Data Structure Problem Solving Techniques

- Dheeraj Bhandari

"Go through these slides before your next coding interview to revise problem solving approaches.
You will definitely get benefitted."

This are taken from people already working from big tech company like facebook , Google etc.

And will helpful for you in your next Coding Interview

Keep Coding ☺

Youtube Channel Link

https://www.youtube.com/channel/UChXHt9pHik0rR0iZDOHNdsg

Sorted Arrays

- **Binary Search** is applicable when the problem needs you to:
  find a certain element in the given sorted array, for eg: find a given number in a sorted array

- **Two Pointers** is applicable when the problem needs you to:
  search set of elements with a given condition, for eg: find two number in a "sorted" array that sum up to a given target number
  find triplets or a sub array with a given condition

- **Cyclic Sort** is applicable when the problem needs you to:
  find missing/duplicate/smallest number, where elements of the array are in a given range,

- **Merge** is applicable when the problem needs you to:
  find the smallest/largest/or a specified element from N sorted arrays

- **Backtracking** is applicable when the problem needs you to:
  find some permutations or combinations of the array elements

- **Dynamic Programming when none of the above applies:**
  for eg: maximum subarray

Unsorted Arrays

- **Using a Collection (like HashMap/HashSet/Stack/Queue etc) is applicable when the problem needs you to:**
  find a pair/group of numbers satisfying a given target condition, for eg: find two number in the array that sum up to a given target number

- **Sorting is applicable when the problem needs you to:**
  Rearrange the given array elements in a specified way to obtain the output.

- **Simple Array Traversal is applicable when the problem needs you to:**
  Find a single value or a transformation of the given array in some way

- **Two Pointers is applicable when the problem needs you to:**
  search set of elements with a given condition, for eg: container with most water
  find triplets or a sub array with a given condition

- **Fast and Slow Pointer is applicable when the problem needs you to:**
  find the position of an element or the length of the array, where the array is cyclic/has a loop

- **Two Heaps is applicable when the problem needs you to:**
  find the smallest/largest/median of elements of an array

Unsorted Arrays

- **Subsets** is applicable when the problem needs you to:
  permutation and combination of the given set

- **Top K Elements** is applicable when the problem needs you to:
  find the top K smallest/largest/frequent elements of the array

- **Sliding Window** is applicable when the problem needs you to:
  find the shortest /longest subset of the given array

- **Merge Intervals** is applicable when the problem needs you to:
  find overlapping intervals/mutually exclusive intervals/merge intervals
  given the input is a list/array of intervals [t1,t2]

- **Backtracking** is applicable when the problem needs you to:
  find some permutations or combinations of the array elements

- **Dynamic Programming** when none of the above applies.

Strings

- **Traverse like an array, optionally using a Collection** is applicable when the problem needs you to:
  traverse the given string from start to end, whilst validating or finding something, for eg: validate the parentheses of the given string [{(){[]

- **ASCII values approach** is applicable when the problem needs you to:
  solve something based on the number of occurrences of the alphabets, for eg: find the first repetitive character in a string

- **One/More Pointers Approach** is applicable when the problem needs you to:
  search a substring/another string, for eg: implement strStr()

- **Sliding Window** is applicable when the problem needs you to:
  find the smallest/largest/or a conditional substring from the given string, for eg: find the longest substring without a repeating character

- **Backtracking** is applicable when the problem needs you to:
  find all permutation of a string

- **Dynamic Programming when none of the above applies.**

Linked Lists

- **Traverse + optional Collection** is applicable when the problem needs you to
  traverse the given linked list from start to end, whilst validating or finding something, for eg: Adding two numbers represented by linked lists

- **Two Pointers approach** is applicable when the problem needs you to:
  perform an operation, given Nth node from the end of the list
  list is cyclic
  given are two lists that are merged

- **Link Manipulation** is applicable when the problem needs you to:
  modify the given linked list in-place, for eg: reverse a linked list
  perform an operation on two linked lists, for eg: merge two sorted lists to give a single sorted list

- **Recursion** is applicable when the problem needs you to:
  perform the same operation till a base case has been reached, for eg: reverse a linked list in groups of a given size

- **k-way merge** is applicable when the problem needs you to:
  given multiple sorted linked lists, merge them into one sorted list

Graphs Traversal Methods

- **Depth First Search Traversal** starts at the root of the graph and explores as far as possible along each path in the graph based on the condition for traversal/what is being searched.

- **Breadth First Search Traversal** starts at the root of the graph but explores it's neighbour nodes in the graph before moving to next level of neighbours.

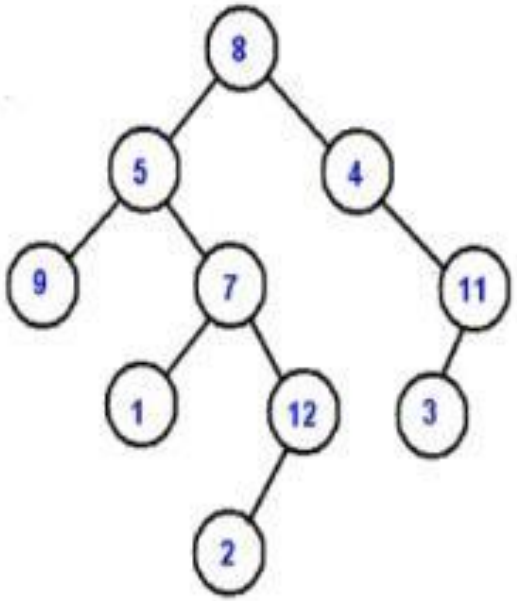
Trees

- **Recursion is applicable when the problem needs you to:**
  find an output that depends on smaller instance of the same problem that build up to the final solution

- **4 traversal techniques of a tree:**
  Level Order
  Pre-Order
  Post-Order
  In-Order

As an example consider the following tree and its four traversals:

PreOrder - 8, 5, 9, 7, 1, 12, 2, 4, 11, 3
InOrder - 9, 5, 1, 7, 2, 12, 8, 4, 3, 11
PostOrder - 9, 1, 2, 12, 7, 5, 3, 11, 4, 8
LevelOrder - 8, 5, 4, 9, 7, 11, 1, 12, 3, 2



Youtube Channel Link

https://www.youtube.com/channel/UChXHt9pHik0rR0iZDOHNdsg

I have posted problems and their detailed solution for each of these problems solving techniques on my channel.

Please comment on my videos if you want me to solve any other problem.