THE QUAIDE MILLETH COLLEGE FOR MEN

Medavakkam, Chennai-600100.

B.Com(ISM)

Data Structure
MBT1A

Question Bank and Answer Key

UNIT – I

Part - A

1. Define data structure.

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

2. What is a data type?

Data type is way to classify various types of data such as integer, string etc. which determines the values that can be used with the corresponding type of data, the type of operations that can be performed on the corresponding type of data.

3. What do you mean by primitive data?

Primitive data types are predefined types of data, which are supported by the programming language. For example, integer, character, and string are all primitive data types.

4. What is composite data type? Give an example.

Composite data types are not defined by the programming language, but are instead created by the programmer. Composite types are derived from more than one primitive type. This can be done in a number of ways. For example, arrays, stack, queue, tree, graph etc.

5. What is an array?

Array is a data structure used to store homogeneous elements at contiguous locations. Size of an array must be provided before storing data. An array is a number of elements in a specific order, typically all of the same type.

6. Differentiate between list and array.

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.

In array, each element is independent, no connection with previous element or with its location. In Linked list, location or address of elements is stored in the link part of previous element/node.

Arrays allow both; direct and sequential access, while lists allow only sequential access.

7. What are the operations performed on arrays?

- Insertion
- Deletion
- Traversing
- Searching
- Sorting
- Update

8. Define graph.

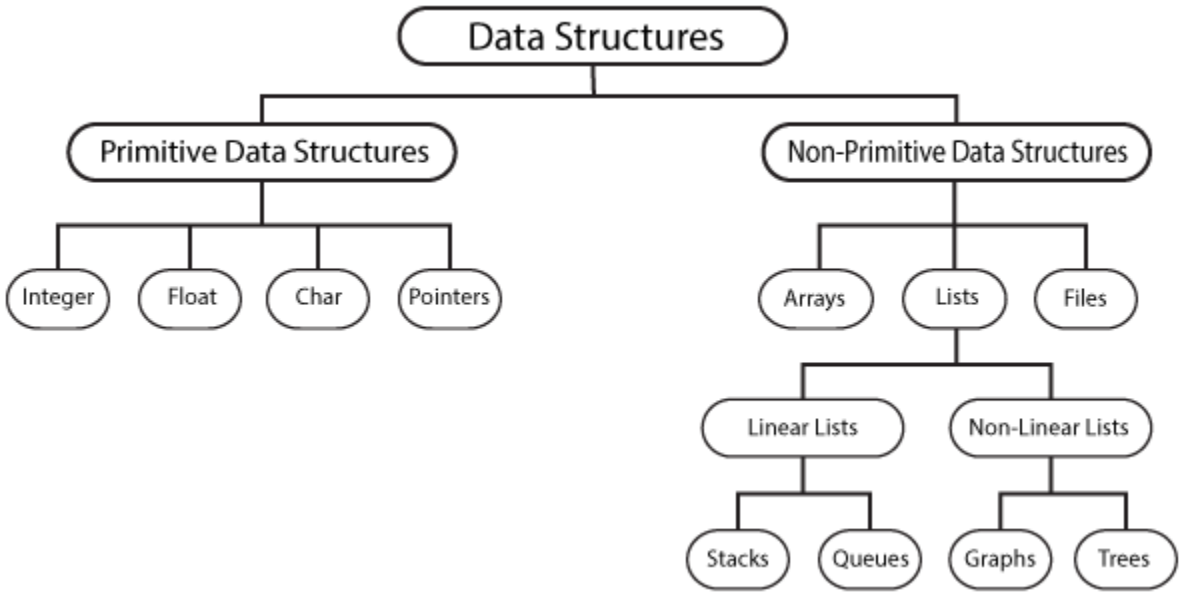
A graph data structure consists of a finite (and possibly mutable) set of vertices or nodes or points, together with a set of unordered pairs of these vertices for an undirected graph or a set of ordered pairs for a directed graph.

Part - B

1. Describe primitive and composite data types.

Types of Data Structures

- Primitive data structures
- Non-primitive data structures



Primitive Data Structures

Primitive Data Structures are the basic data structures that directly operate upon the machine instructions. They have different representations on different computers.

Integers, Floating point numbers, Character constants, String constants and Pointers come under this category.

Composite (Non-primitive) Data Structures

Composite data structures are more complicated data structures and are derived from primitive data structures. They emphasize on grouping same or different data items with relationship between each data item.

Arrays, Lists and Files come under this category.

2. Explain two dimensional arrays.

A two-dimensional array contains two subscripts. A two-dimensional array can be considered as a table made up of rows and columns and can be declared as x[i][j].

The general form of declaring the two-dimensional array is

storage-class datatype array[size1][size2];

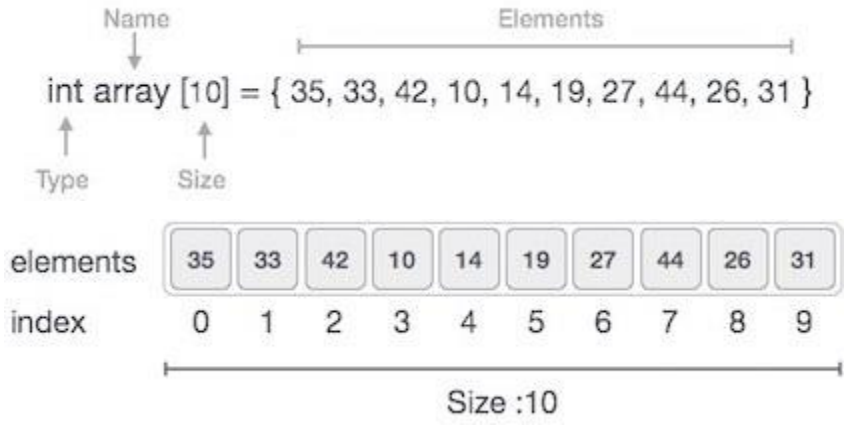
Two-dimensional array can be initialized with value when declaring the array.

