

Data Structure & Algorithm

Course: Data Structure and Algorithm (3-0-1)

Course Code: CMP 224.3

Lecture Hour : 48 hrs.

Methods of teaching: lecture, presentation demo and lab.

Evaluation: 50+50(External written exam and internal lab(20) and exam, CP, DR, Assessment (30))



UNIT ONE: INTRODUCTION

- In this unit we will discuss about following topics
- 1. Introduction of DSA,
- 2. Abstract Data Types(ADTs) &its scope,
- 3. Data structure and its types,
- 4. Brief introduction to Recursion



1. Introduction of DSA

Data structure is representation of the logical relationship existing between individual elements of data.

In other words “a data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other”

Data structure mainly specify following four things, they are



Cont..

1. Organization of data.
 2. Accessing methods
 3. Degree of associativity
 4. Processing alternatives for information
-

Data structure are the building block of the program and hence selection of particular data structure stresses on the following two things.

1. The data structure must be rich enough in structure to reflect the relationship existing between the data



Cont..

2. And the structure should be simple so that we can process the data effectively wherever required.

The data structure affects the design of both structure and functional aspect of a program.

Algorithm + Data Structure = Program

Algorithm and the best design of data structure yields a good program



Algorithm

An algorithm is a set of rules for carrying out calculation either by hand or on machine.

An algorithm is a sequence of computational steps that transform input into output

Algorithm is a step by step procedure to solve a particular problems/function. That is it is the set of instruction written to carry out certain task. Algorithms are mainly two types:



Cont..

1. Iterative (Repetitive): use loop and condition statements

2. Recursive: use divide and conquer method.

Algorithm must have the following properties

- Input: An algorithm must receive some input data supplied externally.
- Output: An algorithm must produce at least one output as the result



Cont..

- Finiteness: the algorithm must terminate after a finite number of steps.
- Definiteness: each step of the algorithm must be clear.
- Effectiveness: one must be able to perform the steps in the algorithm without applying any intelligence.



Types of Data Structure

There are mainly two types of data structure they are as following

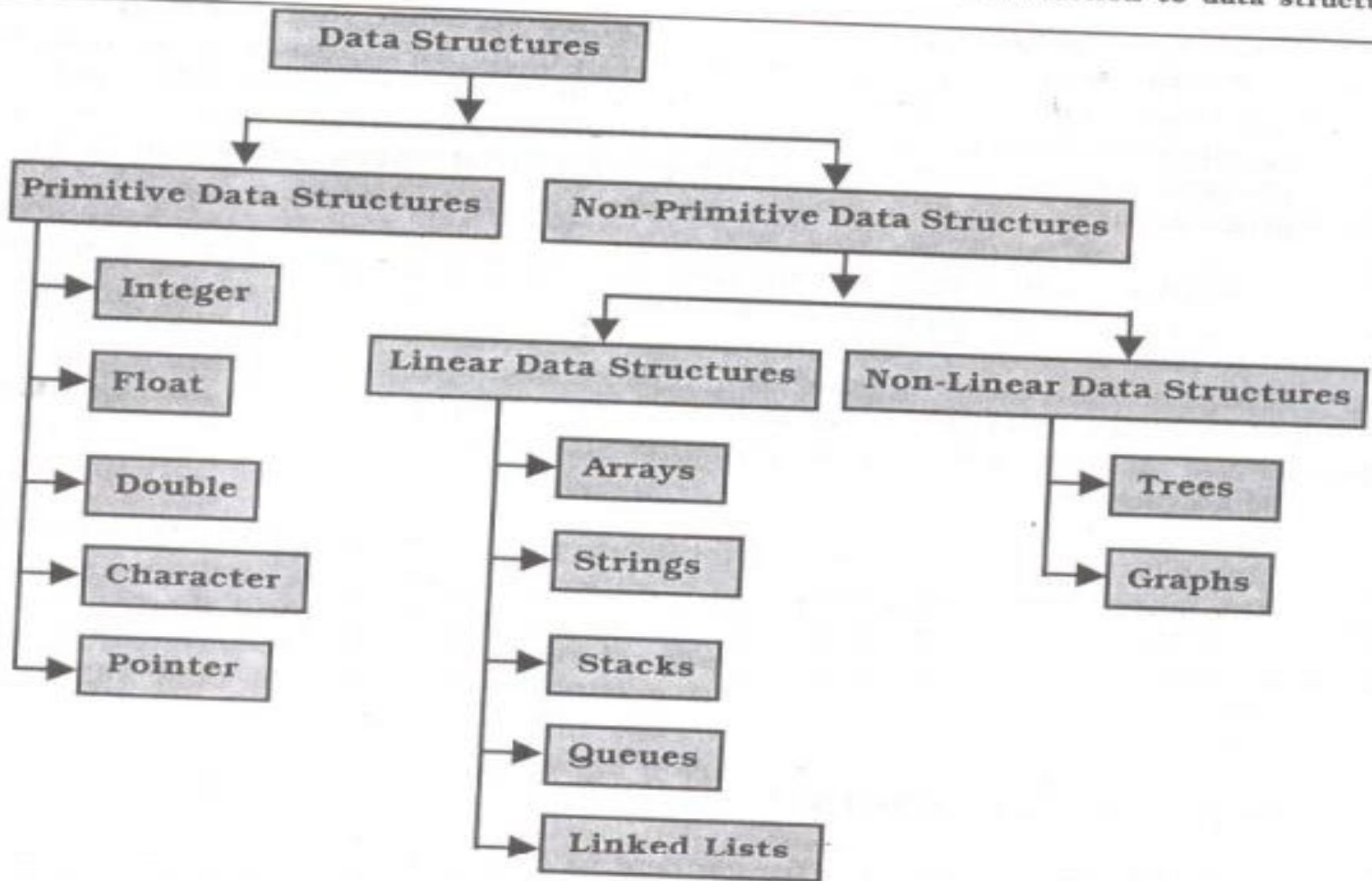
1. Primitive data structure
 2. Non primitive data structure
- **Primitive data structure:** these are basic structure and are directly operated upon by the machine instructions. These in general, have different representation on different computer. int float, char, string and pointer fall in this category.



