23CSE203
DATA STRUCTURES AND ALGORITHMS

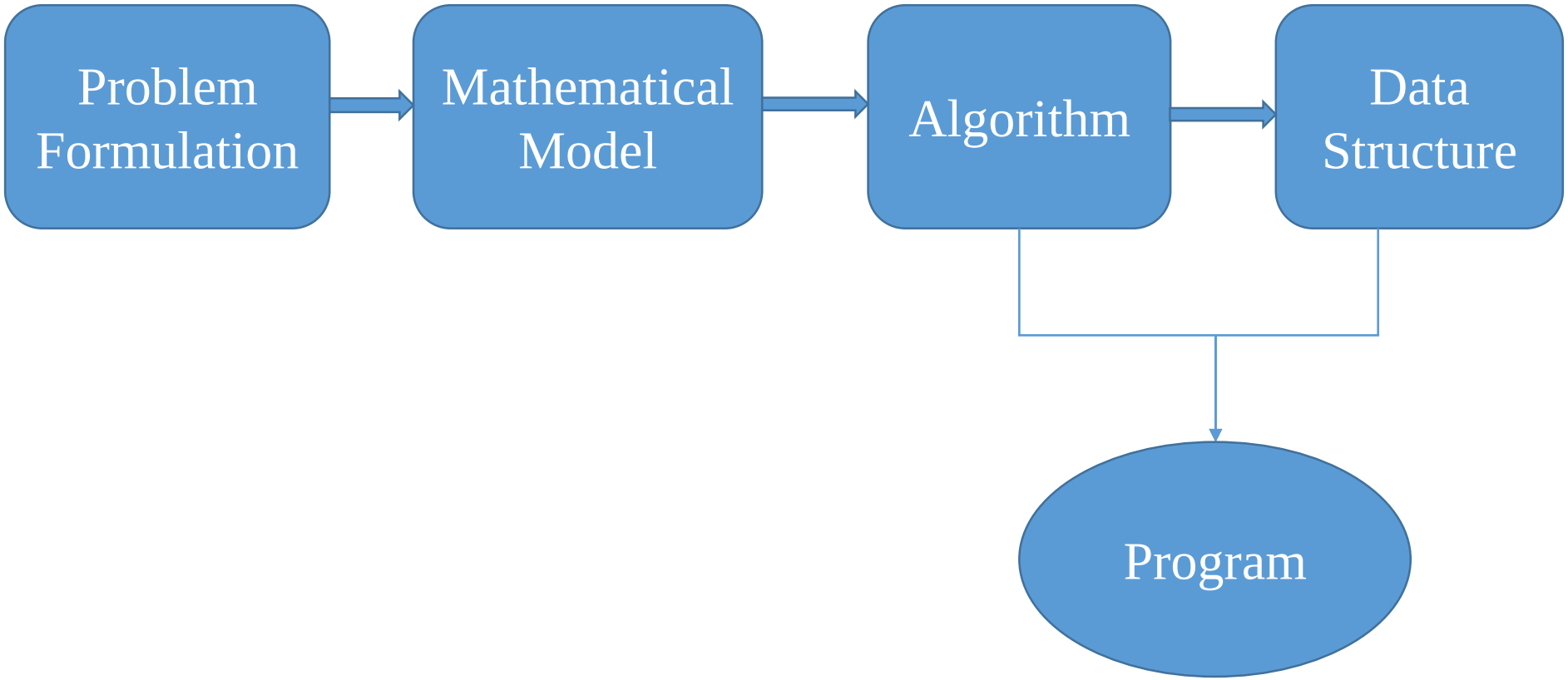
Dr. Anjali Patel

Dept. of Computer Science and Engineering,

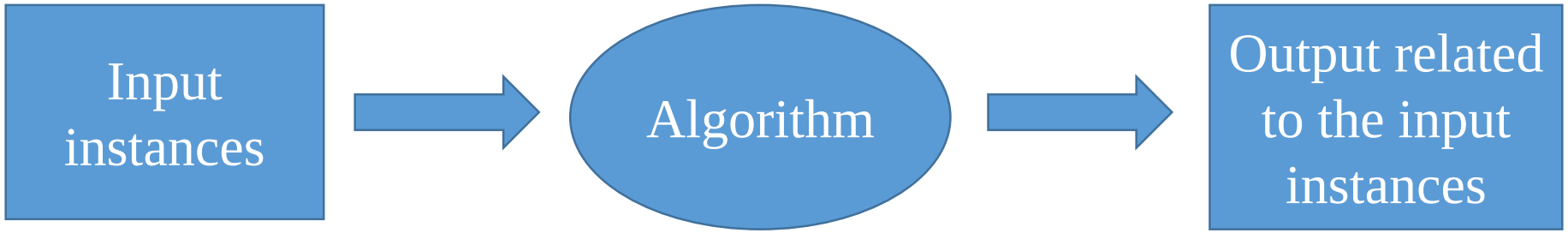
Amrita Vishwa Vidyapeetham,

Bengaluru

How to solve a problem ?



