

Topic:

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Array

Matrix

Matrix

Matrix

Matrix

Matrix

Matrix

Matrix

Matrix

Matrix

Matrix

String

String

String

String

String

String

String

String

String

String

String

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

Searching & Sorting

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

LinkedList

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Binary Search Trees

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy

Greedy
Greedy
Greedy
Greedy
Greedy
Greedy
Greedy
Greedy

BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking
BackTracking

Stacks & Queues
Stacks & Queues

Stacks & Queues

Stacks & Queues

Stacks & Queues

Stacks & Queues

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Heap

Graph

Graph

Graph

Graph

Graph

Graph

Graph

Graph

Graph

Graph

Graph

Trie

Trie

Trie

Trie

Trie

Trie

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

Dynamic Programming

[illegible]

Dynamic Programming

Bit Manipulation

Questions by Love Babbar:

Youtube Channel: <https://www.youtube.com/channel/UCQHLxxBFrbfdrk1jF0moTpw>

Problem:

Reverse the array

Find the maximum and minimum element in an array

Find the "Kth" max and min element of an array

Given an array which consists of only 0, 1 and 2. Sort the array without using a

Move all the negative elements to one side of the array

Find the Union and Intersection of the two sorted arrays.

Write a program to cyclically rotate an array by one.

find Largest sum contiguous Subarray [V. IMP]

Minimise the maximum difference between heights [V.IMP]

Minimum no. of Jumps to reach end of an array

find duplicate in an array of N+1 Integers

Merge 2 sorted arrays without using Extra space.

Kadane's Algo [V.V.V.V.V IMP]

Merge Intervals

Next Permutation

Count Inversion

Best time to buy and Sell stock

find all pairs on integer array whose sum is equal to given number

find common elements In 3 sorted arrays

Rearrange the array in alternating positive and negative items with O(1) extra s

Find if there is any subarray with sum equal to 0

Find factorial of a large number

find maximum product subarray

Find longest coinsecutive subsequence

Given an array of size n and a number k, fin all elements that appear more than

Maximum profit by buying and selling a share atmost twice

Find whether an array is a subset of another array

Find the triplet that sum to a given value

[Trapping Rain water problem](#)

[Chocolate Distribution problem](#)

[Smallest Subarray with sum greater than a given value](#)

[Three way partitioning of an array around a given value](#)

[Minimum swaps required bring elements less equal K together](#)

[Minimum no. of operations required to make an array palindrome](#)

[Median of 2 sorted arrays of equal size](#)

[Median of 2 sorted arrays of different size](#)

[Spiral traversal on a Matrix](#)

[Search an element in a matrix](#)

[Find median in a row wise sorted matrix](#)

[Find row with maximum no. of 1's](#)

[Print elements in sorted order using row-column wise sorted matrix](#)

[Maximum size rectangle](#)

[Find a specific pair in matrix](#)

[Rotate matrix by 90 degrees](#)

[Kth smallest element in a row-column wise sorted matrix](#)

[Common elements in all rows of a given matrix](#)

[Reverse a String](#)

[Check whether a String is Palindrome or not](#)

[Find Duplicate characters in a string](#)

[Why strings are immutable in Java?](#)

[Write a Code to check whether one string is a rotation of another](#)

[Write a Program to check whether a string is a valid shuffle of two strings or not](#)

[Count and Say problem](#)

[Write a program to find the longest Palindrome in a string.\[Longest palindrom](#)

[Find Longest Recurring Subsequence in String](#)

[Print all Subsequences of a string.](#)

[Print all the permutations of the given string](#)

[Split the Binary string into two substring with equal 0's and 1's](#)

[Word Wrap Problem \[VERY IMP\].](#)

[EDIT Distance \[Very Imp\]](#)

[Find next greater number with same set of digits. \[Very Very IMP\]](#)

[Balanced Parenthesis problem.\[Imp\]](#)

[Word break Problem\[Very Imp\]](#)

[Rabin Karp Algo](#)

[KMP Algo](#)

[Convert a Sentence into its equivalent mobile numeric keypad sequence.](#)

[Minimum number of bracket reversals needed to make an expression balanced](#)

[Count All Palindromic Subsequence in a given String.](#)

[Count of number of given string in 2D character array](#)

[Search a Word in a 2D Grid of characters.](#)

[Boyer Moore Algorithm for Pattern Searching.](#)

[Converting Roman Numerals to Decimal](#)

[Longest Common Prefix](#)

[Number of flips to make binary string alternate](#)

