ToppersNotes

GATE

COMPUTER SCIENCE & INFORMATION TECHNOLOGY

VOLUME-IV

DIGITAL LOGIC & ENGINEERING MATHEMATICS

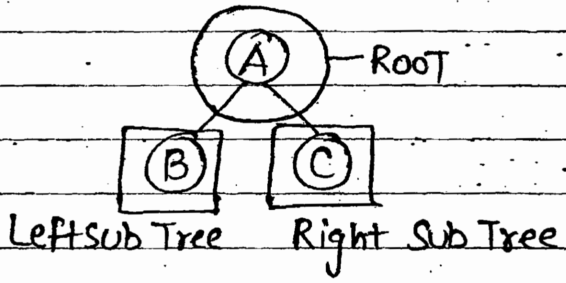
Sierra Innovations Pvt. Ltd.

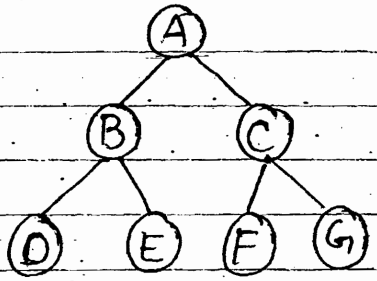
Contents

Tree Traversal & Graph Traversal

(i) Tree Traversal

① Preorder [Root LST RST] [N L R]
② Postorder [LST RST Root] [L R N]
③ Inorder [LST Root RST] [L N R]





① Preorder (root) [N L R]

```
          Preorder (root)  ⇒ T(n)
          {
N             if (root → data) "Print Root"   ⇒ O(1)
L         Preorder (root → LST)               ⇒ T(n/2)
R         Preorder (root → RST)               ⇒ T(n/2)
          }
```

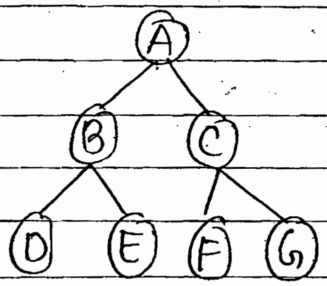
$$2T(n/2) + C$$
$$= O(n)$$

② Inorder [L N R]

```
            Inorder (root)
              {
L               Inorder (root → LST)
            // Inorder ( root →            ⟵ (Don't Consider)
N                 if (root → data) "Paint"
R                 Inorder (root → RST)
              }
```

③ Postorder [L R N]

```
            Postorder (root)
              {
L               Inorder (root → LST)
R               Inorder (root → RST)
N               Inorder (root → Data)  "Paint"
              }
```



[LNR] Inorder = DBE A FCG

[NLR] Preorder = ABDECFG

[LR N] Postorder = DEBFGCA

* Preorder, Postorder, Inorder on N nodes binary Tree will take O(n) Time [WC, BC, AC]

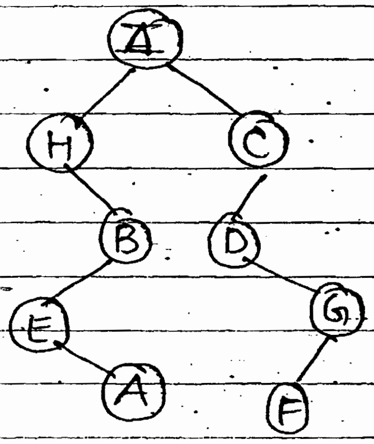
Space Complexity = Present No of levels.

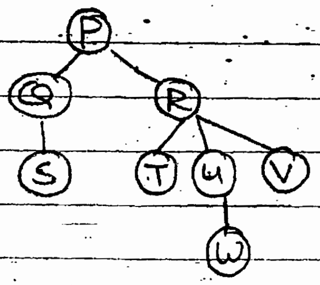
In Best Case Stack space O(logn)

In Worst Case Stack space O(n)

If 3-ary Tree is given then In order Traversal is [L N mid R].

