

ARRAYS

1. [Find 2 elements with given sum](#)
2. [Majority Element](#)
3. [Find the number occurring odd number of times](#)
4. [Merge an array of size n into another of size m + n](#)
5. [Rotate an array](#)
6. [Leaders in an array](#)
7. [Majority element in sorted array](#)
8. [Segregate 0s and 1s in an array](#)
9. [Product array](#)
10. [Find 2 repeating elements](#)
11. [Find the smallest missing number](#)
12. [Find max j-i such that arr\[j\] > arr\[i\]](#)
13. [Find subarray with given sum](#)
14. [Find the smallest positive number missing from an unsorted array](#)
15. [Find 2 numbers with odd occurrence](#)
16. [Largest subarray with equal number of 0s and 1s](#)
17. [Replace every element with the greatest on right side](#)
18. [Stock buy sell to maximize profit](#)
19. [Find common elements in 3 sorted arrays](#)
20. [Nuts and bolts problem](#)
21. [Trapping rain water](#)
22. [Merge 2 sorted arrays in O\(1\) space](#)

STRINGS

1. [Remove duplicates from string](#)
2. [Remove characters from the first string which are present in the second string](#)
3. [Check if strings are rotations of each other](#)
4. [Print all permutations of a given string](#)
5. [Reverse words in a given string](#)
6. [Find the smallest window in a string containing all the characters of the second string](#)
7. [Check whether two strings are anagrams of each other](#)
8. [Write your own atoi\(\)](#)
9. [Rearrange a string so that similar characters become d distance away](#)
10. [Find excel column name from a given column number](#)

LINKED LIST

1. [Get Nth node in a linked list](#)
2. [Delete a node given a pointer to it](#)
3. [Print middle](#)
4. [Find Nth node from the end](#)
5. [Delete linked list](#)
6. [Reverse linked list](#)
7. [Detect loop in a linked list - McKinsey](#)
8. [Check if a singly linked list is a palindrome](#)
9. [Clone a linked list with next and random pointer](#)
10. [Memory efficient doubly linked list](#)
11. [Insert in sorted linked list](#)
12. [Get intersection point of 2 linked lists](#)
13. [Print reverse of a linked list](#)
14. [Remove duplicates from sorted linked list](#)
15. [Remove duplicates from unsorted linked list](#)
16. [Reverse doubly linked list](#)
17. [Merge 2 sorted linked lists](#)
18. [Merge sort for linked lists](#)
19. [Reverse a linked list in groups of given size](#)
20. [Linked list vs Array](#)
21. [Sorted insert for circular linked list](#)
22. [Detect and remove loop in a linked list](#)
23. [Add 2 numbers represented by linked lists](#)
24. [Clone a linked list with next and random pointer | Set 2](#)

TREES

1. [Recursive Tree Traversals](#)
2. [Calculate size of tree](#)
3. [Check if two trees are identical](#)
4. [Height of tree](#)
5. [Delete a tree](#)
6. [Convert a binary tree to its mirror tree](#)
7. [Given two traversal sequences, construct the binary tree](#)
8. [Print all root to leaf paths in a binary tree](#)
9. [Lowest common ancestor in BST](#)
10. [Level order traversal](#)
11. [Count leaf nodes](#)
12. [Spiral level order traversal](#)

13. [Diameter of tree](#)
14. [Inorder traversal without recursion](#)
15. [Root to leaf path sum equal to given number](#)
16. [Construct tree from inorder and preorder traversal](#)
17. [Print nodes at k distance from root](#)
18. [Applications of tree](#)
19. [Check if a binary tree is a subtree of another binary tree](#)
20. [Find inorder successor for all nodes](#)
21. [Vertical sum in a given binary tree](#)
22. [Maximum sum root to leaf path](#)
23. [Check if a binary tree is complete or not](#)
24. [Iterative preorder traversal](#)
25. [Iterative postorder traversal](#)
26. [Reverse level order traversal](#)
27. [Binary tree to doubly linked list](#) OR [Binary tree to double linked list](#)
28. [Find height of tree iteratively](#)
29. [Left view of binary tree](#)
30. [Lowest common ancestor binary tree](#)
31. [Print all nodes at k distance from given node](#)
32. [Right view of binary tree](#)
33. [Check if binary tree is subtree of another binary tree](#)
34. [Print nodes b/w two given levels](#)
35. [Find node with min value in BST](#)
36. [Check if a binary tree is BST](#)
37. [Find kth smallest element in BST](#)
38. [Sorted linked list to balanced BST](#)
39. [Kth largest element in BST](#)
40. [Advantages of BST over hash table](#)
41. [Kth smallest element in BST using O\(1\) space](#)

STACK

1. [Implement queue using stack](#)
2. [Check for balanced parentheses in an expression](#)
3. [Reverse a string using recursion](#)
4. [Design and implement special stack](#)
5. [Implement stack using queues](#)
6. [Expression evaluation](#)

