Modern Education Society's
Wadia College of Engineering, Pune

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR:	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: HPC- 02**

TITLE: Parallel DFS and BFS Algorithms

AIM: Design and implement Parallel Breadth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS

OBJECTIVES:
Students should perform Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP

PRE-REQUISITES:
1. Knowledge of OPENMP programming.
2. Knowledge of parallel programming.
3. Knowledge of BFS and DFS search.

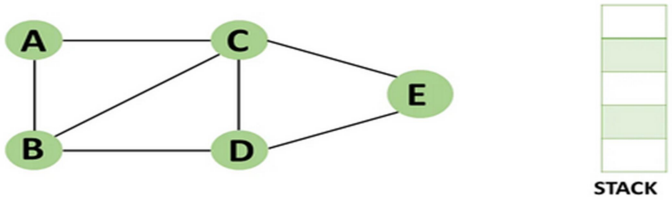
THEORY:-

What is DFS?
DFS stands for Depth-First Search. It is a popular graph traversal algorithm that explores as far as possible along each branch before backtracking. This algorithm can be used to find the shortest path between two vertices or to traverse a graph in a systematic way. The algorithm starts at the root node and explores as far as possible along each branch before backtracking. The backtracking is done to explore the next branch that has not been explored yet. DFS can be implemented using either a recursive or an iterative approach. The recursive approach is simpler to implement but can lead to a stack overflow error for very large graphs. The iterative approach uses a stack to keep track of nodes to be explored and is preferred for larger graphs. DFS can also be used to detect cycles in a graph. If a cycle exists in a graph, the DFS algorithm will eventually reach a node that has already been visited, indicating that a cycle exists.
A standard DFS implementation puts each vertex of the graph into one of two categories:
1. Visited
2. Not Visited
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

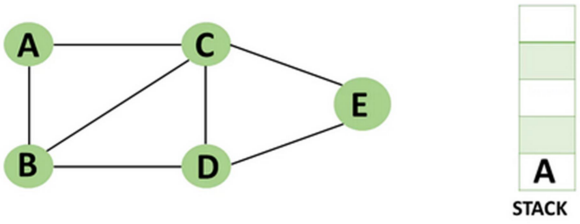
What is BFS?
BFS stands for Breadth-First Search. It is a graph traversal algorithm used to explore all the nodes of a graph or tree systematically, starting from the root node or a specified starting point, and visiting all the neighboring nodes at the current depth level before moving on to the next

depth level. The algorithm uses a queue data structure to keep track of the nodes that need to be visited, and marks each visited node to avoid processing it again. The basic idea of the BFS algorithm is to visit all the nodes at a given level before moving on to the next level, which ensures that all the nodes are visited in breadth-first order.
BFS is commonly used in many applications, such as finding the shortest path between two nodes,solving puzzles, and searching through a tree or graph.

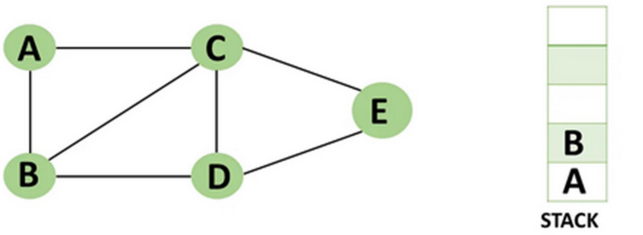
Example of DFS:
To implement DFS traversal, you need to take the following stages.
Step 1: Create a stack with the total number of vertices in the graph as the size.
Step 2: Choose any vertex as the traversal's beginning point. Push a visit to that vertex and add it to the stack.
Step 3 - Push any non-visited adjacent vertices of a vertex at the top of the stack to the top of the stack.
Step 4 - Repeat steps 3 and 4 until there are no more vertices to visit from the vertex at the top of the stack.
Step 5 - If there are no new vertices to visit, go back and pop one from the stack using backtracking.
Step 6 - Continue using steps 3, 4, and 5 until the stack is empty.
Step 7 - When the stack is entirely unoccupied, create the final spanning tree by deleting the graph's unused edges.



Step 1: Mark vertex A as a visited source node by selecting it as a source node.
You should push vertex A to the top of the stack.



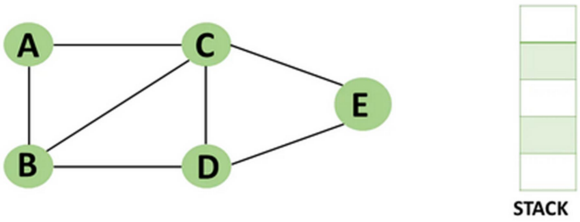
Step 2: Any nearby unvisited vertex of vertex A, say B, should be visited.
You should push vertex B to the top of the stack



Step 3: From vertex C and D, visit any adjacent unvisited vertices of vertex B. Imagine you have chosen vertex C, and you want to make C a visited vertex.

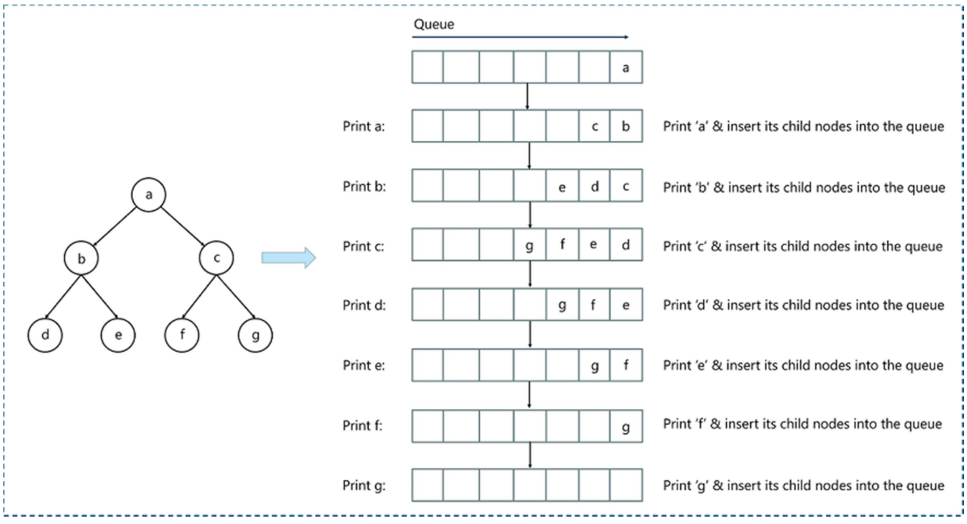
.
.
.

Step 10: All of the nearby vertices of Vertex A, B, and C, have already been visited, so pop vertex A from the stack as well.



Example of BFS

Now let's take a look at the steps involved in traversing a graph by using Breadth-First Search:
Step 1: Take an Empty Queue.
Step 2: Select a starting node (visiting a node) and insert it into the Queue.
Step 3: Provided that the Queue is not empty, extract the node from the Queue and insert its child nodes (exploring a node) into the Queue.
Step 4: Print the extracted node.



QUESTIONS FOR REVIEW:

1. What is BFS and DFS?
2. Write a parallel BFS and DFS algorithm using OpenMP
3. What is the advantage of using parallel programming in DFS and BFS