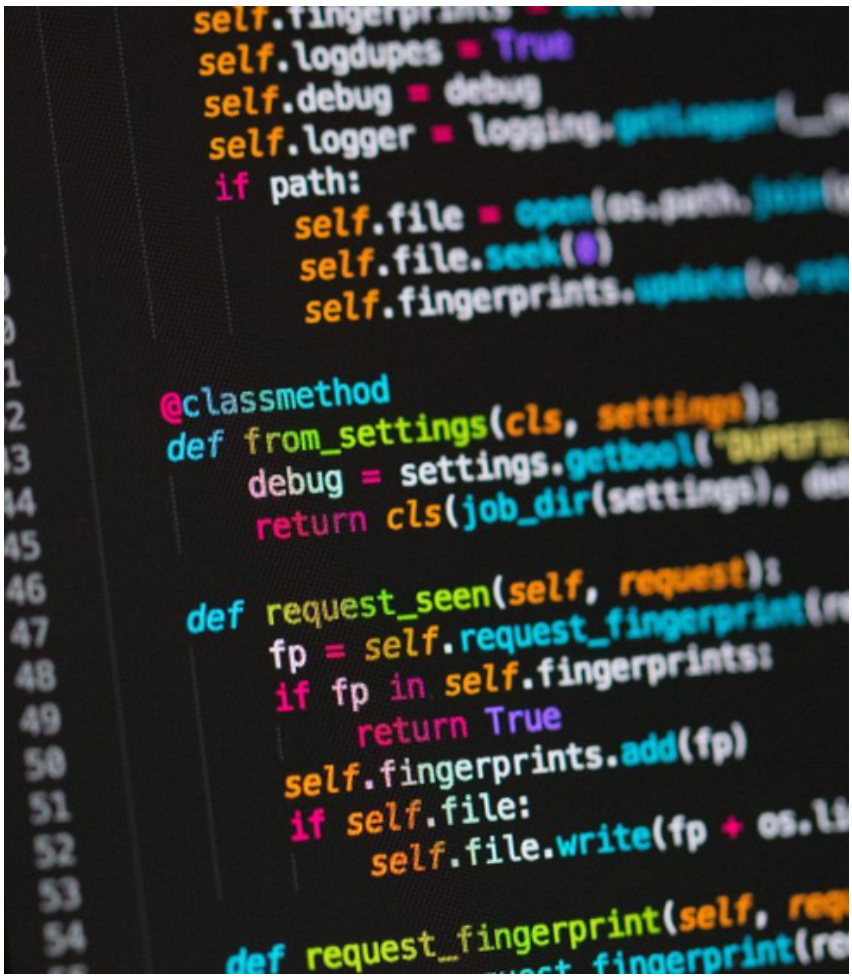
20 CODING PATTERNS

TO MASTER IN TECHNICAL INTERVIEWS



1. Two Pointers

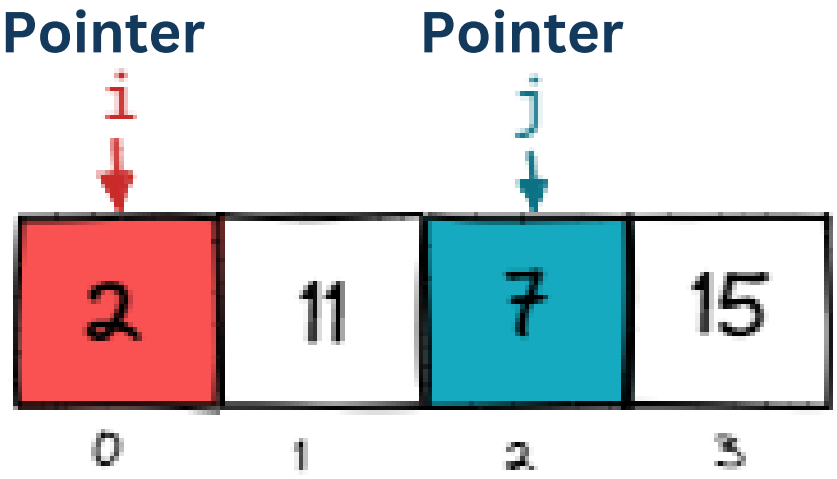
Applications:

It is used when we need to find pairs or sub-arrays in an array that satisfy a certain condition, or when we need to find a specific element in a sorted array.

Typically, the two pointers advance towards opposite ends of the data structure with a fixed increment.

DSA Usages:

Array, String, Linked List



Sample Problems:

- Finding a pair of numbers that add up to a target sum in a sorted or unsorted array.

  Input: [3, 5, 2, 8, 11], target = 10
  Output: [2, 8]

- Finding the longest substring in a string that is a palindrome.

  Input: "babad"
  Output: "bab"

- Finding the maximum sum sub-array of a given array.

  Input: [-2, 1, -3, 4, -1, 2, 1, -5, 4]
  Output: [4, -1, 2, 1]

- Finding the intersection of two sorted arrays.

  Input: [1, 3, 4, 5, 7], [2, 3, 5, 6]
  Output: [3, 5]

2. Fast & Slow Pointers

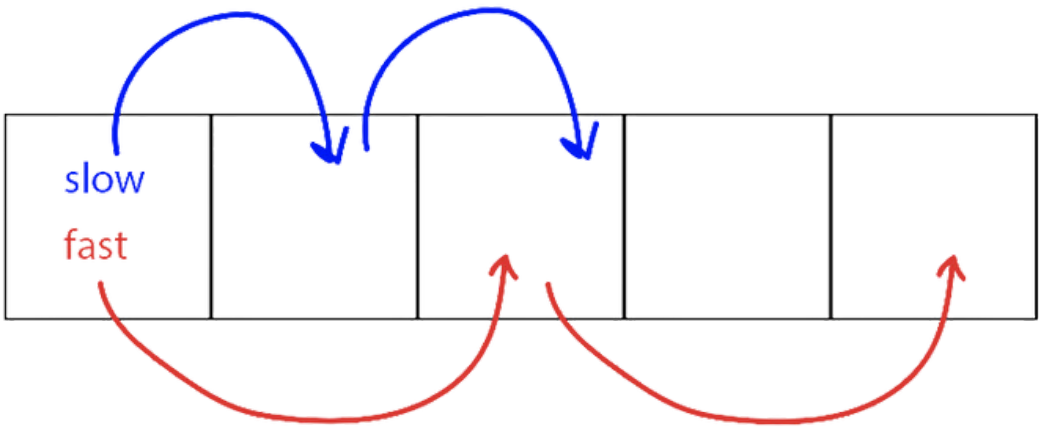
Applications:

It is used in DSA to solve problems that involve linked lists, arrays, or other data structures where we need to iterate through the data in a specific way.

