of a tree</a>	<->
Binary Trees	<a href="#">Diameter of a tree</a>	<->
Binary Trees	<a href="#">Mirror of a tree</a>	<->
Binary Trees	<a href="#">Inorder Traversal of a tree both using recursion and Iteration</a>	<->
Binary Trees	<a href="#">Preorder Traversal of a tree both using recursion and Iteration</a>	<->
Binary Trees	<a href="#">Postorder Traversal of a tree both using recursion and Iteration</a>	<->
Binary Trees	<a href="#">Left View of a tree</a>	<->
Binary Trees	<a href="#">Right View of Tree</a>	<->
Binary Trees	<a href="#">Top View of a tree</a>	<->
Binary Trees	<a href="#">Bottom View of a tree</a>	<->
Binary Trees	<a href="#">Zig-Zag traversal of a binary tree</a>	<->
Binary Trees	<a href="#">Check if a tree is balanced or not</a>	<->
Binary Trees	<a href="#">Diagnol Traversal of a Binary tree</a>	<->
Binary Trees	<a href="#">Boundary traversal of a Binary tree</a>	<->
Binary Trees	<a href="#">Construct Binary Tree from String with Bracket Representation</a>	<->
Binary Trees	<a href="#">Convert Binary tree into Doubly Linked List</a>	<->
Binary Trees	<a href="#">Convert Binary tree into Sum tree</a>	<->
Binary Trees	<a href="#">Construct Binary tree from Inorder and preorder traversal</a>	<->
Binary Trees	<a href="#">Find minimum swaps required to convert a Binary tree into BST</a>	<->
Binary Trees	<a href="#">Check if Binary tree is Sum tree or not</a>	<->
Binary Trees	<a href="#">Check if all leaf nodes are at same level or not</a>	<->
Binary Trees	<a href="#">Check if a Binary Tree contains duplicate subtrees of size 2 or more [ IMP ]</a>	<->
Binary Trees	<a href="#">Check if 2 trees are mirror or not</a>	<->
Binary Trees	<a href="#">Sum of Nodes on the Longest path from root to leaf node</a>	<->
Binary Trees	<a href="#">Check if given graph is tree or not. [ IMP ]</a>	<->
Binary Trees	<a href="#">Find Largest subtree sum in a tree</a>	<->
Binary Trees	<a href="#">Maximum Sum of nodes in Binary tree such that no two are adjacent</a>	<->

Binary Trees	<a href="#">Print all "K" Sum paths in a Binary tree</a>	<->
Binary Trees	<a href="#">Find LCA in a Binary tree</a>	<->
Binary Trees	<a href="#">Find distance between 2 nodes in a Binary tree</a>	<->
Binary Trees	<a href="#">Kth Ancestor of node in a Binary tree</a>	<->
Binary Trees	<a href="#">Find all Duplicate subtrees in a Binary tree [ IMP ]</a>	<->
Binary Trees	<a href="#">Tree Isomorphism Problem</a>	<->
Binary Search Trees	<a href="#">Find a value in a BST</a>	<->
Binary Search Trees	<a href="#">Deletion of a node in a BST</a>	<->
Binary Search Trees	<a href="#">Find min and max value in a BST</a>	<->
Binary Search Trees	<a href="#">Find inorder successor and inorder predecessor in a BST</a>	<->
Binary Search Trees	<a href="#">Check if a tree is a BST or not</a>	<->
Binary Search Trees	<a href="#">Populate Inorder successor of all nodes</a>	<->
Binary Search Trees	<a href="#">Find LCA of 2 nodes in a BST</a>	<->
Binary Search Trees	<a href="#">Construct BST from preorder traversal</a>	<->
Binary Search Trees	<a href="#">Convert Binary tree into BST</a>	<->
Binary Search Trees	<a href="#">Convert a normal BST into a Balanced BST</a>	<->
Binary Search Trees	<a href="#">Merge two BST [ V.V.V&gt;IMP ]</a>	<->
Binary Search Trees	<a href="#">Find Kth largest element in a BST</a>	<->
Binary Search Trees	<a href="#">Find Kth smallest element in a BST</a>	<->
Binary Search Trees	<a href="#">Count pairs from 2 BST whose sum is equal to given value "X"</a>	<->
