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

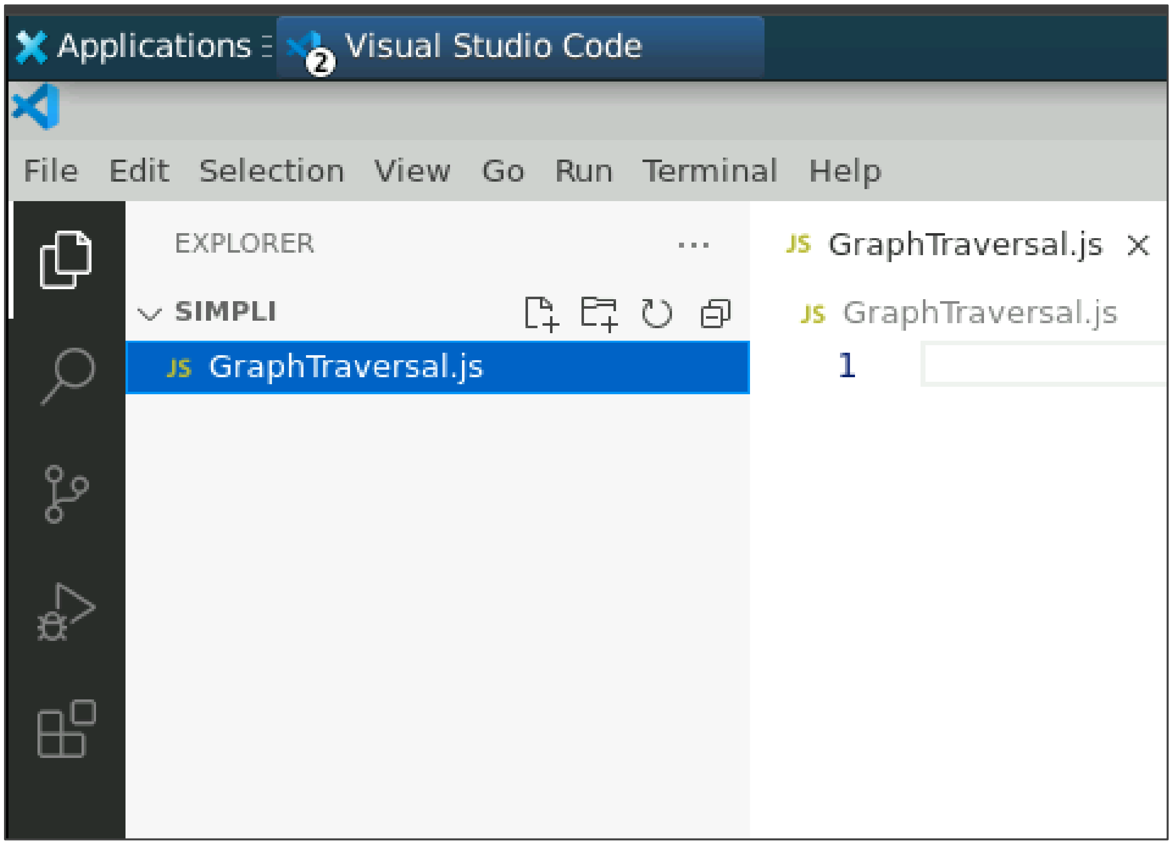
Prerequisites: A basic understanding of data structures and JavaScript

Steps to be followed:

