

Graphs

Memory Diagram

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
}

```

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

```

    Graph<String> graph = new Graph<>();
    graph.addEdge("BUF", "WDC");
    graph.addEdge("WDC", "JFK");
    graph.addEdge("TOR", "BUF");
    ArrayList<String> path1 = new ArrayList<>(
        Arrays.asList("BUF", "WDC", "JFK"));
    System.out.println(graph.validPath(path1));
    ArrayList<String> path2 = new ArrayList<>(
        Arrays.asList("JFK", "WDC", "BUF"));
    System.out.println(graph.validPath(path2));
}

```

Stack		Heap	
Name	Value	Name	Value

- Always start with the main method in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

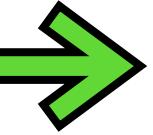
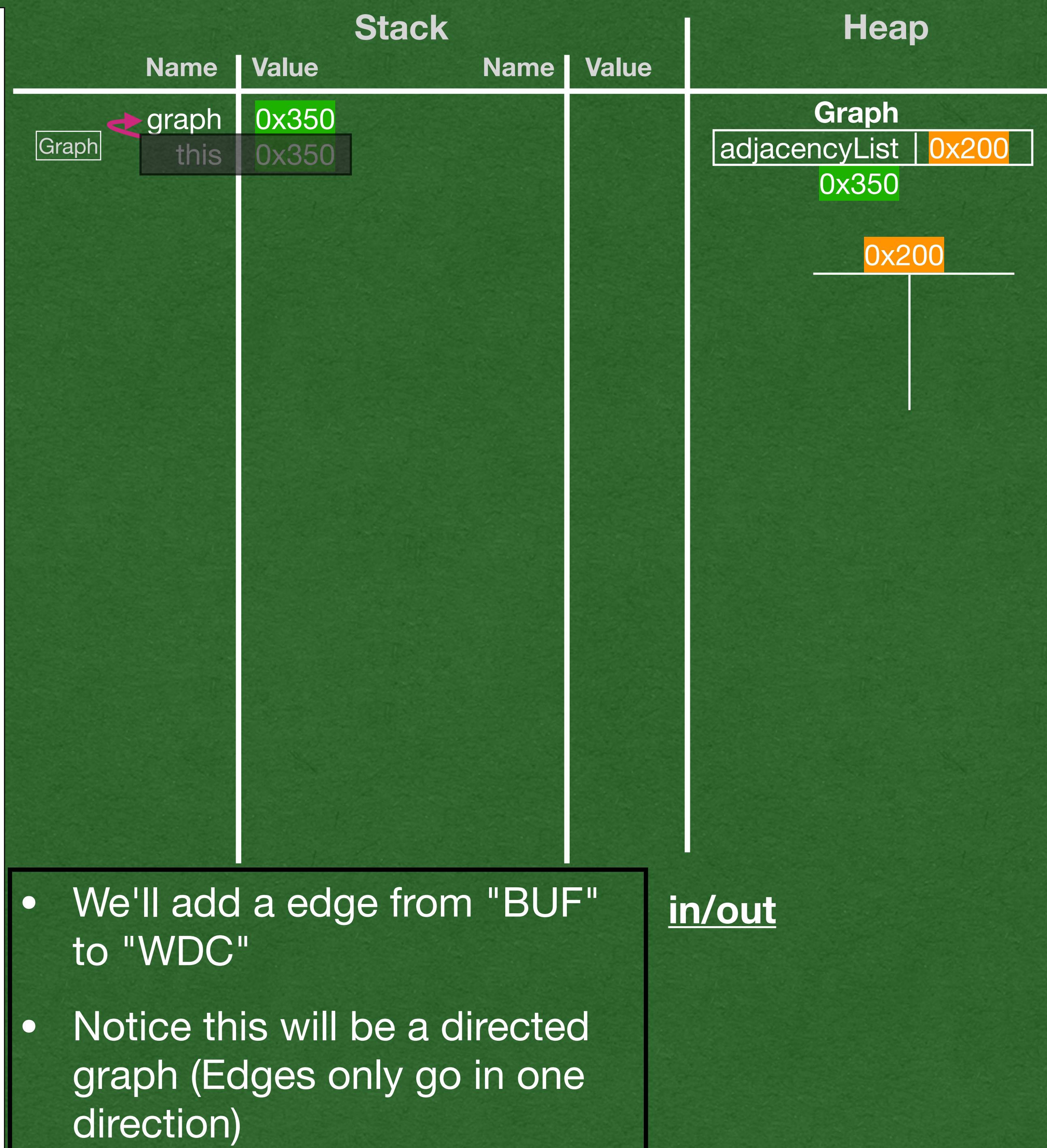
Stack		Heap	
Name	Value	Name	Value
graph	0x350	Graph	adjacencyList 0x200
this	0x350	adjacencyList	0x350
			0x200

- The Graph constructor initializes the adjacency list to a new HashMap
- in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

		Stack		Heap	
	Name	Value		Name	Value
Graph	graph	0x350	addEdge	adjacencyList	0x200
	this	0x350		0x350	0x350
	from	"BUF"		0x200	"BUF"
addEdge	to	"WDC"		"BUF"	0x480
	this	0x350			0x480
addNode	a	"BUF"			

- We haven't seen the node "BUF" yet
- Initialize it in the adjacency list

in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

		Stack		Heap	
	Name	Value		Name	Value
Graph	graph	0x350	addEdge	adjacencyList	0x200
	this	0x350		0x350	0x350
	from	"BUF"		"BUF"	0x480
	to	"WDC"		"WDC"	0x120
addNode	this	0x350	addNode	0x480	
	a	"BUF"		0x120	
addNode	this	0x350			
	a	"WDC"			

- We haven't seen the node "WDC" yet
- Initialize it in the adjacency list

in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
Graph	graph 0x350	Graph	adjacencyList 0x200
	this 0x350		0x350
addEdge	this 0x350	"BUF" 0x480	"BUF" 0x480
	from "BUF"	"WDC" 0x120	"WDC" 0x120
addNode	this 0x350		
	a "BUF"		
addNode	this 0x350		
	a "WDC"		
		0 "WDC"	0 "WDC"
		0x480	0x480
		0x120	0x120

- Add the edge into the adjacency list in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList( "JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

		Stack		Heap	
	Name	Value		Name	Value
Graph	graph	0x350		Graph	adjacencyList 0x200
	this	0x350		adjacencyList	0x350
	this	0x350			0x200
addEdge	from	"BUF"		"BUF" 0x480	
	to	"WDC"		"WDC" 0x120	
	this	0x350			
addNode	a	"BUF"			
	this	0x350			
addNode	a	"WDC"			
	this	0x350			
addEdge	from	"WDC"			
	to	"JFK"			
	this	0x350			