The Fast and Slow Pointer technique involves using two pointers - a "**slow**" pointer and a "**fast**" pointer. The slow pointer moves one step at a time, while the fast pointer moves two steps at a time.

DSA Usages:

Array, String, Linked List



Sample Problems:

- Detecting cycles in a linked list.

  Input: 1 -> 2 -> 3 -> 4 -> 5 -> 2
  Output: True

- Finding the middle element of a linked list.

  Input: 1 -> 2 -> 3 -> 4 -> 5
  Output: 3

- Finding the intersection point of two linked lists.

  Input: 1 -> 2 -> 3 -> 4 -> 5, 8 -> 4 -> 5
  Output: 4

- Finding the kth to last element in a linked list.

  Input: 1 -> 2 -> 3 -> 4 -> 5, k = 2
  Output: 4

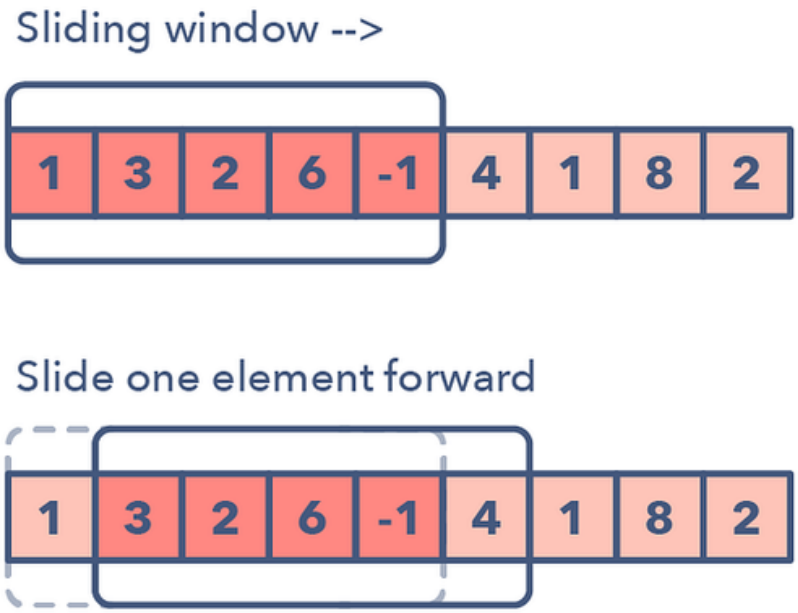
3. Sliding Window

Applications:

It is used in scenarios where we need to find a subarray or a substring that meets certain criteria, such as having a specific sum or product, or containing a certain pattern or character and the input data in a specific window size.

DSA Usages:

Array, String, Linked List



Sample Problems:

- Finding the maximum or minimum sum subarray of a given size.

  Input: [1, 4, 2, 10, 2, 3, 1, 0], k=3
  Output: 15

- Finding the longest substring with distinct characters.

  Input: "abcabcbb"
  Output: "abc"

- Finding the first occurrence of a pattern or substring in a string.

  Input: s= "hello world", pat= "world"
  Output: 6

- Finding the smallest window in a string that contains all characters of another string.

  Input: s1= "this is a test string", s2= "tist"
  Output: "tist"

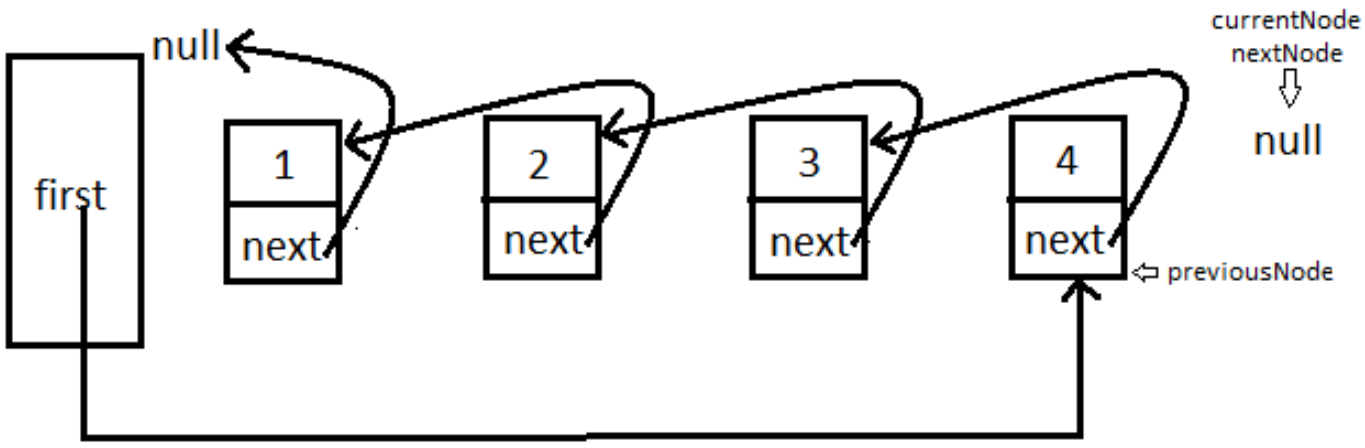
4. In-place Reversal of a LinkedList

Applications:

It is used when we need to reverse the order of the elements in the linked list without using any additional data structures such as arrays or other lists.

