Topics:
1. Types
2. ADT
3. Design of Algorithm
4. Array Implementation
5. Linked List
6. Types of Linked List
7. Single Linked List
8. Double Linked List
9. Circular Linked List
10. Advantage of Linked List over Array
11. Revenue of Linked List
12. Advantages of Doubly Linked list, circular list over single linked list
13. Stack
14. Overflow and Underflow condition
15. Polish Notation
16. Postfix evaluation using stack
17. Queue
18. Insertion & deletion
19. Priority Queue
20. Circular Queue
21. Dequeue
22. Advantages of Circular Queue
23. Sorting
i. Insertion
ii. Selection
iii. Bubble Sort
iv. Quick Sort
v. Binary Tree Sort
vi. Heap Sort
vii. Merge Sort
viii. Radix Sort
ix. External Sort
24. Searching
i. Linear Search
ii. Binary Search
25. Tree
i. Binary Tree
ii. Types of binary tree
iii. Preorder, Inorder, Postorder, External of binary tree
iv. Binary Search Tree
v. AVL tree
vi. Threaded binary tree
vii. B tree
viii. B+ Tree
ix. Heap tree
26. Graph: Spanning Tree, Minimum Tree, Prin’s & Krushkals algorithm to minimum Spanning Tree, Breadth First Search, Depth First Seach.
27. Analysis of Algorithm : Time and Space Complexity.

DSA 14 February, 2023

Data Structure
Primitive- Int, Float, String.
Non Primitive – Array, List, File.
Linear – Stack, Queue.
Non Linear – Tree.

Data Structure: Data Structure is a way of organizing and storing data in a computer so that it can be accused and used efficienctly. There are many types of data structures.
1. Arrays
2. Linked List
3. Stack.
4. Queues
5. Trees
6. Graphs
Note: DataStructure + Algorithm = Program

Primitive : These are basic structure that are directly operated upon basic instructution.
Eg: integer, float, number, character, constraint, string, pointer etc are the example of these

Non Primitive : These are derived from primitice datastructure . The non primitive data structure emphasises on the structure of a group of homogenous or heterogenous data item.
Array, Linked List are the example of these categories.

ADT(Abstract Data Types): An abstract data type commonly called ADT is nothing but a set of operation which is used with the component of the element of that abstract data types.
These are two basic structure used to implement on ADT:
i. Array
ii. Linked List
We can simply take the data type of a list ADT-
i. Insertion
ii. Deletion
iii. Search
iv. Display
Here, the item is component of ADT list can contain member of items. Operation of these item can be Insertion, Deletion, Search and display.

DSA 17 February, 2023
Introduction to Algorithm, designing and data structure
Procedure for problem solving:
i. Understanding the problem
ii. Construction of the list of variables
iii. Output design
iv. Program Development
v. Testing the Program
vi. Validating the Program

Algorithm : An algorithm is a set of instruction or a step by step procedure for solving a problem or achieving a specific goal. It is a series of well defined steps that are used to perform a specific task or to solve a specific task or to solve a specific problem. Algorithm are used in many areas of computer science, including programming , data analysis and artificial intelligence.

Design and Analysis of a Program

For designing of any algorithm some important things should be considered run time, space and simplicity of algorithm.

Some common approach for designing algorithm are:

1. Greedy Algorithm
2. Divide and Conquer
3. Non recursive algorithm
4. Randomize
5. Modular Programming Approach

Greedy Algorithm: This Algorithm work in steps in each step accelates the best available option until all option finished. This approach is widely used in so many places for designing algorithm.
Eg: Sort List, Path algorithm

Divide and Conquer: Here, we divide a problem into smaller subproblems, solve them independently, and then combine the solutions to obtain the final solution to the original problem.
Eg: Quick Sort, where we divide the list into several small list after those small list we combine and get the initial list sorted.

Non recursive algorithm: As we know recursive is very powerful technique supported by C but some compiler do not support them. In some places recursive is not as effective and it can be avoided with iteration concept is used. So, we need to design the algorithm with non recursive approach.

Randomize algorithm: In thse algorithm, we used the feature of random number instead of fixed number. Performance of some algorithm depend upon the input data. It gives different result with different input data.

Modular Programming: Here, we divide, the big problem into smaller one which are totally different from each other, then we combine the solution of all smaller programmer and we get the solution of big problem. Actually here we can use different algorithm for all small problem. These approach for all small problem. These approach gives different module for different problem makes it easier to handle big problem.

Complexity of Algorithm or Analysis of Algorithm

Analysis of algorithm is required to compare the algorithm and recognised the most efficient or best one. Algorithm are generally analysed on their time and space requirement. One way of comparing algorithm is to compare the exact running time of all algorithm, the running time is depend on the language and machine implementation of algorithm. Even if the machine and language are kept same the calculation of exact time wil be very different as it require the count of instruction executed by the hardware and the time taken to execute each instruction. So, the time efficiency is not measured in time units like seconds or microseconds. The running time is generally depend on the size of the input.

Time Complexity: The time complexity of a program is the amount of time an algorithm takes to solve a problem as a function of the input size. Suppose space is fixed for an algorithm then only run for an algorithm then only run time will be considered for obtaining the complexity of algorithm. These are:

1. Best Case: In thse case the algorithm searches the elements in first time itself.
2. Worst Case: In these case, the algorithm we find the elements at the end or when searching of element fails. These could involved comparing the key to its list value for a total or comparision.
3. Average Case: Analysis the average case behaviour algorithm is little bit complex. We take the probality with the list of data. Average number algorithm should be the average number of steps but since data can be at any place so finding exact behaviour of algorithm is difficult. As the volume of data increases average case of algorithm behaves like worst case.

Space Complexity: The space complexity of the program, the amount of memory it needs to runs to complision. The space needed by the program.