What is an algorithm ?



- Algorithm describes actions on the input instances
- It is a step-by-step instructions for performing some task in a finite amount of time
- Infinite many correct algorithm for the same problem

Algorithm to Make Indian tea (Chai)

Input: Water (approx. 1 cup) Milk (approx. 1 cup) Loose tea leaves (1–2 teaspoons) or tea bags Sugar (1–2 teaspoons, as desired) Optional spices: ginger, cardamom, cloves, cinnamon

Step 1: Place a saucepan on the stove.

Step 2: Pour 1 cup of water into the saucepan.

Step 3: If using, add crushed ginger, cardamom, cloves, or cinnamon.

Step 4: Heat water (and spices) until it starts boiling.

Step 5: Add tea leaves or tea bag(s) into the boiling water.

Step 6: Boil for 2–3 minutes to let the tea brew.

Step 7: Pour 1 cup of milk into the saucepan.

Step 8: Add sugar according to taste.

Step 9: Stir the mixture gently.

Step 10: Allow the chai to come to a boil again.

Step 11: Reduce heat and let it simmer for 2–3 minutes to develop flavor.

Step 12: Turn off the heat. Use a strainer to filter chai into a cup or teapot.

Step 13: Pour into cups.

Output: Hot, aromatic chai ready to drink.

List of even numbers among {1,2,3,4,5,6,7,8,9,10}

Algo 1

```
FOR i FROM 1 TO 10 DO
   IF i MOD 2 = 0 THEN
      PRINT i
   END IF
END FOR
```

Algo 2

```
SET i = 1
WHILE i <= 10 DO
   IF i MOD 2 = 0 THEN
      PRINT i
   END IF
   i = i + 1
END WHILE
```

Algo 3

```
FOR i FROM 2 TO 10 STEP 2
DO
   PRINT i
END FOR
```

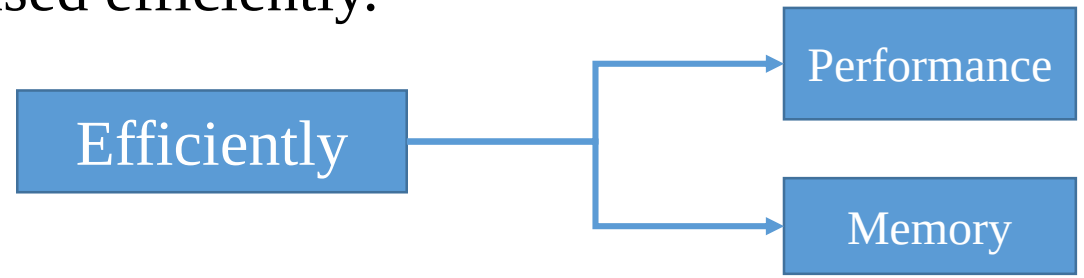
Algorithm	Approach	Efficiency	Readability
Algo 1	Loop 1–10, check MOD each time	10 iterations, 5 checks pass, 5 fail	Very clear and common
Algo 2	Same as above with WHILE	Same as above	Also clear, slightly more verbose
Algo 3	Loop by steps of 2 (start from 2)	5 iterations only No conditional check needed	Most efficient and elegant

Why Data Structure?

- To store and retrive information in computer as fast as possible
- To write efficient algorithms for a given problem
- To compare algorithms written for the same problem

What is data structure?

- A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.



- Data structure aims at two points as follows:

➢ It should be strong enough to reflect the actual relationships of the data in the real situation

➢ It should be simple enough to efficiently process the data whenever necessary

- The way we organize information can have a lot of impact on the performance.

Example: Suppose, you want to have a book on Set Theory from a public library, to do that you have to first go to the maths section, then to set theory section. If these books are not organized in this manner and just distributed randomly then it will be really a cumbersome process to find a book on set theory.

- Store
- Organize
- Access
- Manipulate

The need for data structure

- Using the proper data structure can make the difference between a program running in a few seconds and one requiring many days

- For a given programming problem, there will be many solutions. The solution which solves the problem within the required/available resource constraints is called an efficient

Steps to be followed when selecting a data structure

- Analyze your problem to determine the basic operations that must be supported.

- Examples of basic operations: Inserting a data item into the data structure, deleting a data item from the data structure and finding a specified data item.