addNode	a	"WDC"			

- "WDC" already has an entry in the adjacency list
- Nothing to initialize

in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

		Stack		Heap	
	Name	Value		Name	Value
Graph	graph	0x350		Graph	adjacencyList 0x200
	this	0x350		adjacencyList	0x350
	from	"BUF"			0x200
	to	"WDC"			"BUF" 0x480
addEdge	this	0x350			"WDC" 0x120
	a	"BUF"			"JFK" 0x777
addNode	this	0x350			
	a	"WDC"			
addEdge	this	0x350			0x480
	from	"WDC"			0 "WDC"
	to	"JFK"			
addNode	this	0x350			0x120
	a	"WDC"			
addNode	this	0x350			0x777
	a	"JFK"			

- Create a new entry in the adjacency list for "JFK" in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList( "JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

		Stack		Heap	
	Name	Value		Name	Value
Graph	graph	0x350		Graph	adjacencyList 0x200
	this	0x350		adjacencyList	0x350
	from	"BUF"			0x200
	to	"WDC"			"BUF" 0x480
addEdge	this	0x350			"WDC" 0x120
	a	"BUF"			"JFK" 0x777
addNode	this	0x350			
	a	"WDC"			
addEdge	this	0x350			0x480
	from	"WDC"			0 "WDC"
	to	"JFK"			
addNode	this	0x350			0x120
	a	"WDC"			0 "JFK"
addNode	this	0x350			0x777
	a	"JFK"			

- Add the edge from "WDC" to "JFK" in the adjacency list

in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
Graph	graph 0x350	adjacencyList 0x200	adjacencyList 0x200
	this 0x350	0x350	"BUF" 0x480
	this 0x350	0x350	"WDC" 0x120
addEdge	from "BUF"	"JFK" 0x777	"JFK" 0x777
	to "WDC"	0x808	"TOR" 0x808
	this 0x350	0x480	0 "WDC"
addNode	a "BUF"	0x120	0 "JFK"
	this 0x350	0x777	0x808
addNode	a "WDC"	0 "BUF"	0 "TOR"
	this 0x350	0 "WDC"	
addEdge	from "WDC"		
	to "JFK"		
	this 0x350		
addNode	a "WDC"		
	this 0x350		
addNode	a "JFK"		
	this 0x350		
addEdge	from "TOR"		
	to "BUF"		
	this 0x350		
addNode	a "TOR"		

- Repeat again for the last edge in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
Graph	graph 0x350	Graph	adjacencyList 0x200
	this 0x350	adjacencyList 0x200	0x350
addEdge	this 0x350	"BUF" 0x480	"BUF" 0x480
	from "BUF"	"WDC" 0x120	"WDC" 0x120
	to "WDC"	"JFK" 0x777	"JFK" 0x777
addNode	this 0x350	"TOR" 0x808	"TOR" 0x808
	a "BUF"		
addNode	this 0x350		
	a "WDC"		
addEdge	this 0x350	0x480	0 "WDC"
	from "WDC"		
	to "JFK"		
addNode	this 0x350	0x120	0 "JFK"
	a "WDC"		
addNode	this 0x350	0x777	
	a "JFK"		
addEdge	this 0x350	0x808	0 "BUF"
	from "TOR"		
	to "BUF"		
addNode	this 0x350	0x196	0 "BUF"
	a "TOR"		

-
- Create a possible path in/out
 - This will be checked by our method

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
graph	graph 0x350	adjacencyList	adjacencyList 0x200
