Deccan Education Society's
Fergusson College (Autonomous), Pune
Department of Computer Science

A

Report
on

"Visual Representation Of DFS And BFS Algorithm "

In partial fulfillment of Post Graduate course

in

M.Sc. Computer Science – I

(Semester -I)

CSC-520 Practical I

SUBMITTED BY

Sahil Dattatray Shejal (ROLL NO – 246220)
Gopika G S (ROLL NO –246210)

Index

Case Study Description:

Overview:

This project is a **graph traversal visualizer**. It allows users to input a graph, choose one of the two graph traversal algorithms (**BFS** or **DFS**), and see how these algorithms work visually on a graph. The program displays nodes (vertices) and edges (connections between nodes) on a canvas, and shows the process of visiting each node using **BFS** or **DFS** step-by-step.

In simple terms, the program helps you understand how BFS and DFS explore a graph by showing each step in the process.

1. Algorithm Details

Here's how the two key graph traversal algorithms, **BFS** and **DFS**, work:

Breadth-First Search (BFS)

- **What it is:** BFS is an algorithm used to explore all the nodes of a graph level by level. It starts from a given node and explores all its neighbors (nodes directly connected to it). After visiting all neighbors, it moves to the next level and continues exploring in this way.
- **How it works:**
    1. Start from a given node.
    2. Add it to a queue (a list where nodes wait their turn to be explored).
    3. Dequeue a node from the front, visit it, and add all its neighbors (connected nodes) to the queue.
    4. Repeat this process until all nodes have been visited.
- **Key Points:**
    - BFS uses a **queue** to keep track of nodes.
    - It's good for finding the **shortest path** in an unweighted graph.

Depth-First Search (DFS)

- **What it is:** DFS is an algorithm that explores a graph by going as deep as possible along a branch before backtracking. It explores one node fully before moving to the next.
- **How it works:**
    1. Start from a given node.

2. Add it to a stack (a collection where nodes are processed in a Last In, First Out (LIFO) manner).
3. Pop a node from the stack, visit it, and add all its unvisited neighbors to the stack.
4. Repeat this process until the stack is empty.

- **Key Points:**
    - DFS uses a **stack** to keep track of nodes.
    - It's good for tasks like **detecting cycles** in graphs or **topological sorting**.

2. UI Design Details

The user interface (UI) is designed to be simple and easy to use, with the following sections:

1. **Input Section:**
    - **Vertices Input:** A field where users enter the nodes (e.g., A, B, C, D).
    - **Edges Input:** A field where users enter the connections between nodes (e.g., A-B, B-C).
    - **Start Vertex:** A field where users enter the starting node for the traversal.
    - **Traversal Type:** Radio buttons to select either **BFS** or **DFS** for the traversal algorithm.
2. **Visualization Section:**
    - **Graph Visualization:** A canvas where the graph is drawn. Each node is displayed as a circle, and edges are drawn as lines between connected nodes.
    - **Traversal Steps:** A display that shows the current state of the traversal (the list of nodes currently being explored).
    - **Result Sequence:** Shows the order in which nodes are visited during the traversal.

3. HTML, CSS, and JavaScript Description

HTML:

- **Structure:** The HTML creates the layout of the page, including input fields for vertices and edges, a canvas for visualizing the graph, and buttons to start and clear the graph.
- **Traversal Buttons:** There are buttons to start the BFS or DFS traversal and to clear the graph.

CSS:

- **Design:** The CSS defines how the page looks. It uses a **flexbox layout** to arrange the graph controls and visualization sections side by side. It also makes the page responsive, meaning it adjusts for different screen sizes (desktop, tablet, mobile).
- **Styling:** The page uses a gradient background for an appealing look, and buttons have hover effects to make the page interactive. The navigation bar at the top lets the user switch between BFS and DFS details pages.

JavaScript:

- **Graph Creation:** When the user enters the graph data (vertices and edges), JavaScript dynamically creates a graph object and stores the connections between nodes.
- **Traversal Algorithms:**
  - **BFS:** The bfs() function uses a queue to explore the graph. Each step in the traversal is displayed on the canvas.
  - **DFS:** The dfs() function uses a stack to explore the graph. It also updates the canvas after each step.
- **Graph Drawing:** JavaScript draws the graph on the canvas using the <canvas> element. Nodes are placed in a circular layout, and edges are drawn between them.
- **Asynchronous Visualization:** The graph is updated step-by-step with a short delay, allowing users to see the traversal process in real-time.

