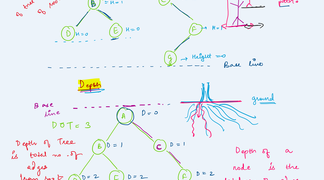
Tuesday, 12 November 2020 9:02 PM

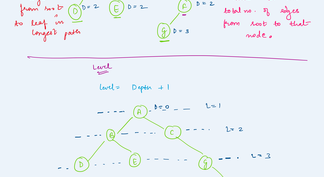
Height

Height of a tree is total no. of edges from root to leaf node.

Height of a node is total no. of edges from leaf to that node in longest path.



Depth



Depth of Tree is total no. of edges from root to leaf in longest path.

Depth of a node is the total no. of edges from root to that node.

Level

level = Depth + 1

Path

Path is a sequence of nodes and edges between two nodes.

source destination

Path (A & J)

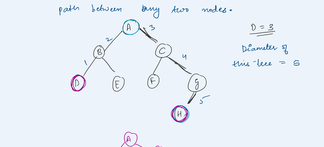
A → B → E → F (4)

Path (C & K)

C → G → K (3)

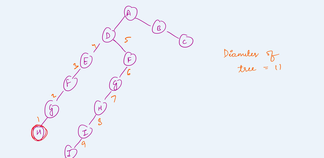
Diameter

Diameter of a tree is the length of the longest path between any two nodes.



D = 3

Diameter of this tree = 5



Diameter of tree = 11

Graph Representations

Three criteria :-

1. How much memory/space is needed for that representation.
2. How long it takes to determine whether a given edge is in the graph.
3. How long it takes to find the neighbors of a given vertex.

1 Edge List

→ List of |E| edges which is called edge list.

[[0,1] , [1,7] , [7,3] , [3 2] , [0 4] , [2,4] ...]

Advantages (1) It is simple.

Criteria : (1) O(|E|)

(2) O(|E|) It takes an edge from 2 to 3

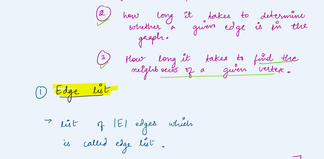
(3) It takes 2 a neighbour of 3 ??? O(|E|)

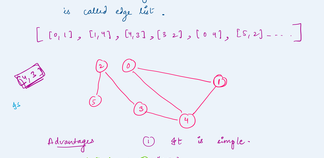
2 Adjacency Matrix

For a graph with |V| vertices, an adjacency matrix |V| × |V| where of 0s and 1s.

cell → [i, j] → 1 if we have an edge from i to j

(1) Here [i,j] → 0





→ It's directed graph

→ undirected 10 vertices 3 edges

(1) O(V²)

(2) O(1) It takes an edge from 0 to 2 ??

(3) O(1) It takes a neighbour of 2 ???

Q→ Find all neighbours of a node x ?? O(V) (2)

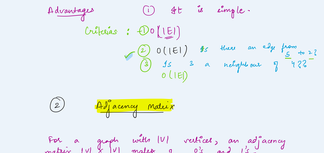
Adjacency List

→ combines adjacency matrices and edge list.

For each vertex i, store an array/Hashmap of vertices adjacent to i.

→ We typically have a Hashmap / array of arrays / Hashmap for each key of Hashmap / value of array.

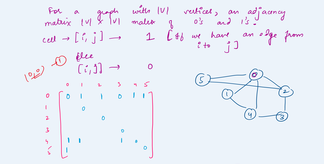
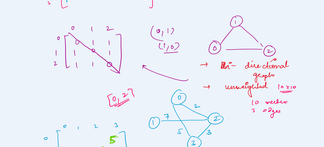
Hashmap of vertices as keys



→ takes less space

(1) O(2|E| + O(V))

(2) O(V) It takes an edge from A to E ??

(3) O(V) It is a neighbour of C ??

Q → Find all neighbours of C ??