Graphs

Memory Diagram

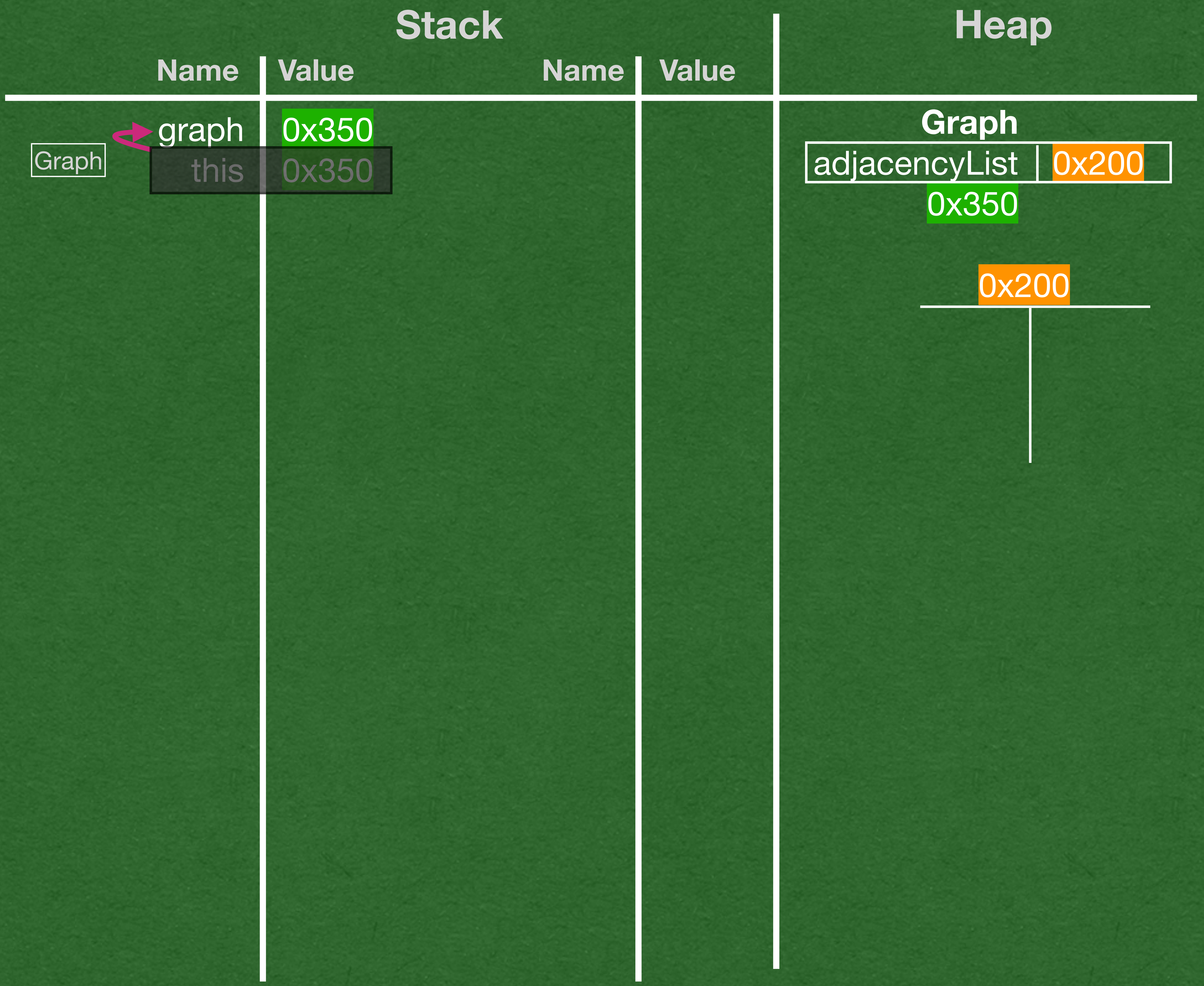
```
public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    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

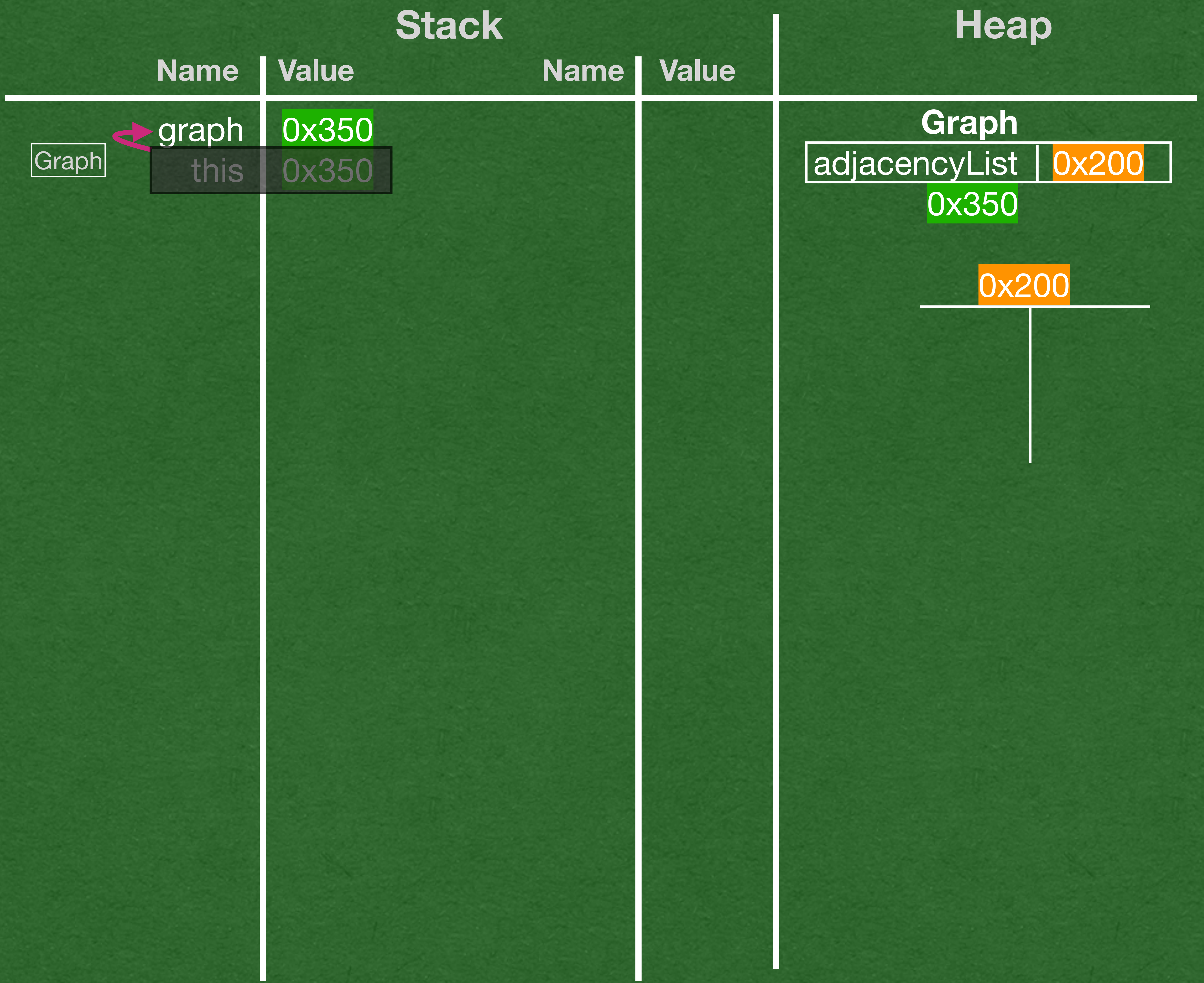
```
public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    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));
    }
}
```



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

in/out

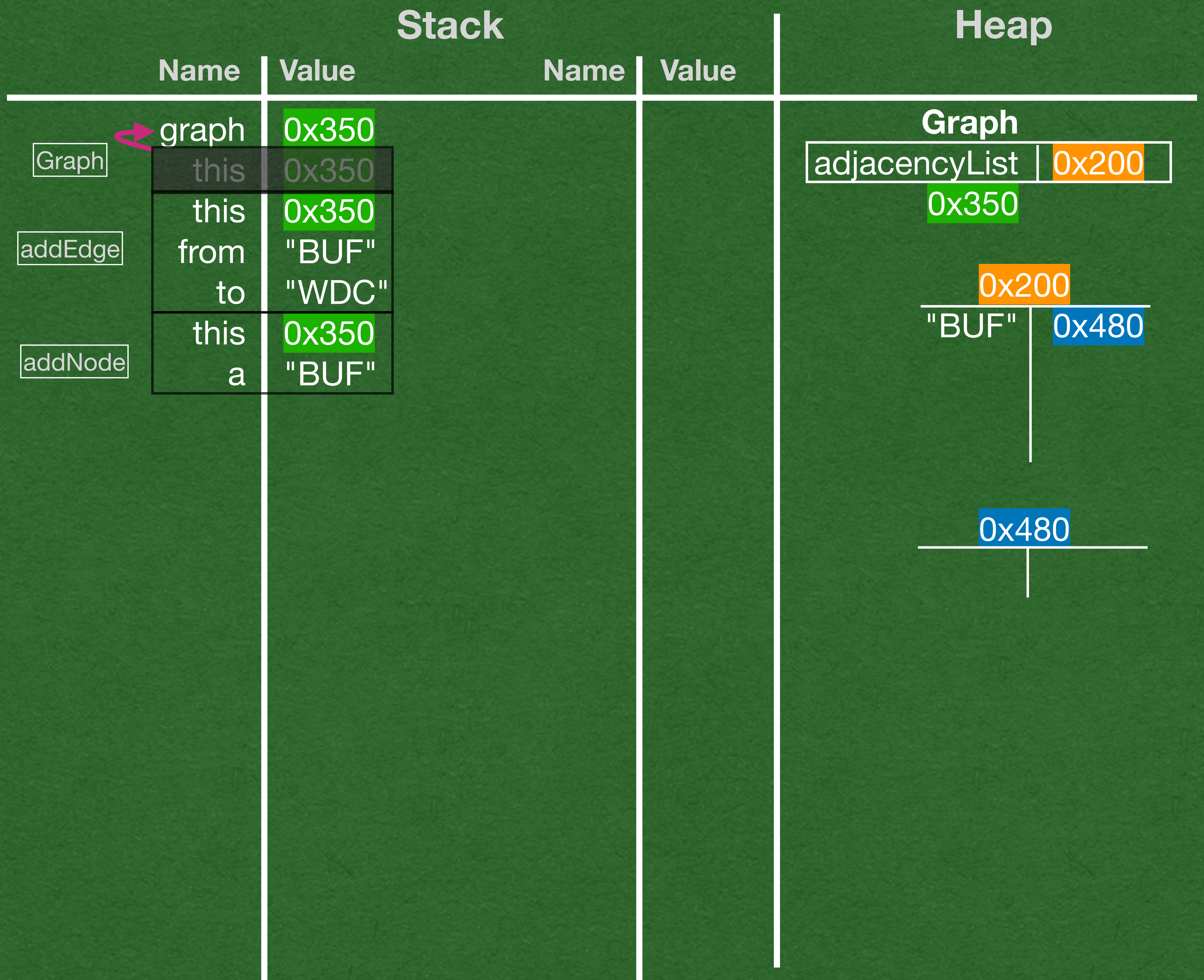
```java
public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    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));
    }
}
```



- We'll add a edge from "BUF" to "WDC"
- Notice this will be a directed graph (Edges only go in one direction)

in/out