Example-2



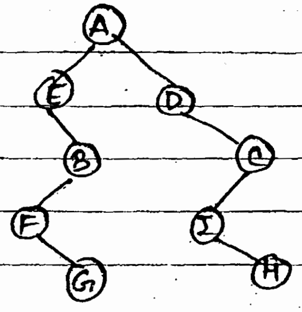


Inorder : SQPTRWUV

[NLR] Preorder = IHBEACDGF
[LRN] Postorder = AEBHFGDCI
[LNR] Inorder = HEABIDFGC

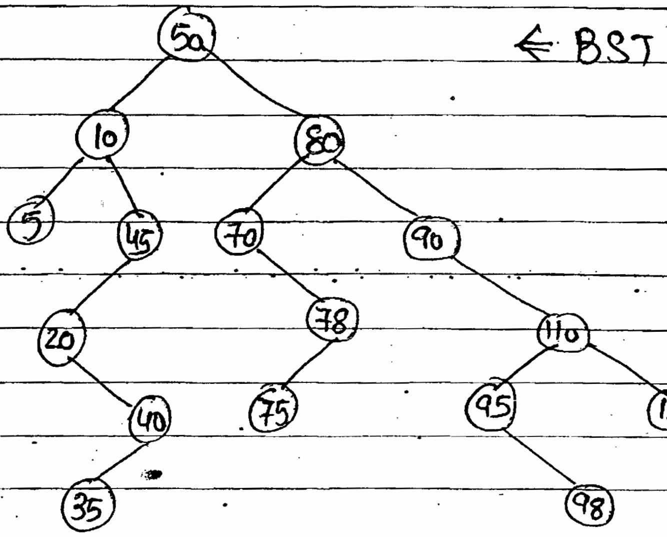
= O(n)

Example-3



Inorder = EFGBADIHC
Preorder = AEBFGDCIH
Postorder = GFBEHICDA

Ex-4



⇐ BST

In order = 5, 10, 20, 35, 40, 45, 50, 70, 75, 78, 80, 90, 95, 98, 110, 120
← smaller → grater

Preorder = 50, 10, 5, 45, 20, 40, 35, 80, 70, 78, 75, 90, 110, 95, 98, 120

Post order = 5, 35, 40, 20, 45, 10, 75, 78, 70, 98, 95, 120, 110, 90, 80, 50

NOTE → Inorder Traversal of the BST in assending order

Important

LST → Root → RST

↓ Small Middle ↓ Greater
(Small → Middle → Greater)

⇓

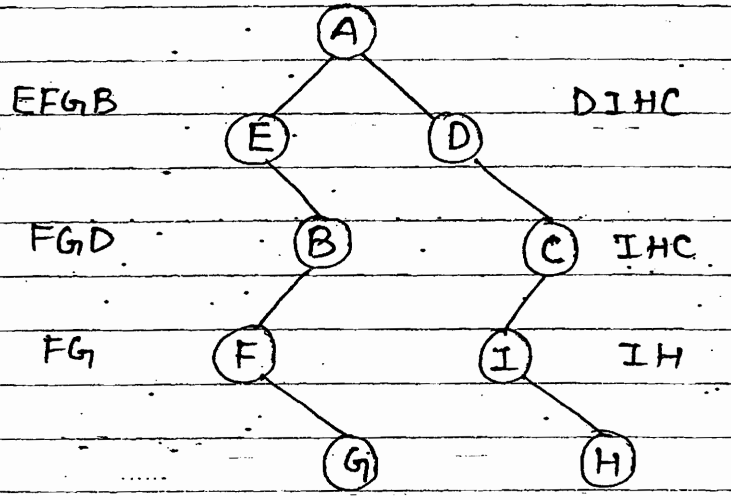
Time Complexity if we have already Created BST $= O(n)$ [otherwise $= O(n^2)$]

Q- Consider the following Binary Tree Data/Information

Inorder : EFGBADIHC (Inorder not sorted

Preorder : AEBFGDCIH

POSTorder = ?