DSA Usages:

Linked List



Sample Problems:

- Reversing a linked list of any size.

  Input: 1->2->3->4->5
  Output: 5->4->3->2->1

- Reversing a linked list from a certain point to the end.

  Input: 1->2->3->4->5, lenght= 3
  Output: 1->2->5->4->3

- Checking if a linked list is a palindrome.

  Input: 1->2->3->2->1
  Output: True

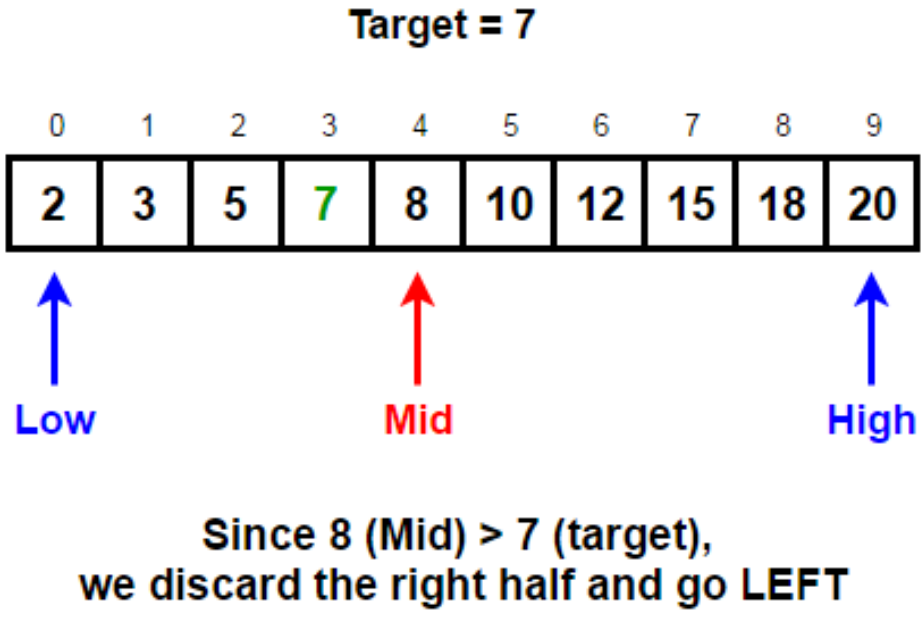
5. Modified Binary Search

Applications:

This method is used when the conventional binary search algorithm cannot be used to solve the problem because the search condition is more complex than simply checking if an element is greater than or less than the middle element.

DSA Usages:

Array



Sample Problems:

- Finding the minimum or maximum element in a rotated sorted array.

  Input: [4, 5, 6, 7, 0, 1, 2]
  Output: 0 (minimum) or 7 (maximum)

- Finding the first occurrence of an element in a sorted array.

  Input: [1, 2, 3, 3, 3, 4, 5], target= 3
  Output: 2

- Searching for a target value in a matrix where each row and column is sorted.

  Input: M= [[1, 4, 7], [2, 5, 8], [3, 6, 9]], t= 5
  Output: True

6. Islands (Matrix Traversal)

Applications:

It is used to solve problems that involve traversing a matrix or grid to find a specific pattern or object. This method is used when the problem requires identifying clusters or groups of cells in the matrix that meet certain criteria.

DSA Usages:

Matrix, Queue

Sample Problems:

- Counting the number of islands in a binary matrix.

  Input: [[1, 1, 0, 0, 0],
  [1, 1, 0, 0, 0],
  [0, 0, 1, 0, 0],
  [0, 0, 0, 1, 1]]
  Output: 3

- Finding the longest path in a matrix with given constraints.

  Input: [[5, 6, 7], [4, 9, 8], [3, 2, 1]]
  Output: [9, 8, 7, 6, 5, 4, 3, 2, 1]

- Finding the largest sub-matrix with all 1's.

  Input: [[1, 1, 0, 1, 0],
  [0, 1, 1, 1, 0],
  [1, 1, 1, 1, 0],
  [0, 1, 1, 1, 1]]
  Output: [[1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1]]