Conclusion

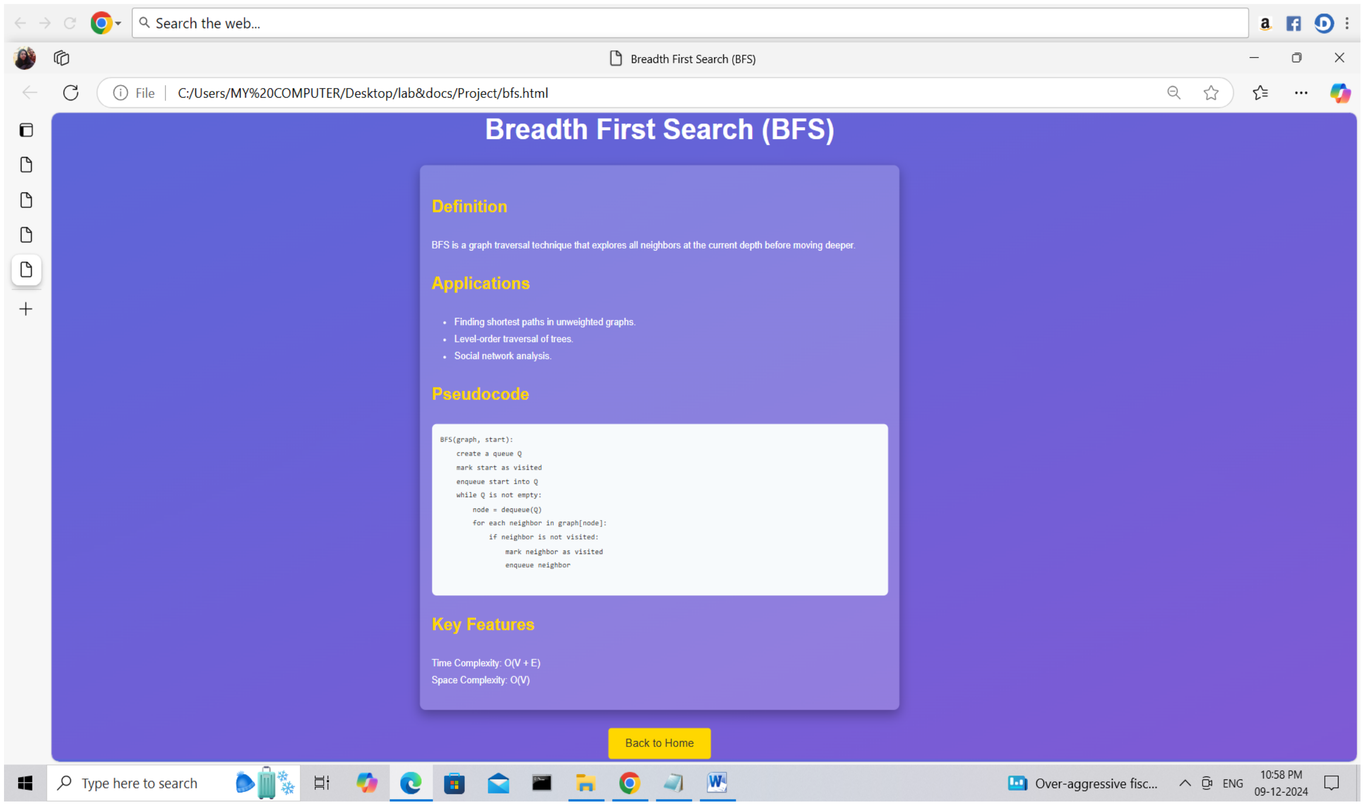
This project helps users understand **BFS** and **DFS** by providing an interactive and visual way to see how these algorithms explore a graph. The key features include:

- **Interactive Graph Creation:** Users can input vertices and edges to create their own graph.
- **Step-by-Step Traversal:** Both BFS and DFS are visualized in real-time, with each traversal step shown on the screen.
- **Simple UI:** The UI is easy to use, with clear instructions and responsive design.

This tool is useful for learning graph traversal algorithms, and it can be extended with features like interactive node/edge manipulation, weighted graphs, or more advanced visualization techniques for educational purposes.

