Trees and Graphs

14 May 2024 14:53

TREES

(i) What is a Tree in DSA?

-> Non-linear Data Structure
-> Used to represent hierarchial relationship among existing data items
-> There is a special node called as root of the Tree

(ii) Terminology-

(a.) Parent Node : A node that has a child is called a child's parent node
(b.) Internal Node / Inner Node : Any node of a tree that has child nodes
(c.) Non-terminal Nodes : Any node whose degree is not zero.
(d.) Leaf / Terminal Node : No child
(e.) Siblings : Node with same parents
(f.) Level
(g.) Edge : Line drawn from one node to another
(h.) Path : Sequence of consecutive edges from source to destination node
(i.) Forest : Set of disjoint trees (no root node)
(j.) Height of a Tree : No. of Levels
(k.) Degree of a Node : No. of Children
(l.) Degree of Tree : Degree of Root Node

(iii) Traversal in Tree

-> Pre Order : Root - Left Subtree - Right Subtree
-> In Order : Left Subtree - Root - Right Subtree
-> PostOrder : Left Subtree - Right Subtree - Root

(iv) Binary Search Tree (BST)

-> The value of the key of left child or left subtree is lesser than the value of the parent key or root.
-> The value of the key of right child or right subtree is lesser than the value of the parent key or root.
-> Also called as ordered or sorted Binary Tree.

→ Also called as ordered or sorted Binary Tree.

(v) Algorithm to Search for a Node in BST

Step 1 : Start

Step 2 : Start at the root of the BST

2.1 : If the root is null or the value of root matches the search key, return the root

Step 3 : If the search key is less than root value, recursively search in the left subtree

Step 4 : If the search key is greater than the root value, recursively search in the right subtree

Step 5 : Repeat steps 3 and 4 until search key is found

Step 6 : Stop / End

(vi) Algorithm to Delete a Node in BST

Step 1 : Start

Step 2 : Search for the node to be deleted using the Search Algorithm

Step 3 : If the Node has no children (Leaf Node), simply delete it.

Step 4 : If the Node has one child

4.1 : Set the parent's pointer to the node's child

Step 5 : If the Node has two children

5.1 : Find the minimum node in the right subtree (or maximum node in left subtree).

5.2 : Replace the value of the node to be deleted with the value of the minimum (or maximum) node.

5.3 : Recursively delete the minimum (or maximum) node from the right (or left) subtree.

Step 6 : Stop / End

(vii) Threaded Binary Tree

→ Type of Binary Tree where the empty left and right child nodes are replaced with threads that link nodes directly to their in-order predecessor or successor.

→ A way to traverse the tree without using Recursion or Stack.

successor.

→ A way to traverse the tree without using Recursion or Stack.

→ Two types – Single Threaded
– Double Threaded

GRAPH

(i) A graph can be defined as groups of vertices & edges that are used to connect these vertices.
A graph can be seen as a cyclic tree where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship.

(ii) Types of Graph – Directed Graph
– Undirected Graph

(iii) Terminologies

(a.) Path : sequence of nodes that are followed in order to reach some terminal node

(b.) Closed Path : If initial node is same as terminal node; $V_0 = V_n$

(c.) Simple Path : If all the nodes are distinct with an exception $V_0 = V_n$

(d.) Cycle : Path which has no repeated edges or vertices except the first and last vertices.

(e.) Connected Graph : Some path exists between every two vertices. There are no isolated vertices.

(f.) Weighted Graph : Each edge is assigned some data such as length or weight.

(g.) Digraph : Directed Graph in which each edge of the graph is associated with some directions & the traversing can be done only in the specified direction.

(h.) Loop : Edge that is associated with the similar end points

(i.) Adjacent Nodes : Neighbour or connected nodes.

(j.) Degree of the Node : No. of edges that are connected with that node.
A node with degree 0 is called an isolated node.

(iv) Graph Representation – Adjacency List
– Adjacency Matrix
– Edge Pairs

(v) BFS (Breadth First Search) Algorithm (using Queue)

(v) BFS (Breadth First Search) Algorithm (using Queue)

Step1 : Start

Step2 : Enqueue the root node into a queue

Step3 : While the queue is not empty:

3.1 : Dequeue a node from the queue

3.2 : Process the node (eg. print its value)

3.3 : Enqueue all the adjacent (child) nodes of the dequeued node into the queue.

Step4 : Repeat until the Queue is empty

Step5 : Stop / End

(vi) DFS (Depth First Search) Algorithm (using Stack)

Step1 : Start

Step2 : Push the root node into a stack

Step3 : While the stack is not empty

3.1 : Pop a node from the stack

3.2 : Process the node (eg. print its value)

3.3 : Push all the adjacent (child) nodes of the popped node onto the stack

Step4 : Repeat until the stack is empty

Step5 : Stop / End

(vii) Applications of BFS

-> Unweighted Graphs
-> P2P Networks
-> Web Crawlers
-> Navigation System
-> Network Broadcasting

(viii) Applications of DFS

-> Weighted Graphs
-> Detecting a cycle in a Graph
-> Path Finding
-> Topological Sorting
-> Searching Strongly connected components of Graph
-> Solving puzzles with only one solution