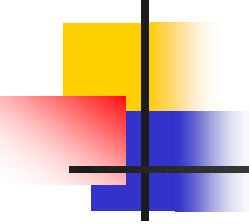
Cont..

- Non-primitive data structure: these are more sophisticated data structures, they are derived from the primitive data structure. Non-primitive data structure emphasize on structuring of a group of homogeneous or heterogeneous data items. Array lists, and files are examples.

In non-primitive data types we can perform the sets of operations on it so they are also called the ADT's (abstract data types). The most common used operation on data structure are *create, delete, insert, update and search*.



Brief introduction to Recursion



Recursion is defined as anything in terms of itself. Recursion is used to solve problems involving iterations (multiple execution), in reverse order. Recursion is an alternative to iteration in making a function execution repeatedly

Recurrence from the Latin, re= back+ currernce=to run to happen again, especially at repeated intervals.

Cont..

The technique is useful both for the definition of mathematical functions and for definition of data structures.

Data structures also may be recursively defined.

One of the most important classes of structure-tree allows recursive definition which lead to simple recursive function for manipulating them. **Recursion:** when a method call itself

- Classic example: factorial function:
- $N! = 1.2.3.4.....(n-1).n$
- Recursive definition:
- $f(n) = 1$ and if $n = 0$ else
- $f(n) = n * f(n-1)$



Disadvantages of recursion

- It consumes more storage space because the recursive calls along with automatic variables are stored on the stack.
- the computer may run out of memory if recursive calls are not checked.
- It is not more efficient in terms of speed and execution time.
- Recursion does not offer any concrete advantages over other methods



Basic requirement/rules for recursion

For the implementing and design the good recursive program we must make certain assumptions as follows:

- **Base case:** we must always have some base cases which can be solved with recursion.
- **Make progress:** The recursive call must always be to a case that make progress towards a base case.
- **Design rule :** Assume that all the recursive calls works.
- **Compound interest rule:** Never duplicate works by solving the same instance of a problem in separate recursive calls.



TOH problem (Tower of Hanoi)

In this game there are three different sized of disks. Each disk has a hole in the center. So that it can be stacked on any of the pegs. Call three pegs as x, y, and z. At the beginning of the game, the disks are stacked on the x pegs, that is the largest sized disk on the bottom and the smallest sized disk on top for n-3 disks