[Find the first repeated word in string.](#)

[Minimum number of swaps for bracket balancing.](#)

[Find the longest common subsequence between two strings.](#)

[Program to generate all possible valid IP addresses from given string.](#)

[Write a program to find the smallest window that contains all characters of string](#)

[Rearrange characters in a string such that no two adjacent are same](#)

[Minimum characters to be added at front to make string palindrome](#)

[Given a sequence of words, print all anagrams together](#)

[Find the smallest window in a string containing all characters of another string](#)

[Recursively remove all adjacent duplicates](#)

[String matching where one string contains wildcard characters](#)

[Function to find Number of customers who could not get a computer](#)

[Transform One String to Another using Minimum Number of Given Operation](#)

[Check if two given strings are isomorphic to each other](#)

[Recursively print all sentences that can be formed from list of word lists](#)

[Find first and last positions of an element in a sorted array](#)

[Find a Fixed Point \(Value equal to index\) in a given array](#)

[Search in a rotated sorted array](#)

[square root of an integer](#)

[Maximum and minimum of an array using minimum number of comparisons](#)

[Optimum location of point to minimize total distance](#)

[Find the repeating and the missing](#)

[find majority element](#)

[Searching in an array where adjacent differ by at most k](#)

[find a pair with a given difference](#)

[find four elements that sum to a given value](#)

[maximum sum such that no 2 elements are adjacent](#)

[Count triplet with sum smaller than a given value](#)

[merge 2 sorted arrays](#)

[print all subarrays with 0 sum](#)

[Product array Puzzle](#)

[Sort array according to count of set bits](#)

[minimum no. of swaps required to sort the array](#)

[Bishu and Soldiers](#)

[Rasta and Kheshtak](#)

[Kth smallest number again](#)

[Find pivot element in a sorted array](#)

[K-th Element of Two Sorted Arrays](#)

[Aggressive cows](#)

[Book Allocation Problem](#)

[EKOSPOJ:](#)

[Job Scheduling Algo](#)

[Missing Number in AP](#)

[Smallest number with atleastn trailing zeroes infactorial](#)

[Painters Partition Problem:](#)

[ROTI-Prata SPOJ](#)

[DoubleHelix SPOJ](#)

Subset Sums

Find the inversion count

Implement Merge-sort in-place

Partitioning and Sorting Arrays with Many Repeated Entries

Write a Program to reverse the Linked List. (Both Iterative and recursive)

Reverse a Linked List in group of Given Size. [Very Imp]

Write a program to Detect loop in a linked list.

Write a program to Delete loop in a linked list.

Find the starting point of the loop.

Remove Duplicates in a sorted Linked List.

Remove Duplicates in a Un-sorted Linked List.

Write a Program to Move the last element to Front in a Linked List.

Add "1" to a number represented as a Linked List.

Add two numbers represented by linked lists.

Intersection of two Sorted Linked List.

Intersection Point of two Linked Lists.

Merge Sort For Linked lists.[Very Important]

Quicksort for Linked Lists.[Very Important]

Find the middle Element of a linked list.

Check if a linked list is a circular linked list.

Split a Circular linked list into two halves.

Write a Program to check whether the Singly Linked list is a palindrome or not.

Deletion from a Circular Linked List.

Reverse a Doubly Linked list.

Find pairs with a given sum in a DLL.

Count triplets in a sorted DLL whose sum is equal to given value "X".

Sort a "k" sorted Doubly Linked list.[Very IMP]

Rotate Doubly Linked list by N nodes.

Rotate a Doubly Linked list in group of Given Size.[Very IMP]

Can we reverse a linked list in less than $O(n)$?

Why Quicksort is preferred for. Arrays and Merge Sort for LinkedLists ?

[Flatten a Linked List](#)

[Sort a LL of 0's, 1's and 2's](#)

[Clone a linked list with next and random pointer](#)

[Merge K sorted Linked list](#)

[Multiply 2 no. represented by LL](#)

[Delete nodes which have a greater value on right side](#)

[Segregate even and odd nodes in a Linked List](#)

[Program for n'th node from the end of a Linked List](#)

[Find the first non-repeating character from a stream of characters](#)

[level order traversal](#)

[Reverse Level Order traversal](#)

[Height of a tree](#)

[Diameter of a tree](#)

[Mirror of a tree](#)

[Inorder Traversal of a tree both using recursion and Iteration](#)

[Preorder Traversal of a tree both using recursion and Iteration](#)

[Postorder Traversal of a tree both using recursion and Iteration](#)