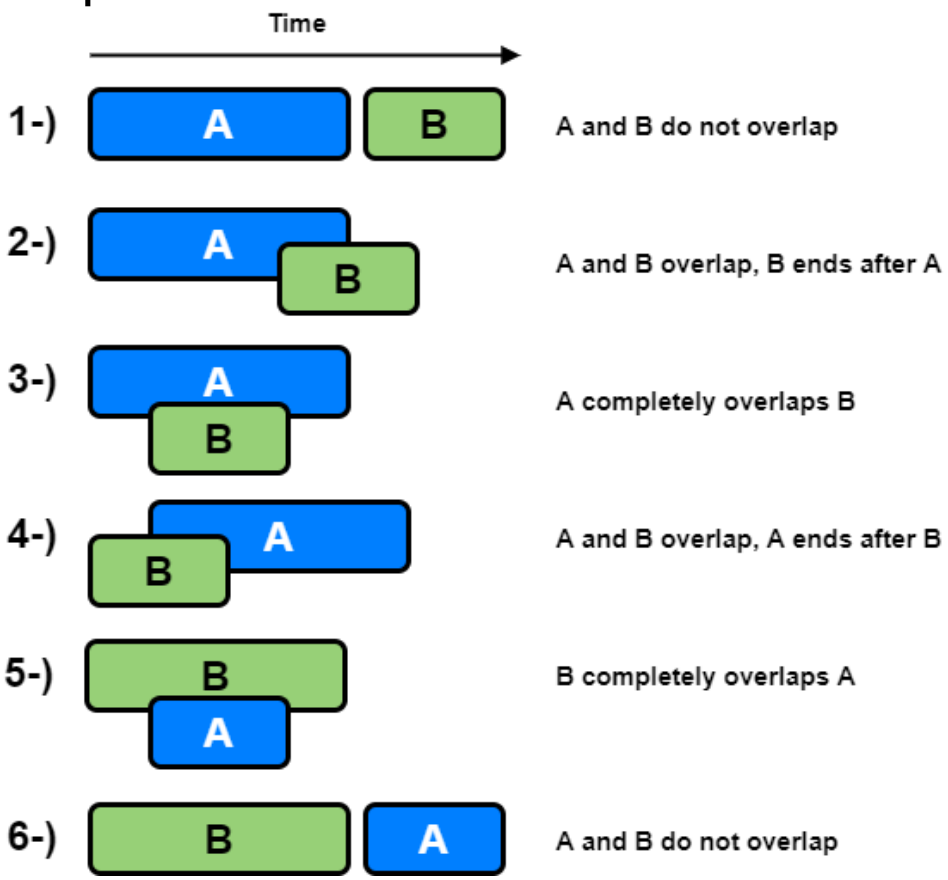
7. Merge Intervals

Applications:

The Merge Interval is a popular method for solving problems that involve merging or overlapping intervals. It is used when the problem requires combining or merging intervals that share a common overlap or intersection.

DSA Usages:

Array, Heap



Sample Problems:

- Given a collection of intervals, merge all overlapping intervals.

  Input: [[1, 3], [2, 6], [8, 10], [15, 18]]
  Output: [[1, 6], [8, 10], [15, 18]]

- Given a list of meeting intervals, determine the minimum number of meeting rooms required to schedule all the meetings.

  Input: [[0, 30], [5, 10], [15, 20]]
  Output: 2

- Given a set of intervals, find the union of all the intervals. The union is defined as the set of intervals that covers all the points in the input intervals.

  Input: [[1, 3], [2, 4], [5, 7], [6, 8]]
  Output: [[1, 4], [5, 8]]

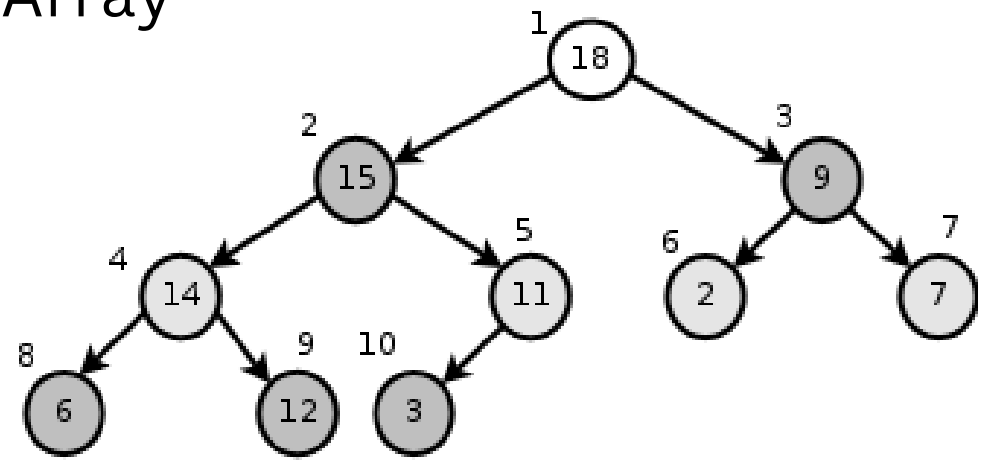
8. Two Heaps

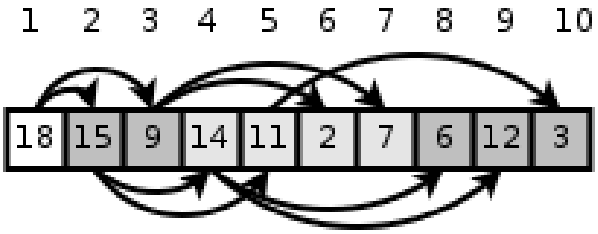
Applications:

The Two Heaps is a popular method for solving problems that involve managing and processing elements in two separate heaps simultaneously. It is used when the problem requires maintaining two sets of elements with specific ordering or prioritization properties.

DSA Usages:

Heap, Array





Sample Problems:

- Find the median of streaming data.

  Input: [2, 5, 1, 3, 6, 4]
  Output: [2, 3.5, 2, 2.5, 3, 3.5]

- Find the kth smallest element in an array.

  Input: [3, 7, 1, 4, 5, 2] and k = 3
  Output: 3

- Given a continuous stream of data and a window size, process the data efficiently within the sliding window.

  Input: [4, 1, 3, 2, 5], window size = 3
  Output: [4, 4, 4, 3, 5]