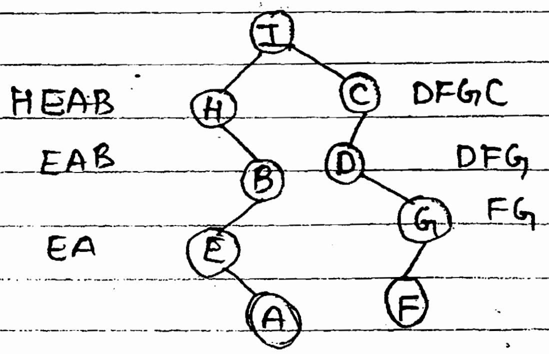


Postorder : GFBEHICDA

Q- Consider the following BT Data/Information

INorder : HEABIDFGC

Postorder : AEBHFGDCI



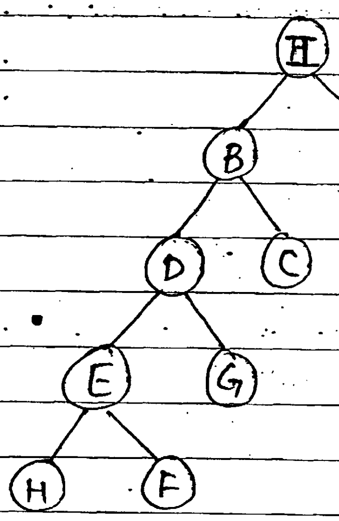
Preorder
↓
IHBAACDGF

$T(n) = O(n^2)$

Q— Consider the following BT Data.

Preorder = I B D E H F G C A

Inorder = H E F D G B C I A



$T(n) = O(n^2)$

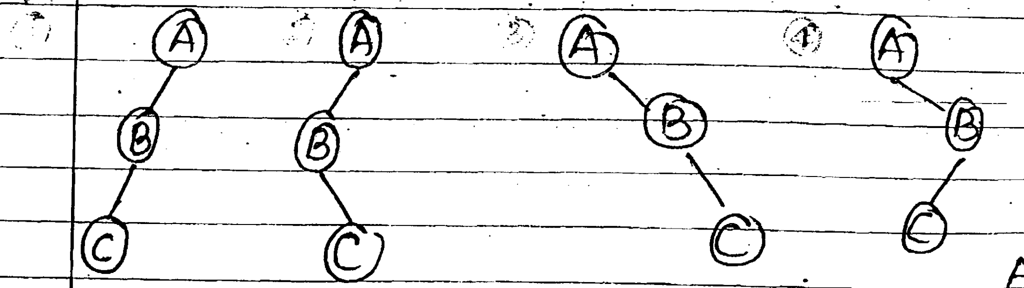
Postorder → HFEGDCBAI

Q- Consider the following BT Data

Preorder ABC

Postorder CBA

$= T(n) = O(2^{n}$



ACB
↑
Inorder

Inorder → CBA

BCA
↑
Inorder

A ~~C~~ BC
↑
Inorder

Time Complexity $= O(2^{n})$

NOTE

→ To Create unique Binary Tree Inorder Compulsory

Inorder } unique BT
Preorder }

Inorder } unique
Postorder }

* Preorder } Binary Tree possible
Postorder } But not unique Binary Tree

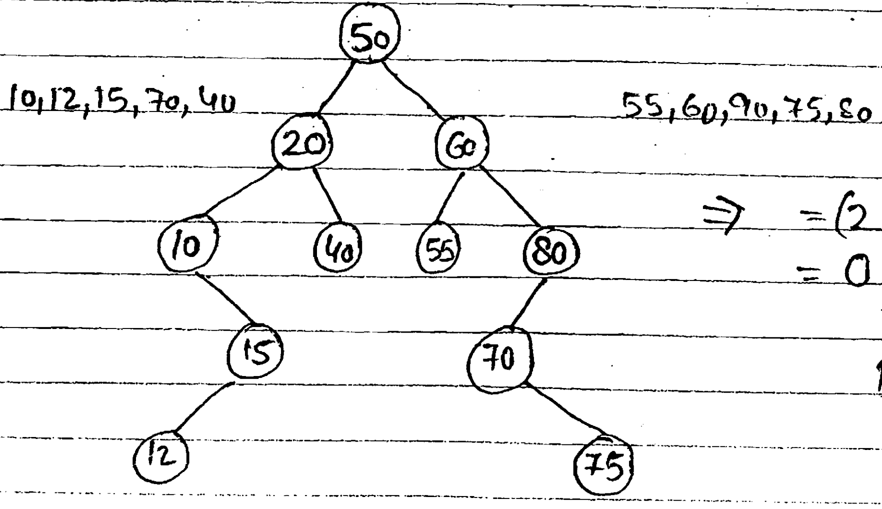
2015 GATE Consider the following BST Data.