Example :

The value set to the initialized value with refers to the individual element.

3. Explain the operations possible on arrays.

Array is a container which can hold a fix number of items and these items should be of the same type.



Following are the basic operations supported by an array.

- Insertion − Adds an element at the given index.
- Deletion − Deletes an element at the given index.
- Traverse − Print all the array elements one by one.
- Search − Searches an element using the given index or by the value.
- Sorting – Sort the element of the array in ascending or descending order.
- Update − Updates an element at the given index.

4. Explain the various asymptotic notations with examples.

Asymptotic analysis of an algorithm refers to defining the mathematical boundation / framing of its run-time performance. Asymptotic analysis is input bound i.e., if there's no input to the algorithm, it is concluded to work in a constant time.

Asymptotic analysis refers to computing the running time of any operation in mathematical units of computation. For example, the running time of one operation is computed as f(n) and may be for another operation it is computed as $g(n^2)$.

Following are the commonly used asymptotic notations to calculate the running time complexity of an algorithm.

- O Notation
- Ω Notation
- θ Notation

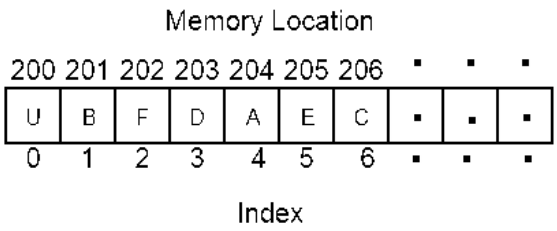
Part - C

1. Explain the features and uses of Array.

Array is a data structure used to store homogeneous elements at contiguous locations. Size of an array must be provided before storing data.

An array is collection of items stored at continuous memory locations. The idea is to store multiple items of same type together. This makes it easier to calculate the position of each element, i.e., the memory location of the first element of the array (generally denoted by the name of the array).

Elements are accessed using an integer index to specify which element is required Typical implementations allocate contiguous memory words for the elements of arrays. Arrays may be fixed-length or resizable.



Types of arrays:

→ One-dimensional array : The first element of the array is indexed by subscript of 0.

→ Two-dimensional array : The first element of the array is indexed by subscript of 1.

→ Multi-dimensional array : The base index of an array can be freely chosen. Usually programming languages allowing n-based indexing and other scalar data types like enumerations, or characters may be used as an array index.

Arrays can be used to store linear data of similar types, but arrays have following limitations.

1) The size of the arrays is fixed. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.
2) Inserting a new element in an array of elements is expensive, because space has to be created for the new elements and to create space existing elements have to shifted.

For example, in a system if we maintain a sorted list in an array a[ ].

a[ ] = {1000, 1010, 1050, 2000, 2040}.

UNIT – II

Part - A

1. What is a stack?

Stack is a linear data structure which follows a particular order in which the operations are performed with LIFO (Last In First Out) order.

2. Write the operations performed on stack.

- Push: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- Pop: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.

3. Give any two applications of stack.

1. Recursion
2. Infix to Postfix expression conversion

4. Write the applications of queue.

Queue is used to process in First In First Out order.

This property of Queue makes useful in the following kind of applications.

1) When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.
2) When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

5. Write the postfix form of the expression.

X – Y * Z / A.

Postfix form : XYZ * A / –

6. Write the postfix form of the expression.

a * b + c + d * e * f + g.

Postfix form : ab * c + de * f * + g +

7. Write the postfix forms for the expression.

A + B * (C – D) / (P – R).

Postfix form : ABCD – * PR – / +

8. What is recursion?

Recursion is the type of function which calls by itself. When there is the logical situation appears, the function returns to main function. A condition should be provided within the recursive function.

9. Define queue.

Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO).

A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.

Operations on Queue:

Front: Remove the front item from queue.

Rear: Insert the item in the end of queue.

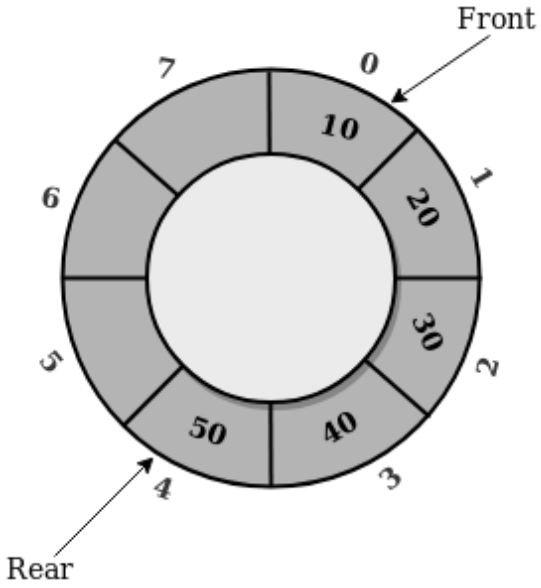
10. Distinguish between a stack and a queue.

The difference between stacks and queues is in removing. In a stack, we remove the item the most recently added. In a queue, we remove the item the least recently added.

Stack is a collection of objects that works in LIFO (Last in First out) mechanism while Queue is FIFO (First in First out).