```java
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

	Name	Value	Name	Value
	graph	0x350		
Graph	this	0x350		
	this	0x350		
addEdge	from	"BUF"		
	to	"WDC"		
	this	0x350		
addNode	a	"BUF"		

Heap

Graph

adjacencyList	0x200

0x350

0x200

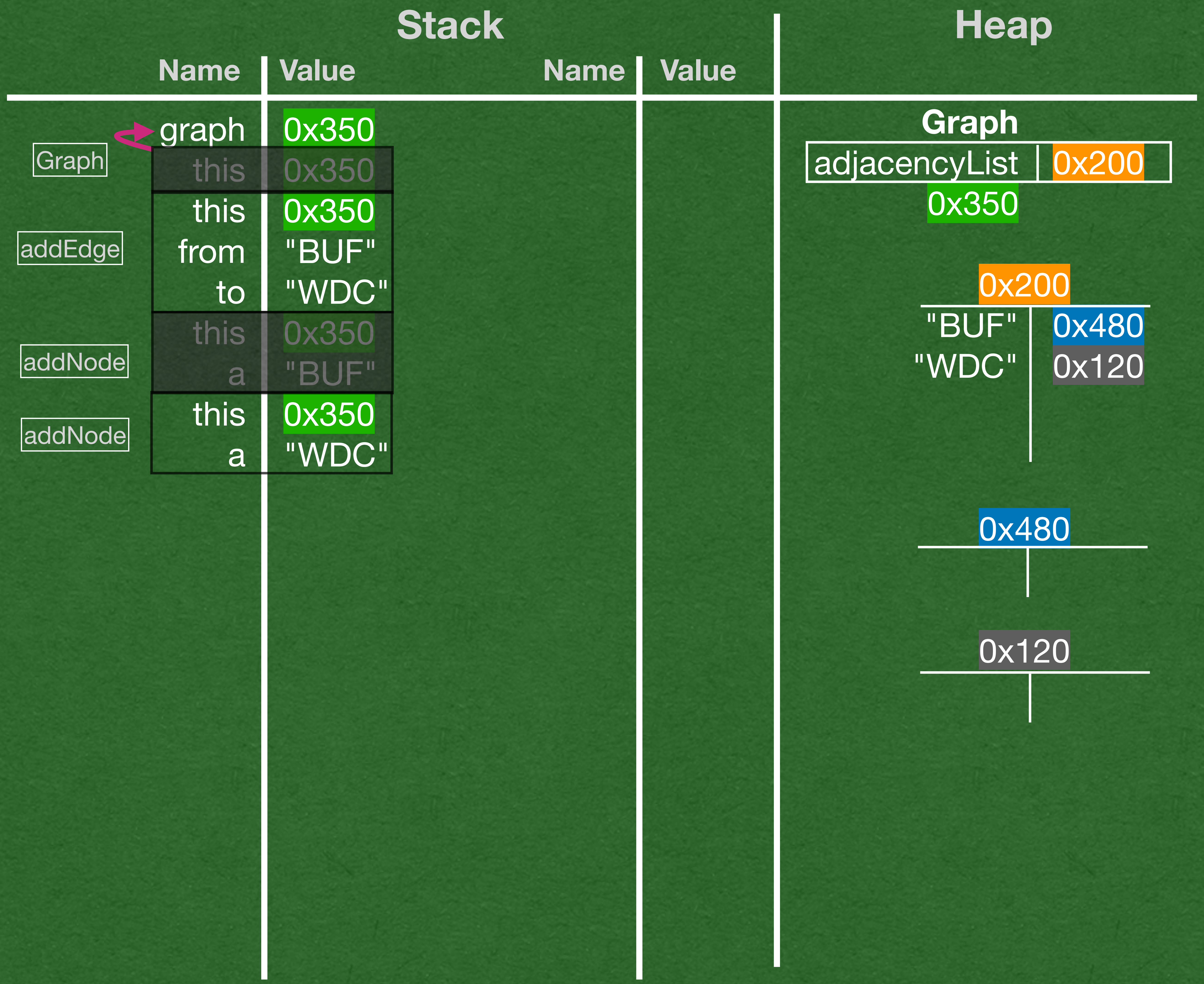
"BUF"	0x480

0x480

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

in/out

```java
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

	Name	Value	Name	Value
	graph	0x350		
Graph	this	0x350		
addEdge	this	0x350		
	from	"BUF"		
	to	"WDC"		
addNode	this	0x350		
	a	"BUF"		
addNode	this	0x350		
	a	"WDC"		

Heap

Graph

adjacencyList	0x200

0x350

0x200

"BUF"	0x480
"WDC"	0x120

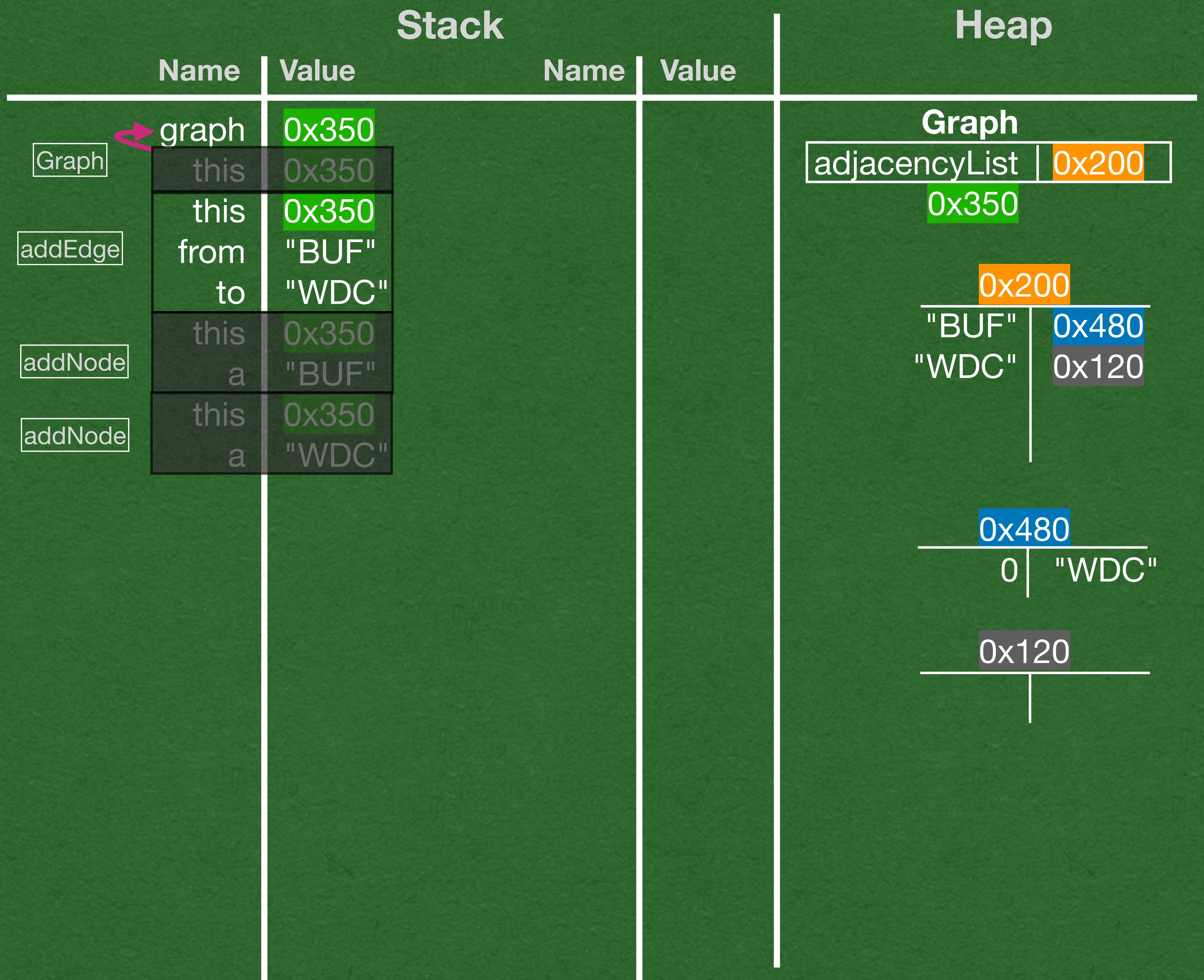
0x480

0x120

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

in/out

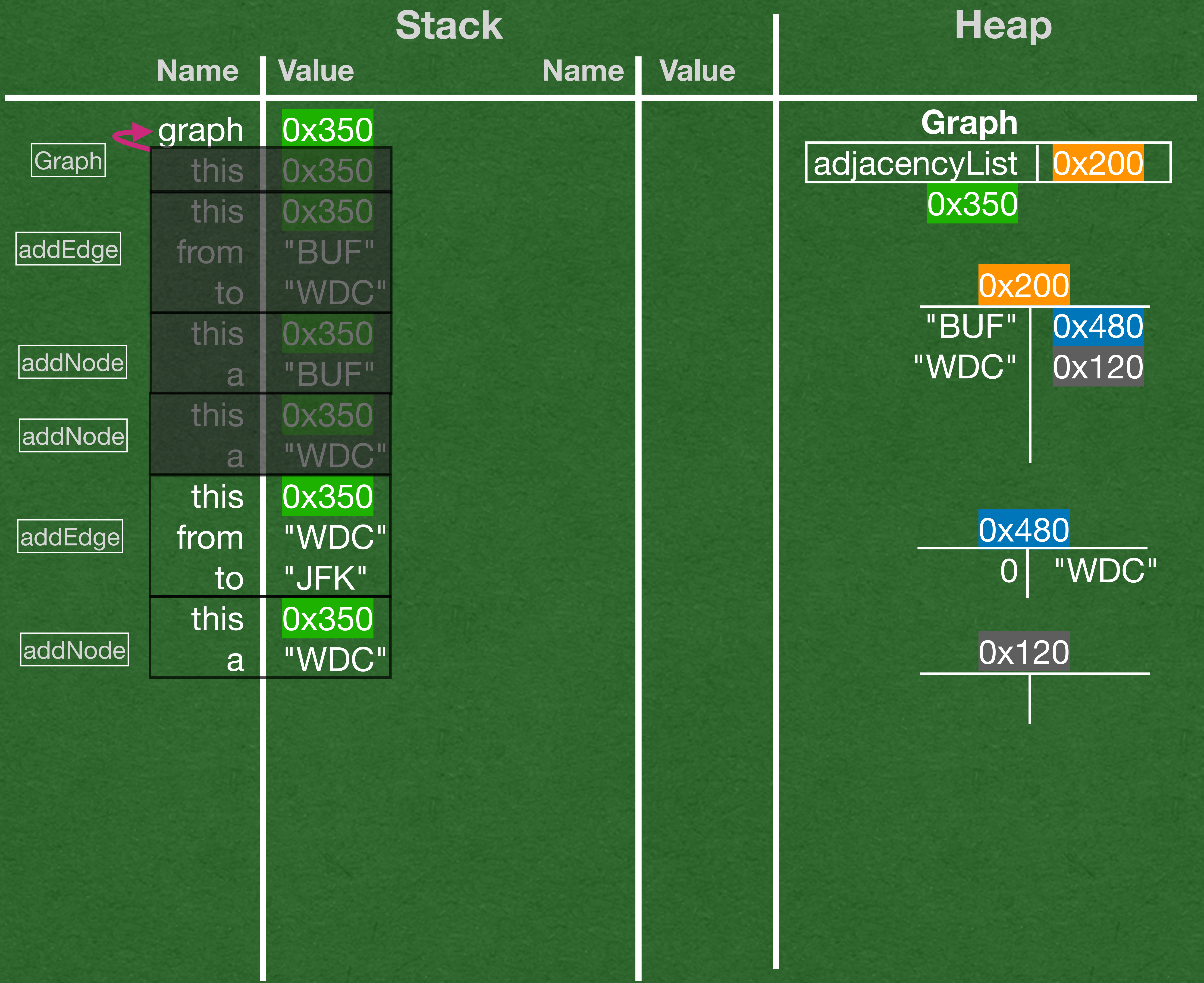
```java
public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    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));
    }
}
```



