

```

package DAY7;
import java.util.*;
public class BFS_GRAPH {

//Class representing a graph

private int vertices; // Number of vertices
private LinkedList<Integer> adjacencyList[]; // Adjacency list

// Constructor
BFS_GRAPH (int vertices) {
    this.vertices = vertices;
    adjacencyList = new LinkedList[vertices];
    for (int i = 0; i < vertices; i++) {
        adjacencyList[i] = new LinkedList<>();
    }
}

// Method to add an edge to the graph
void addEdge(int source, int destination) {
    adjacencyList[source].add(destination);
    // For undirected graph, add the reverse edge as well
    // adjacencyList[destination].add(source);
}

// Method to perform BFS traversal from a given source node
void BFS(int startNode) {
    boolean visited[] = new boolean[vertices]; // Array to keep track of visited nodes
    LinkedList<Integer> queue = new LinkedList<>(); // Queue for BFS

    visited[startNode] = true; // Mark the starting node as visited
    queue.add(startNode); // Enqueue the starting node

    while (!queue.isEmpty()) {
        startNode = queue.poll(); // Dequeue a node
        System.out.print(startNode + " "); // Print the dequeued node

        // Get all adjacent vertices of the dequeued node
        // If an adjacent has not been visited, mark it visited and enqueue it
        Iterator<Integer> iterator = adjacencyList[startNode].listIterator();
        while (iterator.hasNext()) {
            int nextNode = iterator.next();
            if (!visited[nextNode]) {
                visited[nextNode] = true;
                queue.add(nextNode);
            }
        }
    }
}

// Main method to test the BFS implementation
public static void main(String args[]) {
    BFS_GRAPH graph = new BFS_GRAPH(6);

    // Adding edges to the graph
    graph.addEdge(0, 1);
    graph.addEdge(0, 2);
}

```

```
graph.addEdge(1, 3);
graph.addEdge(2, 4);
graph.addEdge(3, 5);
graph.addEdge(4, 5);
```

```
System.out.println("BFS traversal starting from node 0:");
graph.BFS(0);
```

```
}
}
```

```
39 // If an adjacent has not been visited, mark it visited and enqueue it
40 Iterator<Integer> iterator = adjacencyList[startNode].listIterator();
41 while (iterator.hasNext()) {
42     int nextNode = iterator.next();
43     if (!visited[nextNode]) {
44         visited[nextNode] = true;
45         queue.add(nextNode);
46     }
47 }
48 }
49 }
50
51 // Main method to test the BFS implementation
52 public static void main(String args[]) {
53     BFS_GRAPH graph = new BFS_GRAPH(6);
54
55     // Adding edges to the graph
56     graph.addEdge(0, 1);
57     graph.addEdge(0, 2);
58     graph.addEdge(1, 3);
59     graph.addEdge(2, 4);
60     graph.addEdge(3, 5);
61     graph.addEdge(4, 5);
62
63     System.out.println("BFS traversal starting from node 0:");
64     graph.BFS(0);
65 }
66 }
67 }
```

DAY7

- BFS_GRAPH
 - vertices : int
 - adjacencyList : LinkedList<Integer>[]
 - BFS_GRAPH(int)
 - addEdge(int, int) : void
 - BFS(int) : void
 - main(String[]) : void

Markers Properties Terminal Console Coverage

<terminated> BFS_GRAPH [Java Application] C:\Users\Nikita\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Jun 4, 2024, 11:53:34 AM - 11:53:34 AM)

BFS traversal starting from node 0:

0 1 2 3 4 5