

## Lesson 03 Demo 05

### Traversing a Graph

**Objective:** To demonstrate graph traversal using depth-first search (DFS) in JavaScript and an adjacency list to illustrate how graph exploration works programmatically

**Tools required:** Visual Studio Code and Node.js

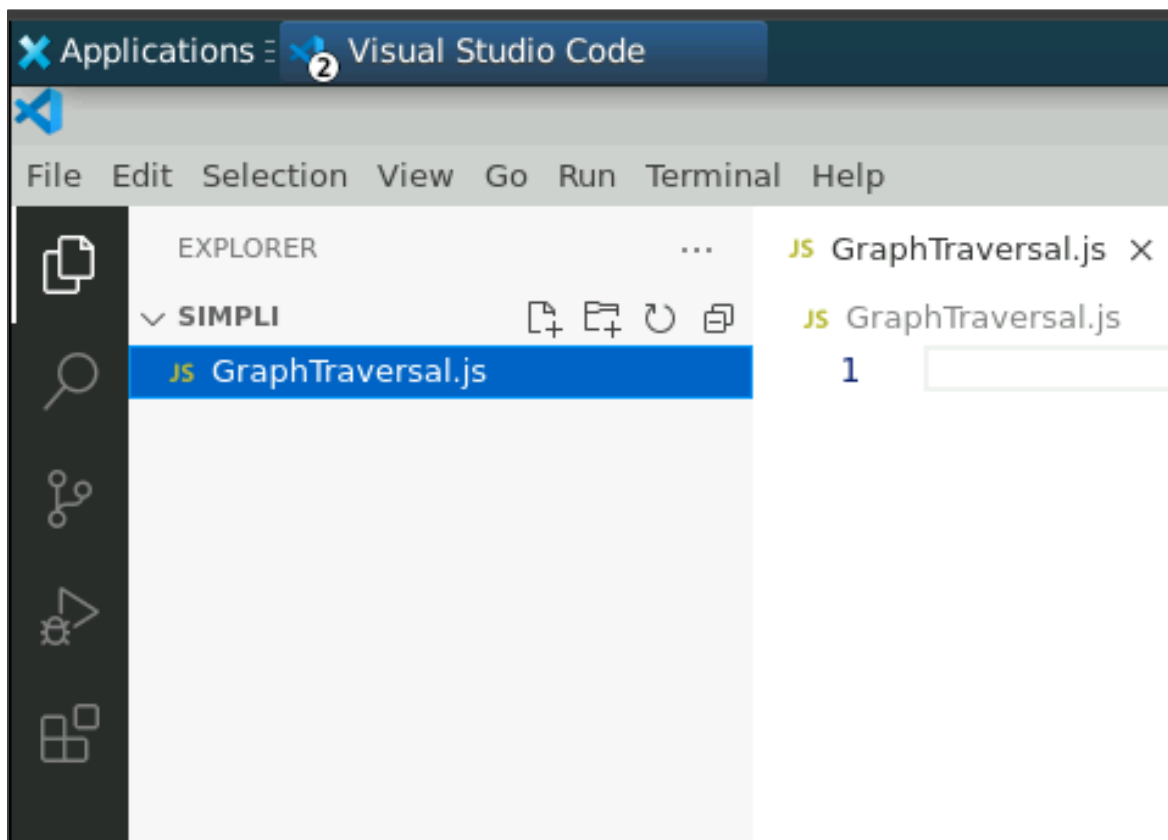
**Prerequisites:** A basic understanding of data structures and JavaScript

Steps to be followed:

1. Create a JavaScript file and execute it

#### Step 1: Create a JavaScript file and execute it

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



1.2 Add the following code to the 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 > ...
1  function Graph() {
2    this.vertices = [];
3    this.adjacencyList = new Map();
4  }
5
6  // Method to add a vertex
7  Graph.prototype.addVertex = function(vertex) {
8    this.vertices.push(vertex);
9    this.adjacencyList.set(vertex, []);
10 };
11
12 // Method to add an edge
13 Graph.prototype.addEdge = function(vertex1, vertex2) {
14   this.adjacencyList.get(vertex1).push(vertex2);
15   this.adjacencyList.get(vertex2).push(vertex1); // If the graph is undirected
16 };
17
```