	this 0x350	0x350	"BUF" 0x480
addEdge	this 0x350	"WDC" 0x120	"WDC" 0x120
	from "BUF"	"JFK" 0x777	"JFK" 0x777
	to "WDC"	"TOR" 0x808	"TOR" 0x808
addNode	this 0x350	0x200	
	a "BUF"	0x480	
addNode	this 0x350	0x120	
	a "WDC"	0x777	
addEdge	this 0x350	0x808	
	from "WDC"	0x196	
	to "JFK"	0 "WDC"	
addNode	this 0x350	0 "JFK"	
	a "WDC"	0x480	
addNode	this 0x350	0x120	
	a "JFK"	0x777	
addEdge	this 0x350	0x808	
	from "TOR"	0 "BUF"	
	to "BUF"	1 "WDC"	
addNode	this 0x350	2 "JFK"	
	a "TOR"		

- Start the loop that will check if every edge in the path exists in the graph

in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
Name	Value	Name	Value
graph	0x350	this	0x350
this	0x350	a	"BUF"
this	0x350	path1	0x196
from	"BUF"	this	0x350
to	"WDC"	path	0x196
this	0x350	i	0
a	"BUF"	this	0x350
this	0x350	areConnect	
a	"WDC"	from	"BUF"
this	0x350	to	"WDC"
from	"WDC"		
to	"JFK"		
this	0x350		
a	"WDC"		
this	0x350		
a	"JFK"		
this	0x350		
from	"TOR"		
to	"BUF"		
this	0x350		
a	"TOR"		

- There is an edge between "BUF" and "WDC", so the method returns true
- Conditional in validPath is false

in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
graph	graph 0x350	this 0x350	adjacencyList 0x200
	this 0x350	a "BUF"	0x350
addEdge	this 0x350	path1 0x196	0x200
	from "BUF"	this 0x350	"BUF" 0x480
	to "WDC"	path 0x196	"WDC" 0x120
GraphNode	this 0x350	i 0	"JFK" 0x777
	a "BUF"	1 1	"TOR" 0x808
addNode	this 0x350	this 0x350	0x480
	a "WDC"	from 0x350	0 "WDC"
addEdge	this 0x350	to 0x350	0x120
	from "WDC"	from "JFK"	0 "JFK"
	to "JFK"	this 0x350	0x777
GraphNode	this 0x350	a "WDC"	0x808
	a "JFK"	this 0x350	0 "BUF"
addEdge	this 0x350	from 0x350	0x196
	from "TOR"	to 0x350	"BUF" 0x480
	to "BUF"	this 0x350	"WDC" 0x120
GraphNode	this 0x350	a "TOR"	"JFK" 0x777
	a "TOR"		"TOR" 0x808

- Increment i and check the next two nodes

in/out

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
Name	Value	Name	Value
graph	0x350	this	0x350
this	0x350	a	"BUF"
this	0x350	path1	0x196
from	"BUF"	this	0x350
to	"WDC"	path	0x196
this	0x350	i	0 1
a	"BUF"	this	0x350
this	0x350	from	"BUF"
a	"WDC"	to	"WDC"
this	0x350	this	0x350
from	"WDC"	from	"WDC"
to	"JFK"	to	"JFK"
this	0x350		
a	"WDC"		
this	0x350		
a	"JFK"		
this	0x350		
from	"TOR"		
to	"BUF"		
this	0x350		
a	"TOR"		

- There is an edge between "WDC" and "JFK", so the method returns true
- Conditional in validPath is false

in/out

adjacencyList	0x200
0x350	
0x200	
"BUF"	0x480
"WDC"	0x120
"JFK"	0x777
"TOR"	0x808
0x480	
0	"WDC"
0x120	
0	"JFK"