11. What do you mean by circular queue?

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called 'Ring Buffer'**.**

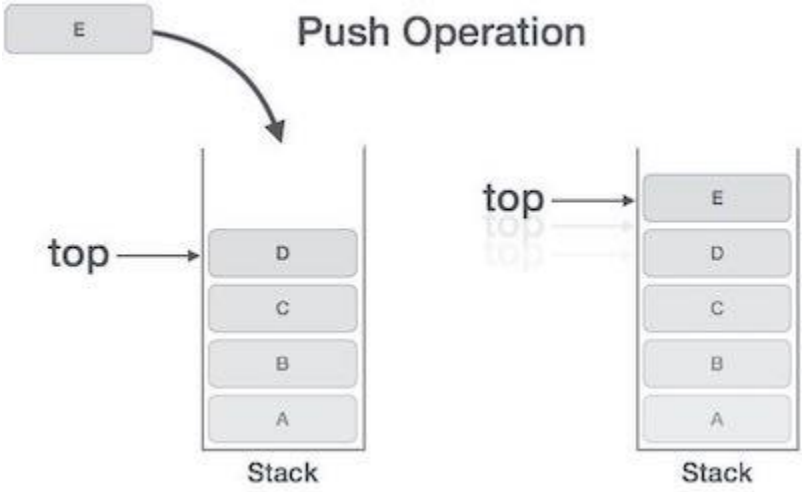


Part - B

1. Discuss the algorithm for adding an element to the stack.

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps.

- Step 1 − Checks if the stack is full.
- Step 2 − If the stack is full, produces an error and exit.
- Step 3 − If the stack is not full, increments top to point next empty space.
- Step 4 − Adds data element to the stack location, where top is pointing.
- Step 5 − Returns success.



Algorithm :

```
begin procedure push: stack, data
   if stack is full
      return null
   endif
   top ← top + 1
   stack[top] ← data
end procedure
```

2. **How is an infix expression converted to postfix expression?**

   Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands.

   Postfix expression: The expression of the form a b op. When an operator is followed for every pair of operands.

   Algorithm :

   Example :

3. Describe an algorithm for producing postfix from infix expression.

The postfix form is abc*+d+. The postfix expressions can be evaluated easily using a stack.

Algorithm :

1. Scan the infix expression from left to right.
2. If the scanned character is an operand, output it.
3. Else,
    a) If the precedence of the scanned operator is greater than the precedence of the operator in the stack (or the stack is empty), push it.
    b) Else, Pop the operator from the stack until the precedence of the scanned operator is less-equal to the precedence of the operator residing on the top of the stack. Push the scanned operator to the stack.
4. If the scanned character is an '(', push it to the stack.
5. If the scanned character is an ')', pop and output from the stack until an '(' is encountered.
6. Repeat steps 2-6 until infix expression is scanned.
7. Pop and output from the stack until it is not empty.

Example :

4. Write an algorithm to insert an element in a queue.

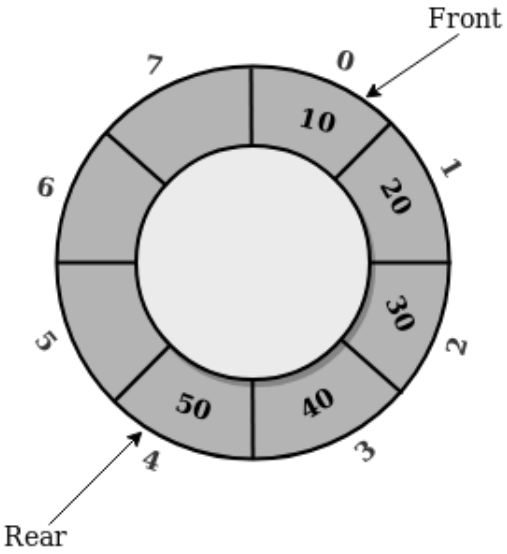
A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle. An excellent example of a queue is a line of students in the food court.

Example :

Algorithm :

5. Briefly explain circular queue.

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called 'Ring Buffer'.



Operations on Circular Queue:

- Front : Get the front item from queue.
- Rear : Get the last item from queue.
- enQueue(value) : This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at Rear position.
- deQueue() : This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from front position.

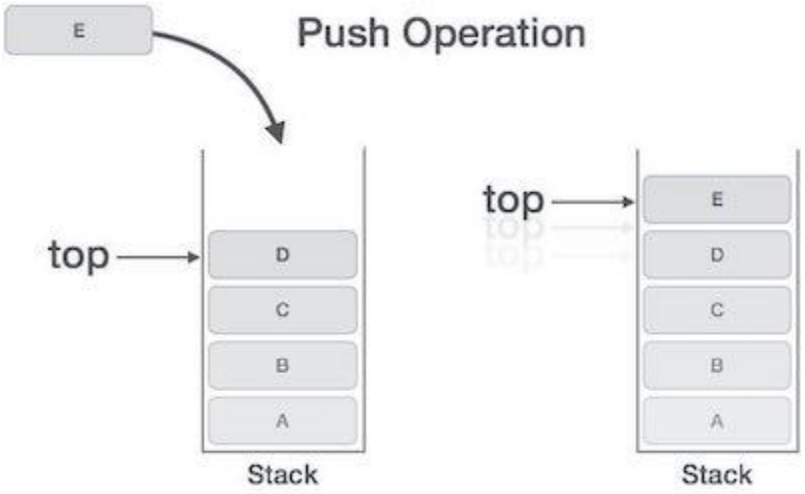
Part - C

1. **Describe the various stack operations.**

Push operation :

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps.

- Step 1 − Checks if the stack is full.
- Step 2 − If the stack is full, produces an error and exit.
- Step 3 − If the stack is not full, increments top to point next empty space.
- Step 4 − Adds data element to the stack location, where top is pointing.
- Step 5 − Returns success.