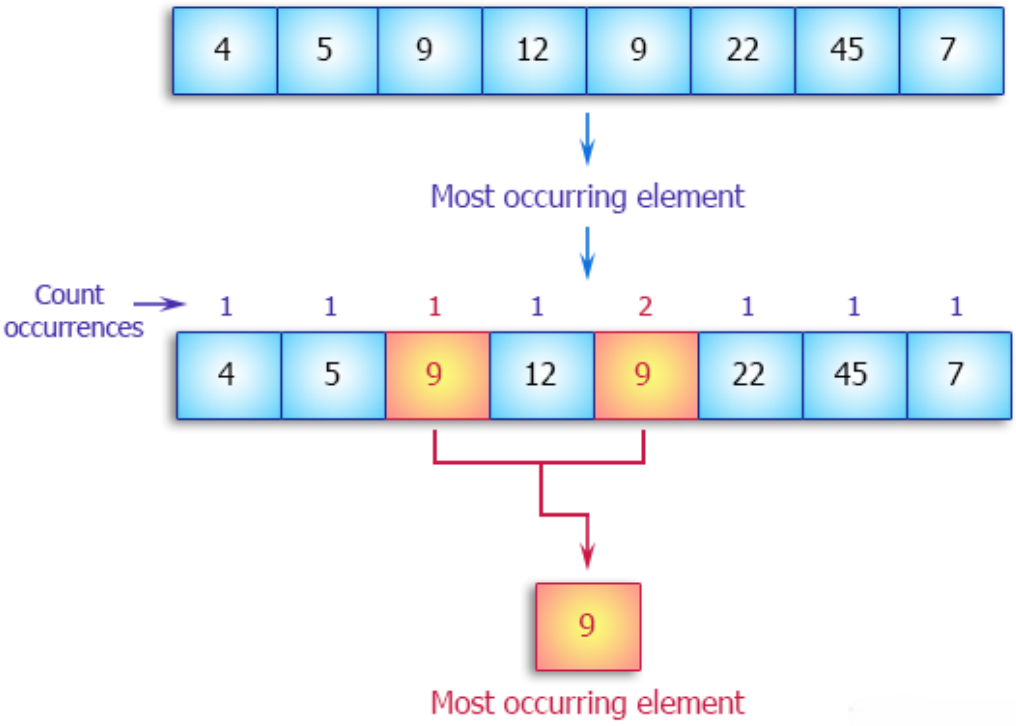
9. Top ‘K’ Elements

Applications:

The Top 'K' Elements is a popular method for solving problems that involve finding or managing the K largest or smallest elements from a collection of elements. It is used when the problem requires identifying the elements with the highest or lowest values based on a specific criterion.

DSA Usages:

Heap, Array



Sample Problems:

- Maximum distinct elements after removing k elements.

  Input: [5, 7, 5, 5, 1, 2, 2], K = 3
  Output: 4

- Find the top 'K' frequent elements in an array.

  Input: [3, 7, 1, 4, 5, 2] = 3
  Output: 3

- Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane and an integer k, return the k closest points to the origin (0, 0)

  Input: [[1,3],[-2,2]], K = 1
  Output: [[-2,2]]

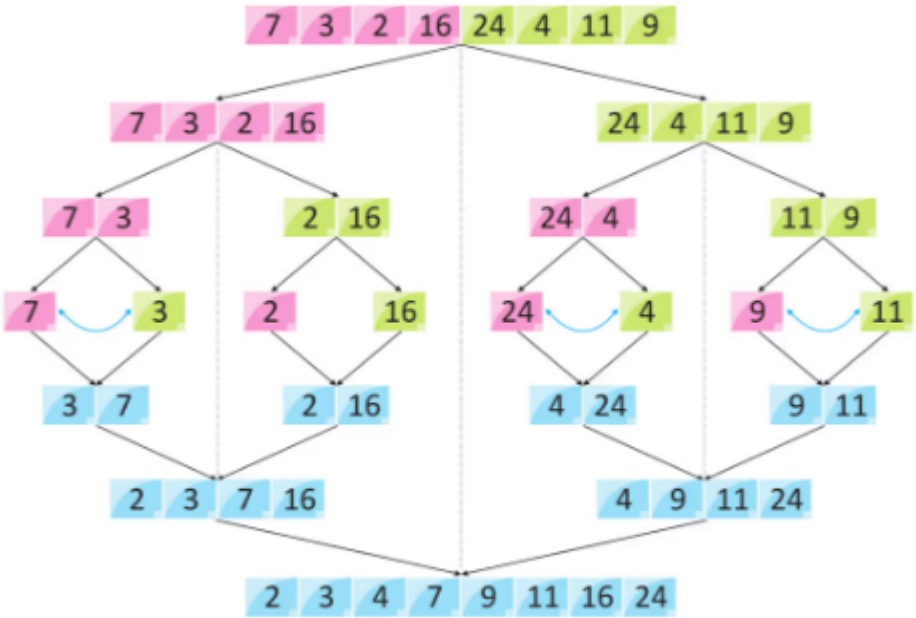
10. K-way Merge

Applications:

The K-way Merge is a popular method for solving problems that involve merging multiple sorted lists or arrays into a single sorted list or array. It is used when the problem requires combining and sorting elements from multiple sources.

DSA Usages:

Array, Queue, Heap



Sample Problems:

- Merging K sorted arrays into a single sorted array.

  Input: [1, 4, 7], [2, 5, 8], and [3, 6, 9]
  Output: [1, 2, 3, 4, 5, 6, 7, 8, 9]

- Find Kth smallest value in M sorted arrays.

  Input: [[1, 3], [2, 4, 6], [0, 9, 10, 11]], K = 5
  Output: 4

- Merging K-sorted streams into a single sorted stream.

  Input: [1, 5, 9], [2, 6, 10], and [3, 7, 11]
  Output: [1, 2, 3, 5, 6, 7, 9, 10, 11]

- Kth Smallest Element in a Sorted Matrix.

  Input: [[1,5,9],[10,11,13],[12,13,15]], K = 8
  Output: 13

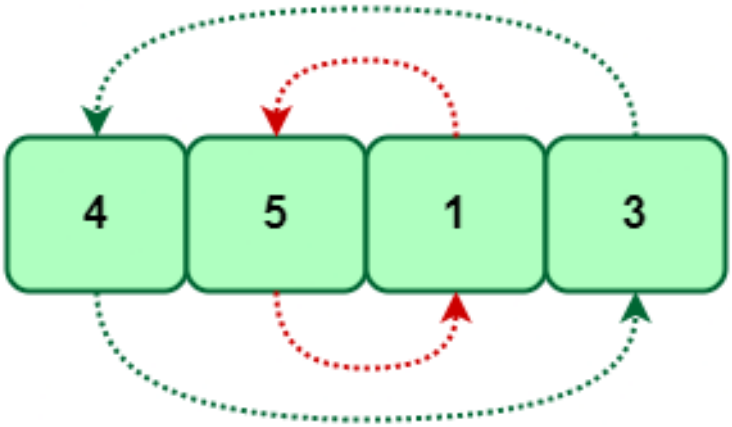
11. Cyclic Sort

Applications:

The Cyclic Sort is a popular method for solving problems that involve sorting an array of numbers in a specific range. It is used when the problem requires arranging the elements of the array in a cyclic manner to achieve the desired order.