[Left View of a tree](#)

[Right View of Tree](#)

[Top View of a tree](#)

[Bottom View of a tree](#)

[Zig-Zag traversal of a binary tree](#)

[Check if a tree is balanced or not](#)

[Diagnol Traversal of a Binary tree](#)

[Boundary traversal of a Binary tree](#)

[Construct Binary Tree from String with Bracket Representation](#)

[Convert Binary tree into Doubly Linked List](#)

[Convert Binary tree into Sum tree](#)

[Construct Binary tree from Inorder and preorder traversal](#)

[Find minimum swaps required to convert a Binary tree into BST](#)

[Check if Binary tree is Sum tree or not](#)

[Check if all leaf nodes are at same level or not](#)
[Check if a Binary Tree contains duplicate subtrees of size 2 or more \[IMP \]](#)
[Check if 2 trees are mirror or not](#)
[Sum of Nodes on the Longest path from root to leaf node](#)
[Check if given graph is tree or not. \[IMP \]](#)
[Find Largest subtree sum in a tree](#)
[Maximum Sum of nodes in Binary tree such that no two are adjacent](#)
[Print all "K" Sum paths in a Binary tree](#)
[Find LCA in a Binary tree](#)
[Find distance between 2 nodes in a Binary tree](#)
[Kth Ancestor of node in a Binary tree](#)
[Find all Duplicate subtrees in a Binary tree \[IMP \]](#)
[Tree Isomorphism Problem](#)

[Find a value in a BST](#)
[Deletion of a node in a BST](#)
[Find min and max value in a BST](#)
[Find inorder successor and inorder predecessor in a BST](#)
[Check if a tree is a BST or not](#)
[Populate Inorder successor of all nodes](#)
[Find LCA of 2 nodes in a BST](#)
[Construct BST from preorder traversal](#)
[Convert Binary tree into BST](#)
[Convert a normal BST into a Balanced BST](#)
[Merge two BST \[V.V.V>IMP \]](#)
[Find Kth largest element in a BST](#)
[Find Kth smallest element in a BST](#)
[Count pairs from 2 BST whose sum is equal to given value "X"](#)
[Find the median of BST in O\(n\) time and O\(1\) space](#)
[Count BST nodes that lie in a given range](#)
[Replace every element with the least greater element on its right](#)
[Given "n" appointments, find the conflicting appointments](#)

[Check preorder is valid or not](#)
[Check whether BST contains Dead end](#)
[Largest BST in a Binary Tree \[V.V.V.V.V IMP \]](#)
[Flatten BST to sorted list](#)

[Activity Selection Problem](#)
[Job Sequencing Problem](#)
[Huffman Coding](#)
[Water Connection Problem](#)
[Fractional Knapsack Problem](#)
[Greedy Algorithm to find Minimum number of Coins](#)
[Maximum trains for which stoppage can be provided](#)
[Minimum Platforms Problem](#)
[Buy Maximum Stocks if i stocks can be bought on i-th day](#)
[Find the minimum and maximum amount to buy all N candies](#)
[Minimize Cash Flow among a given set of friends who have borrowed money f](#)
[Minimum Cost to cut a board into squares](#)
[Check if it is possible to survive on Island](#)
[Find maximum meetings in one room](#)
[Maximum product subset of an array](#)
[Maximize array sum after K negations](#)
[Maximize the sum of \$arr\[i\]*i\$](#)
[Maximum sum of absolute difference of an array](#)
[Maximize sum of consecutive differences in a circular array](#)
[Minimum sum of absolute difference of pairs of two arrays](#)
[Program for Shortest Job First \(or SJF\) CPU Scheduling](#)
[Program for Least Recently Used \(LRU\) Page Replacement algorithm](#)
[Smallest subset with sum greater than all other elements](#)
[Chocolate Distribution Problem](#)
[DEFKIN -Defense of a Kingdom](#)
[DIEHARD -DIE HARD](#)
[GERGOVIA -Wine trading in Gergovia](#)

[Picking Up Chicks](#)

[CHOCOLA –Chocolate](#)

[ARRANGE -Arranging Amplifiers](#)

[K Centers Problem](#)

[Minimum Cost of ropes](#)

[Find smallest number with given number of digits and sum of digits](#)

[Rearrange characters in a string such that no two adjacent are same](#)

[Find maximum sum possible equal sum of three stacks](#)

[Rat in a maze Problem](#)

[Printing all solutions in N-Queen Problem](#)

[Word Break Problem using Backtracking](#)

[Remove Invalid Parentheses](#)