- 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

	Name	Value	Name	Value
	graph	0x350		
Graph	this	0x350		
addEdge	this	0x350		
	from	"BUF"		
	to	"WDC"		
addNode	this	0x350		
	a	"BUF"		
addNode	this	0x350		
	a	"WDC"		
addEdge	this	0x350		
	from	"WDC"		
	to	"JFK"		
addNode	this	0x350		
	a	"WDC"		

Heap

Graph

adjacencyList	0x200

0x350

0x200

"BUF"	0x480
"WDC"	0x120

0x480

0	"WDC"

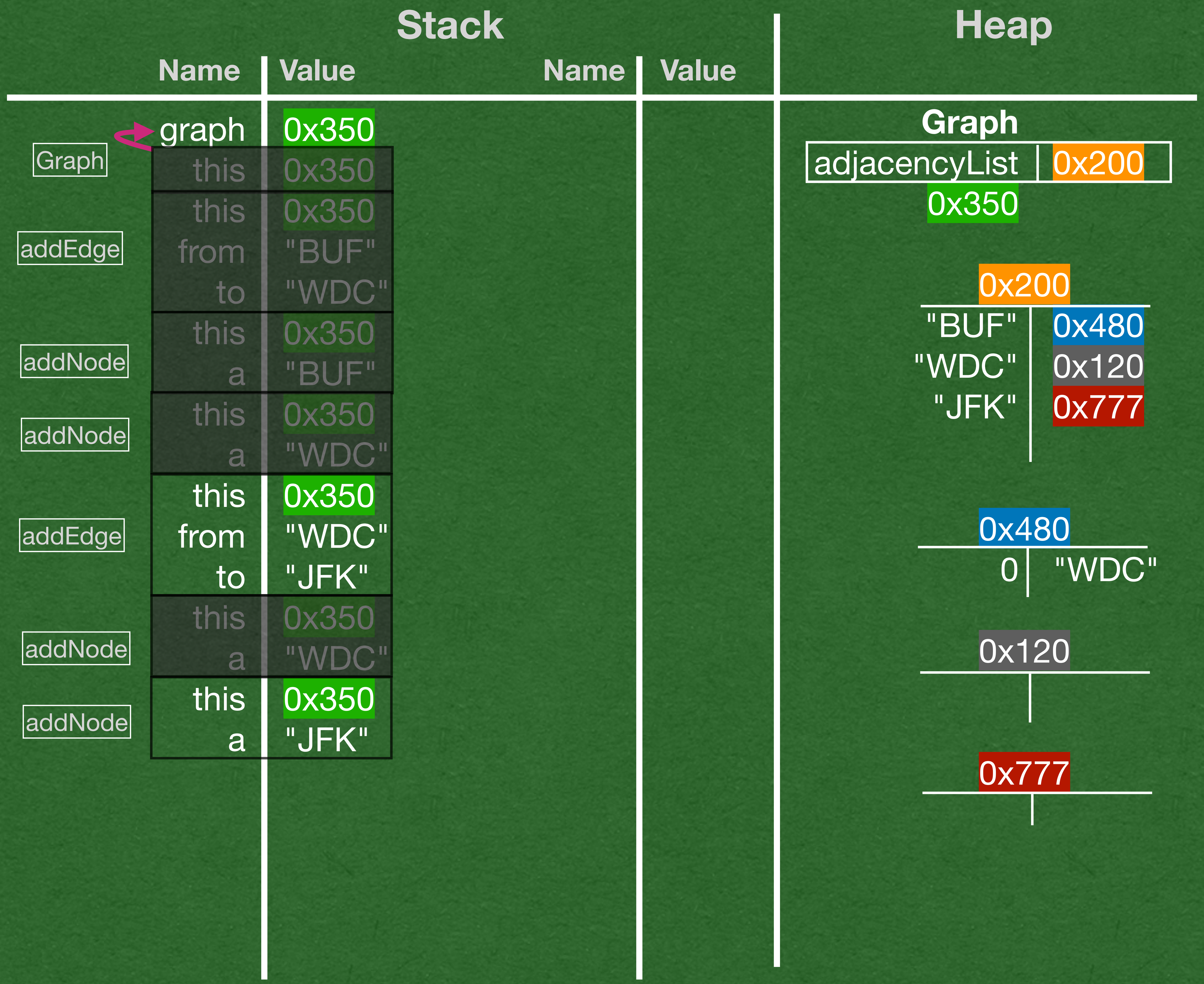
0x120

- "WDC" already has an entry in the adjacency list

- Nothing to initialize

in/out

```java
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

	Name	Value	Name	Value
	graph	0x350		
Graph	this	0x350		
addEdge	this	0x350		
	from	"BUF"		
	to	"WDC"		
addNode	this	0x350		
	a	"BUF"		
addNode	this	0x350		
	a	"WDC"		
addEdge	this	0x350		
	from	"WDC"		
	to	"JFK"		
addNode	this	0x350		
	a	"WDC"		
addNode	this	0x350		
	a	"JFK"		

Heap

Graph

adjacencyList | 0x200

0x350

0x200

"BUF"	0x480
"WDC"	0x120
"JFK"	0x777

0x480

0	"WDC"