Binary Search Trees	<a href="#">Find the median of BST in O(n) time and O(1) space</a>	<->
Binary Search Trees	<a href="#">Count BST nodes that lie in a given range</a>	<->
Binary Search Trees	<a href="#">Replace every element with the least greater element on its right</a>	<->
Binary Search Trees	<a href="#">Given "n" appointments, find the conflicting appointments</a>	<->
Binary Search Trees	<a href="#">Check preorder is valid or not</a>	<->
Binary Search Trees	<a href="#">Check whether BST contains Dead end</a>	<->
Binary Search Trees	<a href="#">Largest BST in a Binary Tree [ V.V.V.V.V IMP ]</a>	<->
Binary Search Trees	<a href="#">Flatten BST to sorted list</a>	<->
Greedy	<a href="#">Activity Selection Problem</a>	<->
Greedy	<a href="#">Job Sequencing Problem</a>	<->
Greedy	<a href="#">Huffman Coding</a>	<->
Greedy	<a href="#">Water Connection Problem</a>	<->
Greedy	<a href="#">Fractional Knapsack Problem</a>	<->
Greedy	<a href="#">Greedy Algorithm to find Minimum number of Coins</a>	<->
Greedy	<a href="#">Maximum trains for which stoppage can be provided</a>	<->
Greedy	<a href="#">Minimum Platforms Problem</a>	<->
Greedy	<a href="#">Buy Maximum Stocks if i stocks can be bought on i-th day</a>	<->

Greedy	<a href="#">Find the minimum and maximum amount to buy all N candies</a>	<->
Greedy	<a href="#">Minimize Cash Flow among a given set of friends who have borrowed money from each other</a>	<->
Greedy	<a href="#">Minimum Cost to cut a board into squares</a>	<->
Greedy	<a href="#">Check if it is possible to survive on Island</a>	<->
Greedy	<a href="#">Find maximum meetings in one room</a>	<->
Greedy	<a href="#">Maximum product subset of an array</a>	<->
Greedy	<a href="#">Maximize array sum after K negations</a>	<->
Greedy	<a href="#">Maximize the sum of arr[i]*i</a>	<->
Greedy	<a href="#">Maximum sum of absolute difference of an array</a>	<->
Greedy	<a href="#">Maximize sum of consecutive differences in a circular array</a>	<->
Greedy	<a href="#">Minimum sum of absolute difference of pairs of two arrays</a>	<->
Greedy	<a href="#">Program for Shortest Job First (or SJF) CPU Scheduling</a>	<->
Greedy	<a href="#">Program for Least Recently Used (LRU) Page Replacement algorithm</a>	<->
Greedy	<a href="#">Smallest subset with sum greater than all other elements</a>	<->
Greedy	<a href="#">Chocolate Distribution Problem</a>	<->
Greedy	<a href="#">DEFKIN -Defense of a Kingdom</a>	<->
Greedy	<a href="#">DIEHARD -DIE HARD</a>	<->
Greedy	<a href="#">GERGOVIA -Wine trading in Gergovia</a>	<->
Greedy	<a href="#">Picking Up Chicks</a>	<->
Greedy	<a href="#">CHOCOLA –Chocolate</a>	<->
Greedy	<a href="#">ARRANGE -Arranging Amplifiers</a>	<->
Greedy	<a href="#">K Centers Problem</a>	<->
Greedy	<a href="#">Minimum Cost of ropes</a>	<->
Greedy	<a href="#">Find smallest number with given number of digits and sum of digits</a>	<->
Greedy	<a href="#">Rearrange characters in a string such that no two adjacent are same</a>	<->
Greedy	<a href="#">Find maximum sum possible equal sum of three stacks</a>	<->
BackTracking	<a href="#">Rat in a maze Problem</a>	<->
BackTracking	<a href="#">Printing all solutions in N-Queen Problem</a>	<->
BackTracking	<a href="#">Word Break Problem using Backtracking</a>	<->
BackTracking	<a href="#">Remove Invalid Parentheses</a>	<->
BackTracking	<a href="#">Sudoku Solver</a>	<->
BackTracking	<a href="#">m Coloring Problem</a>	<->
