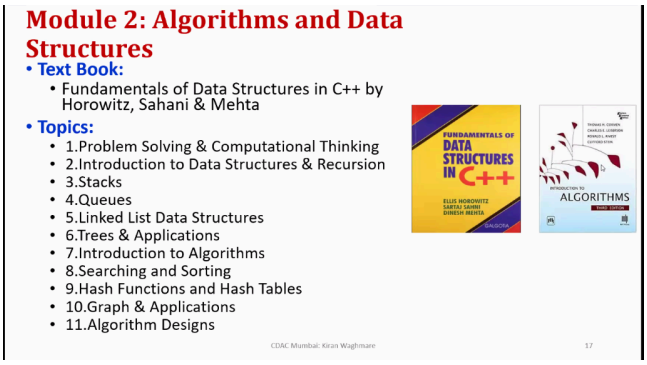
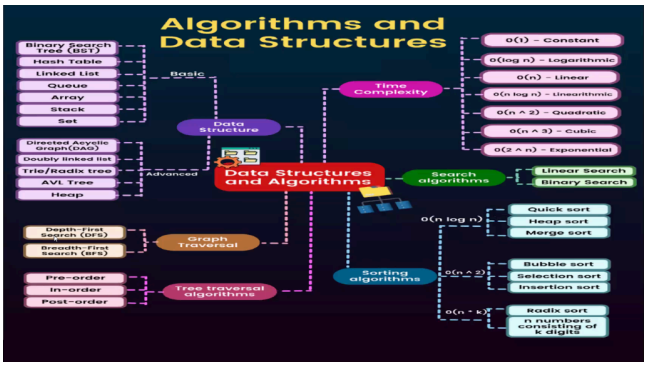
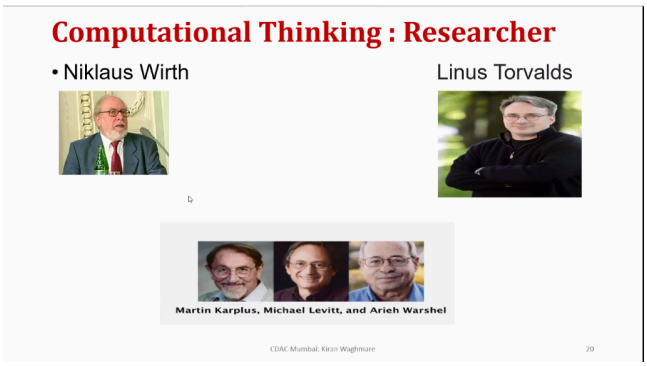
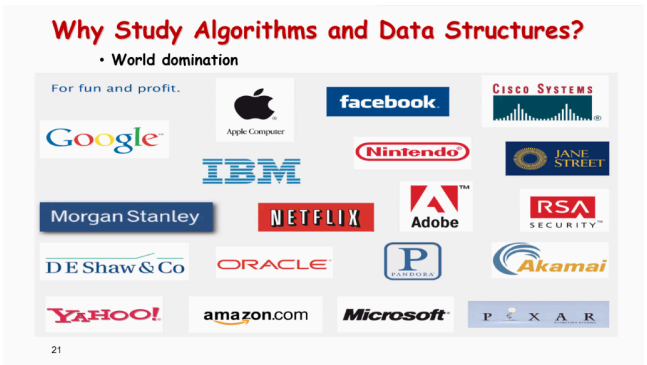
Day1_ADSA_Sanket_Shalukar

Tuesday, September 16, 2025 10:57 AM









Logic :

build a logic

Data :

Collection of Raw facets

Algorithm :

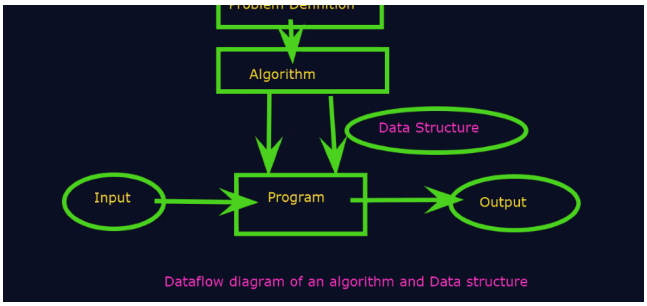
The essence of computational procedure, step by step instructions.

Program:

An implementation of an algorithm in some programing language is called as program

Data Structure :

Organization of data needed to solve the problem.



Algorithm :

- The algorithm is a sequence of unambiguous instruction of

Characteristic of Algorithm:

Finite:
must terminate after a finite number of steps

Definite:
- each step must be precisely defined

Input :
- Takes zero or more number of inputs

Output:
- produces at least one output
- Effective : each step must be basic and achievable

Types of / Strategis of Algorithm:

Greedy Algorithm:
- Makes the locally optimal choice at each step with the hope of finding a global optimum

Divide and Conquer :
- Break the problem into smaller subproblems solves them recursively and combines the results

Dynamic Programing:
- Solves complex problems into smaller subproblems solves them recursively
- And simplify it into similar subproblems and solve each one of them

Backtracking :
- Tries all possible solutions among those that fails to satisfy the problem's constraint.

