

## 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 duplicates in O\(n\) time and O\(1\) space](#)
12. [Linked list vs Array](#)
13. [Find the smallest missing number](#)
14. [Find max j-i such that arr\[j\] > arr\[i\]](#)
15. [Find subarray with given sum](#)
16. [Find the smallest positive number missing from an unsorted array](#)
17. [Find 2 numbers with odd occurrence](#)
18. [Largest subarray with equal number of 0s and 1s](#)
19. [Replace every element with the greatest on right side](#)
20. [Stock buy sell to maximize profit](#)
21. [Find common elements in 3 sorted arrays](#)
22. [Nuts and bolts problem](#)
23. [Trapping rain water](#)
24. [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. [Longest Common Subsequence](#)
11. [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](#)
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. [XOR linked list](#)
24. [Add 2 numbers represented by linked lists](#)
25. [Write C functions that modify head pointer of linked list](#)
26. [Sorted linked list to balanced BST](#)
27. [Sorted DLL to BST](#)
28. [XOR linked list | Set 2](#)
29. [Flatten a linked list](#)
30. [Add 2 numbers represented by linked lists | Set 2](#)
31. [Flatten a multilevel linked list](#)( Geeks solution has a bug. They are not doing cur->child = NULL after appending it to the tail->next. Looks like there is another bug. In the end we need to check for the child of tail node also i.e child of node 15 in the given case. The loop termination condition should not be cur!=tail. It should be cur!=NULL. You better see [this](#) solution instead )
32. [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. [Serialize and deserialize a binary tree](#)
36. [Serialize and deserialize an N-ary tree](#)
37. [Expression tree](#)
38. [Check if binary tree is symmetric](#)
39. [Mirror of N-ary tree](#)
40. [Find node with min value in BST](#)
41. [Check if a binary tree is BST](#)

42. [Find kth smallest element in BST](#)
43. [Sorted linked list to balanced BST](#)
44. [Kth largest element in BST](#)
45. [Advantages of BST over hash table](#)
46. [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](#)
7. [Iterative DFS](#)

## **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. [Find the number of islands](#)
9. [Detect cycle in undirected graph](#)
10. [Kruskal's Algorithm](#)
11. [Graph and its representations](#)
12. [Prim's algorithm](#)
13. [Prim's algorithm 2](#)
14. [Dijkstra's algorithm](#)
15. [Dijkstra's algorithm 2](#)
16. [Bellman-Ford Algorithm](#)
17. [Transitive closure of a graph](#)
18. [Topological sorting](#)
19. [Shortest path in directed acyclic graph](#)
20. [Strongly connected components](#)
21. [Connectivity in directed graph](#)
22. [Detect cycle in an undirected graph 2](#)

## 23. [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. [Dungeon game](#)( See [this](#) solution )
12. [Given a boolean matrix. Find k such that all elements in the kth row are 0 and the kth column are 1](#)
13. [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. [BFS](#)
6. [LRU Cache](#)( You might want to look at one of [these](#) solutions if geeks solution looks too complicated )
7. [Implement stack using queues](#)
8. [First circular tour that visits all petrol pumps](#)
9. [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](#)
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. [BST vs Hash table](#)
7. [Special data structure](#)
8. [Find itinerary from a given list of tickets](#)
9. [Largest subarray with 0 sum](#)

## **BST**

1. [Find min element](#)
2. [Check if binary tree is BST](#)
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](#)
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](#)

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](#)

## **DYNAMIC PROGRAMMING**

1. [Maximum sum subarray](#)( Read second solution )
2. [Maximum size square sub-matrix with all 1s](#)
3. [Fibonacci numbers](#)
4. [LIS](#)
5. [LCS](#)
6. [Edit distance](#)
7. [Minimum cost path](#)
8. [Minimum number of jumps to reach end](#)( See [this](#) )
9. [Coin change problem](#)
10. [0-1 Knapsack](#)
11. [Longest palindromic subsequence](#)
12. [Maximum sum increasing subsequence](#)
13. [Floyd warshall algorithm](#)
14. [Partition problem](#)
15. [Maximum length chain of pairs](#)
16. [Variations of LIS](#)
17. [Bellman-Ford algorithm](#)
18. [Subset sum problem](#)
19. [Maximum sum rectangle in a 2-D matrix](#)
20. [Minimum insertions to form a palindrome](#)
21. [Find if a string is interleaving of 2 other strings](#)
22. [Count possible decodings of a given digit sequence](#)( Read geeks solution. Then read [this](#). If we are at character i, then r1 gives the answer upto i-1 and r2 gives the answer upto i-2 )
23. [Count possible ways to construct buildings](#)
24. [Find minimum number of coins that make a given value](#)
25. [Minimum number of initial points to reach destination](#)

## **DESIGN QUESTIONS**

1. [LRU cache](#)