

DSA Sheet to Leetcode map

Binary Search

- Find the Minimum length Unsorted Subarray, sorting which makes the array sorted(I do not know why this is in binary search) : [GFG](#) , [Leetcode](#)
- Maximum element in an array which is increasing and then decreasing: [GFG](#) , [Leetcode](#)
- Find the minimum element/(or search for an element) in a sorted and rotated array: [GFG](#) , [Leetcode](#)
- Merge two sorted arrays into a single sorted array: [GFG](#) , [Leetcode](#)
- Longest increasing subsequence: [GFG](#) , [Leetcode](#)
- Check for Majority Element in a sorted array: [GFG](#) , [Leetcode](#)
- Search in Rotated Sorted Array: [Leetcode](#)
- Majority Element in Unsorted Array: [Leetcode](#)
- Find First and Last Position of Element in Sorted Array: [Leetcode](#)
- Shortest Unsorted Continuous Subarray: [Leetcode](#)
- Find Pivot Index: [Leetcode](#)
- Find Peak Element: [Leetcode](#)
- Merge Sorted Rotated Array: [Leetcode](#)
- Median of 2 sorted Arrays: [Leetcode](#)

Divide And Conquer

- Maximum Subarray Sum(prefer kadane over divide and conquer): [GFG](#), [Leetcode](#)
- Fibonacci with divide and conquer : [Leetcode](#)
- Pow(X,N) in logN : [Leetcode](#)
- Closest Pair Of Points (GFG is pretty shitty and I couldn't find any problems for this so I'm putting links to learn): [StackOverflow](#), [Youtube](#)

Arrays

- Sorted subsequence of size 3: [GFG](#) , [Leetcode](#)
- Smallest missing positive number : [GFG](#), [LeetCode](#)
- Search in sorted Matrix : [GFG](#), [LeetCode1](#), [LeetCode2](#)
- Construct Product Array without division operator: each element = product of elements in arr[] except arr[i] : [GFG](#), [LeetCode](#)
- Given binary 2D Matrix, for all cells as 0, set corresponding row and column as 0 : [LeetCode](#)
- Rotate Image by 90 degrees: [LeetCode](#)
- Largest Sum Contiguous Subarray: [GFG](#), [LeetCode](#)
- Largest Product Subarray: [GFG](#) , [Leetcode](#)
- Rotate array (Leetcode is right rotate): [GFG left rotate](#), [GFG right rotate](#) [LeetCode](#)
- Max length bitonic subarray(Leetcode one is a bit different): [GFG](#), [LeetCode](#)
- Subarray sum equals K (Leetcode solution is very nice) : [LeetCode](#)
- Largest Subarray with equal number of zeros and ones : [GFG](#), [LeetCode](#)
- Find the Number Occurring Odd Number of Times: [LeetCode](#), [LeetCode](#)
- Best time to buy stock: [Leetcode](#), [Leetcode](#), [Leetcode](#),
- Maximum water trapped in histograms [Leetcode](#) [GFG](#)

Strings

- Reverse a string(Don;t do it normally, use recursion) [GFG](#) [LeetCode1](#) [LeetCode2](#)
- Print all permutations [GFG](#) [Hackerank](#) [LeetCode1](#) [LeetCode2](#)
- Given a string find its first non-repeating character [GFG](#) [LeetCode](#)
- Reverse words in a given string [GFG](#) [LeetCode1](#) [LeetCode2](#)
- Find Lexographical rank of a string [GFG](#) [InterviewBit](#)
- Run Length Encoding [GFG](#) [LeetCode](#)
- ATOI (leetcode one very disliked for some reason) [GFG](#) [LeetCode](#) [HackerRank](#)
- Check whether two strings are anagrams or not [GFG](#) [LeetCode](#) [HackerRank](#)
- Longest substring without repeating characters [GFG](#) [LeetCode](#)

- Find the smallest window of string containing all characters of another string [GFG](#) [LeetCode](#)
- Recursively Remove all adjacent duplicates (leetcode ones do recursively) [GFG](#) [LeetCode1](#) [LeetCode2](#)
- String Matching KMP [GFG](#) [LeetCode](#)

Stacks

- Largest Rectangle in histogram : [Leetcode](#)
- Decode String: [Leetcode](#)
- Stack using queues [Leetcode](#)
- Queues using stacks [Leetcode](#)
- Balanced Parentheses [Leetcode](#)
- LRU Cache [Leetcode](#)
- Stock Span Problem [Leetcode](#)
- First greatest element to the right [Leetcode](#)
- Sliding window maximum [Leetcode](#)

Heaps

- Sort a nearly sorted (or K sorted) array: [GFG](#), [HackerRank](#)
- Find the k most frequent words from a file [GFG](#), [LeetCode](#)
- Merge k sorted arrays [GFG](#), [LeetCode](#)
- Median in a stream of integers [GFG](#), [LeetCode](#)

Graphs

- Detect Cycle in a Directed Graph [GFG](#), [LeetCode](#)
- Find if there is a path between two vertices in a directed graph [GFG](#) , [LeetCode](#)
- Find the number of islands [LeetCode](#)
- Bellman–Ford Algorithm [GFG](#), [LeetCode](#)
- Union Find [GFG](#) [LeetCode](#)
- Topological Sorting [GFG](#) [LeetCode](#)
- Strongly Connected Components [GFG](#) [LeetCode](#)
- Shortest Path in Directed Acyclic Graph [GFG](#) [LeetCode](#)

- Check whether a given graph is Bipartite or not [GFG](#) [LeetCode](#)
- Stable Marriage Problem [GFG](#) [LeetCode](#)
- Travelling Salesman Problem [GFG](#) [LeetCode](#)

DP

- Coin Change: [GFG](#), [LeetCode](#) [LeetCode2](#)
- Rod Cutting [GFG](#) [LeetCode](#)

Tree

- Diameter of a binary tree: [leetcode](#)
- Maximum Width of Binary Tree: [leetcode](#)
- Two trees are identical: [leetcode](#)
- Tree traversal: [leetcode](#) [inorder](#), [leetcode](#) [preorder](#), [leetcode](#) [postorder](#)
- Populate inorder successors: [leetcode](#)
- level order traversal: [leetcode](#)
- reverse level order traversal: [leetcode](#)
- vertical order traversal: [leetcode](#)
- zig zag traversal: [leetcode](#)
- count tree nodes: [leetcode](#)
- lowest common ancestor: [leetcode](#)
- nodes at distance K from root: [leetcode](#)
- Right side view of binary tree: [leetcode](#)
- serialize and deserialize BST: [leetcode](#)
- tree is balanced or not: [leetcode](#)
- check if BT is complete: [leetcode](#)
- symmetric tree: [leetcode](#)
- Kth smallest element in BST: [leetcode](#)
- remove nodes outside given range: [leetcode](#)