Brute Force:
- Tries all possible solutions until the correct one is found.

Recursion :
- Solves a problem by solving the smaller instance of the same problem

Search Algorithms :
- Finds an implement in a data structure

Sorting Algorithms
- Order the element in particular sequence

Data Structures:
- A way of organizing, managing and storing data to enable efficient access and modification.

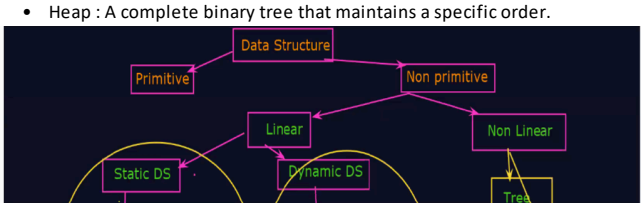
Types of Data Structure

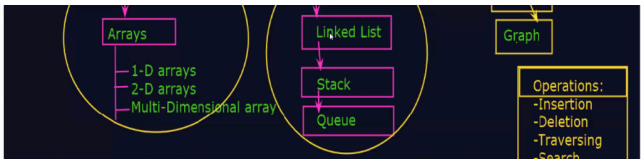
1. **Linear Data Structure**

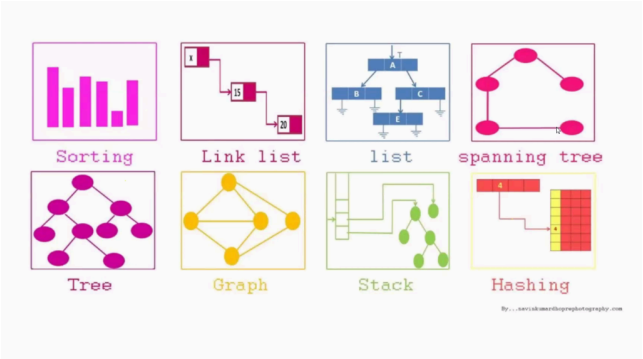
- Data elements are arranged sequentially
- Array- A collection of data elements, stored in contiguous memory locations
- LinkedList – A collections of data elements, where each element points to the next.
- Stack- Follows **Last in first out (LIFO)** principle
- Queue – Follows First in First Out (FIFO) Principle
- Hash Table :
- Stores key-value paires offering fast lookup(Search)

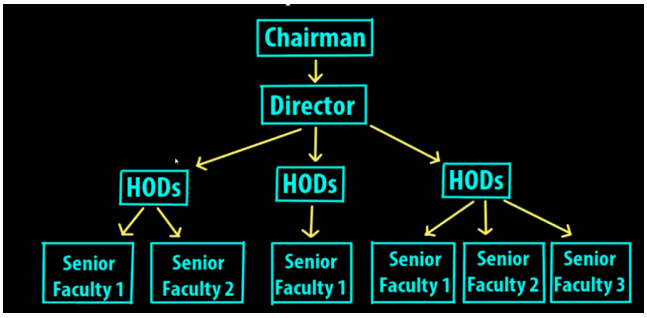
2. **Non liner Data Structure**

- Data elements are arranged in hierarchical or in network (non linearly and in some relationship one to many and many to one
- Trees : A hierarchical structure with root, nodes and edges
- Graph : Consists of nodes(vertical) connected by edges.
- Heap : A complete binary tree that maintains a specific order.

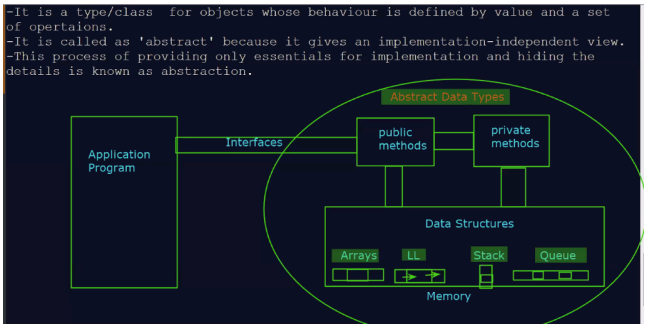








Abstract Data Type (ADT)



- of opertaions.
- It is called 'abstract' because it gives an implementation-independent view.
- This process of providing only essentials for implementation and hiding the details is known as abstraction.

- **Operation on Data structures:**
  - Insertion: Adding new elements
  - Deletion: Removing an elements
  - Traversing: Accessing each element
  - Searching: Finding an element
  - Sorting: Arranging elements in a particular order.
  - Merging: Combining two strucutres into one.

