Unit-4

Basic Traversal and search Techniques.

Que- Explain Tree Traversal techniques in detail with example.

⇒

Tree :-
Tree data structure is a hierarchical structure that is used to represent and organize the data in a way that is easy to navigate and search.

Tree Traversal :-
-Tree traversal is the process of visiting or accessing each node of tree exactly once in a certain order.
- Tree traversal algorithms helps to visit and process all the nodes of the tree.
-A tree data structure can be traversed in following ways :-

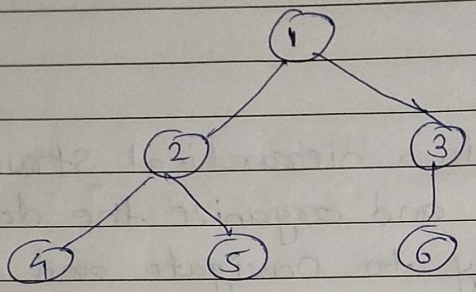
(a) Depth First search (DFS) :-

1) Pre-order Traversal :-
-Pre-order traversal is used to create a copy of a tree.
-Pre-order traversal is used to ~~used~~ get prefix expression on an expression tree.
-Preorder traversal visits the node in the order :-

Root → Left → Right

Algorithm :-
Preorder (tree)
• visit the root

- Traverse the left subtree
- Traverse the right subtree



Preorder Traversal :- 1 → 2 → 4 → 5 → 3 → 6

- Time complexity :- O(N)
- space complexity :- O(1)

2) In-order Traversal :-

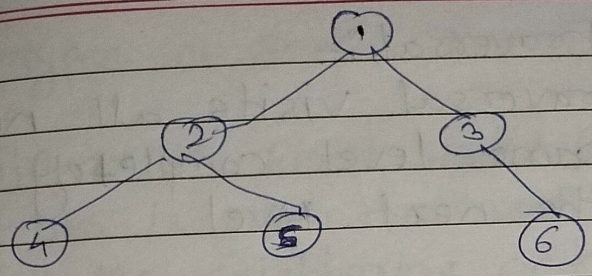
- In case of Binary search Tree, Inorder traversal gives nodes in non-decreasing order.
- To get nodes of BST, in non-increasing order, a variation of Inorder traversal where Inorder traversal is reversed can be used.
- In order traversal can be used to evaluate arithmetic expressions stored in the expression tree

Algorithm :-

Inorder (tree)

- Traverse the left subtree
- Visit the root
- Traverse the right subtree

order :- Left → Root → Right

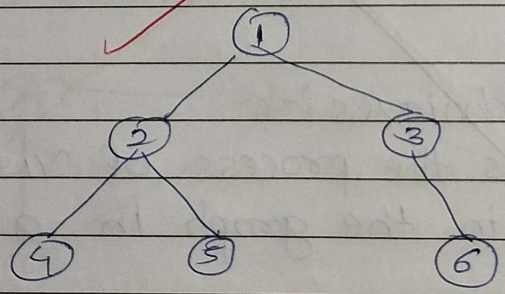


Inorder Traversal :- 4 → 2 → 5 → 1 → 3 → 6

3] Postorder Traversal :-
- Postorder Traversal is useful to get the postfix expression of an expression tree.
- Postorder traversal is also useful to get the postfix expression of an collection algorithm particularly in a systems where manual memory management is used.

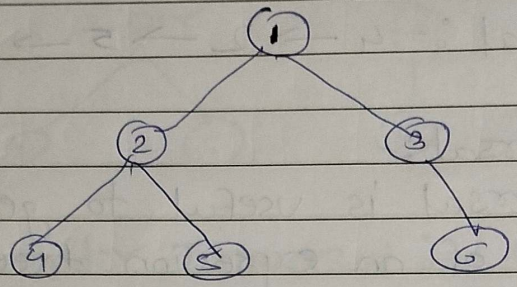
Algorithm :-
Postorder (tree)
- Traverse the left subtree
- Traverse the right subtree
- visit the root

order → Left → Right → Root



Postorder :- 4 → 5 → 2 → 6 → 3 → 1

b) Level order Traversal :-
- Level order traversal visits all nodes present in the same level completely before visiting the next level.



order :- 1 → 2 → 3 → 4 → 5 → 6

- Level order traversal is used in BFS to search or process nodes level-by-level.