DSA Usages:

Array



Sample Problems:

- Sorting an array of distinct numbers.

  Input: [3, 1, 5, 4, 2]
  Output: [1, 2, 3, 4, 5]

- Find the smallest missing positive number.

  Input: [3, 1, -5, 4, 2]
  Output: 5

- Rearranging an array with values in the range 0 to n-1.

  Input: [3, 2, 0, 1]
  Output: [0, 1, 2, 3]

- Find the First K Missing Positive Numbers.

  Input: [1, 2, 3, 0, 4, 9, 7], K = 4
  Output: [5, 6, 8, 10]

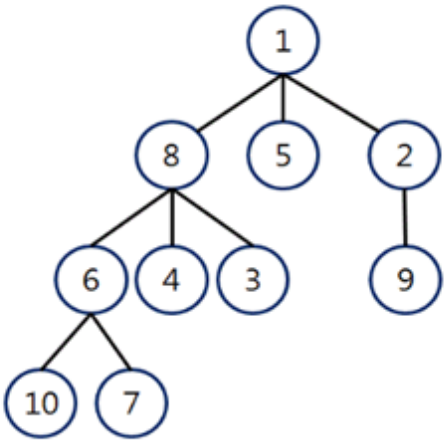
12. Breadth-First Search

Applications:

BFS is a popular method for solving problems related to traversing or searching in a graph or tree data structure. It is used when the problem requires exploring or visiting all the vertices or nodes of the graph in a breadth-wise manner, i.e., exploring the vertices at the same level before moving to the next level.

DSA Usages:

Tree, Graph, Matrix, Queue



BFS: 1 8 5 2 6 4 3 9 10 7

Sample Problems:

- Find the shortest path or distance between two BST nodes.

  Input: [4,2,6,1,3]
  Output: 1

- Given the root of a binary tree, return the level order traversal of its nodes' values. (i.e., from left to right, level by level).

  Input: [3,9,20,null,null,15,7]
  Output: [[3],[9,20],[15,7]]

- Minimum Depth of a Binary Tree.

  Input: [3,9,20,null,null,15,7]
  Output: 2

- Connect nodes at the same Level.

  Input: [1,2,3,4,5,6,7]
  Output: [1,#,2,3,#,4,5,6,7,#]

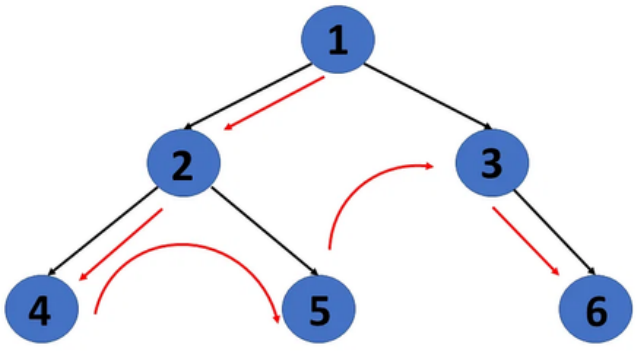
13. Depth-First Search

Applications:

DFS is a popular method for solving problems related to traversing or searching in a graph or tree data structure. It is used when the problem requires exploring or visiting all the vertices or nodes of the graph in a depth-wise manner, i.e., exploring as far as possible along each branch before backtracking.

DSA Usages:

Tree, Graph, Matrix



DFS	1	2	4	5	3	6

Sample Problems:

- Find the connected components in a graph.

  Input: n = 5, edges = [[0, 1], [1, 2], [3, 4]]
  Output: 2

- Detect cycles in a graph.

  Input: n = 5, edges = [[0, 1], [0, 2], [1, 2], [2,0], [2,3], [3,3]]
  Output: [[3],[9,20],[15,7]]

- Check if there is a root-to-leaf path with a given sequence.

  Input: root = [0,1,0,0,1,0,null,null,1,0,0], arr = [0,1,0,1]
  Output: True

- Count Paths for a Sum.

  Input: root =[10,5,-3,3,2,null,11,3,-2,null,1], targetSum = 8
  Output: 3

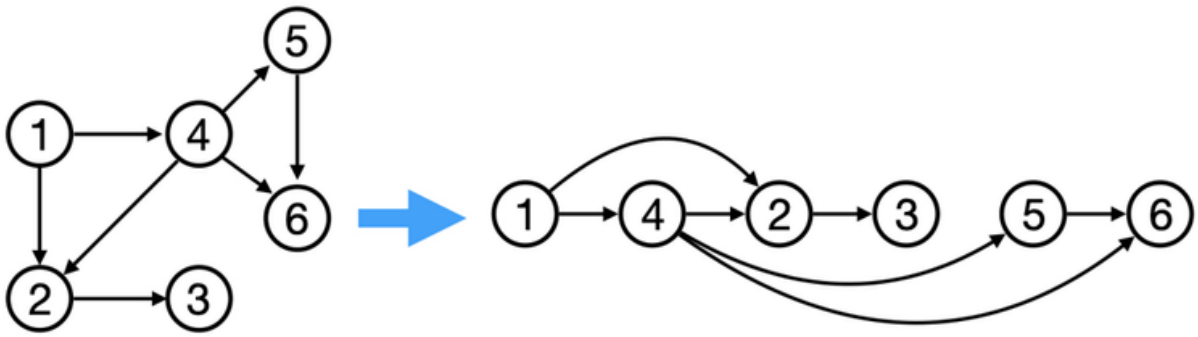
14. Topological Sort

Applications:

The Topological Sort is a popular method for solving problems related to directed acyclic graphs (DAGs). It is used when the problem requires ordering the vertices of a DAG in such a way that for every directed edge (u, v), vertex u comes before vertex v in the ordering.

DSA Usages:

Array, HashTable, Queue, Graph



Sample Problems:

- Given the total number of n tasks and a list of prerequisite pairs of size m. Find the order of tasks you should pick to finish all tasks.

