DATA STRUCTURE ALGORITHMS

RESOURCES USED

@himanshu_shekhar16

1. What is a Data Structure?

The Data Structure is the way data is organized (stored) and manipulated for retrieval and access. It also defines the way different sets of data relate to one another, establishing relationships and forming algorithms.

2. Describe the types of Data Structures?

The following are the types of data structures:

1. Lists: A collection of related things linked to the previous or/and following data items.
2. Arrays: A collection of values that are all the same.
3. Records: A collection of fields, each of which contains data from a single data type.
4. Trees: A data structure that organizes data in a hierarchical framework. This form of data structure follows the ordered order of data item insertion, deletion, and modification.
5. Tables: The data is saved in the form of rows and columns. These are comparable to records in that the outcome or alteration of data is mirrored across the whole table.

3. What is a Linear Data Structure? Name a few examples.

A data structure is linear if all its elements or data items are arranged in a sequence or a linear order. The elements are stored in a non-hierarchical way so that each item has successors and predecessors except the first and last element in the list.

Examples of linear data structures are Arrays, Stack, Strings, Queue, and Linked List.

4. What are some applications of Data Structures?

In terms of data structure interview questions, this is one of the most frequently asked question.

Numerical analysis, operating system, AI, compiler design, database management, graphics, statistical analysis, and simulation.

Array Data Structure

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together.

This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).

The above image can be looked as a top-level view of a staircase where you are at the base of the staircase. Each element can be uniquely identified by their index in the array (in a similar way as you could identify your friends by the step on which they were on in the above example).

- Write a program to cyclically rotate an array by one
- Find the missing integer
- Count Pairs with given sum
- Find duplicates in an array
- Sort an Array using the Quicksort algorithm
- Find common elements in three sorted arrays
- Find the first repeating element in an array of integers
- Find the first non-repeating element in a given array of integers
- Subarrays with equal 1s and 0s
- Rearrange the array in alternating positive and negative items
- Find if there is any subarray with a sum equal to zero
- Find the Largest sum contiguous Subarray
- Find the factorial of a large number
- Find Maximum Product Subarray
- Find the longest consecutive subsequence
- Find the minimum element in a rotated and sorted array

- Max sum in the configuration
- Minimum Platforms
- Minimize the maximum difference between the heights
- Minimum number of jumps to reach the end
- Stock Span problem
- Find a triplet that sums to a given value
- Smallest positive missing number
- Find the row with a maximum number of 1’s
- Print the matrix in a Spiral manner
- Find whether an array is a subset of another array
- Implement two Stacks in an array
- Majority Element
- Wave Array
- Trapping Rainwater
- Maximum Index
- Max sum path in two arrays
- Find Missing And Repeating
- Stock buy and sell Problem
- Pair with the given sum in a sorted array
- Chocolate Distribution Problem
- Partition Equal Subset Sum
- Smallest Positive integer that can’t be represented as a sum
- Coin Change Problem
- Longest Alternating subsequence

String

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'

How String is represented in Memory?

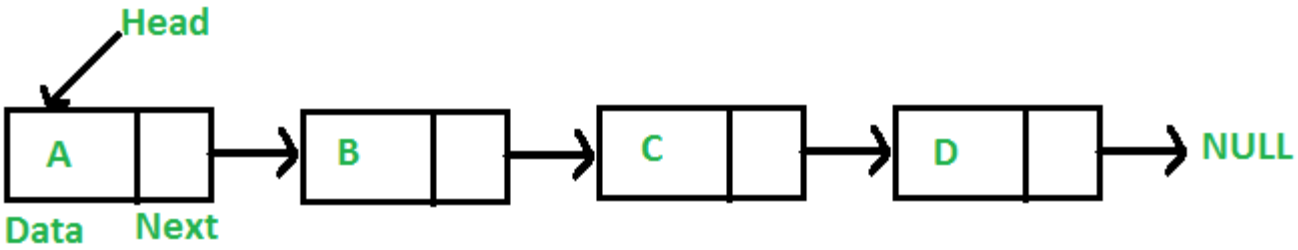
In C, a string can be referred to either using a character pointer or as a character array. When strings are declared as character arrays, they are stored like other types of arrays in C. For example, if str[] is an auto variable then the string is stored in the stack segment, if it's a global or static variable then stored in the data segment, etc.