BackTracking	<a href="#">Print all palindromic partitions of a string</a>	<->
BackTracking	<a href="#">Subset Sum Problem</a>	<->
BackTracking	<a href="#">The Knight's tour problem</a>	<->
BackTracking	<a href="#">Tug of War</a>	<->
BackTracking	<a href="#">Find shortest safe route in a path with landmines</a>	<->
BackTracking	<a href="#">Combinational Sum</a>	<->
BackTracking	<a href="#">Find Maximum number possible by doing at-most K swaps</a>	<->

BackTracking	<a href="#">Print all permutations of a string</a>	<->
BackTracking	<a href="#">Find if there is a path of more than k length from a source</a>	<->
BackTracking	<a href="#">Longest Possible Route in a Matrix with Hurdles</a>	<->
BackTracking	<a href="#">Print all possible paths from top left to bottom right of a mXn matrix</a>	<->
BackTracking	<a href="#">Partition of a set into K subsets with equal sum</a>	<->
BackTracking	<a href="#">Find the K-th Permutation Sequence of first N natural numbers</a>	<->
Stacks & Queues	<a href="#">Implement Stack from Scratch</a>	<->
Stacks & Queues	<a href="#">Implement Queue from Scratch</a>	<->
Stacks & Queues	<a href="#">Implement 2 stack in an array</a>	<->
Stacks & Queues	<a href="#">find the middle element of a stack</a>	<->
Stacks & Queues	<a href="#">Implement "N" stacks in an Array</a>	<->
Stacks & Queues	<a href="#">Check the expression has valid or Balanced parenthesis or not.</a>	<->
Stacks & Queues	<a href="#">Reverse a String using Stack</a>	<->
Stacks & Queues	<a href="#">Design a Stack that supports getMin() in O(1) time and O(1) extra space.</a>	<->
Stacks & Queues	<a href="#">Find the next Greater element</a>	<->
Stacks & Queues	<a href="#">The celebrity Problem</a>	<->
Stacks & Queues	<a href="#">Arithmetic Expression evaluation</a>	<->
Stacks & Queues	<a href="#">Evaluation of Postfix expression</a>	<->
Stacks & Queues	<a href="#">Implement a method to insert an element at its bottom without using any other data structure.</a>	<->
Stacks & Queues	<a href="#">Reverse a stack using recursion</a>	<->
Stacks & Queues	<a href="#">Sort a Stack using recursion</a>	<->
Stacks & Queues	<a href="#">Merge Overlapping Intervals</a>	<->
Stacks & Queues	<a href="#">Largest rectangular Area in Histogram</a>	<->
Stacks & Queues	<a href="#">Length of the Longest Valid Substring</a>	<->
Stacks & Queues	<a href="#">Expression contains redundant bracket or not</a>	<->
Stacks & Queues	<a href="#">Implement Stack using Queue</a>	<->
Stacks & Queues	<a href="#">Implement Stack using Deque</a>	<->
Stacks & Queues	<a href="#">Stack Permutations (Check if an array is stack permutation of other)</a>	<->
Stacks & Queues	<a href="#">Implement Queue using Stack</a>	<->
Stacks & Queues	<a href="#">Implement "n" queue in an array</a>	<->
Stacks & Queues	<a href="#">Implement a Circular queue</a>	<->
Stacks & Queues	<a href="#">LRU Cache Implementationa</a>	<->
Stacks & Queues	<a href="#">Reverse a Queue using recursion</a>	<->
Stacks & Queues	<a href="#">Reverse the first "K" elements of a queue</a>	<->
Stacks & Queues	<a href="#">Interleave the first half of the queue with second half</a>	<->
Stacks & Queues	<a href="#">Find the first circular tour that visits all Petrol Pumps</a>	<->
Stacks & Queues	<a href="#">Minimum time required to rot all oranges</a>	<->
Stacks & Queues	<a href="#">Distance of nearest cell having 1 in a binary matrix</a>	<->