HW: Diff Linear and Nonlinear : 10 ptr
HW: Diff Static and Dynamic

Example 1

```
public static void main(String[] args) {

    ArrayList<String> list = new ArrayList<>();

    //Add : add()
    list.add("Apple");
    list.add("Banana");
    list.add("Chickoo");
    System.out.println("Add List="+list);

    //Update : set()
    list.set(1,"Cherry");
    System.out.println("Update List="+list);

    //Remove by index: remove()
    list.remove(0);
    System.out.println("URemove by index List="+list);

    //Remove by value:remove()
    list.remove("Cherry");
    System.out.println("URemove by value List="+list);

   //Dispaly
   System.out.println("Display List="+list);
  }
}
```

Example 2

```
//To remove duplicates from ArrayList

import java.util.*;

public class TestDemo2 {
    public static void main(String[] args) {

        ArrayList<Integer> list = new ArrayList<>(Arrays.asList(1,2,2,3,1,4,4,5));

        //PReserve insertion order
        ArrayList<Integer> unique = new ArrayList<>(new LinkedHashSet<>(list));

       //Dispaly
       System.out.println("Display original List="+list);
       System.out.println("Display Without duplicates List="+unique);
    }
}
```

Example 3 (Reverse Array list)

```
import java.util.*;

public class TestDemo3 {
    public static void main(String[] args) {

        ArrayList<String> list = new ArrayList<>(Arrays.asList("Apple","Banana",
        "Chickoo"));
        System.out.println("Display original List="+list);

        Collections.reverse(list);
       //Dispaly

       System.out.println("Display Reverse List="+list);
    }
}
```

```
1 error

C:\Test>javac TestDemo3.java

C:\Test>java TestDemo3
Display original List=[Apple, Banana, Chickoo]
Display Reverse List=[Chickoo, Banana, Apple]

C:\Test>
```

Example 4 and 5

```
    //Method 1: For loop with index
    for(int i=0;i<list.size();i++){
        System.out.print(list.get(i)+ " " );
    }

    System.out.println();
    System.out.println("------------------------");
    //Method 2:  for-each
    for(String s : list){
        System.out.print(s+" ");
    }

    System.out.println();
    System.out.println("------------------------");
    //Method 3:
    Iterator<String> it = list.iterator();
    while(it.hasNext()){
        System.out.print(it.next()+" ");
    }


   //System.out.println("Display Reverse List="+list);
}
```

```
        System.out.println();
        System.out.println("------------------------");
        //Method 3: Iterator : Forward direction print karta hai
        Iterator<String> it = list.iterator();
        while(it.hasNext()){
            System.out.print(it.next()+" ");
        }

        System.out.println();
        System.out.println("------------------------");
        //Method 3: ListIterator : Backward direction print karta hai
        ListIterator<String> lit = list.listIterator();
        while(lit.hasPrevious()){
            System.out.print(lit.previous()+" ");
        }


     //System.out.println("Display Reverse List="+list);
    }
}
```

```
C:\Test>java TestDemo4
Display original List=[Apple, Banana, Chickoo]
Apple Banana Chickoo
------------------------
Apple Banana Chickoo
------------------------
Apple Banana Chickoo
------------------------
Apple Banana Chickoo
------------------------
Chickoo Banana Apple
C:\Test>
```

Example 6 (Using lambda function)

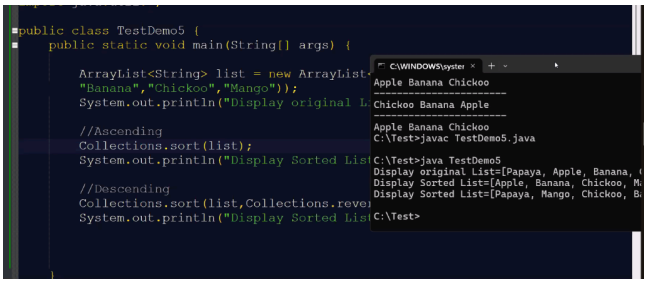
```
        System.out.println();
        System.out.println("------------------------");
        //Method 6:
        list.forEach(x -> System.out.print(x
    }
}
```

```
Display original List=[Apple, Banana,
Apple Banana Chickoo
------------------------
Apple Banana Chickoo
------------------------
Apple Banana Chickoo
------------------------
Apple Banana Chickoo
------------------------
Chickoo Banana Apple
------------------------
AppleBananaChickoo
C:\Test>
```

Example 7 (Sort array and reverse array)

```
//Sort an arraylist in ascending and descending order

import java.util.*;

public class TestDemo5 {
    public static void main(String[] args) {

        ArrayList<String> list = new ArrayList<>(Arrays.asList("Papaya","Apple",
        "Banana","Chickoo","Mango"));
        System.out.println("Display original List="+list);

        //Ascending
        Collections.sort(list);
        System.out.println("Display Sorted List="+list);

        //Descending
        Collections.sort(list,Collections.reverseOrder());
        System.out.println("Display Sorted List="+list);


    }
```

```
//Sort an arraylist in ascending and descending order

import java.util.*;
```



Recursion

Recursion is when a function calls itself to solve a problem.
It keeps calling itself until it reaches a **base case** (a stopping condition).