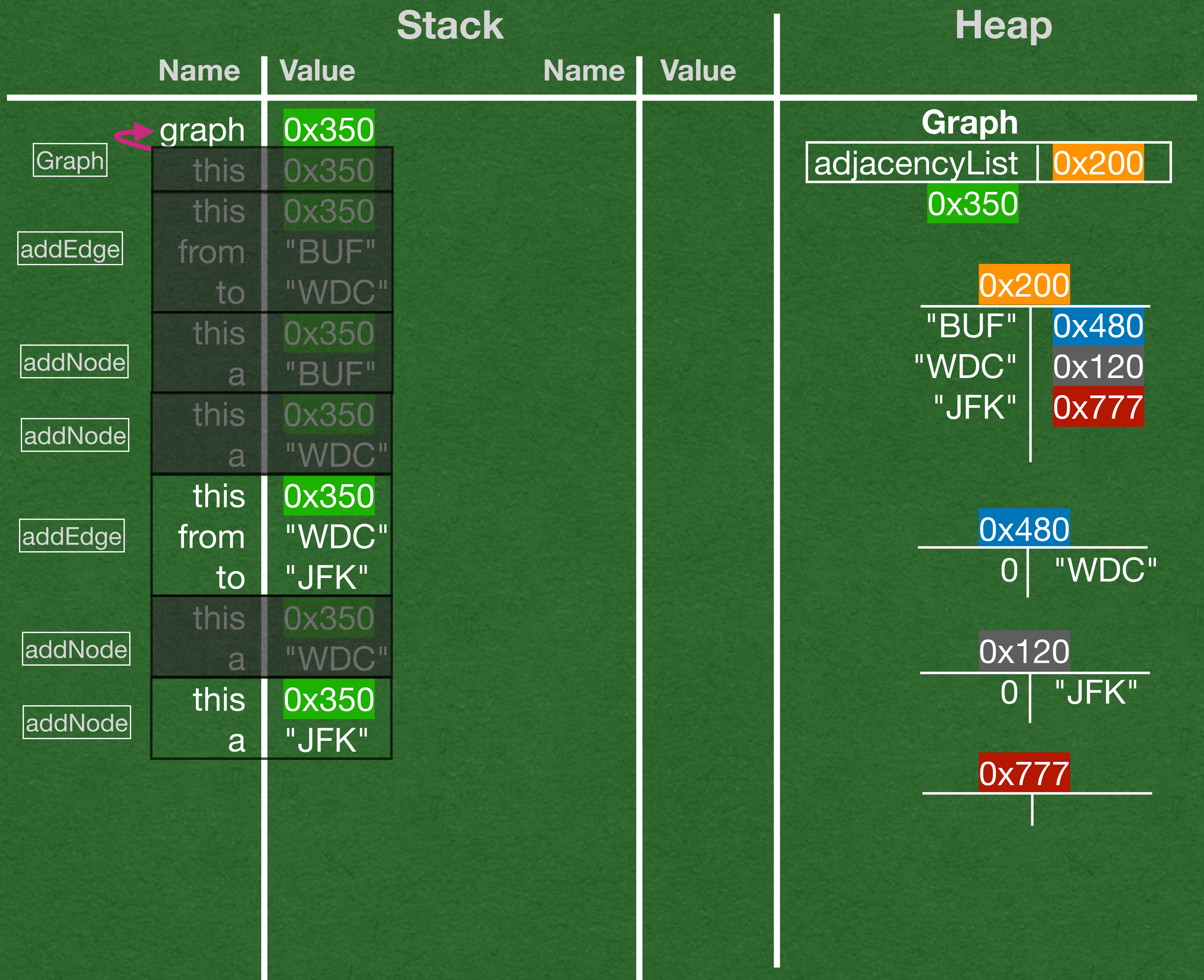
0x120

0x777

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

in/out

```java
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

	Name	Value	Name	Value
	graph	0x350		
Graph	this	0x350		
addEdge	this	0x350		
	from	"BUF"		
	to	"WDC"		
addNode	this	0x350		
	a	"BUF"		
addNode	this	0x350		
	a	"WDC"		
addEdge	this	0x350		
	from	"WDC"		
	to	"JFK"		
addNode	this	0x350		
	a	"WDC"		
addNode	this	0x350		
	a	"JFK"		

Heap

Graph

adjacencyList	0x200

0x350

0x200

"BUF"	0x480
"WDC"	0x120
"JFK"	0x777

0x480

0	"WDC"

0x120

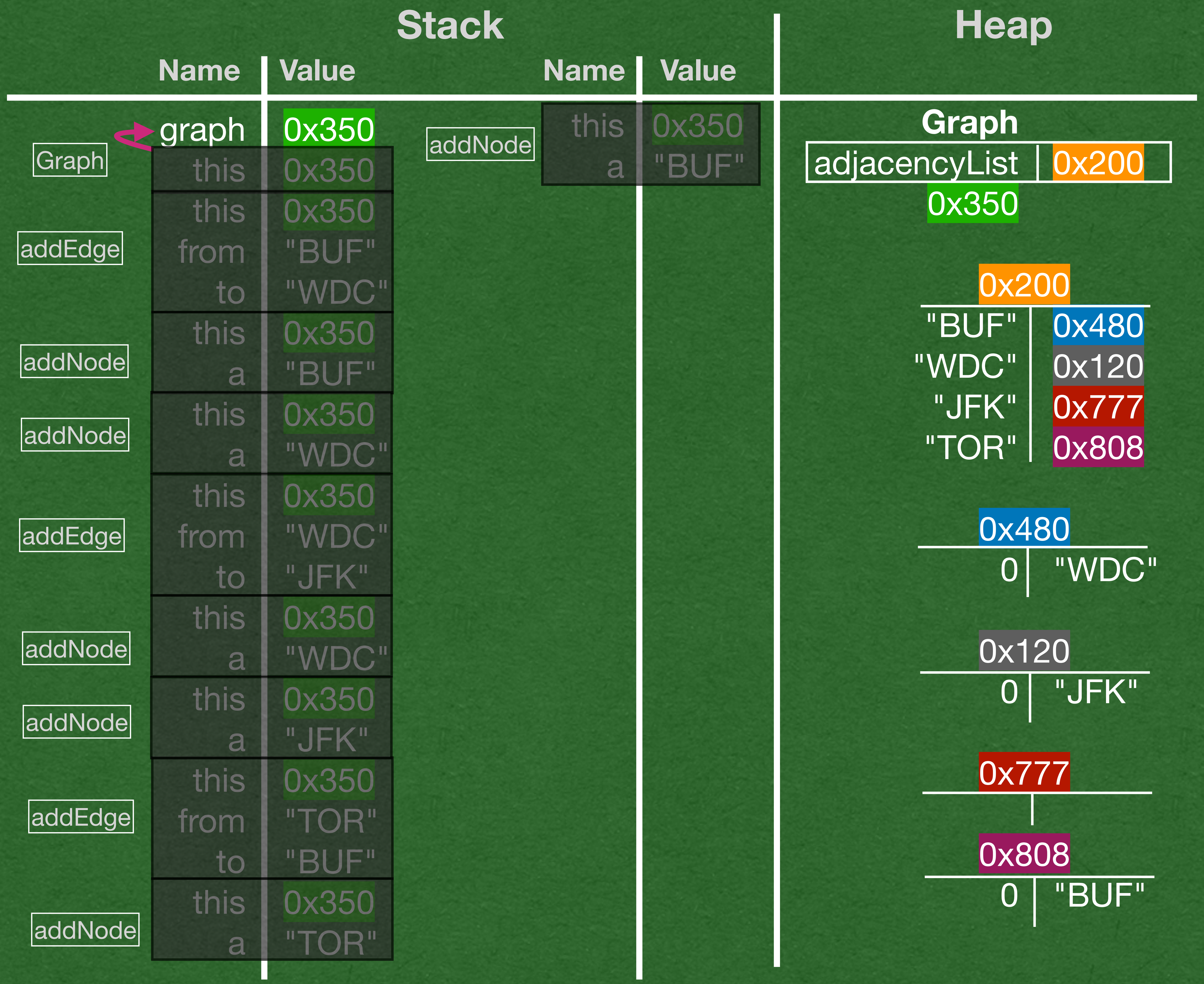
0	"JFK"

0x777

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

in/out

```java
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

	Name	Value		Name	Value
	graph	0x350	addNode	this	0x350
Graph	this	0x350		a	"BUF"
addEdge	this	0x350			
	from	"BUF"			
	to	"WDC"			
addNode	this	0x350			
	a	"BUF"			
addNode	this	0x350			
	a	"WDC"			
addEdge	this	0x350			
	from	"WDC"			
	to	"JFK"			
addNode	this	0x350			
	a	"WDC"			
addNode	this	0x350			
	a	"JFK"			
addEdge	this	0x350			
	from	"TOR"			
	to	"BUF"			
addNode	this	0x350			
	a	"TOR"			

Heap

Graph

adjacencyList	0x200

0x350

0x200

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

0x480

0	"WDC"

0x120

0	"JFK"

0x777

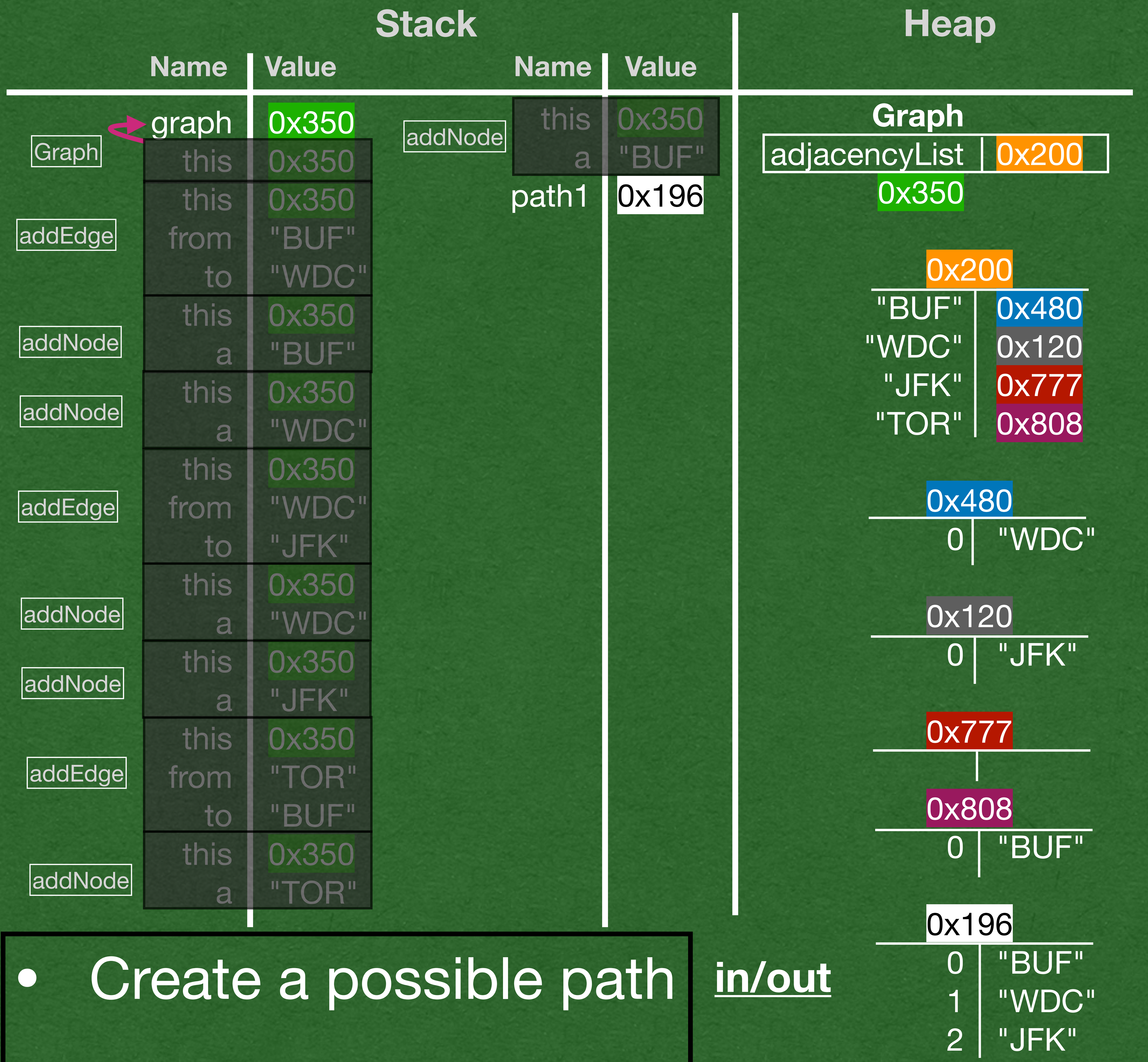
0x808

0	"BUF"

- Repeat again for the last edge

in/out

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196

Heap

Graph

adjacencyList	0x200

0x350

0x200

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

0x480

0	"WDC"

0x120

0	"JFK"

0x777

0x808

0	"BUF"