Que Write short note Graph Traversal Techniques?

⇒

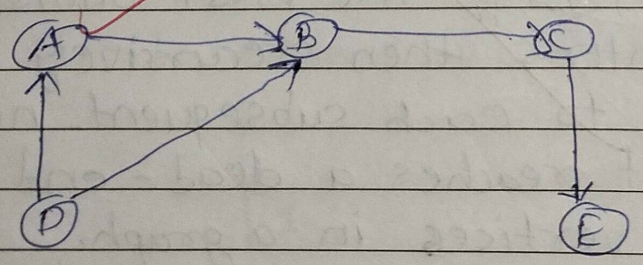
Graph :-
- Graph data structure is a collection of nodes connected by edges.
- It is also used to represent relationships between different entities.

Graph Traversal Techniques :-
- Graph traversal is the process of visiting all the vertices (nodes) in the graph in a systematic way.
- It start involves exploring of traversing each vertex in a graph by following the edge that connect the vertices.

Types of Graph Traversal :-

1] Breadth First search (BFS) :-
- BFS is graph traversal algorithm that explores all the neighboring nodes at the current depth before moving on to nodes at next depth level.
- It starts at the root node and explores all the vertices in a graph.

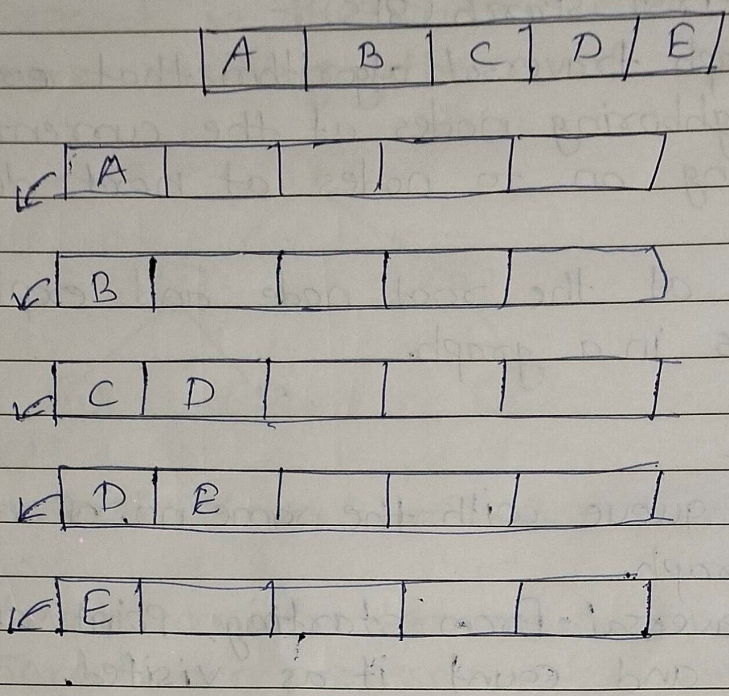
Algorithm :-
1) Define a queue with the same no. of nodes as the graph.
2) Start traversal from starting point and put it into queue and count it as visited.
3] Enqueue all the neighboring nodes of visited node that have not been visited yet.
4] When there are no more vertices to visit Dequeue that node from queue.
5] Repeat step 3 & 4 until queue is empty.
6] When the queue is empty make the final spanning tree by removing unused lines from the graph.



1] Adjacent nodes

A → B
B → C, D
C → E
D → A
E - -

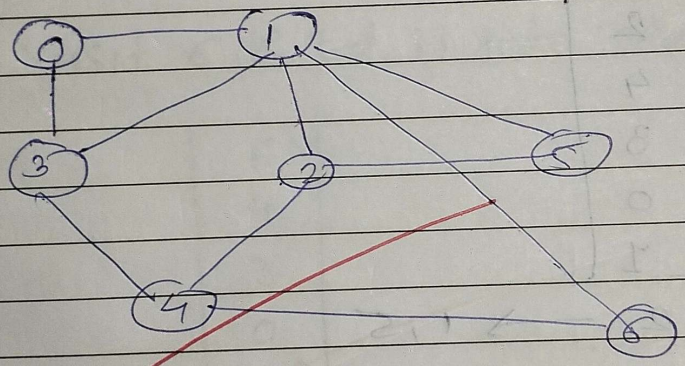
Step-2 :-
Visited queue



2). Depth First search (DFS) :-
- DFS is a simple algorithm used for traversing or searching a tree or graph data structure.
- The algorithm works by exploring a path as far as possible before backtracking.
- starting at a root node or vertex the algorithm explores each branch of the tree or graph by visiting the first unvisited a node it encounters then recursively applying the same rule to each subsequent node or vertex until it reaches a dead-end or visit all the vertices in a graph.

