PDF-XChange Click to buy NOW! www.tracker-software.com

1. Introduction:

1.1. Data Structures Definition

Data structure is the structural representation of logical relationships between elements of data. In other words a data structure is a way of organizing data items by considering its relationship to each other.
Data structure mainly specifies the structured organization of data, by providing accessing methods with correct degree of associativity. Data structure affects the design of both the structural and functional aspects of a program.
Algorithm + Data Structure = Program
Data structures are the building blocks of a program; here the selection of a particular data structure will help the programmer to design more efficient programs as the complexity and volume of the problems solved by the computer is steadily increasing day by day. The programmers have to strive hard to solve these problems. If the problem is analyzed and divided into sub problems, the task will be much easier *i.e.*, divide, conquer and combine.
A complex problem usually cannot be divided and programmed by set of modules unless its solution is structured or organized. This is because when we divide the big problems into sub problems, these sub problems will be programmed by different programmers or group of programmers. But all the programmers should follow a standard structural method so as to make easy and efficient integration of these modules. Such type of hierarchical structuring of program modules and sub modules should not only reduce the complexity and control the flow of program statements but also promote the proper structuring of information. By choosing a particular structure (or data structure) for the data items, certain data items become friends while others loses its relations.

1.2. Types of Data Structures

Data types vs. Data Structures
A data *type* is a well-defined collection of data with a well-defined set of operations on it.
Data types
Primitive types

- Boolean
- Character
- Integer
- String
- Double
- Float

Data structures
There are two types of data structure Linear and Nonlinear
Linear data structure:
In linear data structures, values are arranged in linear fashion. Arrays, lists, stacks and queue are examples of linear data structures in which values are stored in a sequence.

- **Array**
  Arrays are most frequently used in programming. Mathematical problems like matrix, algebra and etc can be easily handled by arrays. An array is a collection of homogeneous data elements described by a single name. Each element of an array is referenced by a subscripted variable or value, called subscript or index enclosed in parenthesis. If an element of an array is referenced by single subscript, then the array is known as one dimensional array or linear array and if two subscripts are required to reference an element, the array is known as two dimensional array and so on. Analogously the arrays whose elements are referenced by two or more subscripts are called multi dimensional arrays.

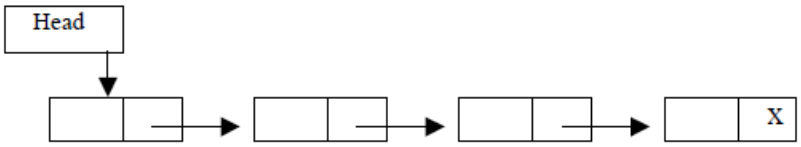
A[0]	A[1]	A[2]	.....	A[9]

1 dimensional array

A_{00}	A_{01}	A_{02}						A_{08}	A_{09}
A_{10}	A_{11}								A_{19}
A_{20}									
A_{30}									
									A_{69}
A_{70}								A_{78}	A_{79}
A_{80}	A_{81}						A_{87}	A_{88}	A_{89}
A_{90}	A_{91}	A_{92}				A_{96}	A_{97}	A_{98}	A_{99}

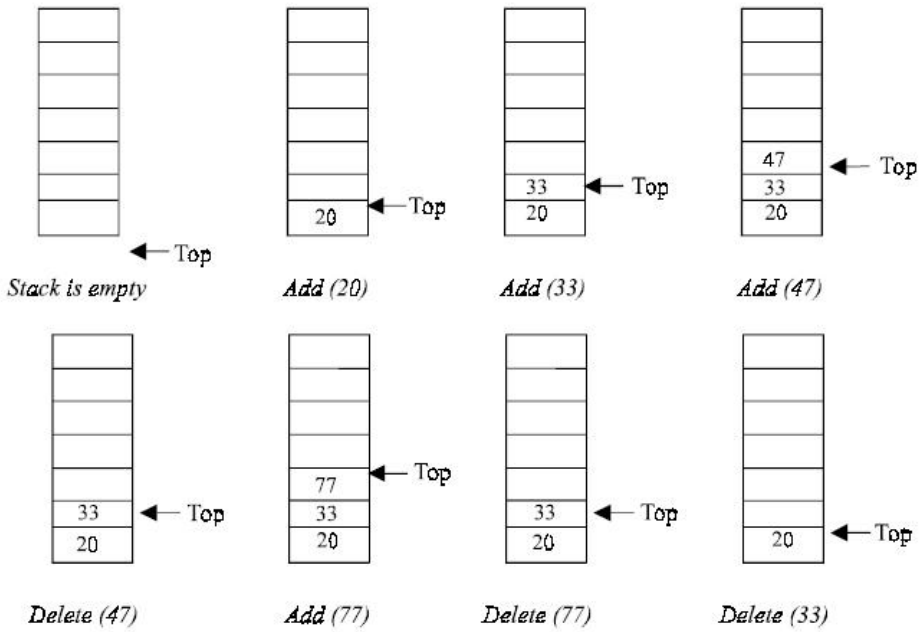
multidimensional array

- **List**
  A list is an ordered set consisting of a varying number of elements to which insertion and deletion can be made. A list represented by displaying the relationship between the adjacent elements is said to be a linear list. Any other list is said to be non linear. List can be implemented by using pointers. Each element is referred to as nodes; therefore a list can be defined as a collection of nodes as shown below :



a list

- **Stack**
  A **stack** is one of the most important and useful non-primitive linear data structure in computer science. It is an ordered collection of items into which new data items may be added/inserted and from which items may be deleted at only one end, called the *top* of the stack. As all the addition and deletion in a stack is done from the top of the stack, the last added element will be first removed from the stack. That is why the stack is also called ***Last-in-First-out (LIFO)*** .Note that the most frequently accessible element in the stack is the top most elements, whereas the least accessible element is the bottom of the stack. The operation of the stack can be illustrated as in figure below:



The insertion (or addition) operation is referred to as *push*, and the deletion (or remove) operation as *pop*. A stack is said to be *empty* or *underflow*, if the stack contains no elements. At this point the top of the stack is present at the bottom of the stack. And it is *overflow* when the stack becomes full, *i.e.*, no other elements can be pushed onto the stack. At this point the top pointer is at the highest location of the stack.