1. Create a JavaScript file and execute it

Step 1: Create a JavaScript file and execute it

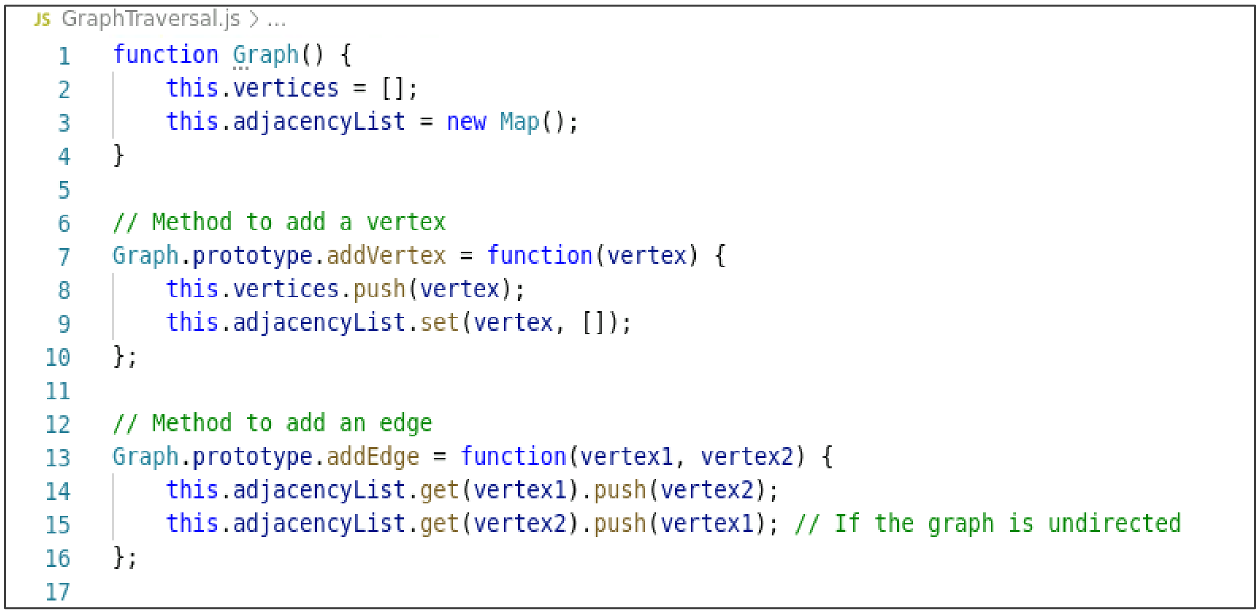
1.1 Open the Visual Studio Code editor and create a JavaScript file named **GraphTraversal.js**



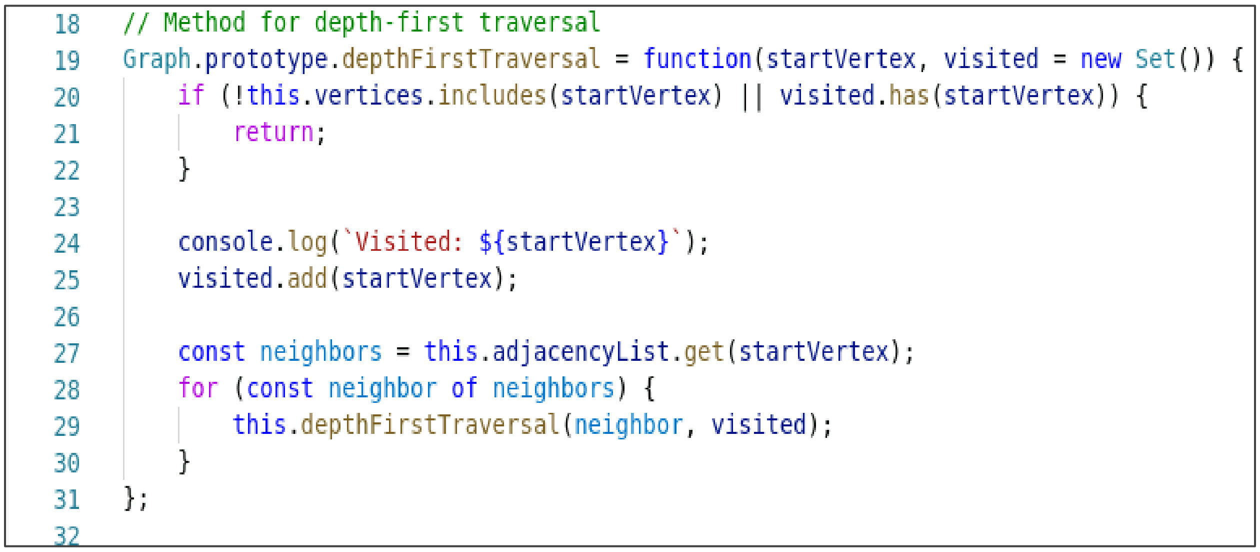
1.2 Add the following code to the file:

```
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));
```