Algorithm for Push Operation :

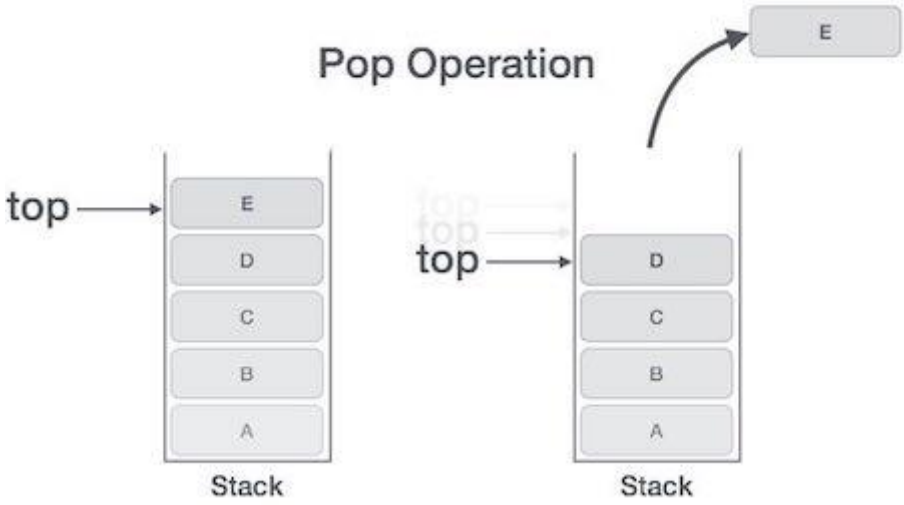
```
begin procedure push: stack, data
   if stack is full
      return null
   endif
   top ← top + 1
   stack[top] ← data
end procedure
```

Pop operation :

Accessing the content while removing it from the stack, is known as a Pop Operation. Top is decremented to a lower position in the stack to point to the next value.

A Pop operation may involve the following steps.

- Step 1 − Checks if the stack is empty.
- Step 2 − If the stack is empty, produces an error and exit.
- Step 3 − If the stack is not empty, accesses the data element at which top is pointing.
- Step 4 − Decreases the value of top by 1.
- Step 5 − Returns success.



Algorithm for Pop Operation

```
begin procedure pop: stack
   if stack is empty
      return null
   endif
   data ← stack[top]
   top ← top - 1
   return data
end procedure
```

2. Briefly explain the applications of stack.

1) Check for balanced parentheses
2) Syntax checker
3) Infix to Postfix
4) Recursion
5) Towers of Hanoi

3. Explain with an example how an infix expression is converted into postfix expression.

Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands.

Postfix expression: The expression of the form a b op. When an operator is followed for every pair of operands.

The postfix form is abc*+d+. The postfix expressions can be evaluated easily using a stack.

Algorithm :

1. Scan the infix expression from left to right.
2. If the scanned character is an operand, output it.
3. Else,
    a) If the precedence of the scanned operator is greater than the precedence of the operator in the stack (or the stack is empty), push it.
    b) Else, Pop the operator from the stack until the precedence of the scanned operator is less-equal to the precedence of the operator residing on the top of the stack. Push the scanned operator to the stack.
4. If the scanned character is an '(', push it to the stack.
5. If the scanned character is an ')', pop and output from the stack until an '(' is encountered.
6. Repeat steps 2-6 until infix expression is scanned.
7. Pop and output from the stack until it is not empty.

Example :

4. Explain various operations available on a queue.

Mainly the following four basic operations are performed on queue:

Enqueue: Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.

Dequeue: Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.

Front: Get the front item from queue.

Rear: Get the last item from queue.

5. Explain the important operations on stacks and queues.

Stack :

a) Push

b) Pop

c) Top

Queue :

a) Enqueue

b) Dequeue

c) Front

d) Rear

UNIT – III

Part - A

1. **Define linked list.**
2. **Define list.**

A linked list is a linear data structure (like arrays) where each element is a separate object. Each element (that is node) of a list is comprising of two items – the data and a reference to the next node.

A linked list (also just called list) is a linear collection of data elements of any type, called nodes, where each node has itself a value, and points to the next node in the linked list.

3. **State the types of linked list.**

i) Singly Linked List
ii) Doubly Linked List
iii) Circular Linked List

4. **Name the operations performed on linked list.**

A node can be added in three ways:

1) At the front of the linked list
2) In between the list
3) At the end of the linked list.

5. **What is doubly linked list?**

In this type of Linked list, there are two references associated with each node, One of the reference points to the next node and one to the previous node.

Eg. NULL ← 1 ↔ 2 ↔ 3 → NULL.

Doubly Linked List contains a link element called first and last. Each link carries a data field(s) and two link fields called next and prev.

Part - B

1. Explain the important operations on linked lists.

A node can be added in three ways:

1) At the front of the linked list
2) In between the list
3) At the end of the linked list.

2. What is singly linked list? Explain.

A linked list is a linear data structure (like arrays) where each element is a separate object. Each element (that is node) of a list is comprising of two items – the data and a reference to the next node.

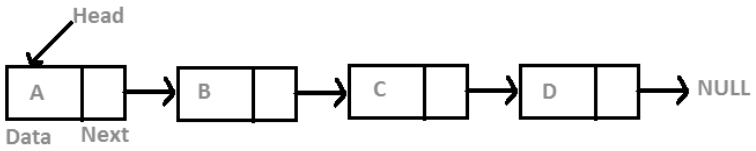
A linked list (also just called list) is a linear collection of data elements of any type, called nodes, where each node has itself a value, and points to the next node in the linked list.

A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL. Each node in a list consists of at least two parts:

1) Data
2) Pointer to the next node.

In this type of linked list, every node stores address or reference of next node in list and the last node has next address or reference as NULL.