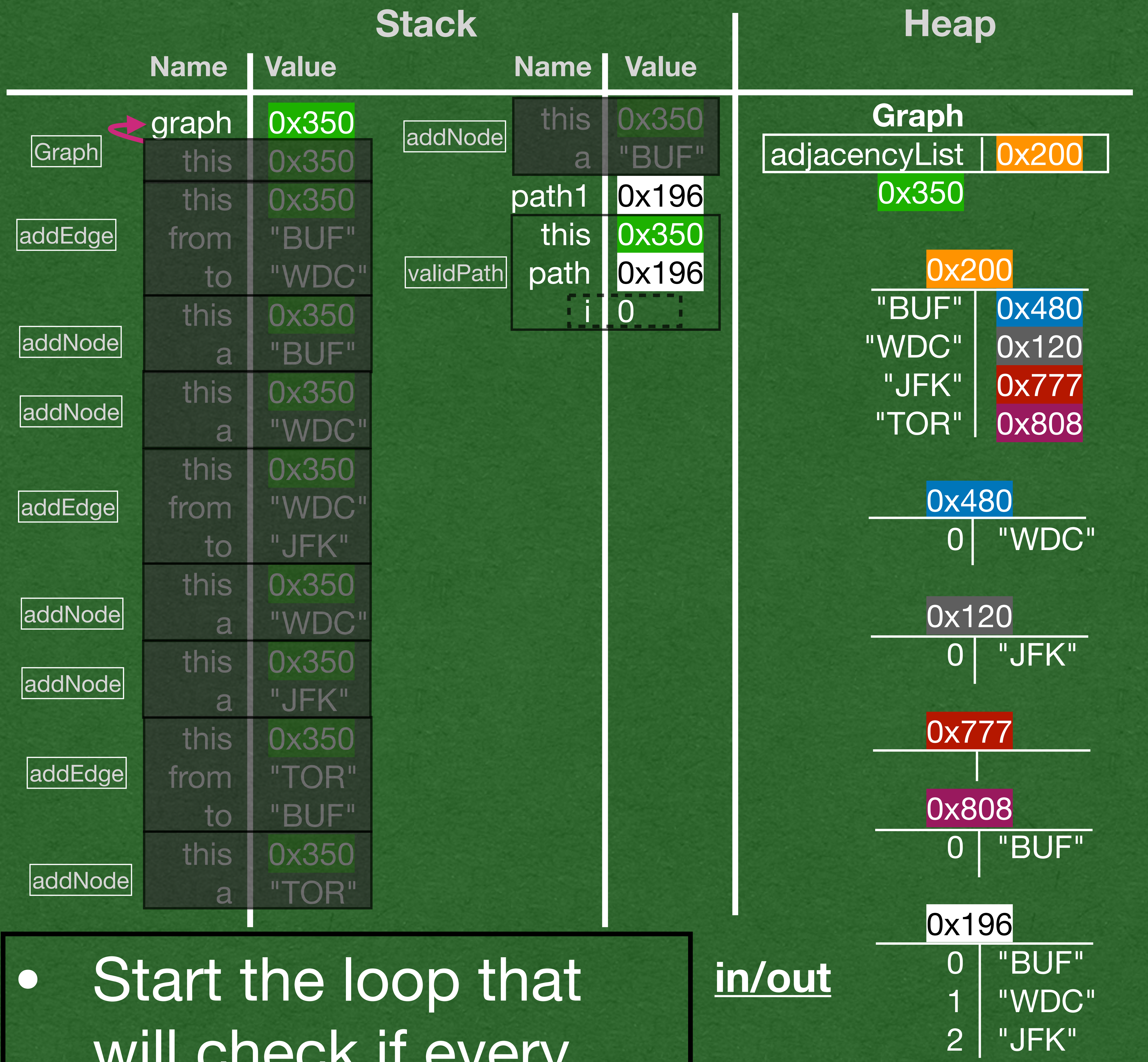
0x196

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

- Create a possible path
- This will be checked by our 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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	0

Heap

Graph

adjacencyList	0x200

0x350

0x200

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

0x480

0	"WDC"

0x120

0	"JFK"

0x777

0x808

0	"BUF"

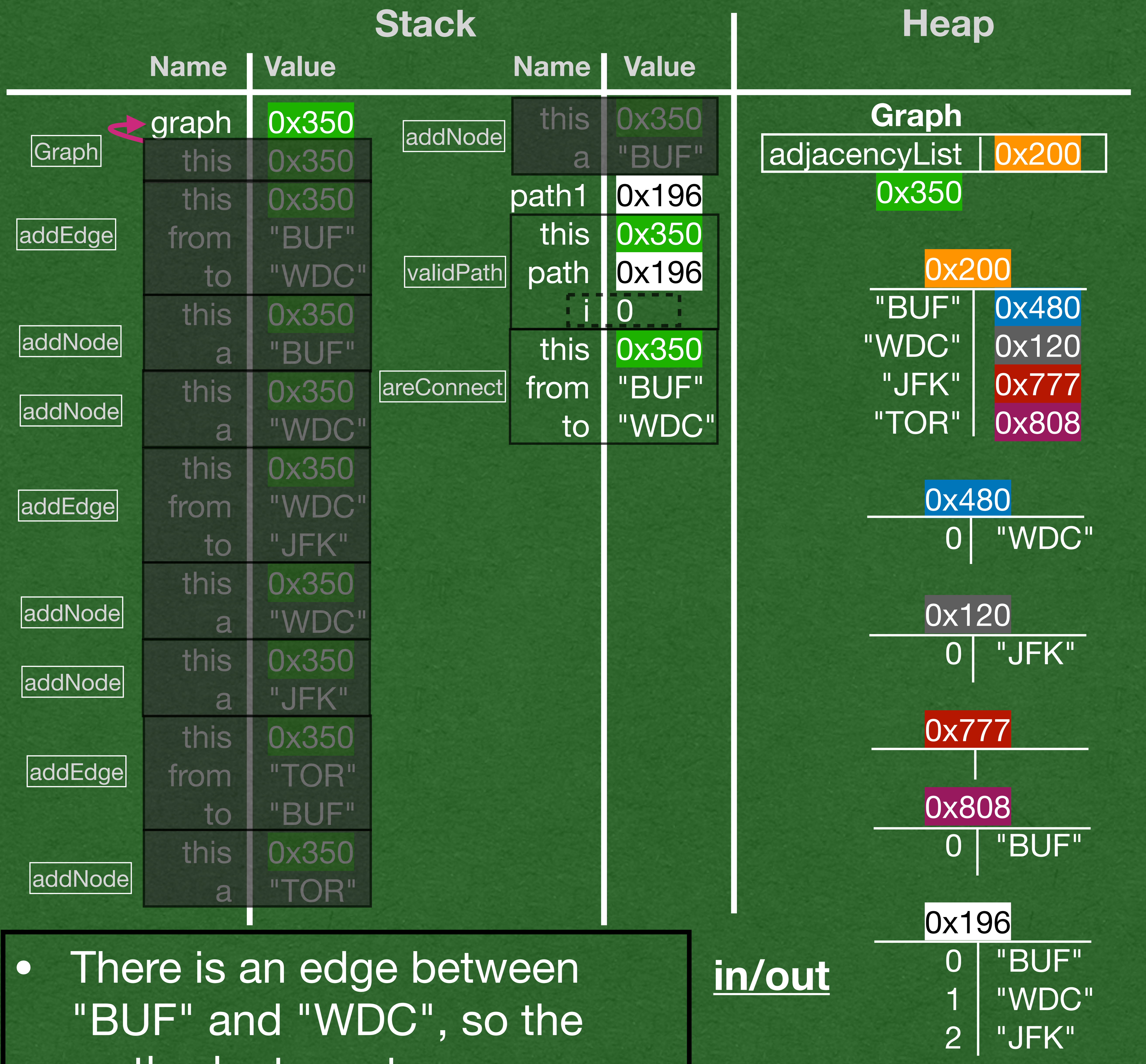
0x196

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

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

in/out

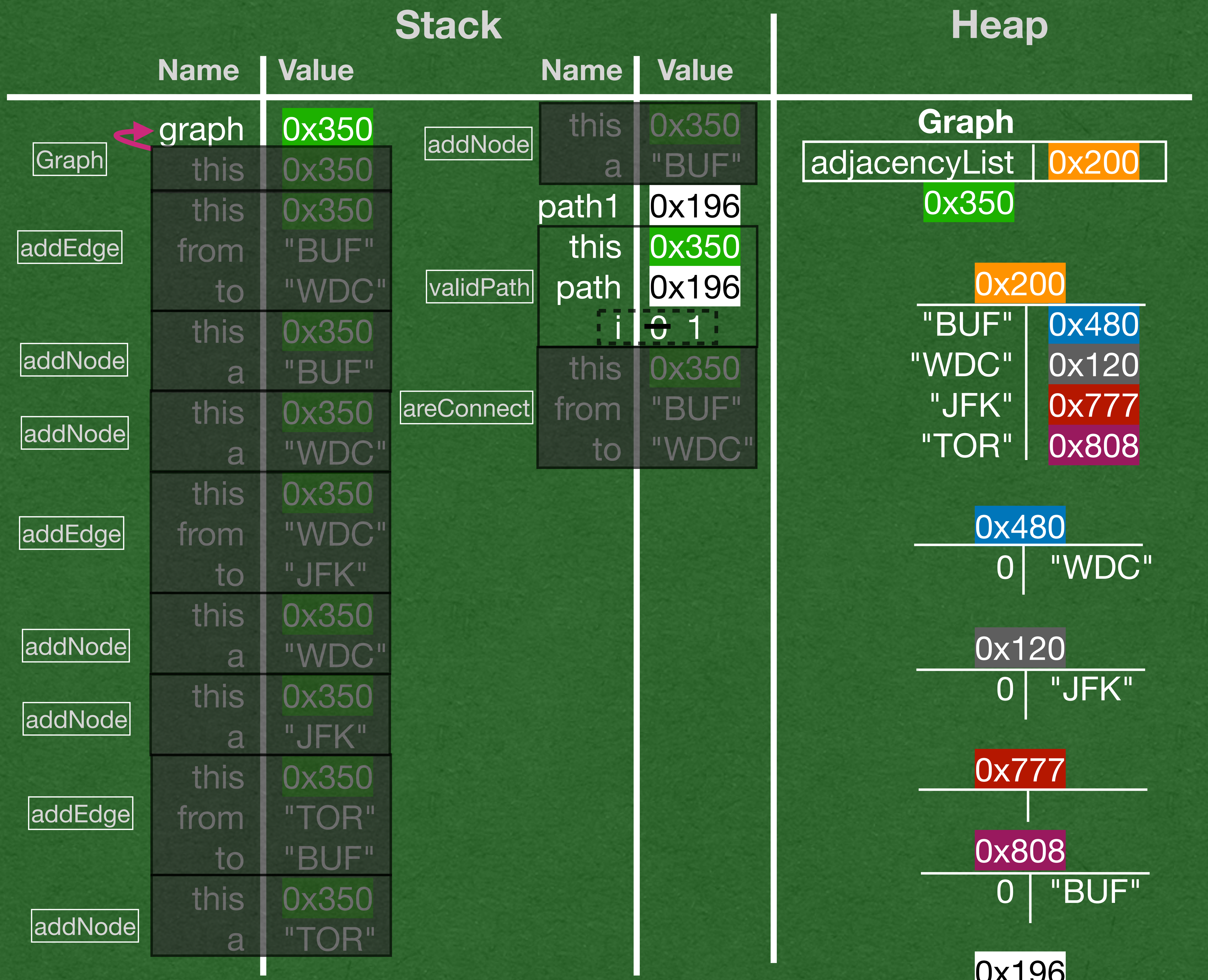
```
public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    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));
    }
}
```