- Reverse words in a given string
- Longest Common Prefix
- Roman Number to Integer
- Integer to Roman
- Closest Strings
- Divisible by 7
- Encrypt the String – II
- Equal point in a string of brackets
- Isomorphic Strings
- Check if two strings are k-anagrams or not
- Panagram Checking
- Minimum Deletions
- Number of Distinct Subsequences
- Check if string is rotated by two places
- Implement Atoi
- Validate an IP address
- License Key Formatting
- Find largest word in dictionary
- Equal 0,1, and 2
- Find and replace in String
- Add Binary Strings
- Sum of two large numbers
- Multiply two strings

- Look and say Pattern
- Minimum times A has to be repeated to make B a Substring
- Excel Sheet – I
- Form a Palindrome
- Find the N-th character
- Next higher palindromic number using the same set of digits
- Length of longest prefix suffix
- Longest K unique characters substring
- Smallest window in string containing all characters
- Longest Palindromic Subsequence
- Longest substring without repeating characters
- Substrings of length k with k-1 distinct elements
- Count number of substrings
- Interleaved Strings
- Print Anagrams together
- Rank the permutation
- A Special Keyboard

Linked List Data Structure

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

- Print the Middle of a given linked list
- Flattening a linked list
- Delete the elements in an linked list whose sum is equal to zero
- Delete middle of linked list
- Remove duplicate elements from sorted linked list
- Add 1 to a number represented as a linked list
- Reverse a linked list in groups of given size
- Detect loop in linked list
- Remove loop in linked list
- Find nth node from the end of linked list
- Function to check if a singly linked list is a palindrome
- Reverse alternate k node in a singly linked list
- Delete last occurrence of an item from linked list
- Rotate a linked list.
- Delete n nodes after m nodes of a linked list.
- Merge a linked list into another linked list at alternate positions.
- Write a function to delete a linked list.
- Write a function to reverse the nodes of a linked list.
- Why quicksort is preferred for arrays and merge sort for linked lists.

Stack Data Structure

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

- Infix to Postfix Conversion using Stack
- Prefix to Infix Conversion
- Prefix to Postfix Conversion
- Postfix to Prefix Conversion
- Postfix to Infix
- Convert Infix To Prefix Notation
- The Stock Span Problem
- Check for balanced parentheses in an expression
- Next Greater Element
- Next Greater Frequency Element
- Number of NGEs to the right
- Maximum product of indexes of next greater on left and right
- The Celebrity Problem
- Expression Evaluation
- Arithmetic Expression Evalution
- Evaluation of Postfix Expression
- Iterative Tower of Hanoi
- Print next greater number of Q queries

Queue

What is Queue?

A queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.

We define a queue to be a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end. The element which is first pushed into the order, the operation is first performed on that.

FIFO Principle of Queue:

- A Queue is like a line waiting to purchase tickets, where the first person in line is the first person served. (i.e. First come first serve).
- Position of the entry in a queue ready to be served, that is, the first entry that will be removed from the queue, is called the **front** of the queue(sometimes, **head** of the queue), similarly, the position of the last entry in the queue, that is, the one most recently added, is called the **rear** (or the **tail**) of the queue. See the below figure.

Characteristics of Queue:

- Queue can handle multiple data.
- We can access both ends.
- They are fast and flexible.

- Check if a queue can be sorted into another queue using a stack
- Breadth First Traversal or BFS for a Graph
- Level Order Tree Traversal
- Reverse a path in BST using queue
- Construct Complete Binary Tree from its Linked List Representation
- Program for Page Replacement Algorithms | Set 2 (FIFO)
- Check whether a given Binary Tree is Complete or not | Set 1 (Iterative Solution)
- Number of siblings of a given Node in n-ary Tree
- ZigZag Tree Traversal
- FIFO (First-In-First-Out) approach in Programming
- FIFO vs LIFO approach in Programming
- LIFO (Last-In-First-Out) approach in Programming

Hashing

Hashing is a technique or process of mapping keys, and values into the hash table by using a hash function. It is done for faster access to elements. The efficiency of mapping depends on the efficiency of the hash function used.

Let a hash function H(x) maps the value at the index **x%10** in an Array. For example if the list of values is [11,12,13,14,15] it will be stored at positions {1,2,3,4,5} in the array or Hash table respectively.

- Find Itinerary from a given list of tickets
- Find number of Employees Under every Employee
- Count divisible pairs in an array
- Check if an array can be divided into pairs whose sum is divisible by k
- Longest subarray with sum divisible by k
- Subarray with no pair sum divisible by K
- Print array elements that are divisible by at-least one other
- Find three element from different three arrays such that that a + b + c = sum
- Find four elements a, b, c and d in an array such that a+b = c+d
- Find the largest subarray with 0 sum
- Printing longest Increasing consecutive subsequence
- Longest Increasing consecutive subsequence
- Longest subsequence such that difference between adjacents is one | Set 2
- Longest Consecutive Subsequence
- Largest increasing subsequence of consecutive integers
- Count subsets having distinct even numbers
- Count distinct elements in every window of size k
- Maximum possible sum of a window in an array such that elements of same window in other array are unique
- Distributing items when a person cannot take more than two items of same type
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Check if array contains contiguous integers with duplicates allowed
- Length of the largest subarray with contiguous elements
- Find if there is a subarray with 0 sum
- Print all subarrays with 0 sum
- Find subarray with given sum | Set 2 (Handles Negative Numbers)
- Find four elements that sum to a given value

- Implementing our Own Hash Table with Separate Chaining in Java
- Implementing own Hash Table with Open Addressing Linear Probing in C++
- Vertical Sum in a given Binary Tree
- Group Shifted String
- Minimum insertions to form a palindrome with permutations allowed
- Check for Palindrome after every character replacement Query
- Maximum length subsequence with difference between adjacent elements as either 0 or 1 | Set 2
- Maximum difference between frequency of two elements such that element having greater frequency is also greater

Dynamic Programming

Dynamic Programming is mainly an optimization over plain recursion. Wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using Dynamic Programming. The idea is to simply store the results of subproblems, so that we do not have to re-compute them when needed later. This simple optimization reduces time complexities from exponential to polynomial.