0x777	
0x808	
0	"BUF"
0x196	
0	"BUF"
1	"WDC"
2	"JFK"

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
Name	Value	Name	Value
graph	0x350	this	0x350
this	0x350	a	"BUF"
this	0x350	path1	0x196
from	"BUF"	this	0x350
to	"WDC"	path	0x196
this	0x350	[i]	-1
a	"BUF"	this	0x350
this	0x350	areConnect	
a	"WDC"	this	0x350
this	0x350	from	"BUF"
from	"WDC"	to	"WDC"
to	"JFK"	this	0x350
this	0x350	from	"WDC"
a	"WDC"	to	"JFK"
this	0x350	this	0x350
a	"JFK"	from	"WDC"
this	0x350	to	"JFK"
from	"TOR"	this	0x350
to	"BUF"	from	"WDC"
this	0x350	to	"JFK"
a	"TOR"		

- The loop condition is false
- We made it through the loop without returning false, therefor all the edges exist and we can return true

in/out

Graph	
adjacencyList	0x200
0x350	0x200
"BUF"	0x480
"WDC"	0x120
"JFK"	0x777
"TOR"	0x808
0	"WDC"
0x120	0
0	"JFK"
0x777	
0x808	0
0	"BUF"
0x196	0
1	"WDC"
2	"JFK"

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
Name	Value	Name	Value
graph	0x350	this	0x350
Graph	graph	a	"BUF"
this	0x350	path1	0x196
this	0x350	this	0x350
addEdge	from	path	0x196
this	"BUF"	i	-1
this	0x350	this	0x350
to	"WDC"	from	"BUF"
addNode	this	to	"WDC"
this	0x350	areConnect	0x350
a	"BUF"	this	0x350
addNode	this	from	"WDC"
a	"WDC"	to	"JFK"
addEdge	this	areConnect	0x350
from	"WDC"	this	0x350
to	"JFK"	from	"WDC"
addNode	this	to	"JFK"
a	"WDC"	this	0x350
addNode	this	from	"JFK"
a	"JFK"	to	"WDC"
addEdge	this	this	0x350
from	"TOR"	from	"WDC"
to	"BUF"	to	"JFK"
addNode	this	this	0x350
a	"TOR"	from	"JFK"
a	"TOR"	to	"BUF"

- Print true to the screen

in/out
true

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
graph	graph 0x350	this	adjacencyList 0x200
	this 0x350	a	0x350
addEdge	this 0x350	path1	"BUF"
	from "BUF"	this	0x196
	to "WDC"	path	0x350
addNode	this 0x350	[i]	0x196
	a "BUF"	this	0x350
addNode	this 0x350	areConnect	0x350
	a "WDC"	from	"BUF"
addEdge	this 0x350	to	0x350
	from "WDC"	this	"WDC"
	to "JFK"	from	0x350
addNode	this 0x350	to	"JFK"
	a "WDC"	path2	0x296
addNode	this 0x350		
	a "JFK"		
addEdge	this 0x350		
	from "TOR"		
	to "BUF"		
addNode	this 0x350		
	a "TOR"		

- Create a new ArrayList with another possible path to check
- in/out
true

0x200	"BUF"
0x480	"WDC"
0x120	"JFK"
0x777	"TOR"
0x808	
0x480	0 "WDC"
0x120	0 "JFK"
0x777	0x808
0x808	0 "BUF"
0x196	0 "BUF"
0x196	1 "WDC"
0x196	2 "JFK"
0x296	0 "JFK"
0x296	1 "WDC"
0x296	2 "BUF"

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
graph	graph 0x350	this 0x350	Graph adjacencyList 0x200
	this 0x350	a "BUF"	0x350
addEdge	this 0x350	path1 0x196	0x200