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

- Conditional in validPath is false

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"

Heap

Graph

adjacencyList	0x200

0x350

0x200

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

0x480

0	"WDC"

0x120

0	"JFK"

0x777

0x808

0	"BUF"

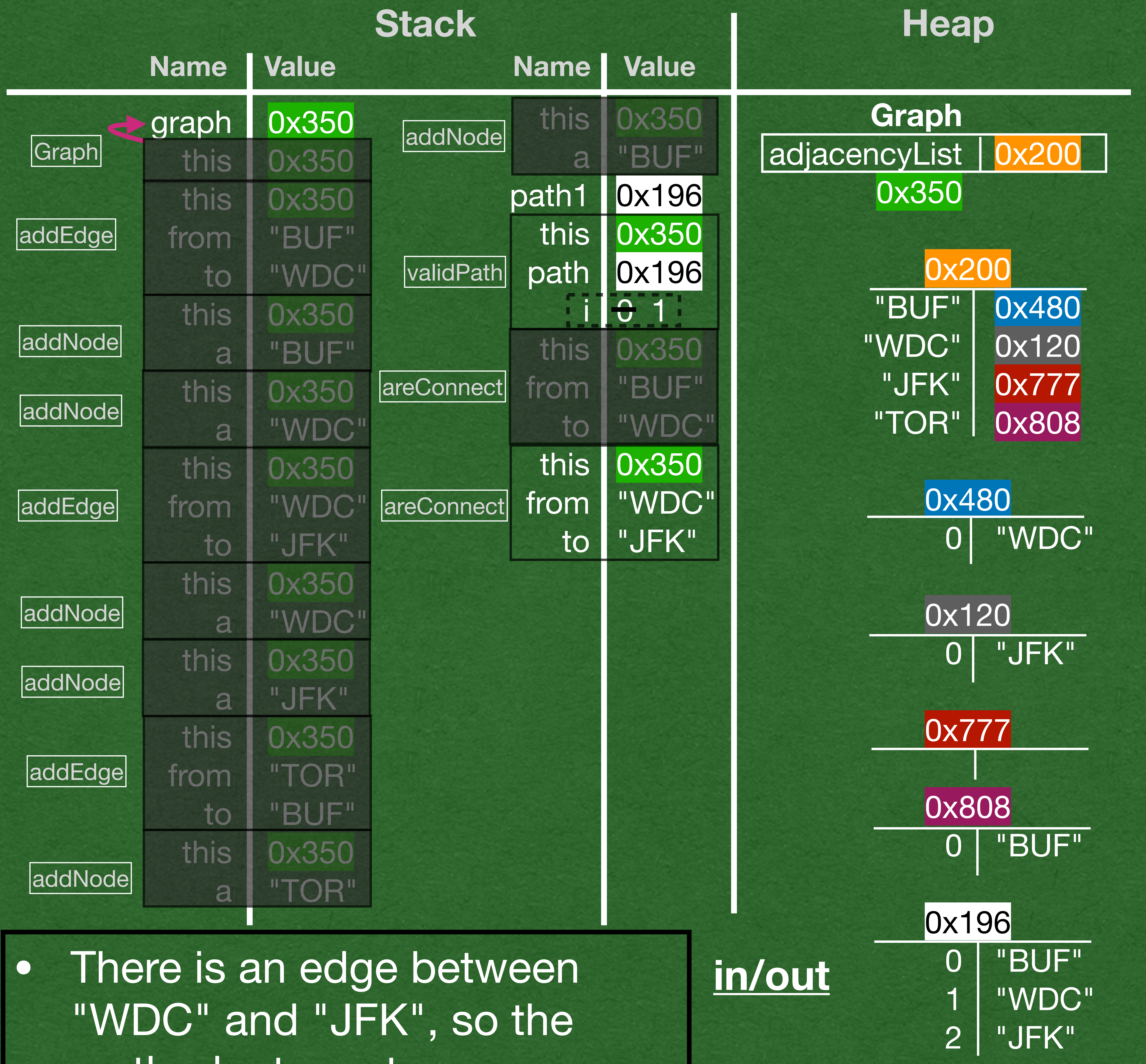
0x196

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

- 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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"

Heap

Graph

adjacencyList	0x200

0x350

0x200

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

0x480

0	"WDC"

0x120

0	"JFK"

0x777

0x808

0	"BUF"

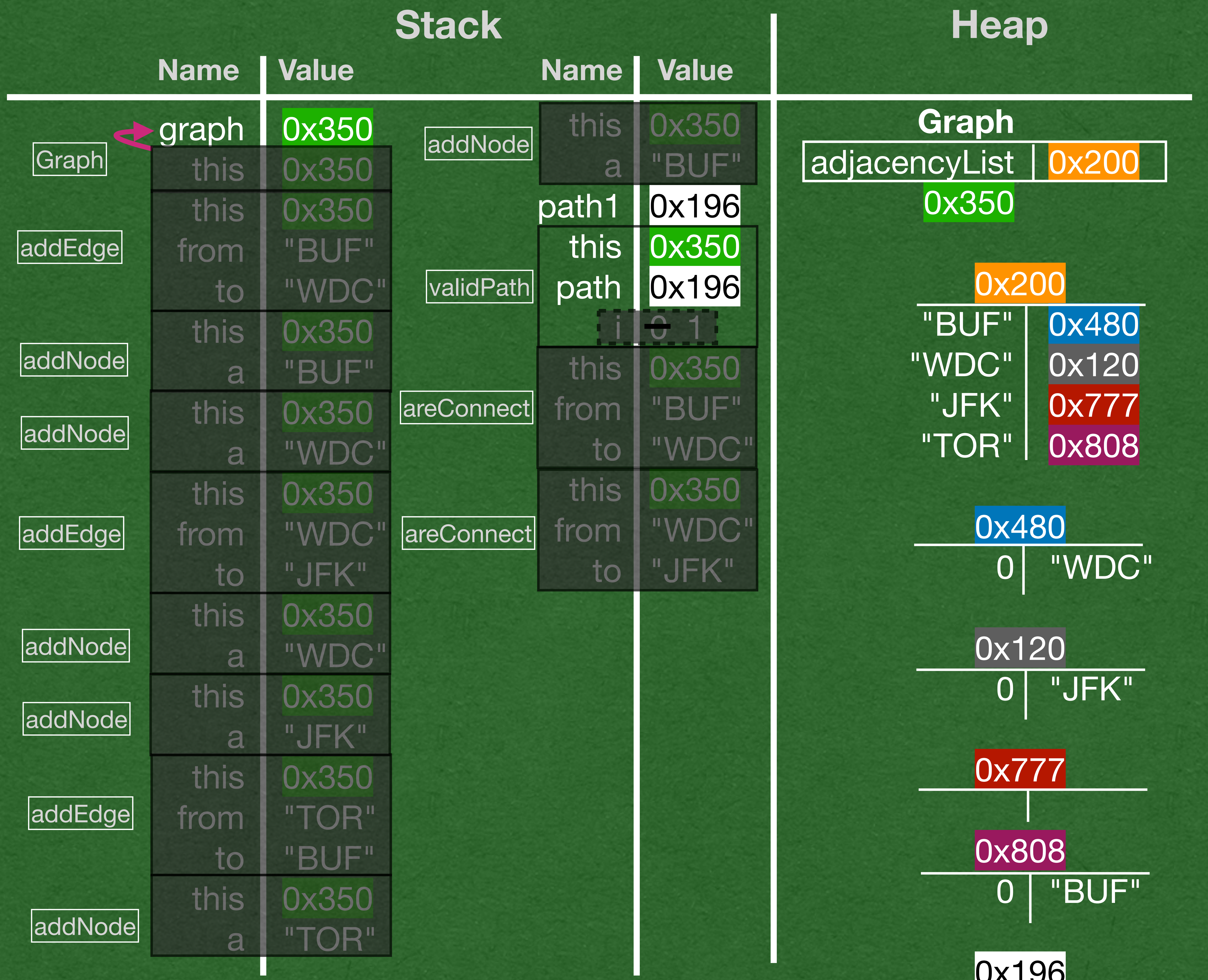
0x196

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

in/out

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

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"

Heap

Graph

adjacencyList	0x200

0x350

0x200

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

0x480

0	"WDC"

0x120

0	"JFK"

0x777

0x808

0	"BUF"