- Quantify the resource constraints for each operation.

Note: Data structures manage how data is stored and accessed, while algorithms focus on processing this data.

Data Structure (DS)

Primitive DS

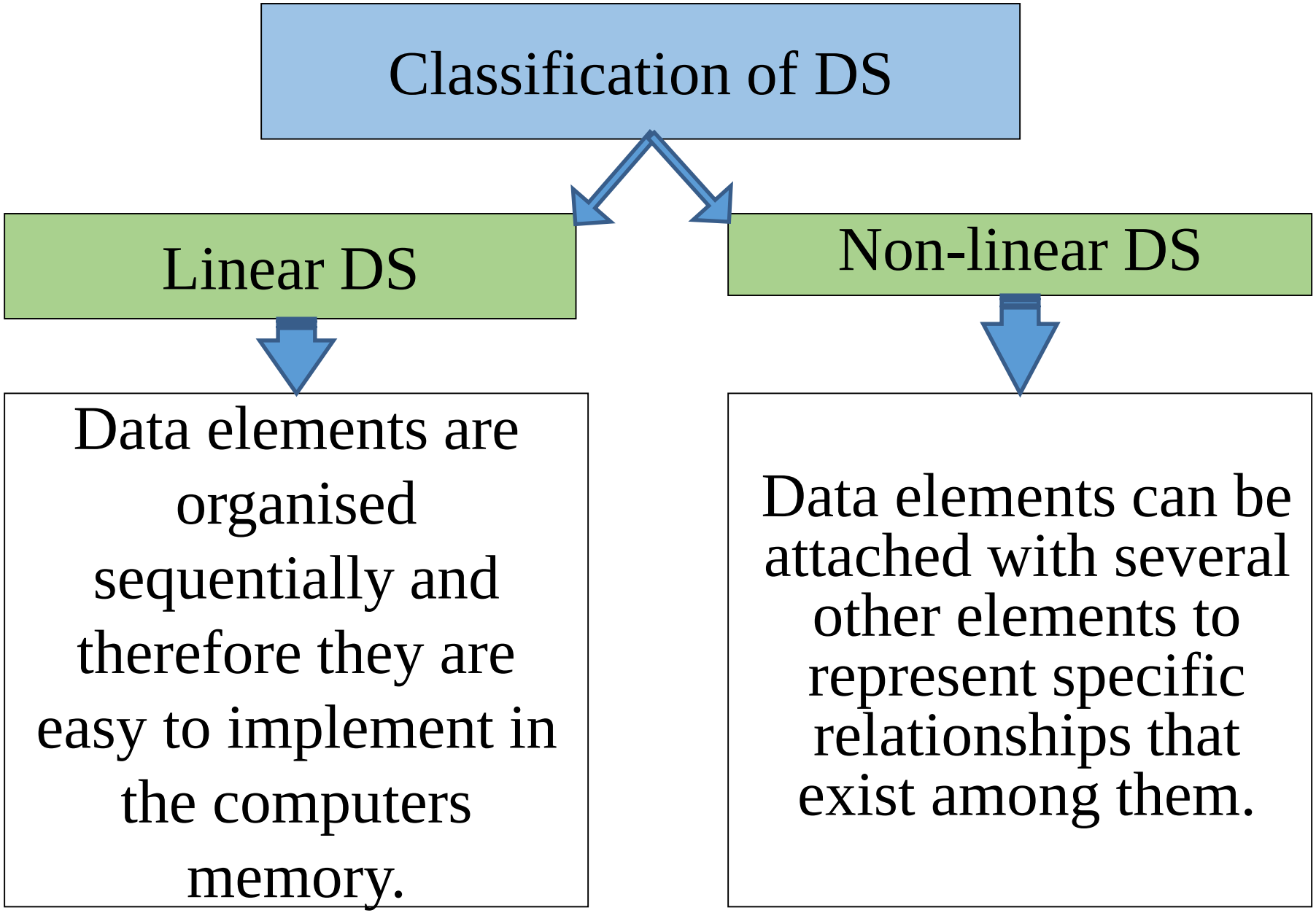
- A single value and are typically predefined in most programming languages.
- Simple and cannot be further divided.
- Also known as built in data structure.
- Ex. int, float, char

Non-primitive DS

- Complex data structures derived from primitive types.
- Store multiple values and often involve relationships between elements.
- Also known as user define data structure.
- Ex. Array, linked list, stack, queue, tree

To study

- Classification of data structures
- Commonly used Data structure
- Costs & benefits associated with every DS
- How to measure effectiveness of DS or algorithm


Classification of DS
Linear DS
Non-linear DS
Data elements are organised sequentially and therefore they are easy to implement in the computers memory.
Data elements can be attached with several other elements to represent specific relationships that exist among them.