Screenshots

1.Breadth-First Search(BFS) info :



Breadth First Search (BFS)

Definition

BFS is a graph traversal technique that explores all neighbors at the current depth before moving deeper.

Applications

- Finding shortest paths in unweighted graphs.
- Level-order traversal of trees.
- Social network analysis.

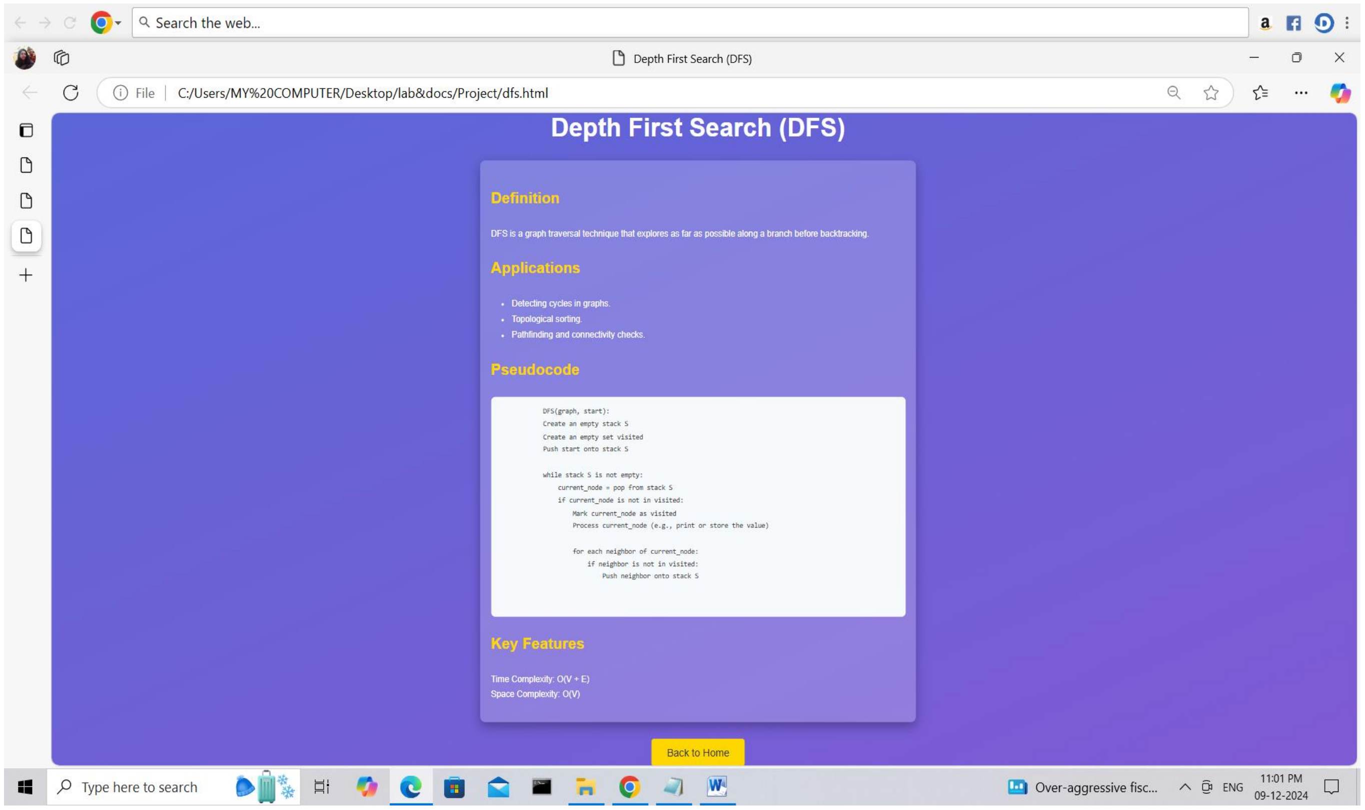
Pseudocode

```
BFS(graph, start):
    create a queue Q
    mark start as visited
    enqueue start into Q
    while Q is not empty:
        node = dequeue(Q)
        for each neighbor in graph[node]:
            if neighbor is not visited:
                mark neighbor as visited
                enqueue neighbor
```

Key Features

Time Complexity: O(V + E)
Space Complexity: O(V)

2. Depth-First Seatch(DFS) info :



Depth First Search (DFS)

Definition

DFS is a graph traversal technique that explores as far as possible along a branch before backtracking.