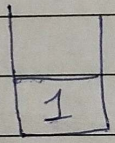
Algorithm :-
1) Define a stack of size equal to the number of vertices of the graph.

2) choose any vertex as starting point for traversal. visit that vertex and push it onto the stack.
3) Push any non-visited adjacent matrix of a top vertex into the stack.
4) Repeat step 3 until there is no new vertex to visit from the top of the stack.
5) Backtrack and remove one vertex from the stack if there are no new vertices.
6) Repeat steps 3, 4 & 5 until stack becomes empty.
7) when the stack is empty. remove unused graph edges to create their final spanning tree.



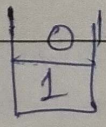
steps :-
1) select any node as source node.

i) visit node 1 & put it into stack



• Adjacent to 1 → 0, 3, 2, 5, 6

ii) Visit 0 & put it into stack



Adjacent to 0 → 4, 3

iii) 1 is already visited then put 3 into stack

3
0
1

Adjacent to 3 → 1, 4, 0

iv) visit 4 & put into stack

4
3
0
1

Adjacent to 4 → 2, 6, 3

v) 3 is visited, then visit 2 & put it into stack

2
4
3
0
1

Adjacent to 2 → 1, 5

vi) 1 is already visited & Puts into stack

5
2
4
3
0
1

Adjacent to 5 → 1, 2 are already visited

vii) Pop 5 from stack

2 X
4
3
0
1

Adjacent to 2 → 5, 1 are already visited

viii) Pop 2 from stack

4
3
0
1

Adjacent to 4 → 2, 6

ix) visit 6 & Put it into stack

6 X
4
3
0
1

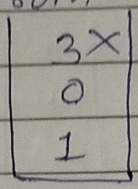
Adjacent to 6 → 4, 1 are already visited

x) Pop 6 from stack

4 X
3
0
1

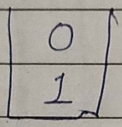
Adjacent to 4 → 2, 6 are already Popped.

xi] Pop 4 from stack



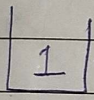
Adjacent to 3 → 0, 1, 4 are already visited

xii] Pop 3 from stack



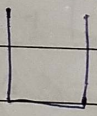
Adjacent to 0 → 1, 3 are already visited

xiii] Pop 0 from stack



Adjacent to 1 → 0, 3, 2, 5, 6 are already Visited

xiiii] Pop 1 & stack become empty.



∴ Path is → 1 → 0 → 3 → 4 → 6 → 2 → 5

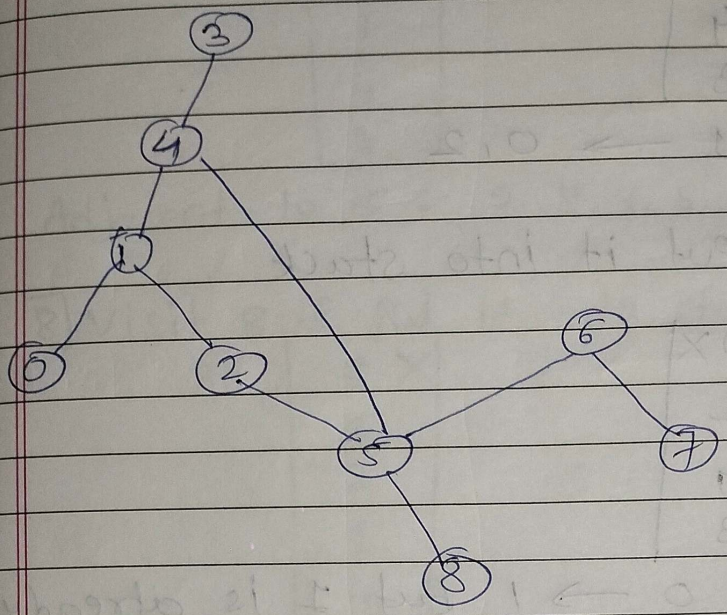
Que- What is DFS. Find the DFS traversal for Following graph?