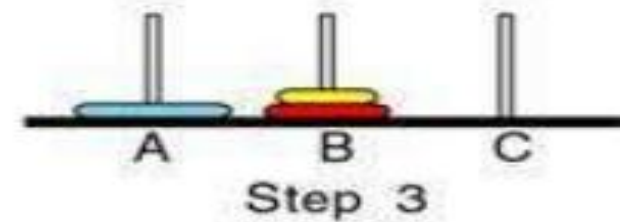
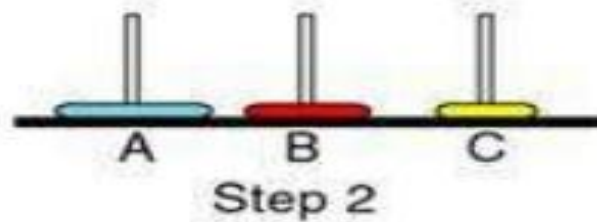
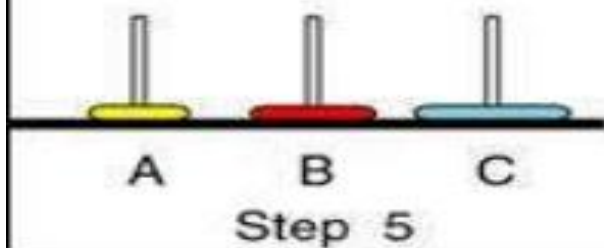
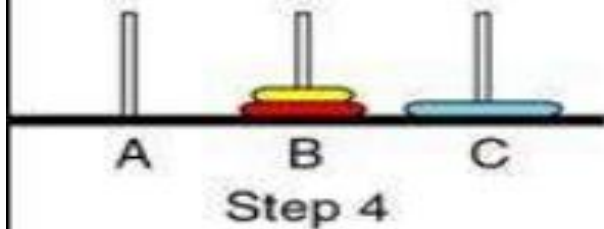
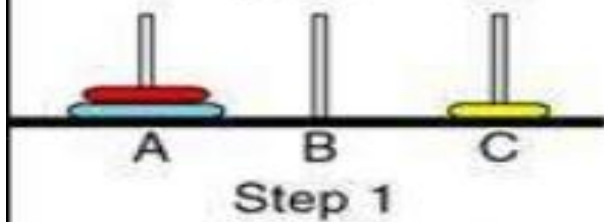
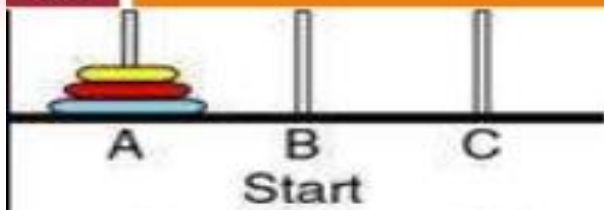
Rules of the games

1. Transferring the disks from the source peg to the destination peg such that at any points of transformation no larger size disk is placed on the smaller one.
2. Only one disk may be moved at a time.
3. Each disk must be stacked on any one of the pegs.

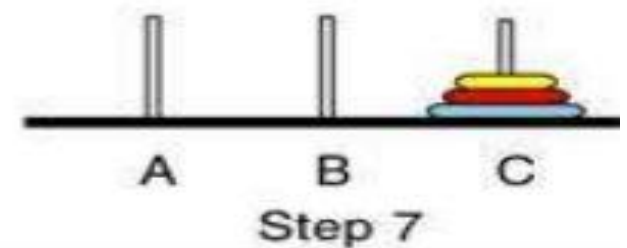
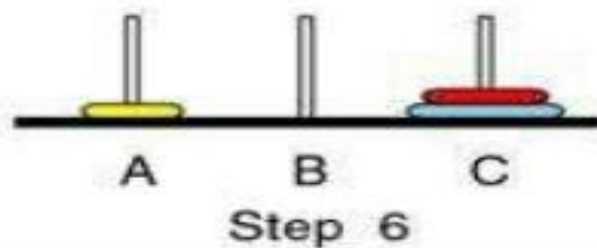
The process is as shown below.

Towers of Hanoi

23



Move one disk from source to destination.





So the general solution is

- If $n == 1$ move the single disk from S to D and stop.
- Move the top $n-1$ disks from S to T, using D as auxiliary.
- Move the remaining disk from S to D.
- Move the $n-1$ disks from T to D, using S as auxiliary.



Function for TOH.

- tofh(int ndisk,char source,char temp, char dest)
- {
- if(ndisk>0)
- {
- tofh(ndisk-1,source,dest,temp);
- printf("move disk %d %c->%c\n",ndisk,source,dest);
- tofh(ndisk-1,temp,source,dest);
- }
- }

Assessments



1. Write a program using recursion function to print the factorial of given number.
2. Write a program using recursion function to display fibonnic series up to 50
3. Write a program using recursion to print the sum of natural number upto n.



THANK YOU