[Sudoku Solver](#)

[m Coloring Problem](#)

[Print all palindromic partitions of a string](#)

[Subset Sum Problem](#)

[The Knight's tour problem](#)

[Tug of War](#)

[Find shortest safe route in a path with landmines](#)

[Combinational Sum](#)

[Find Maximum number possible by doing at-most K swaps](#)

[Print all permutations of a string](#)

[Find if there is a path of more than k length from a source](#)

[Longest Possible Route in a Matrix with Hurdles](#)

[Print all possible paths from top left to bottom right of a mXn matrix](#)

[Partition of a set intoK subsets with equal sum](#)

[Find the K-th Permutation Sequence of first N natural numbers](#)

[Implement Stack from Scratch](#)

[Implement Queue from Scratch](#)

[Implement 2 stack in an array](#)

[find the middle element of a stack](#)

[Implement "N" stacks in an Array](#)

[Check the expression has valid or Balanced parenthesis or not.](#)

[Reverse a String using Stack](#)

[Design a Stack that supports getMin\(\) in O\(1\) time and O\(1\) extra space.](#)

[Find the next Greater element](#)

[The celebrity Problem](#)

[Arithmetic Expression evaluation](#)

[Evaluation of Postfix expression](#)

[Implement a method to insert an element at its bottom without using any other](#)

[Reverse a stack using recursion](#)

[Sort a Stack using recursion](#)

[Merge Overlapping Intervals](#)

[Largest rectangular Area in Histogram](#)

[Length of the Longest Valid Substring](#)

[Expression contains redundant bracket or not](#)

[Implement Stack using Queue](#)

[Implement Stack using Deque](#)

[Stack Permutations \(Check if an array is stack permutation of other\)](#)

[Implement Queue using Stack](#)

[Implement "n" queue in an array](#)

[Implement a Circular queue](#)

[LRU Cache Implementation](#)

[Reverse a Queue using recursion](#)

[Reverse the first "K" elements of a queue](#)

[Interleave the first half of the queue with second half](#)

[Find the first circular tour that visits all Petrol Pumps](#)

[Minimum time required to rot all oranges](#)

[Distance of nearest cell having 1 in a binary matrix](#)

[First negative integer in every window of size "k"](#)

[Check if all levels of two trees are anagrams or not.](#)

[Sum of minimum and maximum elements of all subarrays of size "k".](#)

[Minimum sum of squares of character counts in a given string after removing “](#)
[Queue based approach or first non-repeating character in a stream.](#)
[Next Smaller Element](#)

[Implement a Maxheap/MinHeap using arrays and recursion.](#)

[Sort an Array using heap. \(HeapSort\)](#)

[Maximum of all subarrays of size k.](#)

[“k” largest element in an array](#)

[Kth smallest and largest element in an unsorted array](#)

[Merge “K” sorted arrays. \[IMP \]](#)

[Merge 2 Binary Max Heaps](#)

[Kth largest sum continuous subarrays](#)

[Leetcode- reorganize strings](#)

[Merge “K” Sorted Linked Lists \[V.IMP\]](#)

[Smallest range in “K” Lists](#)

[Median in a stream of Integers](#)

[Check if a Binary Tree is Heap](#)

[Connect “n” ropes with minimum cost](#)

[Convert BST to Min Heap](#)

[Convert min heap to max heap](#)

[Rearrange characters in a string such that no two adjacent are same.](#)

[Minimum sum of two numbers formed from digits of an array](#)

[Create a Graph, print it](#)

[Implement BFS algorithm](#)

[Implement DFS Algo](#)

[Detect Cycle in Directed Graph using BFS/DFS Algo](#)

[Detect Cycle in UnDirected Graph using BFS/DFS Algo](#)

[Search in a Maze](#)

[Minimum Step by Knight](#)

[flood fill algo](#)

[Clone a graph](#)

[Making wired Connections](#)

[word Ladder](#)

[Dijkstra algo](#)

[Implement Topological Sort](#)

[Minimum time taken by each job to be completed given by a Directed Acyclic G](#)

[Find whether it is possible to finish all tasks or not from given dependencies](#)

[Find the no. of Islands](#)

[Given a sorted Dictionary of an Alien Language, find order of characters](#)

[Implement Kruksal's Algorithm](#)

[Implement Prim's Algorithm](#)

[Total no. of Spanning tree in a graph](#)

[Implement Bellman Ford Algorithm](#)

[Implement Floyd warshall Algorithm](#)

[Travelling Salesman Problem](#)