⇒

Depth First search (DFS):-

- DFS is a simple algorithm used for traversing or searching a tree or graph data structure.
- The algorithm works by exploring a path

as far as possible before backtracking.



steps:- Adjacent Nodes:-

3 → 4	5 → 2, 8, 4, 6
4 → 1, 5	8 → 5
1 → 0, 2	6 → 5, 7
0 → 1	7 → 6
2 → 1, 5	

1] select an arbitary vertex & Put it into stack. visit 3 & Put it into stack.

3

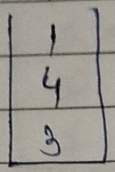
Adjacent to 3 → 4

2] Visit 4 & Put it into stack

4
3

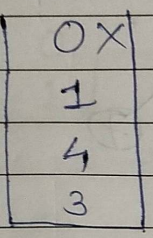
Adjacent to 4 → 1, 5

3) Visit 1 & Put it into stack



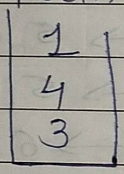
Adjacent to 1 → 0, 2

4) Visit 0 & Put it into stack



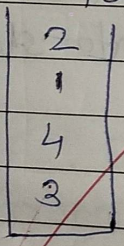
Adjacent to 0 → 1 but 1 is already visited.

5) Pop 0 from the stack



Adjacent to 1 → 0, 2

6) Visit 2 & Put it into stack



Adjacent to 2 → 1, 5

7) Visit 5 & Put it into stack

5
2
1
4
3

Adjacent to 5 :- 2, 8, 4, 6

[8] visit 8 & put it into stack

8 X
5
2
1
4
3

Adjacent to 8 → 5 but 5 is already in the stack.

[9] Pop 8 from the stack

5
2
1
4
3

Adjacent to 5 → 2, 8, 4, 6

[10] visit 6 & put it into stack

6
5
2
1
4
3

Adjacent to 6 → 5, 7.

11) Visit 7 & put it into stack

7 X
6
5
2
1
4
3

Adjacent to 7 → 6 already in the stack.

12) Pop 7 from stack

X 6
5
2
1
4
3

Adjacent to 6 → 5,7 already visited

13) Pop 6 from stack

5 X
2
1
4
3

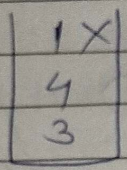
Adjacent to 5 → 2,8,4,6 are already visited.

14) Pop 5 from stack

2
1
4
3

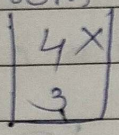
Adjacent to 2 → 1,5 are already visited.

15] Pop 2 from stack



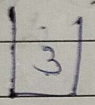
Adjacent to 1 → 0, 2 are already visited

16] Pop 1 from stack



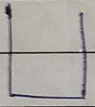
Adjacent to 4 → 1, 5 are already visited

17] Pop 4 from stack



Adjacent to 3 → 4 is already visited

18] Pop 3 from stack, now stack become empty.



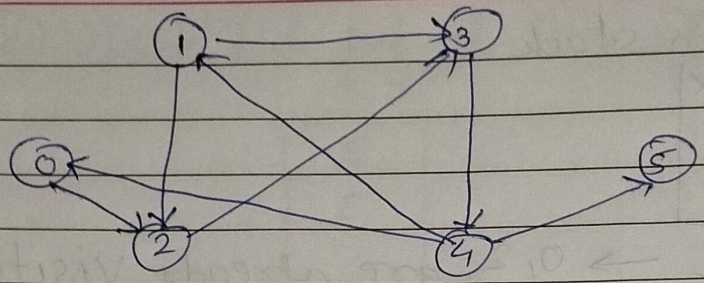
Path :- 3 → 4 → 1 → 2 → 5 → 6 → 7 → 8 → 0.

Que- What is BFS. Find the BFS traversal for given graph?