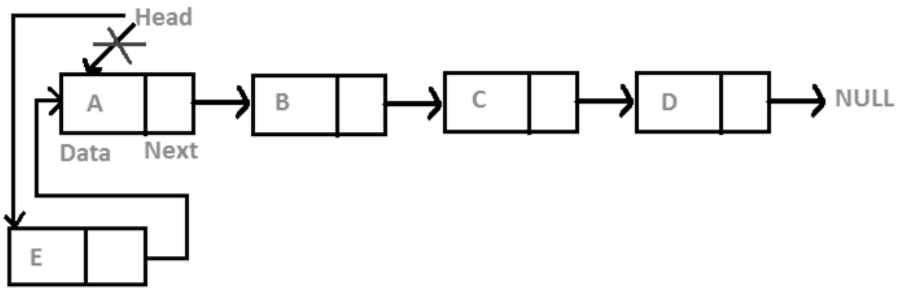
For example 1 → 2 → 3 → 4 → NULL



3. Explain the procedure to insert a node in a singly linked list.

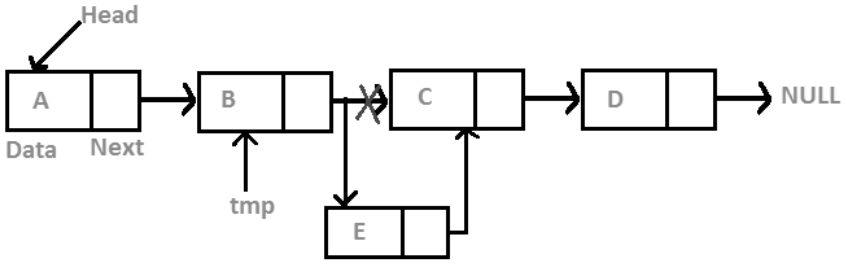
i. At the front of the linked list

The new node is always added before the head of the given Linked List. And newly added node becomes the new head of the Linked List. For example if the given Linked List is 10 → 15 → 20 → 25 and we add an item 5 at the front, then the Linked List becomes 5 → 10 → 15 → 20 → 25. Let us call the function that adds at the front of the list is push(). The push() must receive a pointer to the head pointer, because push must change the head pointer to point to the new node



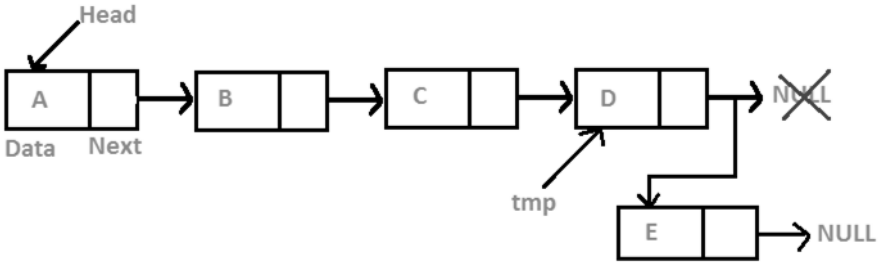
ii. In between the list

The new node is inserted after the given node.



iii. At the end of the linked list

The new node is always added after the last node of the given Linked List. For example if the given Linked List is 5 → 10 → 15 → 20 → 25 and we add an item 30 at the end, then the Linked List becomes 5 → 10 → 15 → 20 → 25 → 30.



4. Write a note on doubly linked list.

In this type of Linked list, there are two references associated with each node, One of the reference points to the next node and one to the previous node. Advantage of this data structure is that we can traverse in both the directions and for deletion we don't need to have explicit access to previous node.

Eg. NULL<-1<->2<->3->NULL.

Diagram:

Part - C

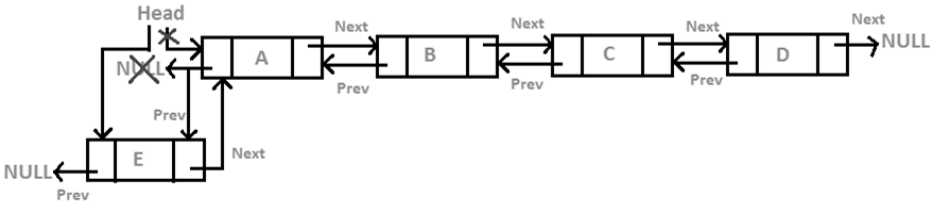
1. Explain about doubly linked list with insertion and deletion algorithms.

Doubly Linked List contains a link element called first and last. Each link carries a data field(s) and two link fields called next and previous. Each link is linked with its next link using its next link. Each link is linked with its previous link using its previous link.

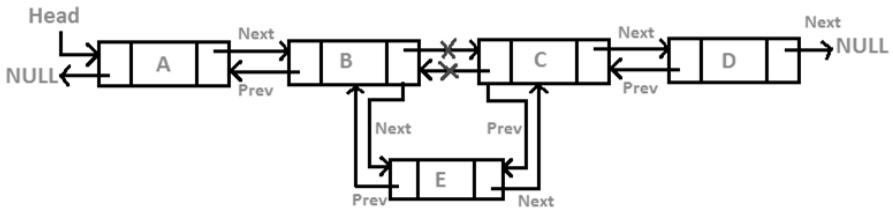
Insertion :

A node can be added in three ways :

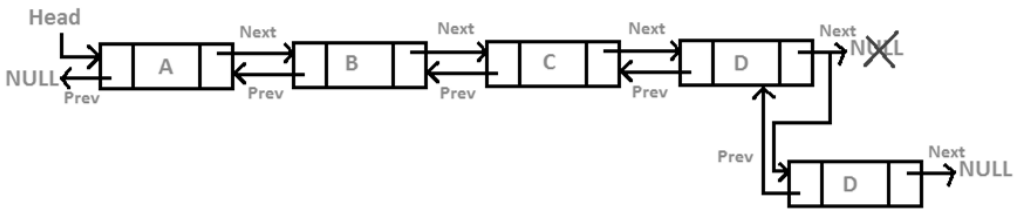
1) At the front of the DLL



2) In between the nodes



3) At the end of the DLL



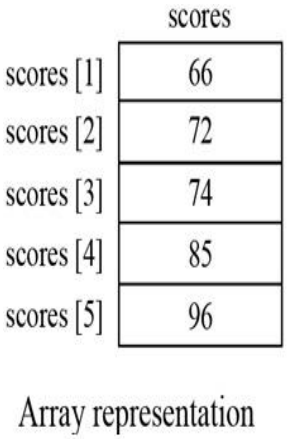
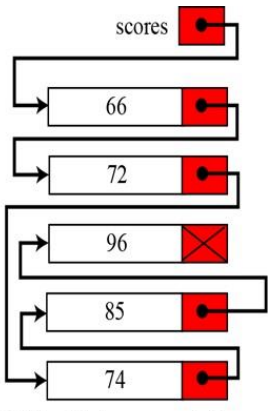
Deletion :