Stacks & Queues	<a href="#">First negative integer in every window of size "k"</a>	<->



Stacks & Queues	<a href="#">Check if all levels of two trees are anagrams or not.</a>	<->
Stacks & Queues	<a href="#">Sum of minimum and maximum elements of all subarrays of size “k”.</a>	<->
Stacks & Queues	<a href="#">Minimum sum of squares of character counts in a given string after removing “k” characters.</a>	<->
Stacks & Queues	<a href="#">Queue based approach or first non-repeating character in a stream.</a>	<->
Stacks & Queues	<a href="#">Next Smaller Element</a>	<->
Heap	<a href="#">Implement a Maxheap/MinHeap using arrays and recursion.</a>	<->
Heap	<a href="#">Sort an Array using heap. (HeapSort)</a>	<->
Heap	<a href="#">Maximum of all subarrays of size k.</a>	<->
Heap	<a href="#">“k” largest element in an array</a>	<->
Heap	<a href="#">Kth smallest and largest element in an unsorted array</a>	<->
Heap	<a href="#">Merge “K” sorted arrays. [ IMP ]</a>	<->
Heap	<a href="#">Merge 2 Binary Max Heaps</a>	<->
Heap	<a href="#">Kth largest sum continuous subarrays</a>	<->
Heap	<a href="#">Leetcode- reorganize strings</a>	<->
Heap	<a href="#">Merge “K” Sorted Linked Lists [V.IMP]</a>	<->
Heap	<a href="#">Smallest range in “K” Lists</a>	<->
Heap	<a href="#">Median in a stream of Integers</a>	<->
Heap	<a href="#">Check if a Binary Tree is Heap</a>	<->
Heap	<a href="#">Connect “n” ropes with minimum cost</a>	<->
Heap	<a href="#">Convert BST to Min Heap</a>	<->
Heap	<a href="#">Convert min heap to max heap</a>	<->
Heap	<a href="#">Rearrange characters in a string such that no two adjacent are same.</a>	<->
Heap	<a href="#">Minimum sum of two numbers formed from digits of an array</a>	<->
Graph	<a href="#">Create a Graph, print it</a>	<->
Graph	<a href="#">Implement BFS algorithm.</a>	<->
Graph	<a href="#">Implement DFS Algo.</a>	<->
Graph	<a href="#">Detect Cycle in Directed Graph using BFS/DFS Algo.</a>	<->
Graph	<a href="#">Detect Cycle in UnDirected Graph using BFS/DFS Algo.</a>	<->
Graph	<a href="#">Search in a Maze</a>	<->
Graph	<a href="#">Minimum Step by Knight</a>	<->
Graph	<a href="#">flood fill algo</a>	<->
Graph	<a href="#">Clone a graph</a>	<->
Graph	<a href="#">Making wired Connections</a>	<->
Graph	<a href="#">word Ladder</a>	<->
Graph	<a href="#">Dijkstra algo</a>	<->
Graph	<a href="#">Implement Topological Sort</a>	<->
Graph	<a href="#">Minimum time taken by each job to be completed given by a Directed Acyclic Graph</a>	<->

Graph	<a href="#">Find whether it is possible to finish all tasks or not from given dependencies</a>	<->
Graph	<a href="#">Find the no. of Islands</a>	<->
Graph	<a href="#">Given a sorted Dictionary of an Alien Language, find order of characters</a>	<->
Graph	<a href="#">Implement Kruksal's Algorithm</a>	<->
Graph	<a href="#">Implement Prim's Algorithm</a>	<->
Graph	<a href="#">Total no. of Spanning tree in a graph</a>	<->
Graph	<a href="#">Implement Bellman Ford Algorithm</a>	<->
Graph	<a href="#">Implement Floyd warshall Algorithm</a>	<->
Graph	<a href="#">Travelling Salesman Problem</a>	<->
Graph	<a href="#">Graph Colouring Problem</a>	<->
Graph	<a href="#">Snake and Ladders Problem</a>	<->
Graph	<a href="#">Find bridge in a graph</a>	<->
Graph	<a href="#">Count Strongly connected Components(Kosaraju Algo)</a>	<->
Graph	<a href="#">Check whether a graph is Bipartite or Not</a>	<->
Graph	<a href="#">Detect Negative cycle in a graph</a>	<->
Graph	<a href="#">Longest path in a Directed Acyclic Graph</a>	<->