⇒

Breadth First search (BFS) :-

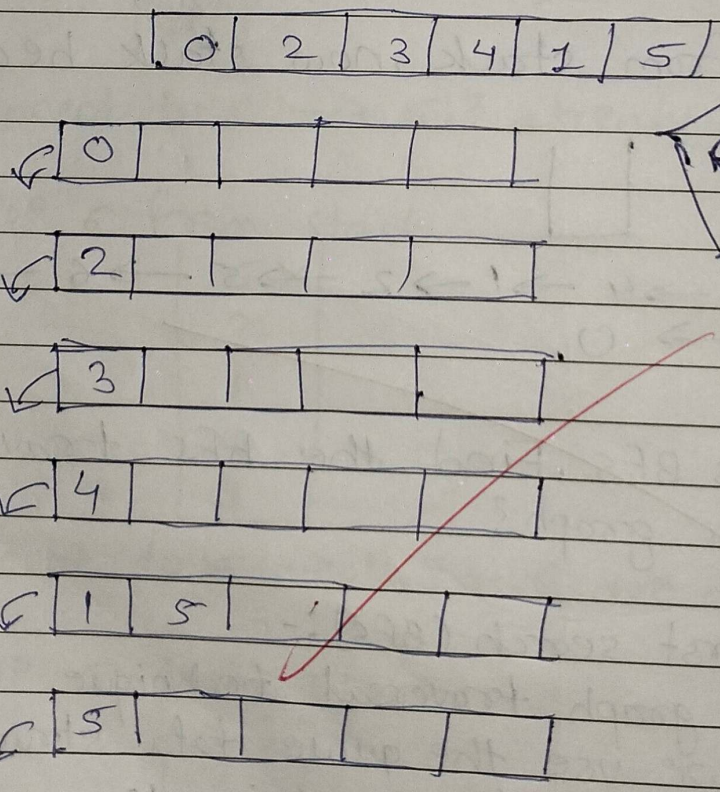
- BFS is a graph traversal technique.
- In BFS, we use the queue data structure & it follows the FIFO/FCFS principle.



Step 1 :- Adjacent Nodes :-

0 → 2
1 → 3, 2
2 → 3
3 → 4
4 → 0, 5, 1
5 → -

step 2 :- visited Queue :



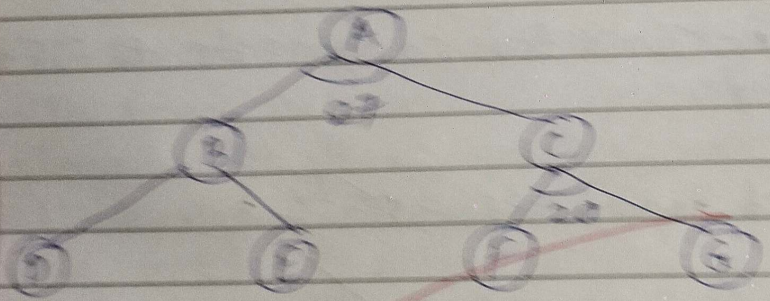
Path :- 0 → 2 → 3 → 4 → 1 → 5

Ques: Explain following concept with details —
i) AND/OR Graph
ii) Connected components

Ans:

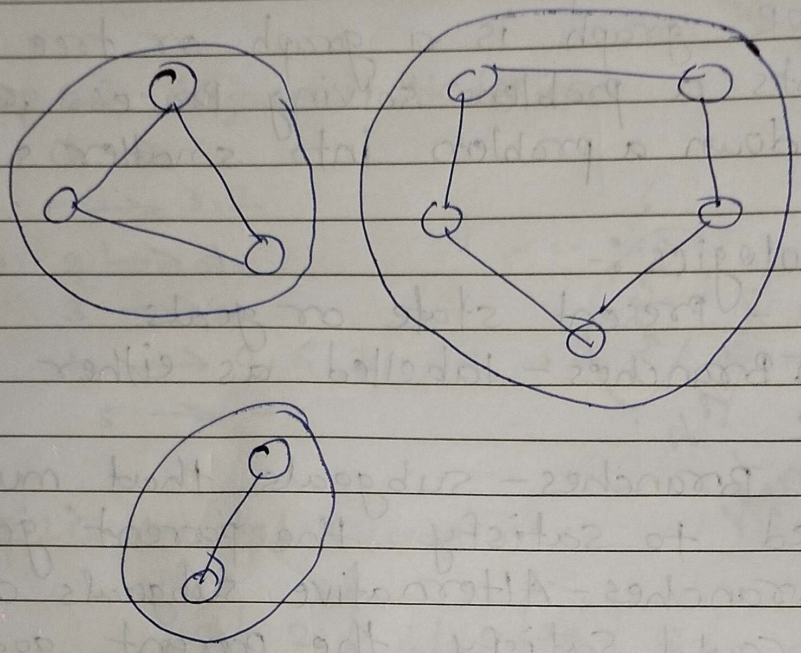
i) AND/OR Graph :-
- AND/OR graph is a graph or tree that represents a problem solving process & how to break down a problem into smaller subproblems.