Applications

- Detecting cycles in graphs.
- Topological sorting.
- Pathfinding and connectivity checks.

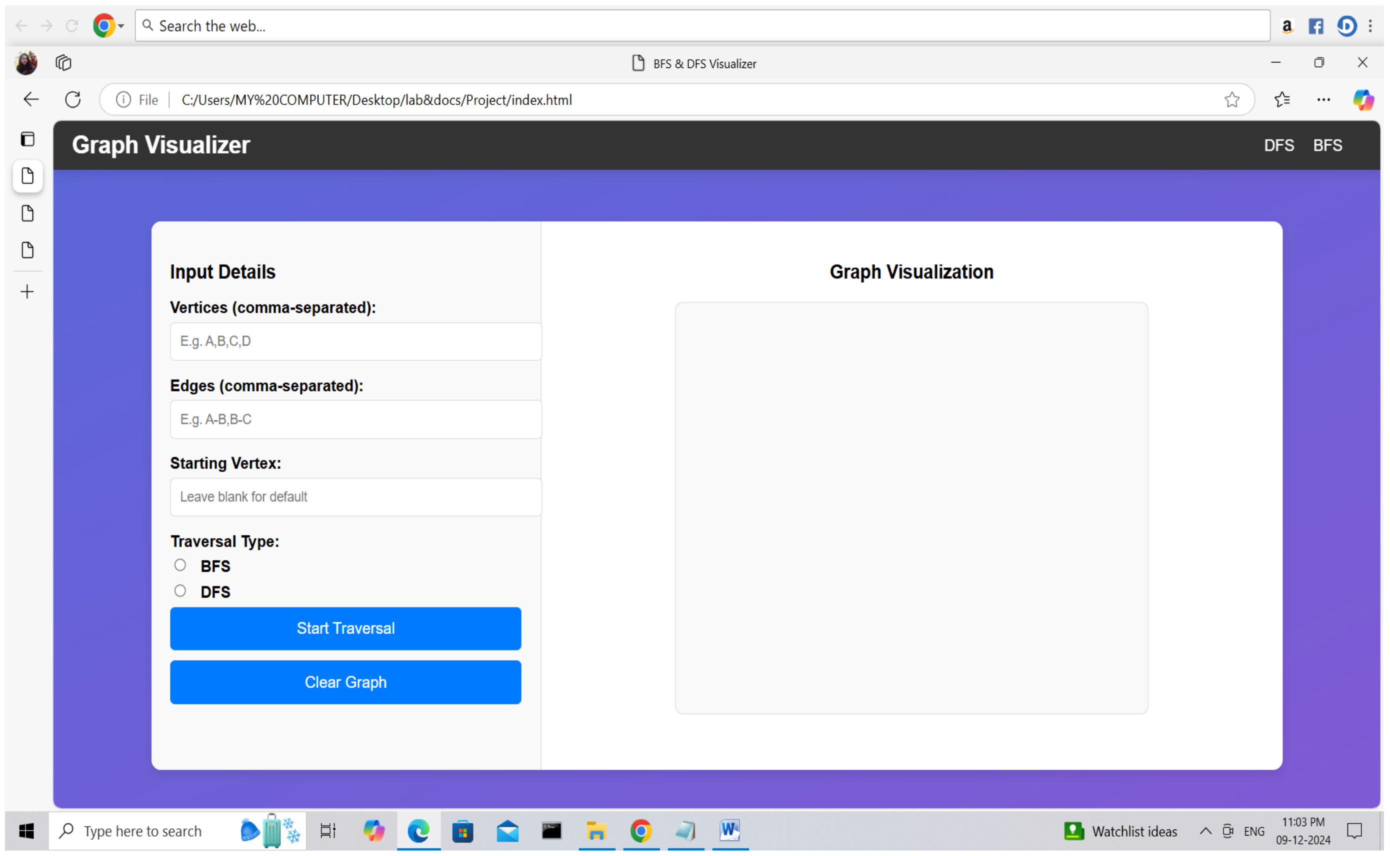
Pseudocode

```
DFS(graph, start):
Create an empty stack S
Create an empty set visited
Push start onto stack S

while stack S is not empty:
    current_node = pop from stack S
    if current_node is not in visited:
        Mark current_node as visited
        Process current_node (e.g., print or store the value)

        for each neighbor of current_node:
            if neighbor is not in visited:
                Push neighbor onto stack S
```