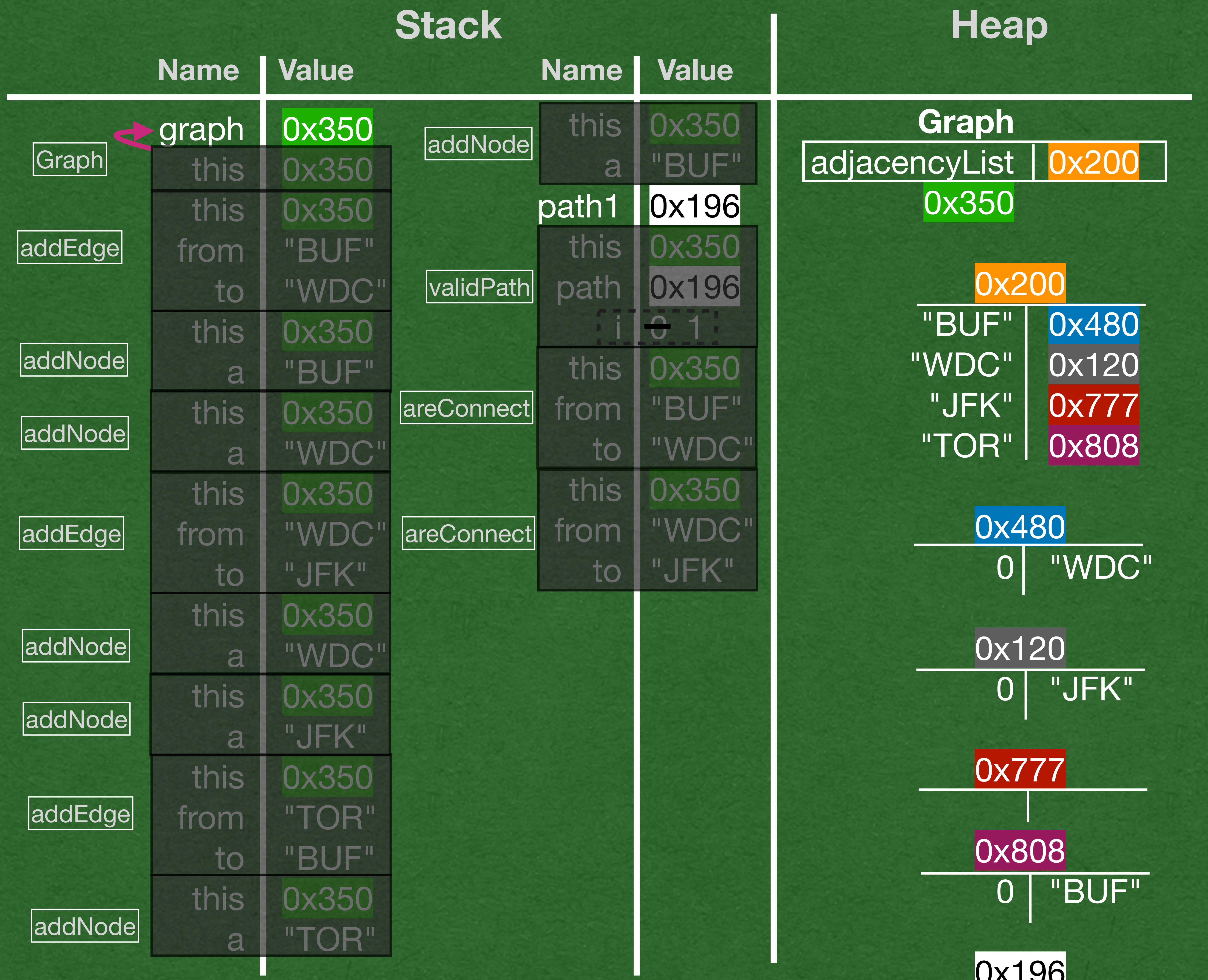
0x196

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

in/out

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

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"

Heap

Graph

adjacencyList	0x200

0x350

0x200

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

0x480

0	"WDC"

0x120

0	"JFK"

0x777

0x808

0	"BUF"

0x196

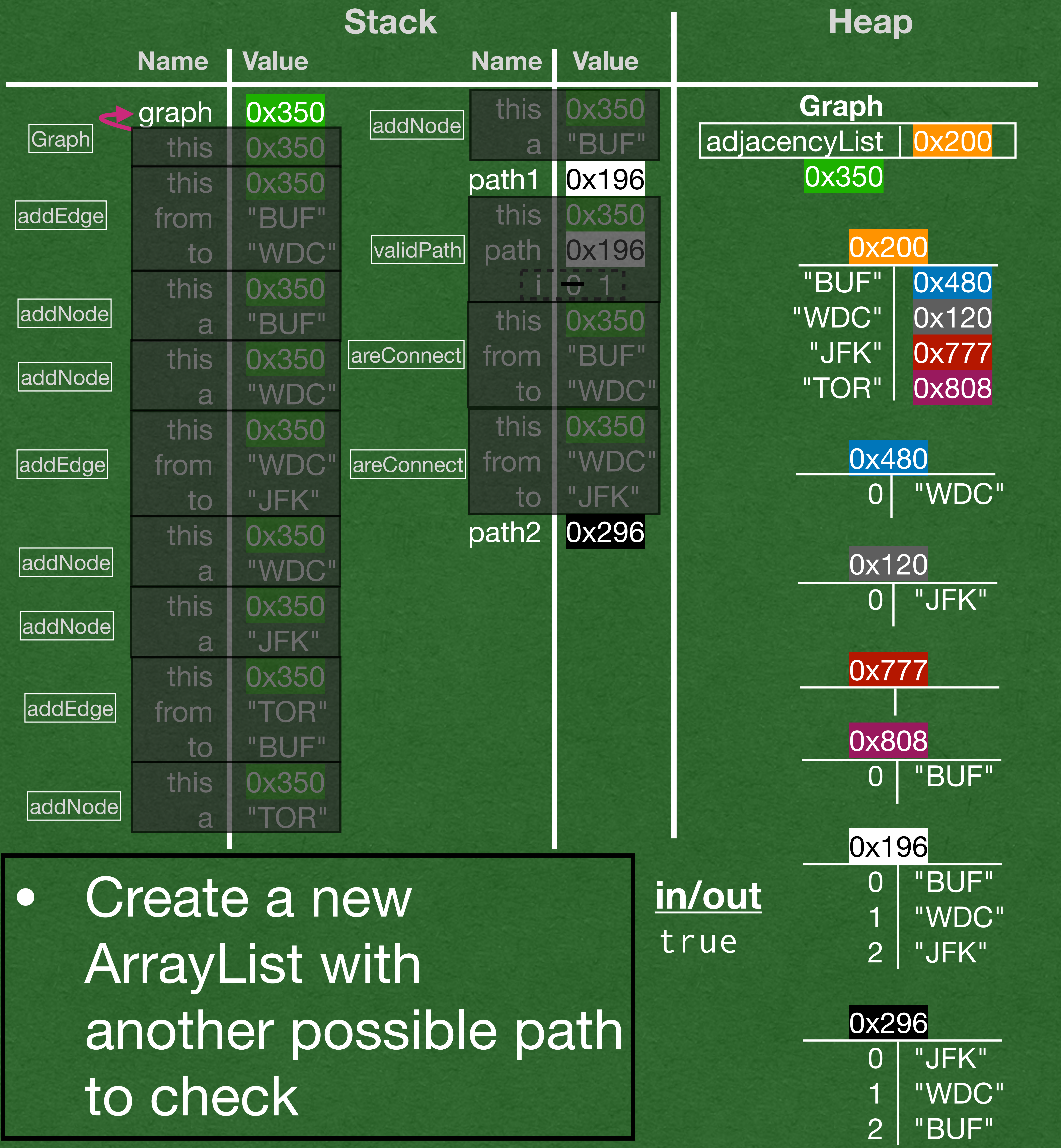
0	"BUF"
1	"WDC"
2	"JFK"

- Print true to the screen

in/out

```
true
```

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"
	path2	0x296

Heap

Graph

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"

0x296

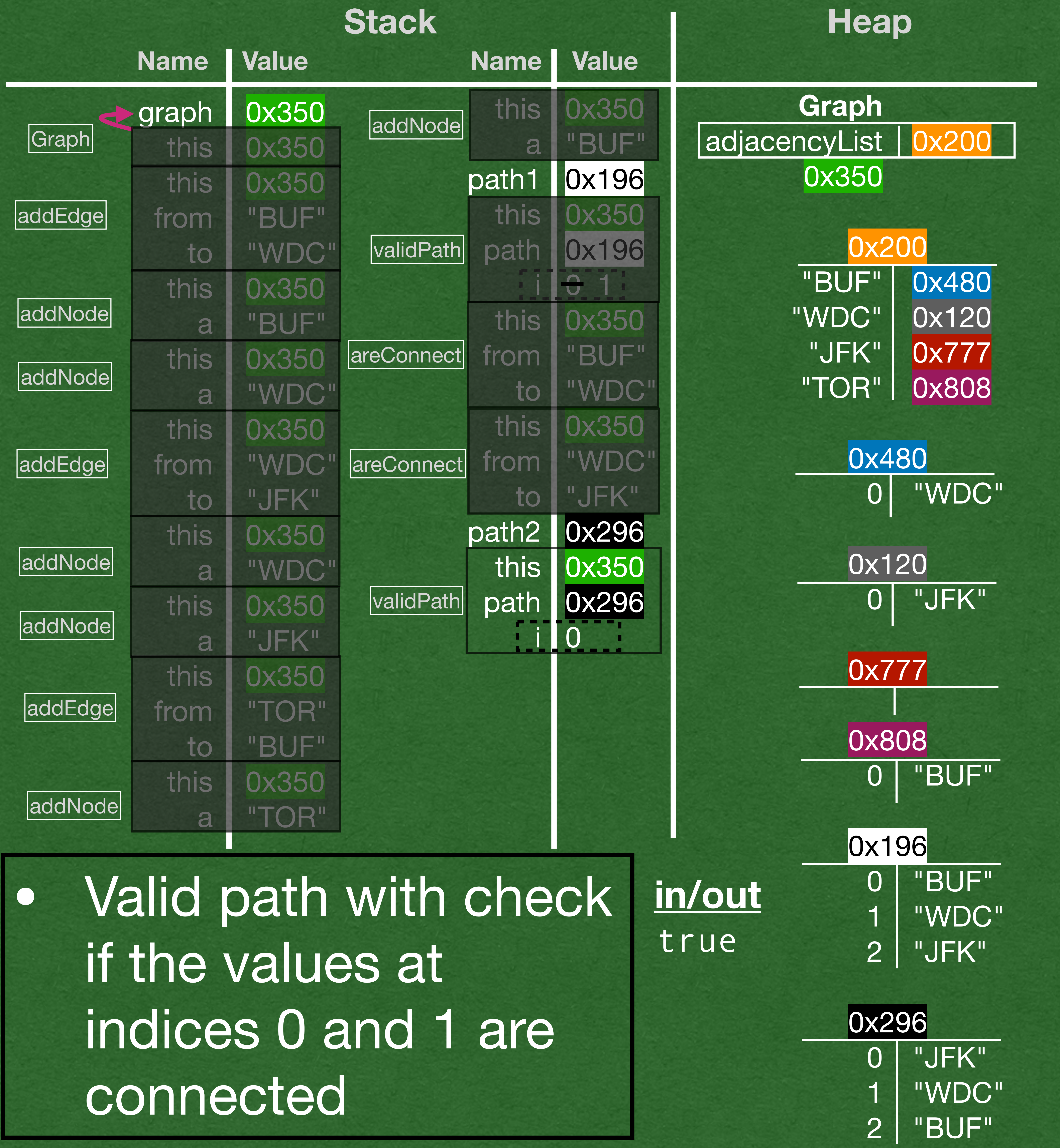
0	"JFK"
1	"WDC"
2	"BUF"

- Create a new ArrayList with another possible path to check

in/out

true

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"
	path2	0x296
validPath	this	0x350
	path	0x296
	i	0

Heap

Graph 0x350

adjacencyList	0x200

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"