	from "BUF"	this 0x350	"BUF" 0x480
	to "WDC"	path 0x196	"WDC" 0x120
addNode	this 0x350	[i 0 1]	"JFK" 0x777
	a "BUF"	this 0x350	"TOR" 0x808
addNode	this 0x350	areConnect 0x296	
	a "WDC"	from 0x350	0x480
addEdge	this 0x350	to 0x350	0 "WDC"
	from "WDC"	from 0x350	0x120
	to "JFK"	to 0x350	0 "JFK"
addNode	this 0x350	path2 0x296	0x777
	a "WDC"	this 0x350	0x808
addNode	this 0x350	path 0x296	0 "BUF"
	a "JFK"	[i 0 1]	0x196
addEdge	this 0x350	this 0x350	0 "BUF"
	from "TOR"	path 0x296	1 "WDC"
	to "BUF"	[i 0 1]	2 "JFK"
addNode	this 0x350	this 0x350	0x296
	a "TOR"	path 0x296	0 "JFK"

- Valid path with check if the values at indices 0 and 1 are connected
- in/out
true

0 "BUF"
1 "WDC"
2 "JFK"
0 "BUF"
1 "WDC"
2 "BUF"

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
graph	graph 0x350	this 0x350	adjacencyList 0x200
	this 0x350	a "BUF"	0x350
addEdge	this 0x350	path1 0x196	0x200
	from "BUF"	this 0x350	"BUF" 0x480
	to "WDC"	path 0x196	"WDC" 0x120
GraphNode	this 0x350	[i] 0	"JFK" 0x777
	a "BUF"	this 0x350	"TOR" 0x808
GraphNode	this 0x350	areConnect	
	a "WDC"	from "BUF"	0x480
GraphNode	this 0x350	to "WDC"	0 "WDC"
	from "WDC"	this 0x350	0x120
GraphNode	to "JFK"	from "WDC"	0 "JFK"
	this 0x350	to "JFK"	0x777
GraphNode	a "WDC"	path2 0x296	0x808
	this 0x350	this 0x350	0 "BUF"
GraphNode	a "JFK"	path 0x296	0x196
	this 0x350	[i] 0	"BUF" 0x480
GraphNode	from "TOR"	this 0x350	"WDC" 0x120
	to "BUF"	from "JFK"	"JFK" 0x777
GraphNode	this 0x350	to "WDC"	"TOR" 0x808
	a "TOR"		

- areConnect returns false

in/out
true

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
		Name	Value
graph	graph	graph	0x350
	this	this	0x350
addEdge	this	path1	0x196
	from	this	0x350
	to	path	0x196
addNode	this	[i]	0 1
	a	this	0x350
addNode	this	from	"BUF"
	a	to	"WDC"
addEdge	this	this	0x350
	from	from	"WDC"
	to	to	"JFK"
addNode	this	path2	0x296
	a	this	0x350
addNode	this	path	0x296
	a	[i]	0 0
addEdge	this	this	0x350
	from	from	"JFK"
	to	to	"WDC"
addNode	this		
	a		

- We reach a return statement
 - The entire stack frame ends and false is returned
- in/out
true

0x296	0	"JFK"
	1	"WDC"
	2	"BUF"

0x196	0	"BUF"
	1	"WDC"
	2	"JFK"

0x120	0	"JFK"
	1	"WDC"

0x200	"BUF"	0x480
	"WDC"	0x120
	"JFK"	0x777
	"TOR"	0x808
0	"WDC"	0 "WDC"

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
graph	graph 0x350	this 0x350	adjacencyList 0x200
	this 0x350	a "BUF"	0x350
addEdge	this 0x350	path1 0x196	0x200
	from "BUF"	this 0x350	"BUF" 0x480
	to "WDC"	path 0x196	"WDC" 0x120
GraphNode	this 0x350	[i] 0	"JFK" 0x777