Pre: 50, 20, 10, 15, 12, 40, 60, 55, 80, 70, 75

Postorder: ?

Because of BST it is easier. Inorder Traversal is in Ascending order.

Inorder: 10, 12, 15, 20, 40, 50, 55, 60, 90, 75, 80



⇒ $= (2\ N\log n)$
$= O(N\log n)$
↑
Becoz Of BST

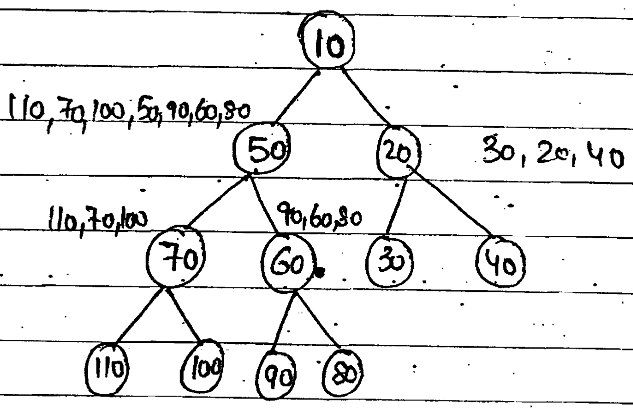
POSTorder: 12, 15, 10, 40, 20, 55, 75, 70, 80, 60, 50

Q- Consider the Inorder of min heap Tree

INorder : 110, 70, 100, 50, 90, 60, 80, 10, 30, 20, 40

Then What will be the Post order?

Always Take min in Case of min heap and make tree.



$\rightarrow O(n)$

$\rightarrow O(\frac{n}{2}) + \frac{n}{2} = O(n)$

$\rightarrow \frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4} \rightarrow O(n)$

$\rightarrow$ $= O(n)$

Every time Root Finding ⇒ O(n log n)

NOTE

1. Binary Tree Pre Inorder, Postorder is given to Construct unique BT $O(n^2)$ Time Required. (N time linear Search to find LST, RST) (Worst Case

2. Binary Search Tree Pre Inorder, Postorder is given to Construct unique Binary Seach Tree O(nlogn) time Required (N Times Binary Search to find out LST, RST)

3. Binary Tree Preorder Post order given to Construct the Binary Tree $O(2^n)$ time Req. becoz of manual Checking

GRAPH TRAVERSAL

Covers all Nodes Exactly once.

① BFT (Breadth first Traversal)
② DFT (Depth first Traversal)

① BFT (Breadth first Traversal)

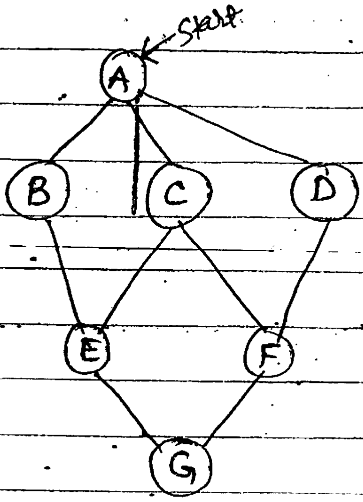
```
BFT(V)
{
    Visited (V) = 1
    add (V, Q) → Queue
    While ( Q is not empty)
    {       ↓ Queue
        x = delete (Q)
        Printf(x);
        for (all W adjacent to x)
        {
            if ( W is not visited )
            {
                Visited (W) = 1
                add (W, Q)
            }
        }
    }
}
```

Here,

Visited (V) → is Flag and Handle of using array.

BFT Use Queue.

Ex-1



visited

0 → Not visited
1 → visited

A	B	C	D	E	F	G
0	0	0	0	0	0	0

Q = Queue

Q

A

↓

	B	C	D

↓

		C	D	E

↓

			D	E	F

↓

				E	F

↓

					F	G

↓

						G

↓

← Empty Queue

output
↓
A B C D E F G

$T(n) = O(V+E)$

* Important Notes

① To implement BFT We are using Queue Data Structure.