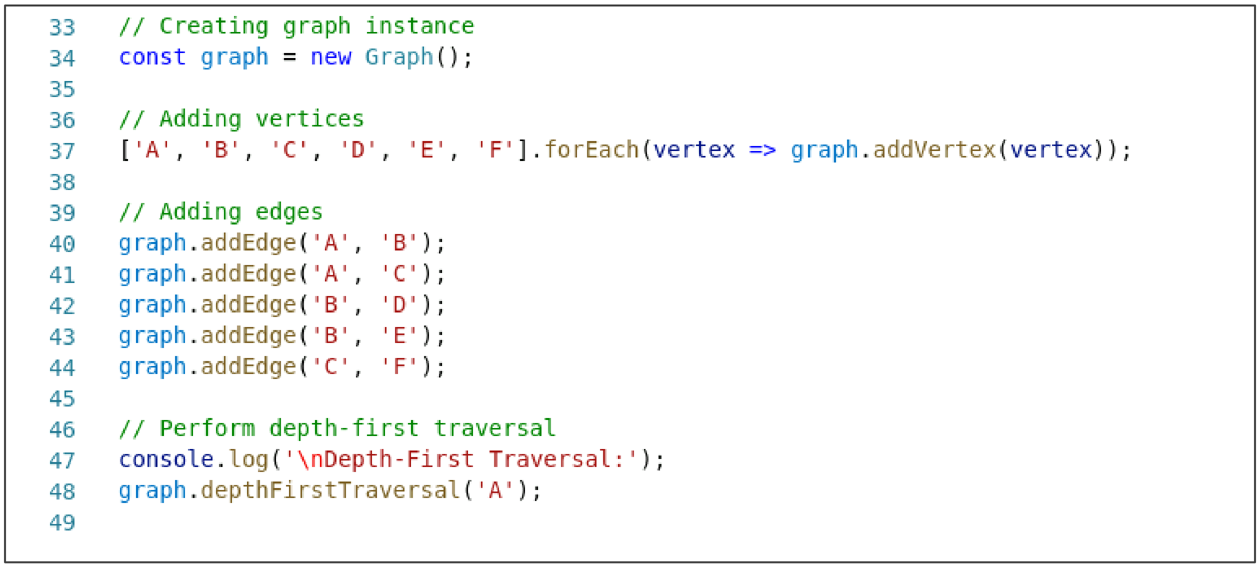
```
// Adding edges
graph.addEdge('A', 'B');
graph.addEdge('A', 'C');
graph.addEdge('B', 'D');
graph.addEdge('B', 'E');
graph.addEdge('C', 'F');

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



```javascript
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
};

```

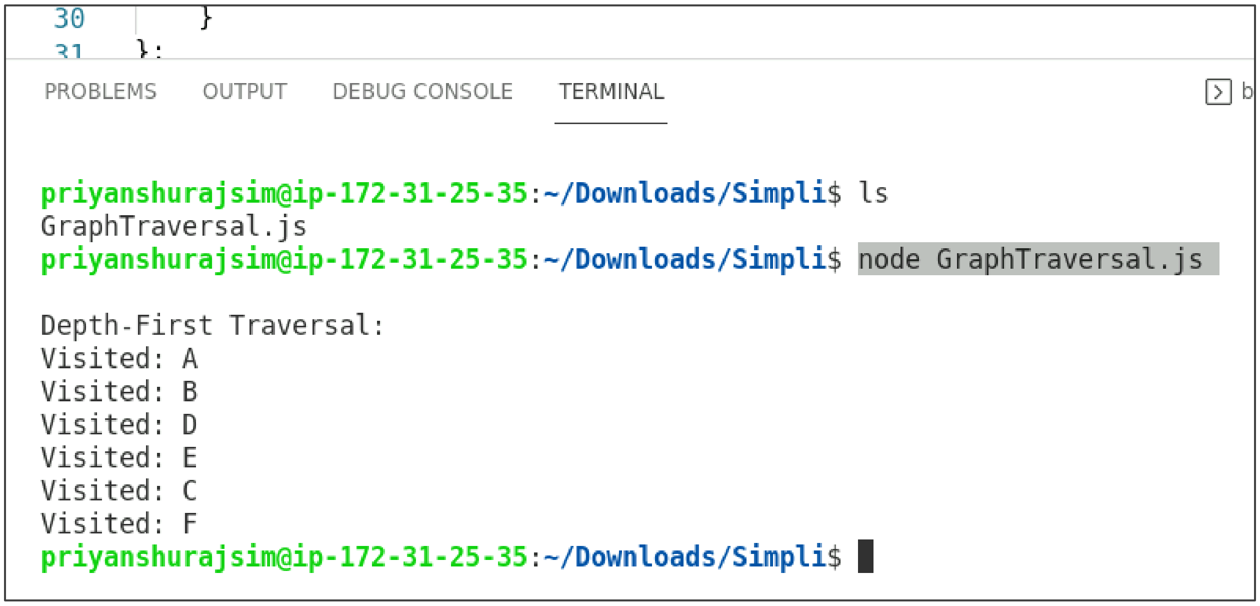


```javascript
// Method for depth-first traversal
Graph.prototype.depthFirstTraversal = function(startVertex, visited = new Set()) {
    if (!this.vertices.includes(startVertex) || visited.has(startVertex)) {
        return;
    }

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

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

```



```javascript
// 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');

```

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



```
        }
    };
```

```
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.