GRAPH

1. [Applications of MST](#)
2. [Applications of DFS](#)
3. [DFS](#)
4. [BFS](#)
5. [Detect cycle in a directed graph](#)
6. [Find if there is a path b/w two vertices in a directed graph](#)
7. [Floyd Warshall Algorithm](#)
8. [Detect cycle in undirected graph](#)
9. [Kruskal's Algorithm](#)
10. [Graph and its representations](#)
11. [Prim's algorithm](#)
12. [Prim's algorithm 2](#)
13. [Dijkstra's algorithm](#)
14. [Dijkstra's algorithm 2](#)
15. [Bellman-Ford Algorithm](#)
16. [Transitive closure of a graph](#)
17. [Topological sorting](#)
18. [Shortest path in directed acyclic graph](#)
19. [Strongly connected components](#)
20. [Connectivity in directed graph](#)
21. [Detect cycle in an undirected graph 2](#)
22. [Applications of BFS](#)

MATRIX

1. [Maximum size square submatrix with all 1s](#)
2. [Turn an image by 90 degree](#)
3. [Search in a row wise and column wise sorted matrix](#)
4. [Print a given matrix in spiral form](#)
5. [A boolean matrix question](#)
6. [Min cost path](#)
7. [Find the row with maximum number of 1s](#)
8. [Find the number of islands](#)
9. [Maximum sum rectangle in a 2D matrix](#)
10. [Rotate matrix clockwise](#)
11. [Given a boolean matrix. Find k such that all elements in the kth row are 0 and the kth column are 1](#)
12. [Maximum size rectangle binary submatrix with all 1s](#)

QUEUE

1. [Level order traversal](#)
2. [Spiral level order traversal](#)
3. [Implement queue using stacks](#)
4. [Applications of queue](#)
5. [Implement stack using queues](#)
6. [First circular tour that visits all petrol pumps](#)
7. [Iterative height of binary tree](#)

HEAP

1. [k largest elements in an array](#)
2. [Applications of heap](#)
3. [Build heap](#)
4. [Median in a stream of integers](#)
5. [Sort a k sorted array](#)
6. [Sort numbers stored on different machines](#)
7. [Merge k sorted arrays](#)
8. [Print all elements in sorted order from row and column wise sorted matrix](#)
9. [kth smallest element in unsorted array](#)
10. [kth largest element in stream](#) -> Zupree
11. [Why prefer heap over BST for priority queue](#)

HASHING

1. [Check for pair in array with sum as x](#)
2. [Vertical sum in binary tree](#)
3. [Largest subarray with equal number of 0s and 1s](#)
4. [Find if there is a subarray with 0 sum](#)
5. [Print binary tree in vertical order](#)
6. [Special data structure](#)
7. [Find itinerary from a given list of tickets](#)
8. [Largest subarray with 0 sum](#)

BST

1. [Find min element](#)
2. [Check if binary tree is BST](#) -> McKinsey
3. [Inorder successor](#)
4. [kth smallest element using order statistics](#)
5. [Sorted linked list to balanced BST](#)
6. [Construct BST from given preorder traversal](#)
7. [Construct BST from given preorder traversal | Set 2](#)

PUZZLES

1. [Measure 1 litre using 2 vessels and infinite water supply](#)(This problem is not asked to be coded. It is asked only as a puzzle. See [this](#) too)
2. [2 eggs 100 floors](#)
3. [Mutilated chessboard problem](#)
4. [100 prisoners, red and blue hats](#)
5. [Measure weight of an elephant](#)
6. [Measure 9 minutes](#)
7. [Shortest path in cube](#)
8. [Angle b/w hour and minute hand](#)
9. [100 doors puzzle](#)
10. [Biased to unbiased coin](#)
11. [Red blue pills](#)
12. [25 horses puzzle](#)
13. [Poisoned bottles](#)
14. [Find the lightest coin](#)
15. [Snail and well problem](#)
16. [Prisoner hat riddle](#)
17. [Cut the cake](#)
18. [3 bulbs and switches problem](#) -> Koo (Indian Twitter)
19. [Ask the question](#)
20. [Cheating husbands](#)
21. [12 marbles and a scale](#)
22. [Socks puzzle](#)
23. [Bee and train puzzle](#)
24. [Will you die](#)
25. [Globe walker](#)
26. [Crossing the river](#)
27. [Changing your mind](#)
28. [Divide cards symmetrically](#)
29. [Where are you](#)
30. [Real and fake coins](#)
31. [Camel and banana puzzle](#)

32. [Probability of observing a car](#)
33. [Red and blue marbles](#)
34. [Warden and 23 prisoners](#)
35. [Crossing a bridge](#)
36. [Age of daughter](#)
37. [Trains and birds](#)
38. [Inverted cards puzzle](#)
39. [Aligned clock hands](#)
40. [3 blind men hat color](#)
41. [Gold bar problem](#)
42. [5 pirates 100 coins](#)
43. [A box of defective balls](#)
44. [Probability of having boy](#)
45. [Days of month using 2 dice](#)
46. [Red and blue balls](#)
47. [Measure 45 seconds](#) -> McKinsey
48. [Water level](#)
49. [8 marbles find heaviest](#)
50. [100 people with sword](#)
51. [Lie tribe and truth tribe](#)
52. [Monty hall problem](#)
53. [Girl counting on fingers](#)