Key Features

Time Complexity: O(V + E)
Space Complexity: O(V)

3.Landing Page



C:/Users/MY%20COMPUTER/Desktop/lab&docs/Project/index.html

Graph Visualizer DFS BFS

Input Details

Vertices (comma-separated): E.g. A,B,C,D

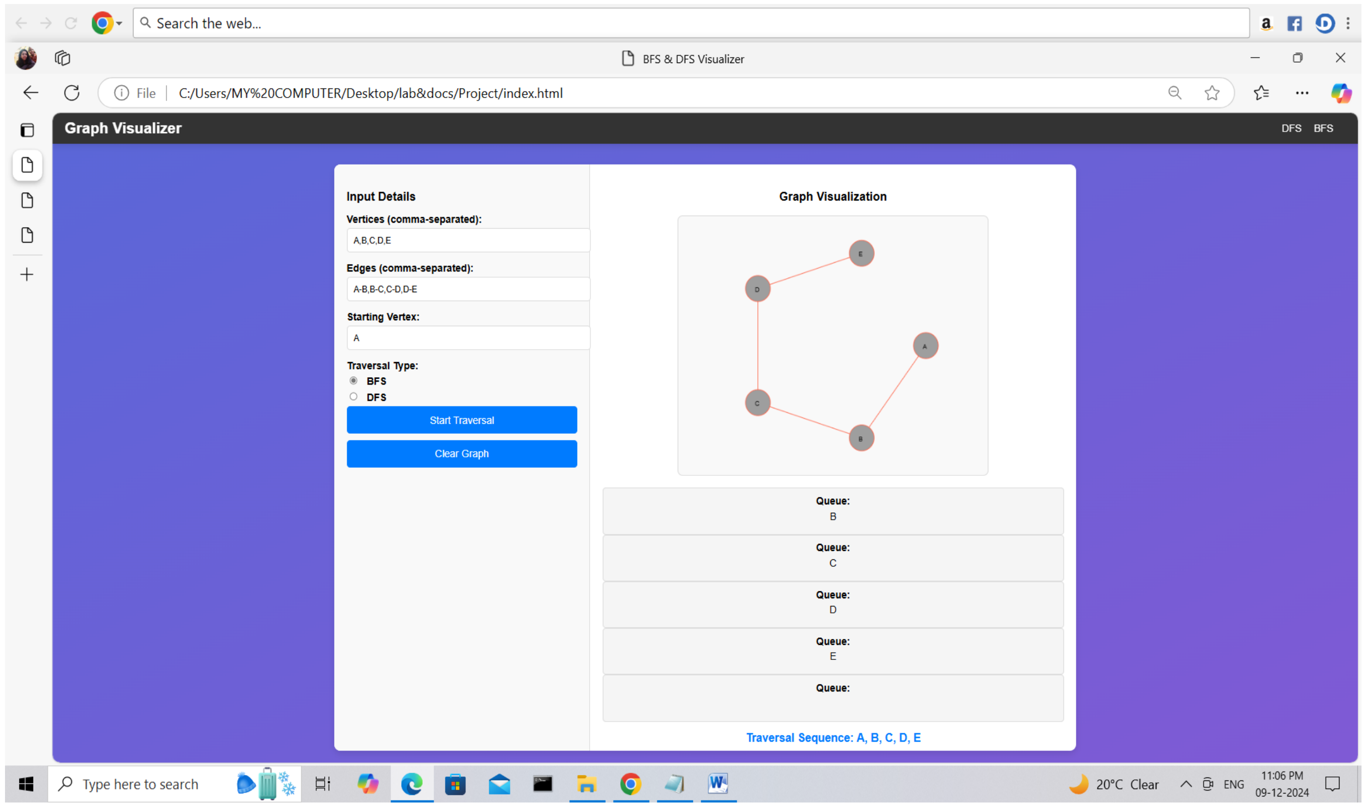
Edges (comma-separated): E.g. A-B,B-C

Starting Vertex: Leave blank for default

Traversal Type: BFS DFS

Graph Visualization

4. BFS Traversal :



C:/Users/MY%20COMPUTER/Desktop/lab&docs/Project/index.html

Graph Visualizer DFS BFS

Input Details

Vertices (comma-separated): A,B,C,D,E