Classification of DS

Linear DS

- Arrays (Linear, Multidimensional)
- Pointers
- Stacks
- Queues
- Linked List (Single, Double, Circular)
- Application of Linked List (Polynomial addition, Sparse matrices)

Non-linear DS

- Graph and its representation
- Tree
- Binary tree
- Red-black tree
- AVL tree
- B tree
- B+ tree
- Spanning tree
- Searching Algorithm
- Hashing

Arrays

- Collection of elements accessible by an index.
- The linear Structure that has the relationship between elements, represented by means of sequential memory location.
- An array is defined as a set of finite number of homogeneous elements or same data items.
- It means an array can contain one type of data only, either all integer, all float-point number or all character.

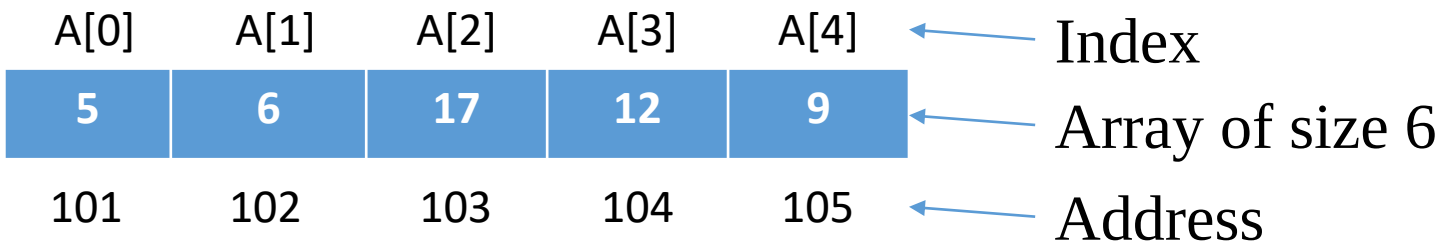
- Simply, declaration of array is as follows:

  int arr[10]

- Where int specifies the data type or type of elements arrays stores.
- “arr” is the name of array & the number specified inside the square brackets is the number of elements an array can store, this is also called sized or length of array.
- The first element of the array has index zero[0]. It means the first element and last element will be specified as: arr[0] & arr[9], respectively.
- The number of elements that can be stored in an array, that is the size of array or its length is given by the following equation:

  (Upperbound-lowerbound)+1

- Example: int A[5]={5,6,17,12,9}



Key notes on array

- *Static and Dynamic Arrays:*

➢Static arrays – Size cannot be changed

➢Dynamic arrays – Size can be changed

- Elements in an array are stored in continuous memory locations.
- Although changing the size of an array is not possible, one can always reallocate it to some bigger memory location. Therefore resizing in an array is a costly operation.

Operation on linear DS

The following operations will be performed on any linear structure, whether it is an array or a linked list.

i. **Traversal**: Processing each element in the list.
ii. **Search**: Finding the location of an element with a given value or the record with a given key.
iii. **Insertion**: Adding a new element to the list
iv. **Deletion**: Removing an element from the list
v. **Sorting**: Arranging the elements in some type of order.
vi. **Merging**: Combining two lists into a single list

Example-1 (Traversing Algo.)

Let there be a linear array (LA) with lower bound LB and upper bound UB.

Algorithm:

1. set k=LB.
2. Is k<= UB, apply process to LA[k] and go to step 3.
   Otherwise go to step 4.
3. set k=k+1 and go to step 2.
4. Exit.

Qn.1 Write an algorithms for inserting an element in a linear array.

Algorithm for Inserting into a Linear Array:

INSERT (LA, N, K, ITEM)

// LA is a linear array with N elements and K is a positive integer such that K<=N. This algorithm inserts an element ITEM in to the K'th Position in LA.

Algorithm: (Inserting)

1. Set J=N.
2. Repeat steps 3 and 4 while J>= K.
3. Set LA[J+1]=LA[J].
4. Set J=J-1 and go to step 2.
5. Set LA[K]=ITEM.
6. Set N=N+1.
7. Exit.

Qn.2 Write an algorithms for deleting an element in a linear array.

Algorithm for Deleting from a Linear Array:

INSERT (LA, N, K, ITEM)

// LA is a linear array with N elements and K is a positive integer such that K<=N. This algorithm deletes the K'th item in LA.

Algorithm: (Deleting)

1. set ITEM= LA[K];
2. Repeat for J=K to N-1
   Set LA[J]=LA[J+1]
3. Set N=N-1.
4. Exit.