A node can be **deleted** in three ways :

1) At the front of the DLL
2) In between the nodes
3) At the end of the DLL

2. Compare arrays and linked lists.

Compare	**Array**	**Linked List**
Define	Array is a collection of elements having same data type with common name.	Linked list is an ordered collection of elements which are connected by links/pointers.
Access	In array, elements can be accessed using index/subscript value, i.e. elements can be randomly accessed like arr[0], arr[3], etc. So array provides fast and random access.	In linked list, elements can't be accessed randomly but can be accessed only sequentially and accessing element takes 0(n) time.
Memory Structure	In array, elements are stored in consecutive manner in memory.	In linked list, elements can be stored at any available place as address of node is stored in previous node.
Insertion &	Insertion & deletion takes more time in array as elements are	Insertion & deletion are fast & easy in linked list as only value of

Deletion	stored in consecutive memory locations.	pointer is needed to change.
Memory Allocation	In array, memory is allocated at compile time i.e. Static Memory Allocation.	In linked list, memory is allocated at run time i.e. Dynamic Memory Allocation.
Types	Array can be single dimensional, two dimension or multidimensional.	Linked list can be singly, doubly or circular linke d list.
Dependency	In array, each element is independent, no connection with previous element or with its location.	In Linked list, location or address of elements is stored in the link part of previous element/node.
Extra Space Figure	In array, no pointers are used like linked list so no need of extra space in memory for pointer.  Array representation	In linked list, adjacency between the elements are maintained using pointers or links, so pointers are used and for that extra memory space is needed.  Linked list representation

UNIT – IV

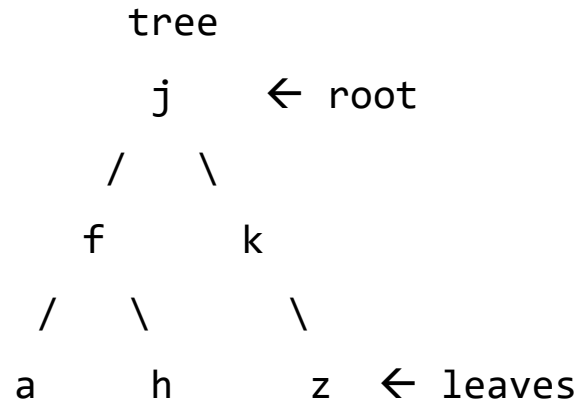
Part - A

1. What is a tree?

Trees are hierarchical data structures. It is one of the non-linear data structures. The topmost node is called root of the tree. The elements that are directly under an element are called its children. The element directly above something is called its parent.

2. Define binary tree.

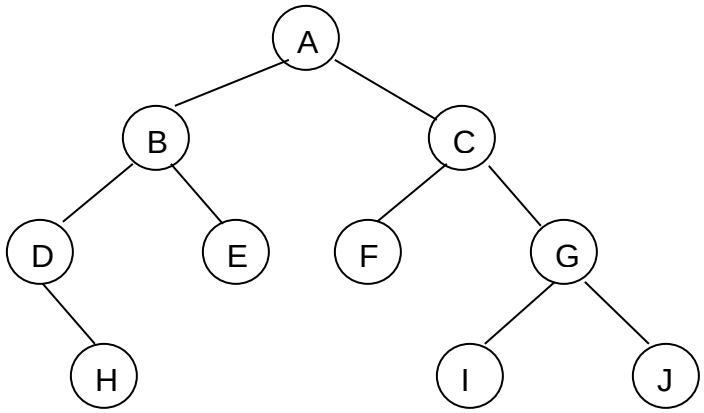
A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.



3. What is meant by traversal?

Tree traversal (also known as tree search) is a form of graph traversal and refers to the process of visiting (checking and/or updating) each node in a tree data structure, exactly once. Such traversals are classified by the order in which the nodes are visited.

4. Traverse the given tree using traversal types.



(a) Inorder – D, H, B, E, A, F, C, I, G, J.

(b) Preorder – A, B, D, H, E, C, F, G, I, J.

(c) Postorder – H, D, E, B, F, I, J, G, C, A.

5. List the three different types of tree traversals.

Trees can be traversed in different ways.

(a) Inorder (Left, Root, Right)

(b) Preorder (Root, Left, Right)

(c) Postorder (Left, Right, Root)

6. Write the procedure for preorder traversal of binary tree.

Algorithm Preorder(tree)

1. Visit the root.
2. Traverse the left subtree
3. Traverse the right subtree

7. What is a forest?

The forest-and-tree model is a logical structure for interconnecting multiple network domains. A tree is a set of domains sharing a common network. A forest consists of one or more trees that do not form a contiguous namespace.

Part - B

1. Explain inorder traversal of a binary tree.

Algorithm Inorder(tree)

1. Traverse the left subtree, i.e., call Inorder (left-subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder (right-subtree)

2. Explain the procedure of postorder traversal of binary tree.

Algorithm Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder (left-subtree)
2. Traverse the right subtree, i.e., call Postorder (right-subtree)
3. Visit the root.