	a "BUF"	this 0x350	"TOR" 0x808
GraphNode	this 0x350	areConnect	
	a "WDC"	from "BUF"	0x480
GraphNode	this 0x350	to "WDC"	0 "WDC"
	from "WDC"	this 0x350	0x120
GraphNode	to "JFK"	from "WDC"	0 "JFK"
	this 0x350	to "JFK"	0x777
GraphNode	a "WDC"	path2 0x296	0x808
	this 0x350	this 0x350	0 "BUF"
GraphNode	a "JFK"	path 0x296	0x196
	this 0x350	[i] 0	"BUF" 0x480
GraphNode	from "TOR"	this 0x350	"WDC" 0x120
	to "BUF"	from "JFK"	"JFK" 0x777
GraphNode	this 0x350	to "WDC"	"TOR" 0x808
	a "TOR"		

- Print false to the screen
- in/out
true
false

```

public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    private void addNode(N a) {
        if (!this.adjacencyList.containsKey(a)) {
            this.adjacencyList.put(a, new ArrayList<>());
        }
    }
    public boolean areConnect(N from, N to){
        return this.adjacencyList.containsKey(from) &&
               this.adjacencyList.get(from).contains(to);
    }
    public boolean validPath(ArrayList<N> path) {
        for (int i=0; i < path.size()-1; i++) {
            if(!this.areConnected(path.get(i), path.get(i+1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Graph<String> graph = new Graph<>();
        graph.addEdge("BUF", "WDC");
        graph.addEdge("WDC", "JFK");
        graph.addEdge("TOR", "BUF");
        ArrayList<String> path1 = new ArrayList<>(
            Arrays.asList("BUF", "WDC", "JFK"));
        System.out.println(graph.validPath(path1));
        ArrayList<String> path2 = new ArrayList<>(
            Arrays.asList("JFK", "WDC", "BUF"));
        System.out.println(graph.validPath(path2));
    }
}

```

Stack		Heap	
	Name Value		Name Value
graph	graph 0x350	this 0x350	adjacencyList 0x200
	this 0x350	a "BUF"	0x350
addEdge	this 0x350	path1 0x196	0x200
	from "BUF"	this 0x350	"BUF" 0x480
	to "WDC"	path 0x196	"WDC" 0x120
GraphNode	this 0x350	[i] 0	"JFK" 0x777
	a "BUF"	this 0x350	"TOR" 0x808
GraphNode	this 0x350	areConnect	
	a "WDC"	from "BUF"	0x480
GraphNode	this 0x350	to "WDC"	0 "WDC"
	from "WDC"	this 0x350	0x120
GraphNode	to "JFK"	from "WDC"	0 "JFK"
	this 0x350	to "JFK"	0x777
GraphNode	a "WDC"	path2 0x296	0x808
	this 0x350	this 0x350	0 "BUF"
GraphNode	a "JFK"	path 0x296	0x196
	this 0x350	[i] 0	"BUF" 0x480
GraphNode	from "TOR"	this 0x350	"WDC" 0x120
	to "BUF"	from "JFK"	"JFK" 0x777
GraphNode	this 0x350	to "WDC"	"TOR" 0x808
	a "TOR"		

• Program ends

in/out
true
false

0 "BUF"	0 "BUF"
1 "WDC"	1 "WDC"
2 "JFK"	2 "JFK"
0x296	0x296
0 "JFK"	0 "JFK"
1 "WDC"	1 "WDC"
2 "BUF"	2 "BUF"