  Input: n = 2, m = 1, prerequisites = {{1, 0}}
  Output: True

- Given a sequence of words written in the alien language, and the order of the alphabet, return true if and only if the given words are sorted lexicographically in this alien language.

  Input: words = ["hello", "leetcode"], order = "hlabcdefgijkmnopqrstuvwxyz"
  Output: True

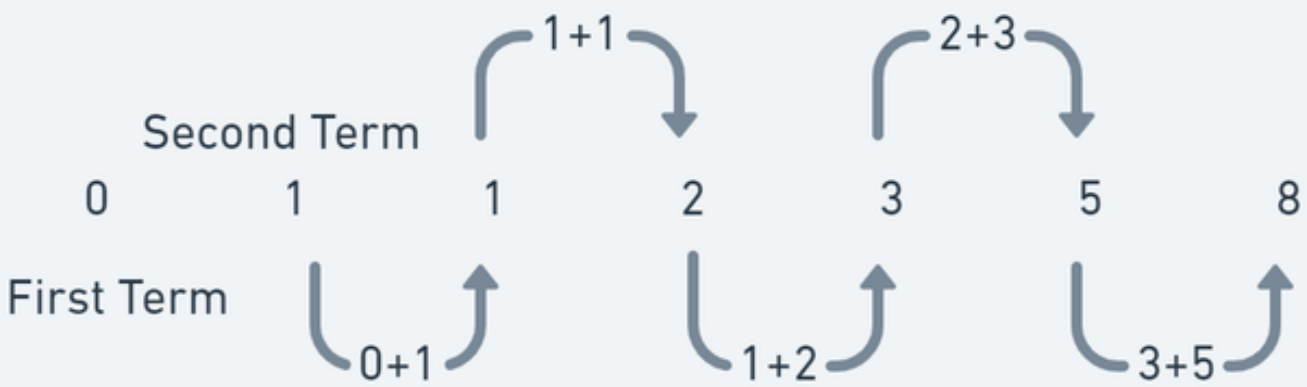
15. Fibonacci Numbers

Applications:

The Fibonacci Numbers method is a mathematical sequence that is frequently used in DSA to solve various problems. It is used when the problem involves calculating or generating the Fibonacci sequence or utilizing the properties of Fibonacci numbers.

DSA Usages:

Array, HashTable



Sample Problems:

- Calculate the nth Fibonacci number.

  Input: n = 10
  Output: 55

- You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

  Input: n = 2
  Output: 2

- You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

  Input: nums = [1,2,3,1]
  Output: 4

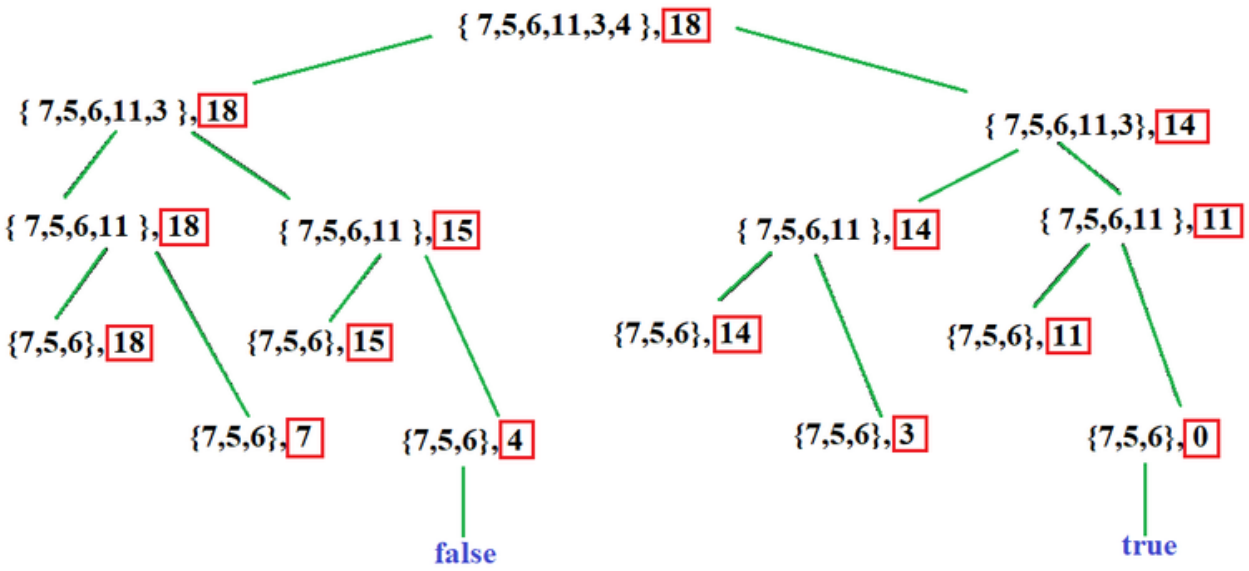
16. Subsets

Applications:

The Subsets method is a common algorithmic approach used in DSA to generate or explore all possible subsets of a given set. It is used when the problem involves finding or generating all combinations or subsets of elements from a set.

DSA Usages:

Queue, Array, String



Sample Problems:

- Given a set of elements, generate all possible subsets, including the empty set and the set itself.

  Input: [1, 2, 3]
  Output: {}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}

- Given a set of non-negative integers and a value sum, the task is to check if there is a subset of the given set whose sum is equal to the given sum.

  Input: [3, 34, 4, 12, 5, 2], sum = 9
  Output: True

- Given an array of nums of distinct integers, return all the possible permutations. You can return the answer in any order.