Edges (comma-separated): A-B,B-C,C-D,D-E

Starting Vertex: A

Traversal Type: [✓ BFS] DFS

Graph Visualization

Queue: B

Queue: C

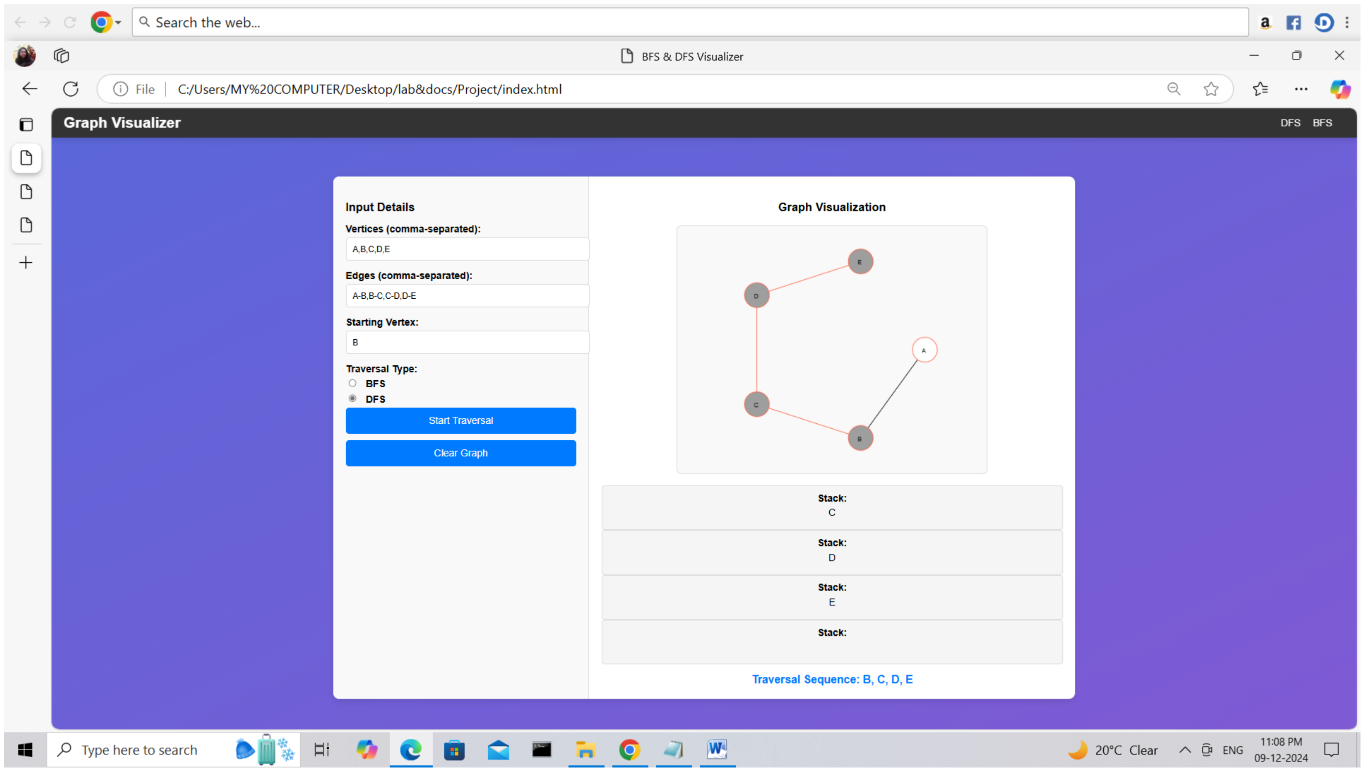
Queue: D

Queue: E

Queue:

Traversal Sequence: A, B, C, D, E

6 DFS Traversal :



Input Details

Vertices (comma-separated): A,B,C,D,E

Edges (comma-separated): A-B,B-C,C-D,D-E

Starting Vertex: B

Traversal Type: BFS [✓ DFS]

Stack: C

Stack: D

Stack: E

Stack:

Traversal Sequence: B, C, D, E

References :

1. https://www.w3schools.com/
2. https://www.tutorialspoint.com/index.htm
3. https://www.geeksforgeeks.org/