**Task 4: Graph Edge Addition Validation**

**Given a directed graph, write a function that adds an edge between two nodes and then checks if the graph still has no cycles. If a cycle is created, the edge should not be added.**

**import** java.util.\*;

**public** **class** DirectedGraph {

**private** **final** Map<Integer, List<Integer>> adjacencyList;

**public** DirectedGraph() {

**this**.adjacencyList = **new** HashMap<>();

}

// Method to add a node to the graph

**public** **void** addNode(**int** node) {

adjacencyList.putIfAbsent(node, **new** ArrayList<>());

}

// Method to add an edge to the graph

**public** **boolean** addEdge(**int** from, **int** to) {

**if** (!adjacencyList.containsKey(from) || !adjacencyList.containsKey(to)) {

**throw** **new** IllegalArgumentException("Both nodes must exist in the graph.");

}

// Add the edge temporarily

adjacencyList.get(from).add(to);

// Check if a cycle is introduced

**if** (hasCycle()) {

// If a cycle is detected, remove the edge and return false

adjacencyList.get(from).remove((Integer) to);

**return** **false**;

}

// No cycle detected, the edge is added successfully

**return** **true**;

}

// Method to check if the graph has a cycle

**private** **boolean** hasCycle() {

Set<Integer> visited = **new** HashSet<>();

Set<Integer> recStack = **new** HashSet<>();

**for** (Integer node : adjacencyList.keySet()) {

**if** (hasCycleUtil(node, visited, recStack)) {

**return** **true**;

}

}

**return** **false**;

}

// Utility method for cycle detection using DFS

**private** **boolean** hasCycleUtil(**int** node, Set<Integer> visited, Set<Integer> recStack) {

**if** (recStack.contains(node)) {

**return** **true**;

}

**if** (visited.contains(node)) {

**return** **false**;

}

visited.add(node);

recStack.add(node);

**for** (Integer neighbor : adjacencyList.get(node)) {

**if** (hasCycleUtil(neighbor, visited, recStack)) {

**return** **true**;

}

}

recStack.remove(node);

**return** **false**;

}

**public** **static** **void** main(String[] args) {

DirectedGraph graph = **new** DirectedGraph();

// Adding nodes

graph.addNode(1);

graph.addNode(2);

graph.addNode(3);

graph.addNode(4);

// Adding edges

System.***out***.println(graph.addEdge(1, 2)); // true

System.***out***.println(graph.addEdge(2, 3)); // true

System.***out***.println(graph.addEdge(3, 4)); // true

// Adding an edge that creates a cycle

System.***out***.println(graph.addEdge(4, 1)); // false (cycle detected)

}

}