[Graph Colouring Problem](#)

[Snake and Ladders Problem](#)

[Find bridge in a graph](#)

[Count Strongly connected Components\(Kosaraju Algo\)](#)

[Check whether a graph is Bipartite or Not](#)

[Detect Negative cycle in a graph](#)

[Longest path in a Directed Acyclic Graph](#)

[Journey to the Moon](#)

[Cheapest Flights Within K Stops](#)

[Oliver and the Game](#)

[Water Jug problem using BFS](#)

[Water Jug problem using BFS](#)

[Find if there is a path of more than length from a source](#)

[M-Colouring Problem](#)

[Minimum edges to reverse to make path from source to destination](#)

[Paths to travel each node using each edge\(Seven Bridges\)](#)

[Vertex Cover Problem](#)

[Chinese Postman or Route Inspection](#)

[Number of Triangles in a Directed and Undirected Graph](#)

[Minimise the cashflow among a given set of friends who have borrowed money](#)

[Two Clique Problem](#)

[Construct a trie from scratch](#)

[Find shortest unique prefix for every word in a given list](#)

[Word Break Problem | \(Trie solution\)](#)

[Given a sequence of words, print all anagrams together](#)

[Implement a Phone Directory](#)

[Print unique rows in a given boolean matrix](#)

[Coin Change Problem](#)

[Knapsack Problem](#)

[Binomial Coefficient Problem](#)

[Permutation Coefficient Problem](#)

[Program for nth Catalan Number](#)

[Matrix Chain Multiplication](#)

[Edit Distance](#)

[Subset Sum Problem](#)

[Friends Pairing Problem](#)

[Gold Mine Problem](#)

[Assembly Line Scheduling Problem](#)

[Painting the Fence problem](#)

[Maximize The Cut Segments](#)

[Longest Common Subsequence](#)

[Longest Repeated Subsequence](#)

[Longest Increasing Subsequence](#)

[Space Optimized Solution of LCS](#)

[LCS \(Longest Common Subsequence\) of three strings](#)

[Maximum Sum Increasing Subsequence](#)

[Count all subsequences having product less than K](#)

[Longest subsequence such that difference between adjacent is one](#)

[Maximum subsequence sum such that no three are consecutive](#)

[Egg Dropping Problem](#)

[Maximum Length Chain of Pairs](#)

[Maximum size square sub-matrix with all 1s](#)

[Maximum sum of pairs with specific difference](#)

[Min Cost Path Problem](#)

[Maximum difference of zeros and ones in binary string](#)

[Minimum number of jumps to reach end](#)

[Minimum cost to fill given weight in a bag](#)

[Minimum removals from array to make \$\max - \min \leq K\$](#)

[Longest Common Substring](#)

[Count number of ways to reach a given score in a game](#)

[Count Balanced Binary Trees of Height h](#)

[Largest Sum Contiguous Subarray \[V>V>V>V IMP\]](#)

[Smallest sum contiguous subarray](#)

[Unbounded Knapsack \(Repetition of items allowed\)](#)

[Word Break Problem](#)

[Largest Independent Set Problem](#)

[Partition problem](#)

[Longest Palindromic Subsequence](#)

[Count All Palindromic Subsequence in a given String](#)

[Longest Palindromic Substring](#)

[Longest alternating subsequence](#)

[Weighted Job Scheduling](#)

[Coin game winner where every player has three choices](#)

[Count Derangements \(Permutation such that no element appears in its original position\)](#)

[Maximum profit by buying and selling a share at most twice \[IMP\]](#)

[Optimal Strategy for a Game](#)

[Optimal Binary Search Tree](#)

[Palindrome Partitioning Problem](#)

[Word Wrap Problem](#)

[Mobile Numeric Keypad Problem \[IMP\]](#)

Boolean Parenthesization Problem

Largest rectangular sub-matrix whose sum is 0

Largest area rectangular sub-matrix with equal number of 1's and 0's [IMP]

Maximum sum rectangle in a 2D matrix

Maximum profit by buying and selling a share at most k times

Find if a string is interleaved of two other strings

Maximum Length of Pair Chain

Count set bits in an integer

Find the two non-repeating elements in an array of repeating elements

Count number of bits to be flipped to convert A to B

Count total set bits in all numbers from 1 to n

Program to find whether a no is power of two

Find position of the only set bit

Copy set bits in a range

Divide two integers without using multiplication, division and mod operator

Calculate square of a number without using *, / and pow()

Power Set

Done [yes or no]

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->



<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->

<->