Graph	<a href="#">Journey to the Moon</a>	<->
Graph	<a href="#">Cheapest Flights Within K Stops</a>	<->
Graph	<a href="#">Oliver and the Game</a>	<->
Graph	<a href="#">Water Jug problem using BFS</a>	<->
Graph	<a href="#">Water Jug problem using BFS</a>	<->
Graph	<a href="#">Find if there is a path of more than length from a source</a>	<->
Graph	<a href="#">M-Colouring Problem</a>	<->
Graph	<a href="#">Minimum edges to reverse to make path from source to destination</a>	<->
Graph	<a href="#">Paths to travel each node using each edge(Seven Bridges)</a>	<->
Graph	<a href="#">Vertex Cover Problem</a>	<->
Graph	<a href="#">Chinese Postman or Route Inspection</a>	<->
Graph	<a href="#">Number of Triangles in a Directed and Undirected Graph</a>	<->
Graph	<a href="#">Minimise the cashflow among a given set of friends who have borrowed money from each other</a>	<->
Graph	<a href="#">Two Clique Problem</a>	<->
Trie	<a href="#">Construct a trie from scratch</a>	<->
Trie	<a href="#">Find shortest unique prefix for every word in a given list</a>	<->
Trie	<a href="#">Word Break Problem   (Trie solution)</a>	<->
Trie	<a href="#">Given a sequence of words, print all anagrams together</a>	<->
Trie	<a href="#">Implement a Phone Directory</a>	<->
Trie	<a href="#">Print unique rows in a given boolean matrix</a>	<->

Dynamic Programming	<a href="#">Coin Change Problem</a>	<->
---------------------	-------------------------------------	-----

Dynamic Programming	<a href="#">Knapsack Problem</a>	<->
Dynamic Programming	<a href="#">Binomial Coefficient Problem</a>	<->
Dynamic Programming	<a href="#">Permutation Coefficient Problem</a>	<->
Dynamic Programming	<a href="#">Program for nth Catalan Number</a>	<->
Dynamic Programming	<a href="#">Matrix Chain Multiplication</a>	<->
Dynamic Programming	<a href="#">Edit Distance</a>	<->
Dynamic Programming	<a href="#">Subset Sum Problem</a>	<->
Dynamic Programming	<a href="#">Friends Pairing Problem</a>	<->
Dynamic Programming	<a href="#">Gold Mine Problem</a>	<->
Dynamic Programming	<a href="#">Assembly Line Scheduling Problem</a>	<->
Dynamic Programming	<a href="#">Painting the Fence problem</a>	<->
Dynamic Programming	<a href="#">Maximize The Cut Segments</a>	<->
Dynamic Programming	<a href="#">Longest Common Subsequence</a>	<->
Dynamic Programming	<a href="#">Longest Repeated Subsequence</a>	<->
Dynamic Programming	<a href="#">Longest Increasing Subsequence</a>	<->
Dynamic Programming	<a href="#">Space Optimized Solution of LCS</a>	<->
Dynamic Programming	<a href="#">LCS (Longest Common Subsequence) of three strings</a>	<->
Dynamic Programming	<a href="#">Maximum Sum Increasing Subsequence</a>	<->
Dynamic Programming	<a href="#">Count all subsequences having product less than K</a>	<->
Dynamic Programming	<a href="#">Longest subsequence such that difference between adjacent is one</a>	<->
Dynamic Programming	<a href="#">Maximum subsequence sum such that no three are consecutive</a>	<->
Dynamic Programming	<a href="#">Egg Dropping Problem</a>	<->
Dynamic Programming	<a href="#">Maximum Length Chain of Pairs</a>	<->
Dynamic Programming	<a href="#">Maximum size square sub-matrix with all 1s</a>	<->
Dynamic Programming	<a href="#">Maximum sum of pairs with specific difference</a>	<->
Dynamic Programming	<a href="#">Min Cost Path Problem</a>	<->
Dynamic Programming	<a href="#">Maximum difference of zeros and ones in binary string</a>	<->
