CHAPTER:1 INTRODUCTION TO DATA STRUCTURE (2 HRS)

By: Er. Pralhad Chapagain

Lecturer

CONTENT

DATA STRUCTURE- OVERVIEW

- Data Structure is a systematic way to organize data in order to use it efficiently. Following terms are the foundation terms of a data structure.
- **Interface**
  - Each data structure has an interface. Interface represents the set of operations that a data structure supports.
  - An interface only provides the list of supported operations, type of parameters they can accept and return type of these operations.
- **Implementation**
  - Implementation provides the internal representation of a data structure.
  - Implementation also provides the definition of the algorithms used in the operations of the data structure.

DATA STRUCTURE- OVERVIEW

Characteristics of a Data Structure:

- **Correctness**
  - Data structure implementation should implement its interface correctly.
- **Time Complexity**
  - Running time or the execution time of operations of data structure must be as small as possible.
- **Space Complexity**
  - Memory usage of a data structure operation should be as little as possible.

DATA STRUCTURE- OVERVIEW

Need for Data Structure:

- As applications are getting complex and data rich, there are three common problems that applications face now-a-days.
- **Data Search**
  - Consider an inventory of 1 million(10^6) items of a store. If the application is to search an item, it has to search an item in 1 million(10^6) items every time slowing down the search. As data grows, search will become slower.
- **Processor speed**
  - Processor speed although being very high, falls limited if the data grows to billion records.
- **Multiple requests**
  - As thousands of users can search data simultaneously on a web server, even the fast server fails while searching the data.

DATA STRUCTURE- OVERVIEW

- To solve the above-mentioned problems, data structures come to rescue.
- Data can be organized in a data structure in such a way that all items may not be required to be searched, and the required data can be searched almost instantly.

Execution Time Cases:

- There are three cases which are usually used to compare various data structure's execution time in a relative manner.
- **Worst Case**
  - This is the scenario where a particular data structure operation takes maximum time it can take.
  - If an operation's worst case time is $f(n)$ then this operation will not take more than $f(n)$ time where $f(n)$ represents function of n.

DATA STRUCTURE- OVERVIEW

- **Average Case**
  - This is the scenario depicting the average execution time of an operation of a data structure.
  - If an operation takes $f(n)$ time in execution, then m operations will take $m*f(n)$ time.
- **Best Case**
  - This is the scenario depicting the least possible execution time of an operation of a data structure.
  - If an operation takes $f(n)$ time in execution, then the actual operation may take time as the random number which would be maximum as $f(n)$.

Basic Terminology:

- **Data** − Data are values or set of values.
- **Data Item** − Data item refers to single unit of values.
- **Group Items** − Data items that are divided into sub items are called as Group Items.

DATA STRUCTURE- OVERVIEW

- **Elementary Items** – Data items that cannot be divided are called as Elementary Items.
- **Attribute and Entity** – An entity is that which contains certain attributes or properties, which may be assigned values.
- **Entity Set** – Entities of similar attributes form an entity set.
- **Field** – Field is a single elementary unit of information representing an attribute of an entity.
- **Record** – Record is a collection of field values of a given entity.
- **File** – File is a collection of records of the entities in a given entity set.

DATA STRUCTURE- CLASSIFICATION

1. **STATIC AND DYNAMIC DATA STRUCTURE:**

- **Static Data Structure** are those whose size is fixed at compile time and doesn't grow or shrink at run time. E.g. Array
- **Dynamic Data Structure** are those whose size is not fixed at compile time and that can grow or shrink at run time so as to make efficient use of memory. E.g. Linked List
- **Arrays and Linked list** are basic data structures that are used to implement other data structure such as stack, queue, trees and graphs. So any other data structure can be static or dynamic depending on whether they are implementing using an array or linked list.

2. PRIMITIVE AND NON-PRIMITIVE DATA STRUCTURE:

- **Primitive Data Structure** are those data types which are provided by a programming language as basic building block. So, primitive data types are predefined types of data, which are supported by programming language. E.g. integer, float, character etc.

DATA STRUCTURE- CLASSIFICATION

- **Non-Primitive Data Structure** are those data types which are not defined by programming language but are instead created by the programmer by using primitive data types. E.g. array, linked list, stack, queue, tree etc.

3. LINEAR AND NON-LINEAR DATA STRUCTURE:

- A data structure is said to be **linear** if its elements form a sequence i.e. linear list. So, here while traversing the data elements sequentially, only one element can directly be reached. Example: Array, stack, queue

- A data structure is said to be **non linear** if its elements do not form a sequence. So, here every data items is attached to several other data items. Tree and Graph are examples of non linear data structure.

DATA TYPES AND ABSTRACT DATA TYPE (ADT)

- A **data type** is a classification of data, which can store specific type of information.
- Data types are primarily used in computer programming in which variables are created to store data.
- Each variable is assigned a data type that determine what type of data the variable may contain.
- Data type is a collection of values and set of operation on those values.
- Example: in the statement int a=5, 'a' is a variable of data type integer which stores integer value 5 and allow different mathematical operations like addition, subtraction, multiplication and division.

ADT:

- We can perform some specific operations with data structure, so data structure with these operations are called **Abstract Data Types (ADT).**
- ADT have their own data type and methods to manipulate data.
- It is a useful tool for specifying the logical properties of a data type.

DATA TYPES AND ABSTRACT DATA TYPE (ADT)

- A data type or data structure is the implementation of ADT.
- So ADT allow us to work with abstract idea behind a data type or data structure without getting bagged down in the implementation detail. The abstraction in this case is called an abstract data type.

Advantage of abstract data type:

- **Encapsulation:**
  - Encapsulation is a process by which we can combine code and data and it manipulates into a single unit.
  - While working with ADT, the user doesn’t require knowing how the implementation works because implementation is encapsulated in a simple interface.
- **Flexibility:**
  - If different implementations of ADT are equivalent then they can be interchangeable in the code. So user have flexibility to select the one which is most efficient in different situations.

DATA TYPES AND ABSTRACT DATA TYPE (ADT)

- **Localization if change:**
  - If ADT is used then if changes are made to the implementation then it doesn’t require any changes in the code.
- Example: Stack is ADT which have pop and push operation for deleting and inserting element.

DATA STRUCTURE OPERATIONS

- **Traversing** :- Accessing each record so that some item in the record may be processed.
- **Searching** :- Finding the location of the record with the given key value
- **Inserting** :- Adding new record to the data structure
- **Deleting** :- Removing record from the data structure
- **Sorting** :- Arranging the record in some logical order
- **Merging** :- Combining the record of different sorted file into single sorted file.

Example:

- An array A having n element is ADT which have following operations
  - Create(A) :- create an array A
  - Insert(A,x) :- insert an element x into an array A in any location
  - Delete(A,x) :- remove element x

DATA STRUCTURE OPERATIONS

- Traverse(A):- access each element of an array A
- Modify(A,x,y) :- Change the old element x with new element y
- Search(A,x) :- search element x in an array A
- Merge (A,B,C) :- Combine the elements of array B and C and store in new array A

Your Queries Please!!!

THANK
YOU