- **Queue**

  A queue is logically a *first in first out (FIFO or first come first serve)* linear data structure.
  The concept of queue can be understood by our real life problems. For example a customer come and join in a queue to take the train ticket at the end (rear) and the ticket is issued from the front end of queue. That is, the customer who arrived first will receive the ticket first. It means the customers are serviced in the order in which they arrive at the service centre.
  It is a homogeneous collection of elements in which new elements are added at one end called *rear*, and the existing elements are deleted from other end called *front*.
  The basic operations that can be performed on queue are
  1. Insert (or add) an element to the queue (push)
  2. Delete (or remove) an element from a queue (pop)

Fig. 4.1. Queue is empty.

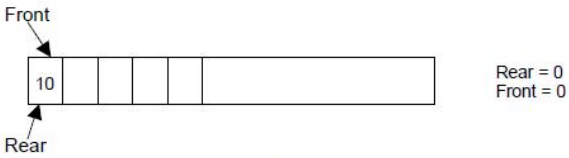


Fig. 4.2. push(10)

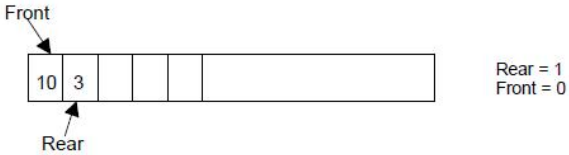


Fig. 4.3. push(3)

- **FILES**

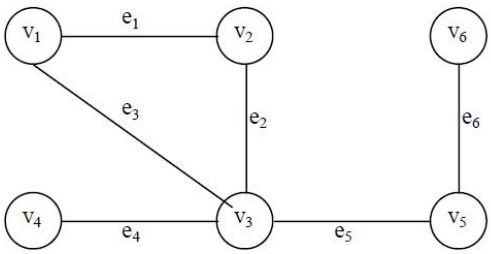
  A file is typically a large list that is stored in the external memory (*e.g.*, a magnetic disk) of a computer.

Nonlinear data structure

This type is opposite to linear. The data values in this structure are not arranged in order.

- **Graph**

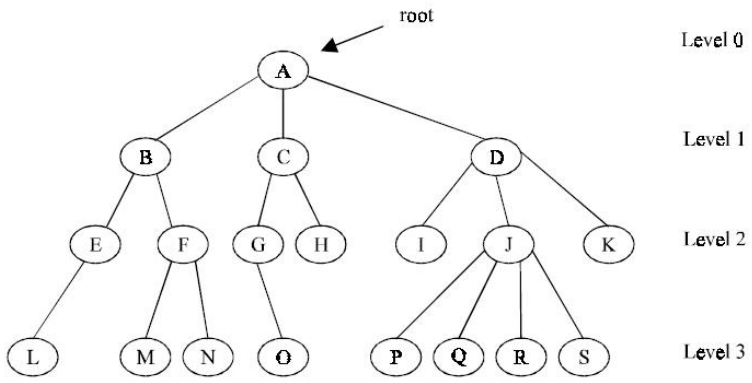
  This chapter discusses another nonlinear data structures, graphs. Graphs representations have found application in almost all subjects like geography, engineering and solving games and puzzles.
  A graph G consist of
  1. Set of vertices V (called nodes), (V = {v_1, v_2, v_3, v_4.....}) and
  2. Set of edges E (*i.e.*, E {e_1, e_2, e_3.....cm}
  A graph can be represents as G = (V, E), where V is a finite and non empty set at vertices and E is a set of pairs of vertices called edges. Each edge '*e*' in E is identified with a unique pair (*a*, *b*) of nodes in V, denoted by *e* = [*a*, *b*].



Consider a graph, G in Fig. 9.1. Then the vertex V and edge E can be represented as:
V = {v_1, v_2, v_3, v_4, v_5, v_6} and E = {e_1, e_2, e_3, e_4, e_5, e_6} E = {(v_1, v_2) (v_2, v_3) (v_1, v_3) (v_3, v_4), (v_3, v_5) (v_5, v_6)}. There are six edges and vertex in the graph

- **Tree**

  one of the important non-liner data structure in computer science, Trees. Many real life problems can be represented and solved using trees.
  Trees are very flexible, versatile and powerful non-liner data structure that can be used to represent data items possessing hierarchical relationship between the grand father
  and his children and grand children as so on.



A tree is an ideal data structure for representing hierarchical data. A tree can be theoretically defined as a finite set of one or more data items (or nodes) such that :
1. There is a special node called the root of the tree.
2. Removing nodes (or data item) are partitioned into number of mutually exclusive (i.e., disjoined) subsets each of which is itself a tree, are called sub tree.

1.3. Selection of Data

There are many considerations to be taken into account when choosing the best data structure for a specific program”

- Size of data.
- Speed and manner data use.
- Data dynamics, as change and edit.
- Size of required storage.
- Fetch time of any information from data structure.