Lesson 03 Demo 05

Traversing a Graph

Objective: To demonstrate the graph traversal using JavaScript

Tools required: Visual Studio Code and Node.js

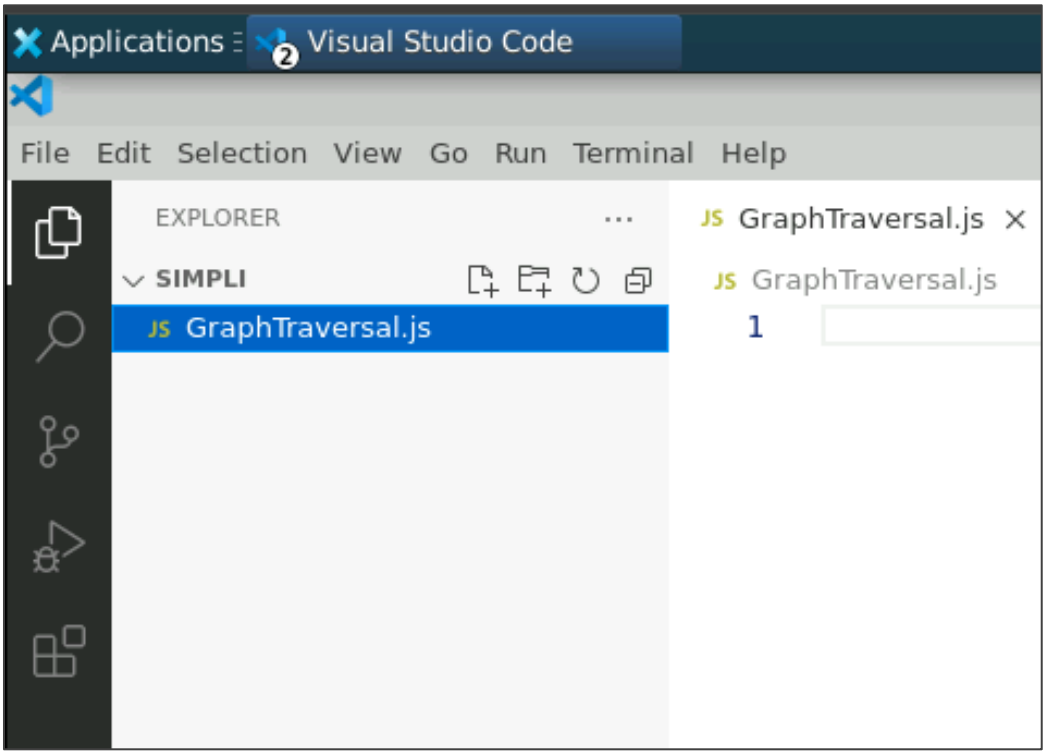
Prerequisites: Basic understanding of data structures and JavaScript

Steps to be followed:

1. Create and execute the JS file

Step 1: Create and execute JS file

1.1 Open the Visual Studio Code editor and create a JavaScript file named **GraphTraversal.js**



1.2 Write the code given below in the **GraphTraversal.js** file:

```
function Graph() {
    this.vertices = [];
    this.adjacencyList = new Map();
}

// Method to add a vertex
Graph.prototype.addVertex = function(vertex) {
    this.vertices.push(vertex);
    this.adjacencyList.set(vertex, []);
};

// Method to add an edge
Graph.prototype.addEdge = function(vertex1, vertex2) {
    this.adjacencyList.get(vertex1).push(vertex2);
    this.adjacencyList.get(vertex2).push(vertex1); // If the graph is undirected
};

// Method for depth-first traversal
Graph.prototype.depthFirstTraversal = function(startVertex, visited = new Set()) {
    if (!this.vertices.includes(startVertex) || visited.has(startVertex)) {
        return;
    }

    console.log(`Visited: ${startVertex}`);
    visited.add(startVertex);

    const neighbors = this.adjacencyList.get(startVertex);
    for (const neighbor of neighbors) {
        this.depthFirstTraversal(neighbor, visited);
    }
};

// Creating graph instance
const graph = new Graph();

// Adding vertices
['A', 'B', 'C', 'D', 'E', 'F'].forEach(vertex => graph.addVertex(vertex));
```

```
// Adding edges
graph.addEdge('A', 'B');
graph.addEdge('A', 'C');
graph.addEdge('B', 'D');
graph.addEdge('B', 'E');
graph.addEdge('C', 'F');

// Perform depth-first traversal
console.log('\nDepth-First Traversal:');
graph.depthFirstTraversal('A');
```

JS GraphTraversal.js > ...

```
function Graph() {
    this.vertices = [];
    this.adjacencyList = new Map();
}

// Method to add a vertex
Graph.prototype.addVertex = function(vertex) {
    this.vertices.push(vertex);
    this.adjacencyList.set(vertex, []);
};

// Method to add an edge
Graph.prototype.addEdge = function(vertex1, vertex2) {
    this.adjacencyList.get(vertex1).push(vertex2);
    this.adjacencyList.get(vertex2).push(vertex1); // If the graph is undirected
};

```

```
// Method for depth-first traversal
Graph.prototype.depthFirstTraversal = function(startVertex, visited = new Set()) {
    if (!this.vertices.includes(startVertex) || visited.has(startVertex)) {
        return;
    }

    console.log(`Visited: ${startVertex}`);
    visited.add(startVertex);

    const neighbors = this.adjacencyList.get(startVertex);
    for (const neighbor of neighbors) {
        this.depthFirstTraversal(neighbor, visited);
    }
};

```

```
33  // Creating graph instance
34  const graph = new Graph();
35
36  // Adding vertices
37  ['A', 'B', 'C', 'D', 'E', 'F'].forEach(vertex => graph.addVertex(vertex));
38
39  // Adding edges
40  graph.addEdge('A', 'B');
41  graph.addEdge('A', 'C');
42  graph.addEdge('B', 'D');
43  graph.addEdge('B', 'E');
44  graph.addEdge('C', 'F');
45
46  // Perform depth-first traversal
47  console.log('\nDepth-First Traversal:');
48  graph.depthFirstTraversal('A');
49
```

1.3 Save the file and execute it in the terminal using the command given below:
node GraphTraversal.js

```
30      }
31  };
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
priyanshurajsim@ip-172-31-25-35:~/Downloads/Simpli$ ls
GraphTraversal.js
priyanshurajsim@ip-172-31-25-35:~/Downloads/Simpli$ node GraphTraversal.js

Depth-First Traversal:
Visited: A
Visited: B
Visited: D
Visited: E
Visited: C
Visited: F
priyanshurajsim@ip-172-31-25-35:~/Downloads/Simpli$
```

This example demonstrates the graph traversal using depth-first traversal in JavaScript.

By following these steps, you have successfully implemented and executed the process of traversing a graph.