② Time Complexity = O(V+E) (For all Case)
(In all graph theory it take min Time

Space Complexity = Input + Extra

$$\underbrace{\text{Input}}_{\text{Adj. List} \to O(V+E)} + \underbrace{\text{Extra}}_{\text{Queue} + \text{Array} \to V + V}$$

$$\underbrace{O(V+E) + 2V}$$

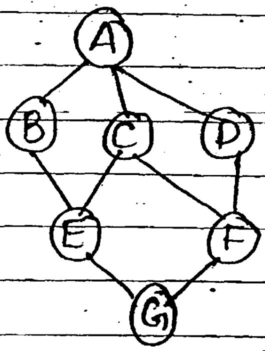
$$3V+E$$

$$\downarrow$$

$$O(V+E)$$

③ BFT is also known as level Order Traversal

Ex- Consider the following graph

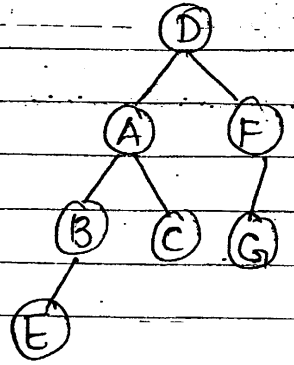


ⓐ ABCDEFG → True
ⓑ ACBDEFG → True
ⓒ CAEFBDG → True
ⓓ DFAGCBE → True
ⓔ CAEGα → False ?
ⓕ ECBGFDA → False
α

Find Correct BFT from the following answers

BFT – Tree for option D

option D → D F A G C B E

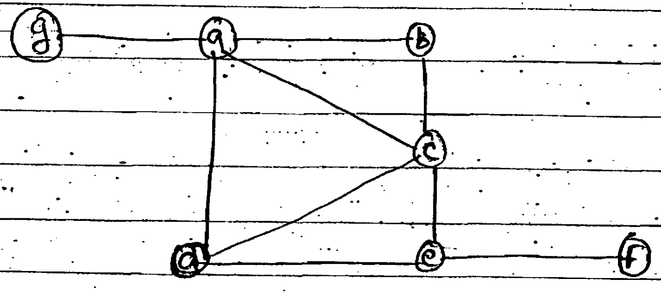


← make using Adjacent of each Node.

→ Here no cycle.

spanning Tree or BFT Tree . Here is a spanning Tree not Shure minimum or not.

Q- Consider the following graph



Correct BFT

(a) a g b d c f e → False

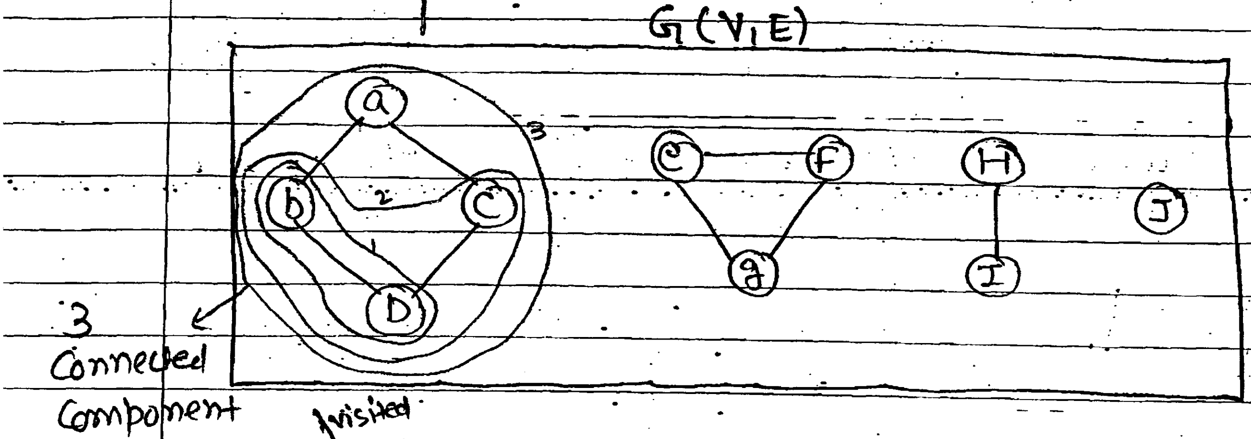
(e) g a b c d e f → True

(b) C e d b a g f → false

(c) F e d c a b g → True

(d) C a b d e F g → False

Applications of BFT



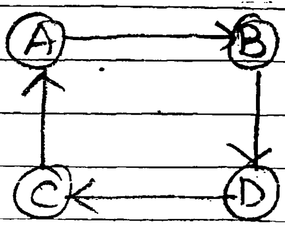
visited									
~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1
a	b	C	d	e	F	g	h	I	J
↑ Initial BFT				↑ Again Apply BFT			↑ Agian BFT		↑ Again BFT

0 → Initial
(Not Cover

① Using BFT we Can verfiy given graph is Connected or disConnected.