Terminologies :-
1) Node - Present state or goals
2) Branches - labelled as either AND or OR
3) AND Branches - Subgoals that must all be achieved to satisfy the parent goal.
4) OR Branches - Alternative subgoals anyone of which could satisfy the parent goal.



ii) Connected components :-
- In data analysis, connected components are a set of nodes in a graph that are connected to the each other by paths.
- Connected components are ~~connected~~ to the ~~of~~ liked a clusters of similar objects.
- Connected components can be used to solve

problems by solving grouping item based on similarity factor. ie
- mainly connected component of a graph is a maximal connected subgraph.



Que- Explain Bi-connected components & DFS with example.

⇒

Bi-connected components & DFS :-

1] Biconnected components (BCC) :-
- A biconnected components is a maximal subgraph in which any two vertices are connected by at least two distinct Points.
- This means that removing any one vertex from the component will not a discovered the graph.
- A graph is said to be biconnected if there are no articulation points (vertices).
- If a graph contains articulation points, it

can be decomposed into multiple biconnected components.

2. Articulation Point :-
- An articulation point (or cut vertex) is a vertex in a graph such that removing this vertex increases the no. of connected components of the graph.
- Identifying these points helps decomposing a graph into biconnected components.

3] DFS :-
- DFS is key algorithm for finding biconnected components.
- DFS is traversal technique that starts a root node & explores as far as possible along each branch before backtracking.

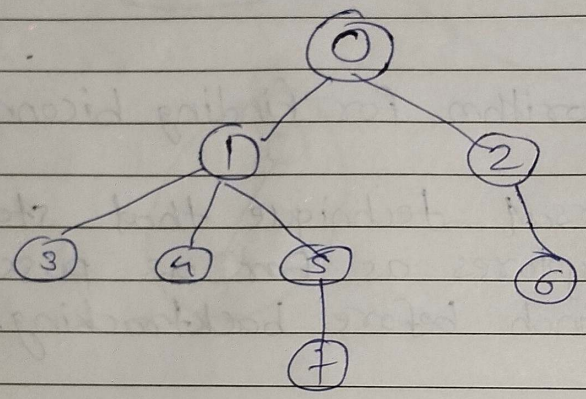
• Biconnected components using DFS :-
- To find biconnected components, we use DFS along with two auxillary arrays :-
a) Discovery Time (disc[]) :- The time at which the vertex is first visited.
b) Low Value (low[]) :- The lowest discovery time reachable from the subtree rooted at that vertex.

Steps :-
1) Start DFS from any vertex. For each vertex, maintain its discovery time & low value.
2) For every DFS tree edge (u,v) :-
• If v is not visited yet, recurse DFS for

v. After recursion, updated low[u] as the minimum of low[u] and low[v].
- If v is already visited for and v is not the parent of u, update low[u] to be the minimum of low[u] & disc[v]

3) For an edge (u,v), if low[v] >= disc[u], then u is an articulation point. The subtree rooted at v is bi-connected component.

- Example :-



steps :-

1) DFS tree :- The start DFS from node 0 with the discovery & low values initialized to infinity.

- visit node 0 → disc[0] = 0, low[0] = 0
- visit node 1 → disc[1] = 1, low[1] = 1
- visit node 3 → disc[3] = 2, low[3] = 2

Backtrack to node 1.

- visit node 5 → disc[5] = 4, low[5] = 4
- visit node 7 → disc[7] = 5, low[7] = 5

Backtrack to node 5

Backtrack to node 1

Backtrack to node 0

- visit node 2 → disc[2] = 6, low[2] = 6
- visit node 6 → disc[6] = 7, low[6] = 7

Backtrack to node 2
Backtrack to node 0

2] Biconnected components :-

- {3, 1}
- {4, 1}
- {5, 7, 1}
- {2, 6, 0}