```
18 // Method for depth-first traversal
19 Graph.prototype.depthFirstTraversal = function(startVertex, visited = new Set()) {
20   if (!this.vertices.includes(startVertex) || visited.has(startVertex)) {
21     return;
22   }
23
24   console.log(`Visited: ${startVertex}`);
25   visited.add(startVertex);
26
27   const neighbors = this.adjacencyList.get(startVertex);
28   for (const neighbor of neighbors) {
29     this.depthFirstTraversal(neighbor, visited);
30   }
31 };
32
```

```

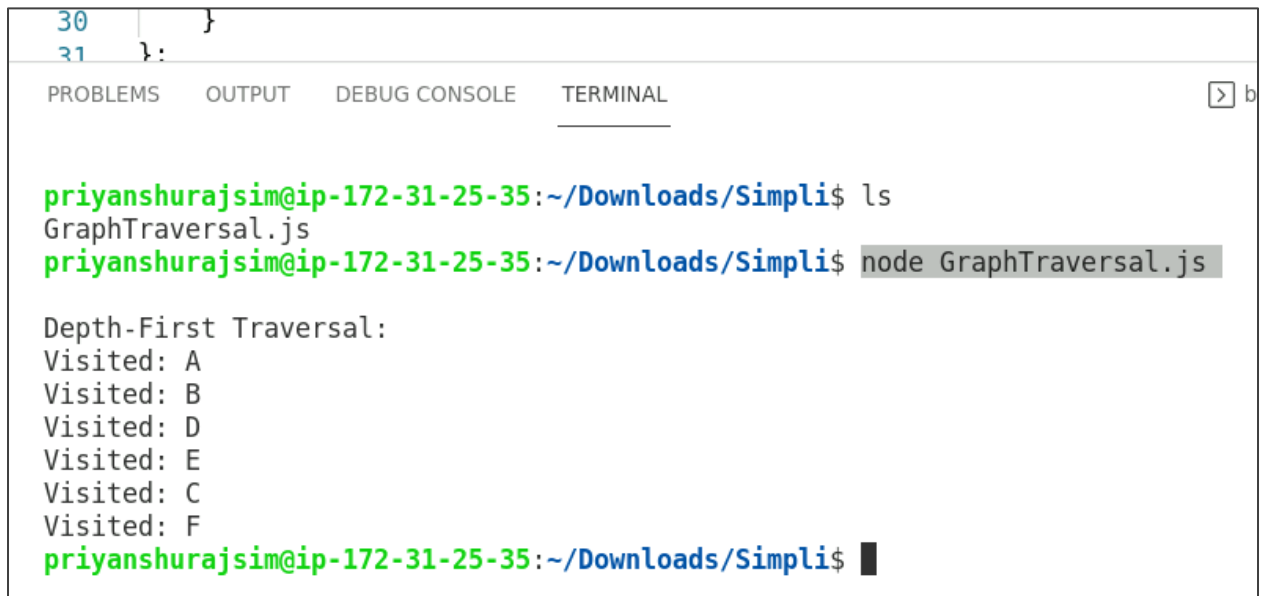
33 // Creating graph instance
34 const graph = new Graph();
35
36 // Adding vertices
37 ['A', 'B', 'C', 'D', 'E', 'F'].forEach(vertex => graph.addVertex(vertex));
38
39 // Adding edges
40 graph.addEdge('A', 'B');
41 graph.addEdge('A', 'C');
42 graph.addEdge('B', 'D');
43 graph.addEdge('B', 'E');
44 graph.addEdge('C',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

```

1.3 Press **Ctrl + S** to save the file and execute it in the **TERMINAL** using the commands given below:

**ls**

**node GraphTraversal.js**



The screenshot shows a code editor with a file named `GraphTraversal.js` containing the JavaScript code from the previous block. The **TERMINAL** tab is active, showing the following commands and output:

```

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$

```

By following these steps, you have successfully implemented a graph and performed a depth-first traversal using JavaScript. By creating a simple yet effective graph representation through vertices and edges and utilizing the depth-first traversal method, you have explored how to navigate through the graph systematically.