Dynamic Programming	<a href="#">Minimum number of jumps to reach end</a>	<->
Dynamic Programming	<a href="#">Minimum cost to fill given weight in a bag</a>	<->
Dynamic Programming	<a href="#">Minimum removals from array to make max – min &lt;= K</a>	<->
Dynamic Programming	<a href="#">Longest Common Substring</a>	<->
Dynamic Programming	<a href="#">Count number of ways to reach a given score in a game</a>	<->
Dynamic Programming	<a href="#">Count Balanced Binary Trees of Height h</a>	<->
Dynamic Programming	<a href="#">Largest Sum Contiguous Subarray [V &gt; V &gt; V &gt; V IMP ]</a>	<->
Dynamic Programming	<a href="#">Smallest sum contiguous subarray</a>	<->
Dynamic Programming	<a href="#">Unbounded Knapsack (Repetition of items allowed)</a>	<->
Dynamic Programming	<a href="#">Word Break Problem</a>	<->
Dynamic Programming	<a href="#">Largest Independent Set Problem</a>	<->
Dynamic Programming	<a href="#">Partition problem</a>	<->
Dynamic Programming	<a href="#">Longest Palindromic Subsequence</a>	<->

Dynamic Programming	<a href="#">Count All Palindromic Subsequence in a given String</a>	<->
Dynamic Programming	<a href="#">Longest Palindromic Substring</a>	<->
Dynamic Programming	<a href="#">Longest alternating subsequence</a>	<->
Dynamic Programming	<a href="#">Weighted Job Scheduling</a>	<->
Dynamic Programming	<a href="#">Coin game winner where every player has three choices</a>	<->
Dynamic Programming	<a href="#">Count Derangements (Permutation such that no element appears in its original position) [ IMPORTANT ]</a>	<->
Dynamic Programming	<a href="#">Maximum profit by buying and selling a share at most twice [ IMP ]</a>	<->
Dynamic Programming	<a href="#">Optimal Strategy for a Game</a>	<->
Dynamic Programming	<a href="#">Optimal Binary Search Tree</a>	<->
Dynamic Programming	<a href="#">Palindrome Partitioning Problem</a>	<->
Dynamic Programming	<a href="#">Word Wrap Problem</a>	<->
Dynamic Programming	<a href="#">Mobile Numeric Keypad Problem [ IMP ]</a>	<->
Dynamic Programming	<a href="#">Boolean Parenthesization Problem</a>	<->
Dynamic Programming	<a href="#">Largest rectangular sub-matrix whose sum is 0</a>	<->
Dynamic Programming	<a href="#">Largest area rectangular sub-matrix with equal number of 1's and 0's [ IMP ]</a>	<->
Dynamic Programming	<a href="#">Maximum sum rectangle in a 2D matrix</a>	<->
Dynamic Programming	<a href="#">Maximum profit by buying and selling a share at most k times</a>	<->
Dynamic Programming	<a href="#">Find if a string is interleaved of two other strings</a>	<->
Dynamic Programming	<a href="#">Maximum Length of Pair Chain</a>	<->
Bit Manipulation	<a href="#">Count set bits in an integer</a>	<->
Bit Manipulation	<a href="#">Find the two non-repeating elements in an array of repeating elements</a>	<->
Bit Manipulation	<a href="#">Count number of bits to be flipped to convert A to B</a>	<->
Bit Manipulation	<a href="#">Count total set bits in all numbers from 1 to n</a>	<->
Bit Manipulation	<a href="#">Program to find whether a no is power of two</a>	<->
Bit Manipulation	<a href="#">Find position of the only set bit</a>	<->
Bit Manipulation	<a href="#">Copy set bits in a range</a>	<->
Bit Manipulation	<a href="#">Divide two integers without using multiplication, division and mod operator</a>	<->
Bit Manipulation	<a href="#">Calculate square of a number without using *, / and pow()</a>	<->
Bit Manipulation	<a href="#">Power Set</a>	<->