  Input: nums = [1,2,3]
  Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

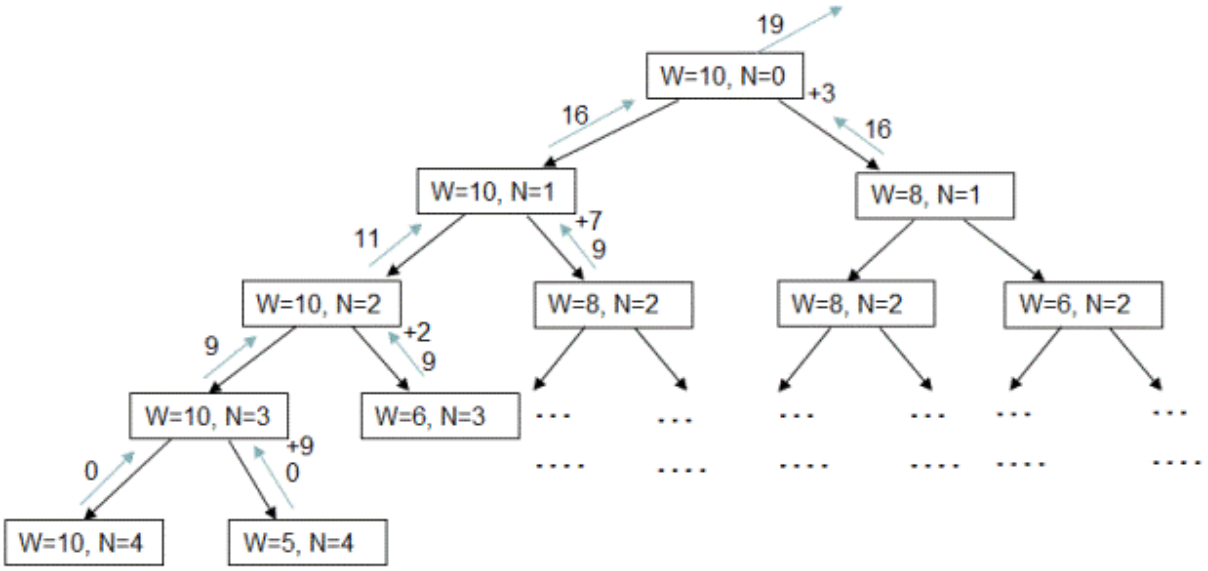
17. 0/1 Knapsack

Applications:

The 0/1 Knapsack method is a well-known algorithmic approach used in DSA to solve optimization problems, particularly those involving resource allocation or selection. It is used when the problem requires maximizing or minimizing a value while considering constraints on the capacity or availability of resources.

DSA Usages:

Array, HashTable



Recursion tree for 0-1 Knapsack problem

Sample Problems:

- Given a list of items with their weights and values, and a knapsack with a weight capacity, find the maximum value that can be obtained by selecting a subset of items to fit into the knapsack without exceeding its weight capacity.

  Input: [2, 3], [3, 4], [4, 5], [5, 8], [9, 10]]
  Output: 17

- Given a set of resources with their respective costs and benefits, and a limited budget or capacity, determine the optimal allocation of resources to maximize the overall benefit or value.

  Input: [[5, 10], [8, 15], [3, 7], [6, 12]]
  Output: 15

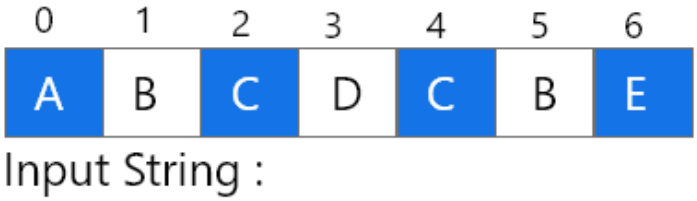
18. Palindromic Subsequence

Applications:

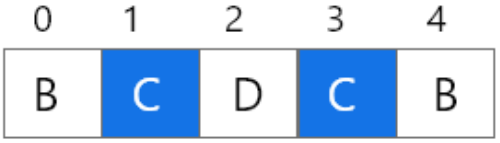
The Palindromic Subsequence method is a technique used in DSA to solve problems that involve identifying and manipulating palindromic subsequences within a given string or sequence. It is used when the problem requires finding or manipulating palindromic subsequences with specific properties or constraints.

DSA Usages:

Array, HashTable



Input String :



Resultant Palindromic Subsequence

Sample Problems:

- Longest Palindromic Subsequence.

  Input: "BBABCBCAB"
  Output: "BABCBAB" with a length of 7

- Given str, count the total no. of distinct palindromic subsequences that can be formed using the characters of the string.

  Input: "ACBCDBAA"
  Output: 19

- Given a string, count the total number of distinct palindromic subsequences that can be formed using the characters of the string.

  Input: "ACBCDBAA"
  Output: 19

- Minimum Deletions in a String to make it a Palindrome.

  Input: "ACBCDBAA"
  Output: 19

19. Longest Common Substring

Applications:

The Longest Common Substring method is used in DSA to solve problems that involve finding the longest common substring between two or more strings. It is used when the problem requires identifying the longest continuous sequence of characters that is common to multiple strings.

DSA Usages:

Array, HashTable

string 1	**a**	**c**	b	**a**	e	**d**
string 2	**a**	b	**c**	**a**	**d**	f

LCS: "acad" with length 4

20. Bitwise XOR

Applications:

The Bitwise XOR (Exclusive OR) method is used in DSA to solve various problems that involve bitwise operations and manipulation of binary representations of numbers. It is used when the problem requires performing operations or extracting information by manipulating individual bits using XOR logic.

DSA Usages:

Array, Bits

Bitwise XOR (^) Table

X	Y	X&Y
0	0	0
0	1	1
1	0	1
1	1	0

Sample Problems:

- Given an array of integers where every element appears twice except for one element, find that single element.

  Input: [2, 4, 6, 2, 4]
  Output: 6

- Given an array in which all numbers except two are repeated once. (i.e. we have 2n+2 numbers and n numbers are occurring twice and the remaining two have occurred once). Find those two numbers in the most efficient way.

  Input: [2, 3, 7, 9, 11, 2, 3, 11]
  Output: 7, 9

- Given two strings s1 and s2, the task is to find the length of the longest common substring of s1 and s2 such that the XOR is maximum.

  Input: s1 = "79567", s2 = "56779"
  Output: 2