Part - C

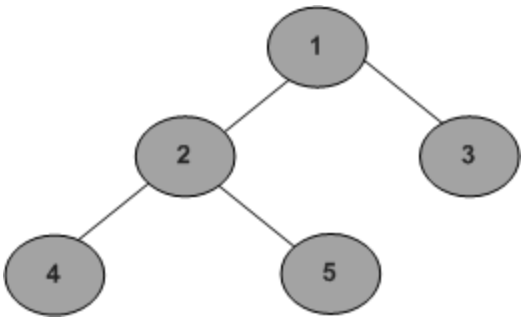
1. Discuss the various traversal techniques of a binary tree.

A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

A Tree node contains following parts:

1. Data (Root)
2. Pointer to left child
3. Pointer to right child.

Trees can be traversed in different ways.



(a) Inorder (Left, Root, Right) : 4 2 5 1 3

(b) Preorder (Root, Left, Right) : 1 2 4 5 3

(c) Postorder (Left, Right, Root) : 4 5 2 3 1

Algorithm :

UNIT – V

Part - A

1. What is a graph?

Graph is a data structure that consists of following two components:

a) A finite set of vertices also called as nodes.

b) A finite set of ordered pair of the form (u, v) called as edge.

2. What is a graph?

A graph is a structure consisting of a set of arrays (also called dimensions) and a set of edges. An edge is a pair of vertices. The two vertices are called the edge endpoints.

3. Name the types of graph.

a) Directed graph

b) Undirected graph

c) Weighted graph

4. What is directed graph?

The number of edges with one endpoint on a given vertex is called that vertex's degree. In a directed graph, the number of edges that point to a given vertex is called its in-degree, and the number that point from it is called its out-degree.

5. State the meaning of Depth first search.

In depth-first search, Start at vertex a, visit its neighbour b, then b's neighbour c and keep going until reach 'a dead end' then iterate back and visit nodes reachable from second last visited vertex and keep applying the same principle.

6. Narrate the meaning of Breadth first search.

Breadth first search visits the nodes neighbours and then the unvisited neighbours of the neighbours, etc. If it starts on vertex a it will go to all vertices that

have an edge from a. If some points are not reachable it will have to start another BFS from a new vertex.

7. Define hashing.

Hashing is the solution that can be used in almost all such situations and performs extremely well compared to above data structures like Array, Linked List etc. Hashing is an improvement over Direct Access Table.

8. What is hashing function?

A function that converts a given big input key to a small practical integer value. The mapped integer value is used as an index in hash table.

9. What is hash table?

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Part - B

1. Define graph. Explain briefly about graph.

Graph is a data structure that consists of following two components:

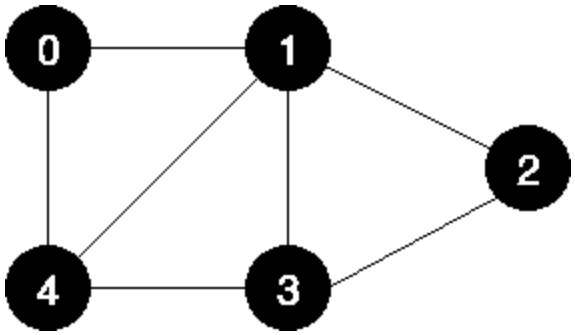
a) A finite set of vertices also called as nodes.

b) A finite set of ordered pair of the form (u, v) called as edge.

The pair is ordered because (u, v) is not same as (v, u) in case of directed graph(di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Graphs are used to represent many real life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, facebook.

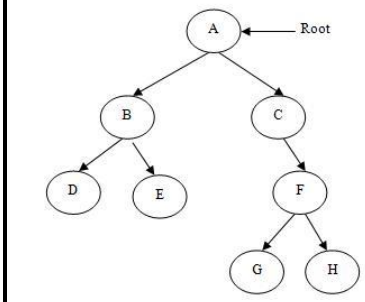
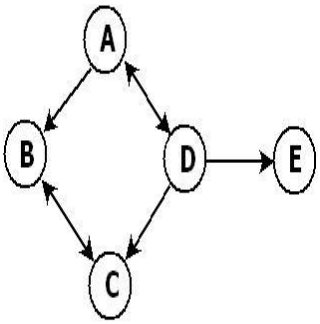
Following is an example undirected graph with 5 vertices.



2. Mention the differences between graph and tree.

Compare	Trees	Graphs
Path	Tree is special form of graph i.e. minimally connected graph and having only one path between any two vertices.	In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes
Loops	Tree is a special case of graph having no loops, no circuits and no self-loops.	Graph can have loops, circuits as well as can have self-loops.
Root Node	In tree there is exactly one root node and every child have only one parent.	In graph there is no such concept of root node.
Parent Child relationship	In trees, there is parent child relationship so flow can be there with direction top to bottom or vice versa.	In Graph there is no such parent child relationship.
Complexity	Trees are less complex then graphs as having no cycles, no self-loops and still connected.	Graphs are more complex in compare to trees as it can have cycles, loops etc