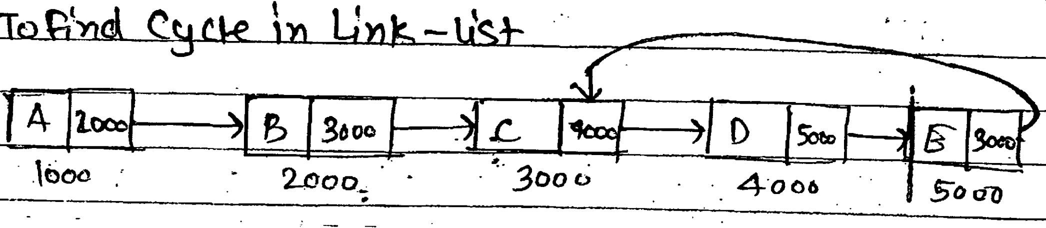
② Using BFT we Can find out no of Connected Componen in given graph (O(V+E) Time Required) ✓

③ Using BFT we Can verify given graph Contain Cycle or not



Use BFT = ABDC→A
↑
cycle
is there

* How To Find Cycle in Link-List



Here link list is is Treated as Directed Graph.
So, apply BFT to find the Cycle.

When apply BFT We get

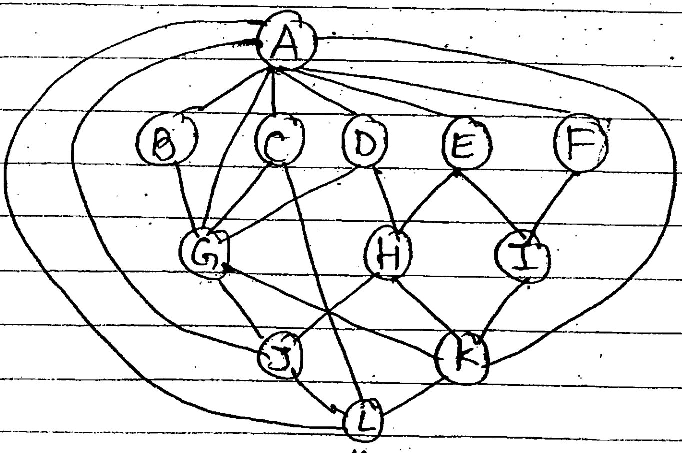
$$A \; B \; C \; D \; E \rightarrow C$$

↑
Already Cover.
So Link List Contain
cycle.

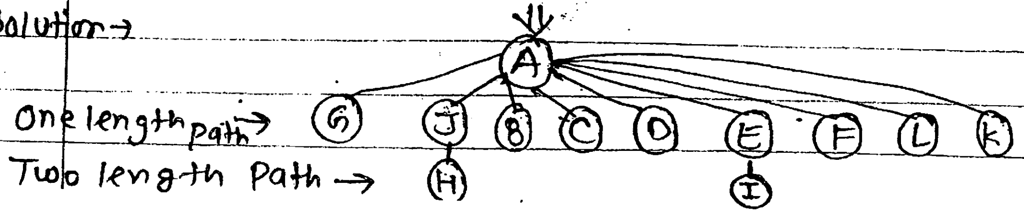
Here also, BFT contain = $\underline{O(V+E)}$

Q- Consider the following undirected graph.

Find Shortest
Path From
Root Node A →



Solution →



④ using BFT We Can find out Shortest path from given Source to every all vertex in the given unweighted graph.

T(n)=O(V+E

Also apply When All edge Nodes Weight are Same.

⑤ We Verify using BFT the given graph is Bipartite graph.

* To implement the Shortest path from given Source to every all vertex in the given unweighted graph Queue Data Structure is used. (Becoz We use BFT)

(No Reccursion In BFT)