For example, if we write simple recursive solution for Fibonacci Numbers, we get exponential time complexity and if we optimize it by storing solutions of subproblems, time complexity reduces to linear.

Coin ChangeProblem

Knapsack Problem

Binomial CoefficientProblem

Permutation CoefficientProblem

Program for nth Catalan Number

Matrix Chain Multiplication

Edit Distance

Subset Sum Problem

Friends Pairing Problem

Gold Mine Problem

Assembly Line SchedulingProblem

Painting the Fenceproblem

Maximize The Cut Segments

Longest Common Subsequence

Longest Repeated Subsequence

Longest Increasing Subsequence

Space Optimized Solution of LCS

LCS (Longest Common Subsequence) of three strings

Maximum Sum Increasing Subsequence

Count all subsequences having product less than K

Longest subsequence such that difference between adjacent is one

Maximum subsequence sum such that no three are consecutive

Egg Dropping Problem

Maximum Length Chain of Pairs

Maximum size square sub-matrix with all 1s

Maximum sum of pairs with specific difference

Min Cost PathProblem

Maximum difference of zeros and ones in binary string
Minimum number of jumps to reach end
Minimum cost to fill given weight in a bag
Minimum removals from array to make max –min <= K
Longest Common Substring
Count number of ways to reacha given score in a game
Count Balanced Binary Trees of Height h
LargestSum Contiguous Subarray [V>V>V>V IMP]
Smallest sum contiguous subarray
Unbounded Knapsack (Repetition of items allowed)
Word Break Problem
Largest Independent Set Problem
Partition problem
Longest Palindromic Subsequence
Count All Palindromic Subsequence in a given String
Longest Palindromic Substring
Longest alternating subsequence
Weighted Job Scheduling
Coin game winner where every player has three choices
Count Derangements (Permutation such that no element appears in its original position) [IMPOR
Maximum profit by buying and selling a share at most twice [IMP]
Optimal Strategy for a Game
Optimal Binary Search Tree
Palindrome PartitioningProblem
Word Wrap Problem
Mobile Numeric Keypad Problem [IMP]
Boolean Parenthesization Problem
Largest rectangular sub-matrix whose sum is 0
Largest area rectangular sub-matrix with equal number of 1's and 0's [IMP]
Maximum sum rectangle in a 2D matrix
Maximum profit by buying and selling a share at most k times
Find if a string is interleaved of two other strings
Maximum Length of Pair Chain

Graph Data Structure And Algorithms

What is Graph in Data Structure and Algorithms?

A Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices(V) and a set of edges(E). The graph is denoted by G(E, V).

Components of a Graph

- **Vertices:** Vertices are the fundamental units of the graph. Sometimes, vertices are also known as vertex or nodes. Every node/vertex can be labeled or unlabelled.
- **Edges:** Edges are drawn or used to connect two nodes of the graph. It can be ordered pair of nodes in a directed graph. Edges can connect any two nodes in any possible way. There are no rules. Sometimes, edges are also known as arcs. Every edge can be labeled/unlabelled.

Graphs are used to solve many real-life problems. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender, locale etc

- Introduction to Graphs
- Graph and its representations
- Types of Graphs with Examples
- Basic Properties of a Graph
- Applications, Advantages and Disadvantages of Graph
- Difference between graph and tree

- **Introduction, DFS and BFS :**
- Breadth First Traversal for a Graph
- Depth First Traversal for a Graph
- Applications of Depth First Search
- Applications of Breadth First Traversal
- Graph representations using set and hash

- Find Mother Vertex in a Graph
- Transitive Closure of a Graph using DFS
- Find K cores of an undirected Graph
- Iterative Depth First Search
- Count the number of nodes at given level in a tree using BFS
- Count all possible paths between two vertices
- Minimum initial vertices to traverse whole matrix with given conditions
- Shortest path to reach one prime to other by changing single digit at a time
- Water Jug problem using BFS
- Count number of trees in a forest
- BFS using vectors & queue as per the algorithm of CLRS
- Level of Each node in a Tree from source node
- Construct binary palindrome by repeated appending and trimming
- Transpose graph
- Path in a Rectangle with Circles
- Height of a generic tree from parent array
- BFS using STL for competitive coding
- DFS for a n-ary tree (acyclic graph) represented as adjacency list
- Maximum number of edges to be added to a tree so that it stays a Bipartite graph
- A Peterson Graph Problem
- Implementation of Graph in JavaScript
- Print all paths from a given source to a destination using BFS
- Minimum number of edges between two vertices of a Graph
- Count nodes within K-distance from all

Feel free to connect with me:
https://www.linkedin.com/in/himanshushekhar16/