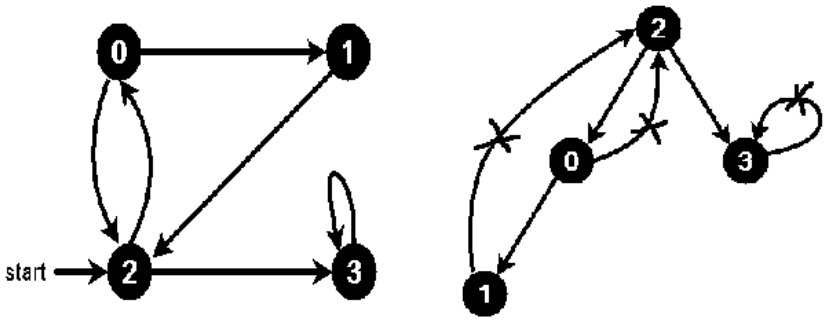
Types of Traversal	Tree traversal is a kind of special case of traversal of graph. Tree is traversed in Pre-Order, In-Order and Post-Order (all three in DFS or in BFS algorithm)	Graph is traversed by DFS: Depth First Search and in BFS: Breadth First Search algorithm
Connection Rules	In trees, there are many rules / restrictions for making connections between nodes through edges.	In graphs no such rules/ restrictions are there for connecting the nodes through edges.
DAG	Trees come in the category of DAG : Directed Acyclic Graphs is a kind of directed graph that have no cycles.	Graph can be Cyclic or Acyclic.
Different Types	Different types of trees are: Binary Tree , Binary Search Tree, AVL tree, Heaps.	There are mainly two types of Graphs : Directed and Undirected graphs.
Applications	Tree applications : sorting and searching like Tree Traversal & Binary Search.	Graph applications : Coloring of maps, in OR (PERT & CPM), algorithms, Graph coloring, job scheduling, etc.
No. of edges	Tree always has n-1 edges.	In Graph, no. of edges depends on the graph.
Model	Tree is a hierarchical model.	Graph is a network model.

Figure		

3. Explain depth-first search with an example.

Depth First Search for a graph is similar to Depth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array.

For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Depth First Traversal of the following graph is 2, 0, 1, 3.

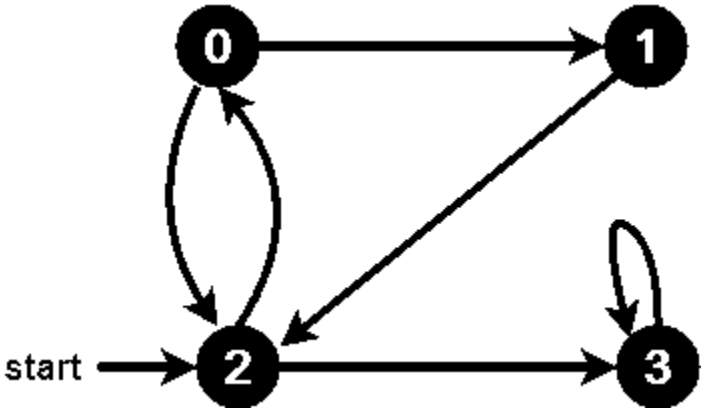


Algorithm :

4. Write an algorithm for breadth-first search on graph.

Breadth First Search for a graph is similar to Breadth First Traversal of a tree. Unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex.

For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Breadth First Traversal of the following graph is 2, 0, 3, 1.



Algorithm :

5. Analyse the features of Hashing function.

Hashing is a technique that is used to uniquely identify a specific object from a group of similar objects. Some examples of how hashing is used in our lives include:

- In universities, each student is assigned a unique roll number that can be used to retrieve information about them.
- In libraries, each book is assigned a unique number that can be used to determine information about the book, such as its exact position in the library or the users it has been issued to etc.

In both these examples the students and books were hashed to a unique number.

In hashing, large keys are converted into small keys by using hash functions. The values are then stored in a data structure called hash table. The idea of hashing is to distribute entries (key/value pairs) uniformly across an array. Each element is assigned a key (converted key). By using that key you can access the element in O(1) time. Using the key, the algorithm (hash function) computes an index that suggests where an entry can be found or inserted.

6. Describe any two popular hash functions.

1) Linear probing technique

In open addressing, instead of in linked lists, all entry records are stored in the array itself. When a new entry has to be inserted, the hash index of the hashed value is computed and then the array is examined (starting with the hashed index). If the slot at the hashed index is unoccupied, then the entry record is inserted in slot at the hashed index else it proceeds in some probe sequence until it finds an unoccupied slot.

The probe sequence is the sequence that is followed while traversing through entries. In different probe sequences, you can have different intervals between successive entry slots or probes.

2) Quadratic probing

Quadratic probing is similar to linear probing and the only difference is the interval between successive probes or entry slots. Here, when the slot at a hashed index for an entry record is already occupied, you must start traversing until you find an unoccupied slot. The interval between slots is computed by adding the successive value of an arbitrary polynomial in the original hashed index.

7. Explain the linear probing technique with an example.

The hashing technique is used to create an already used index of the array. In such a case, we can search the next empty location in the array by looking into the next cell until we find an empty cell. This technique is called linear probing.

Following are the basic primary operations of a hash table.

- Search − Searches an element in a hash table.
- Insert − inserts an element in a hash table.
- Delete − Deletes an element from a hash table.

Part - C

1. **Discuss the types of graphs with examples.**

Directed Graph :

The graph in which all the edges are unidirectional.

Undirected Graph :

The graph in which all the edges are bidirectional.

Weighted Graph :

The Graph in which weight is associated with the edges.

Unweighted Graph :

The Graph in which there is no weight associated to the edges.

Strongly Connected Components of a graph :

A directed graph is called strongly connected if there is a path from each vertex in the graph to every other vertex.

2. **Describe the algorithm of DFS and BFS.**

Depth-First Search

In depth-first search, we go down a path until we get to a dead end; then we backtrack or back up (by popping a stack) to get an alternative path.

- Create a stack
- Create a new choice point
- Push the choice point onto the stack
- while (not found and stack is not empty)
    - Pop the stack
    - Find all possible choices after the last one tried
    - Push these choices onto the stack
- Return

Breadth-First Search

In breadth-first search, we explore all the nearest possibilities by finding all possible successors and enqueue them to a queue.

- Create a queue
- Create a new choice point
- Enqueue the choice point onto the queue
- while (not found and queue is not empty)
    - Dequeue the queue
    - Find all possible choices after the last one tried
    - Enqueue these choices onto the queue
- Return

3. Discuss various hashing functions with examples.

A function that converts a given big input key to a small practical integer value. The mapped integer value is used as an index in hash table.

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.

Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values. Consider an example of hash table of size 20, and the following items are to be stored. Item are in the (key,value) format.