0x296

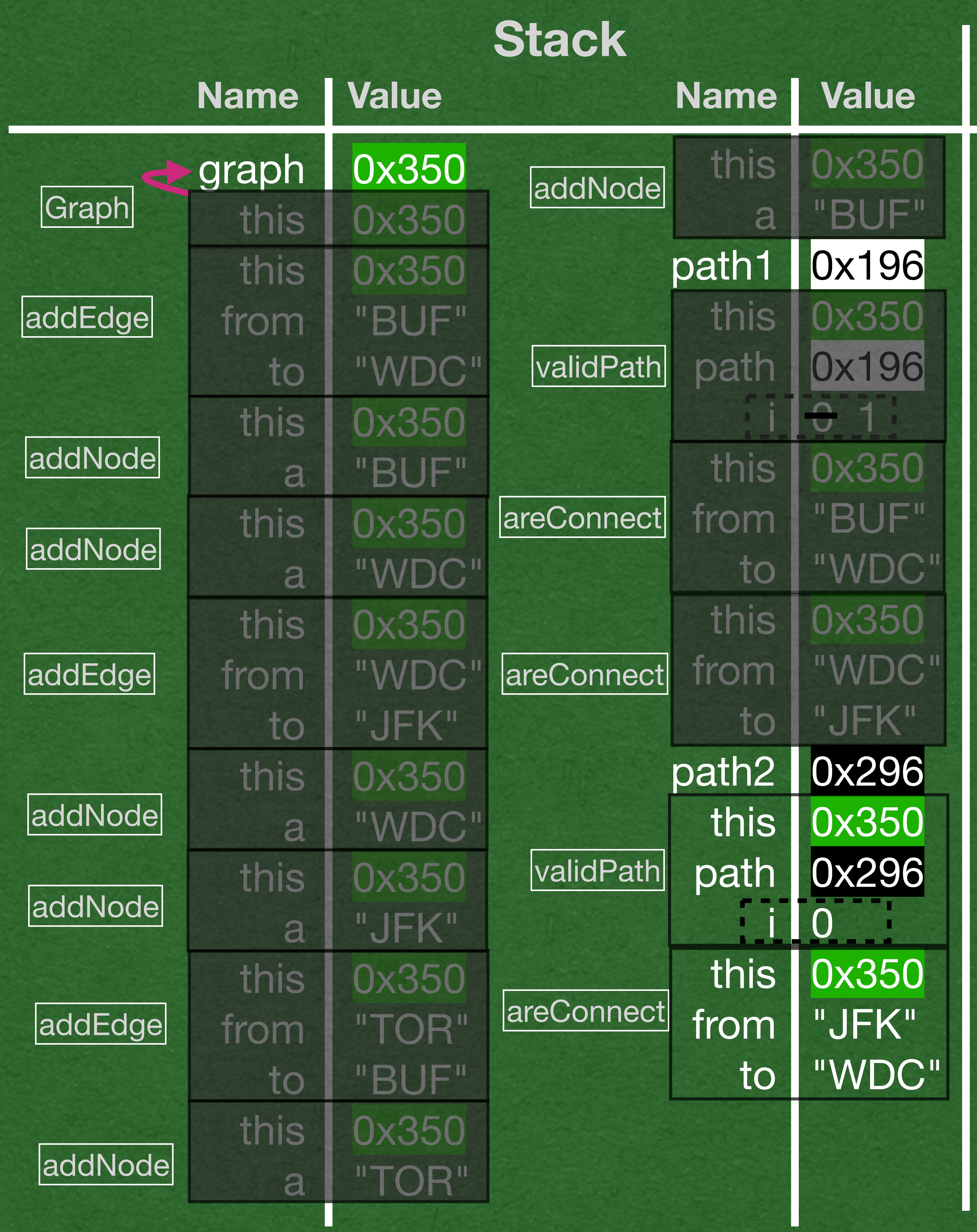
0	"JFK"
1	"WDC"
2	"BUF"

- Valid path with check if the values at indices 0 and 1 are connected

in/out

true

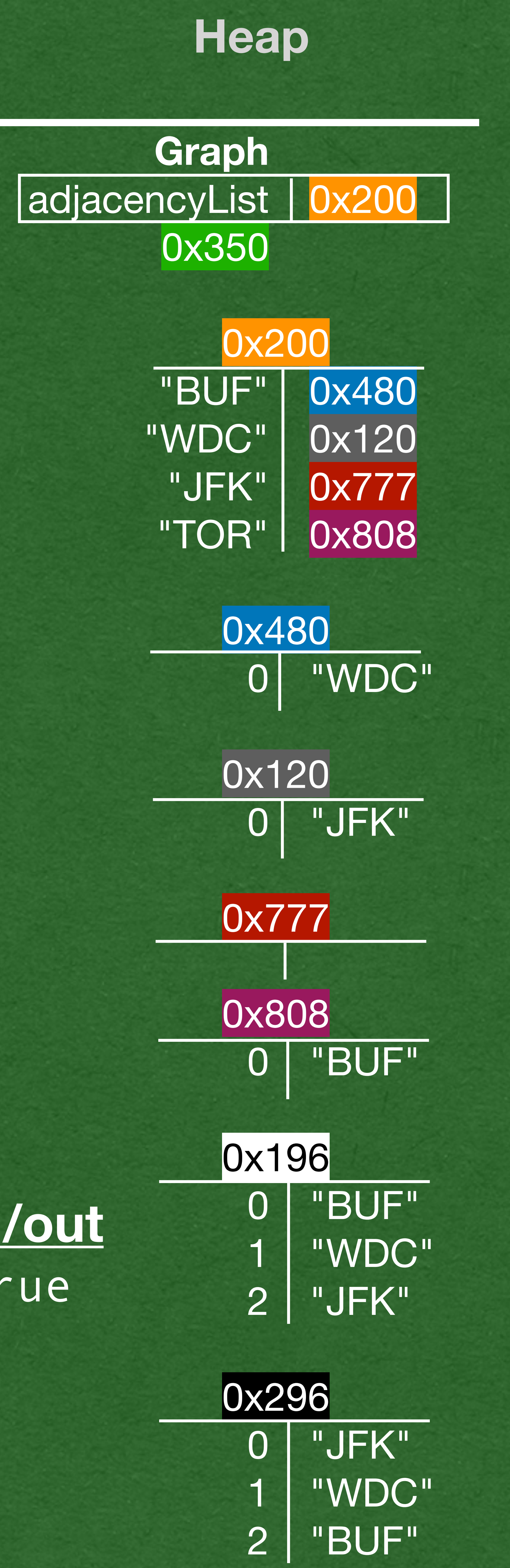
```
public class Graph<N> {
    private HashMap<N, ArrayList<N>> adjacencyList;
    public Graph() {
        this.adjacencyList = new HashMap<>();
    }
    public void addEdge(N from, N to) {
        this.addNode(from);
        this.addNode(to);
        this.adjacencyList.get(from).add(to);
    }
    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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"
	path2	0x296
validPath	this	0x350
	path	0x296
	i	0
areConnect	this	0x350
	from	"JFK"
	to	"WDC"



Heap

Graph

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"

0x296

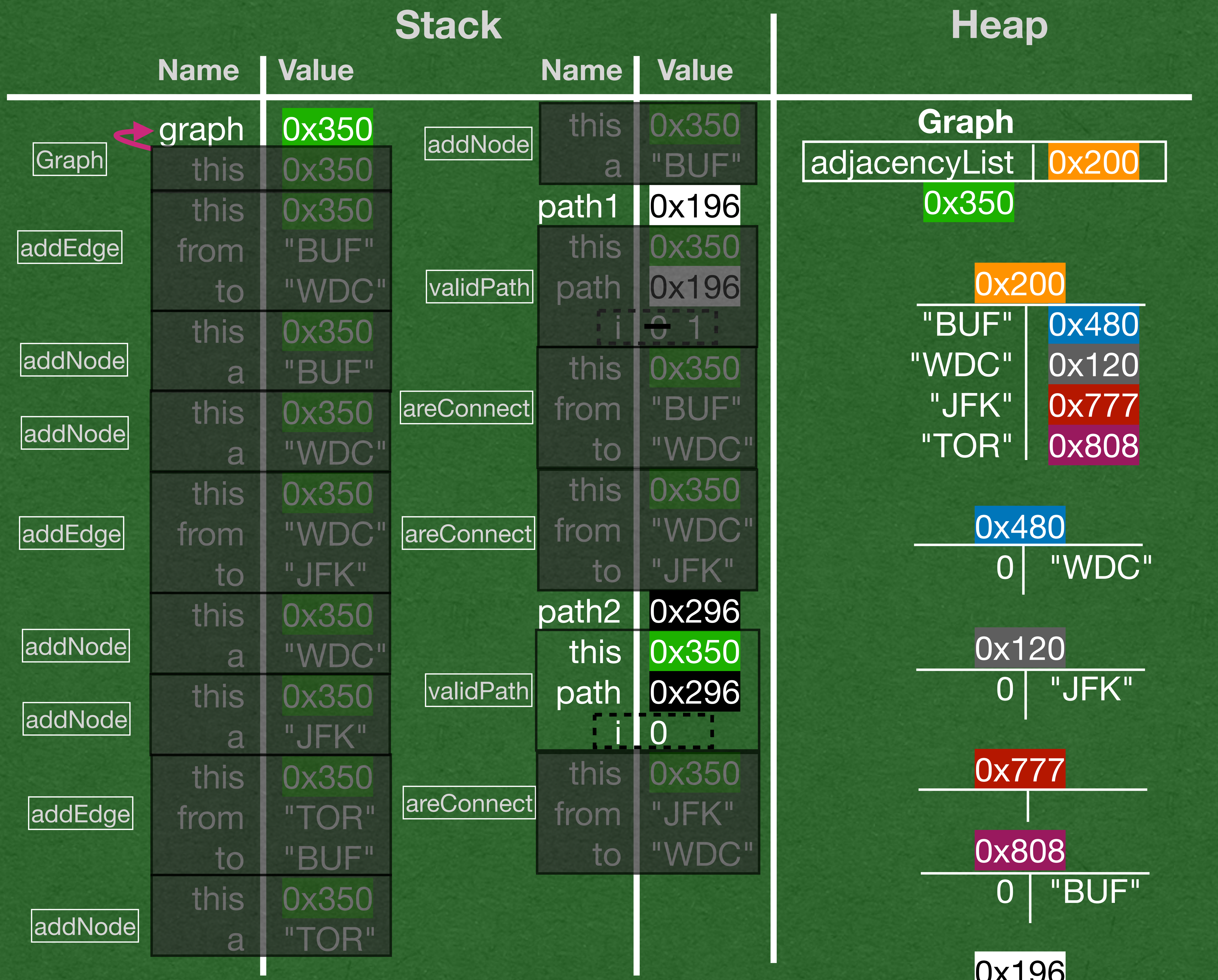
0	"JFK"
1	"WDC"
2	"BUF"

- 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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"
	path2	0x296
validPath	this	0x350
	path	0x296
	i	0
areConnect	this	0x350
	from	"JFK"
	to	"WDC"

Heap

Graph

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"

0x296

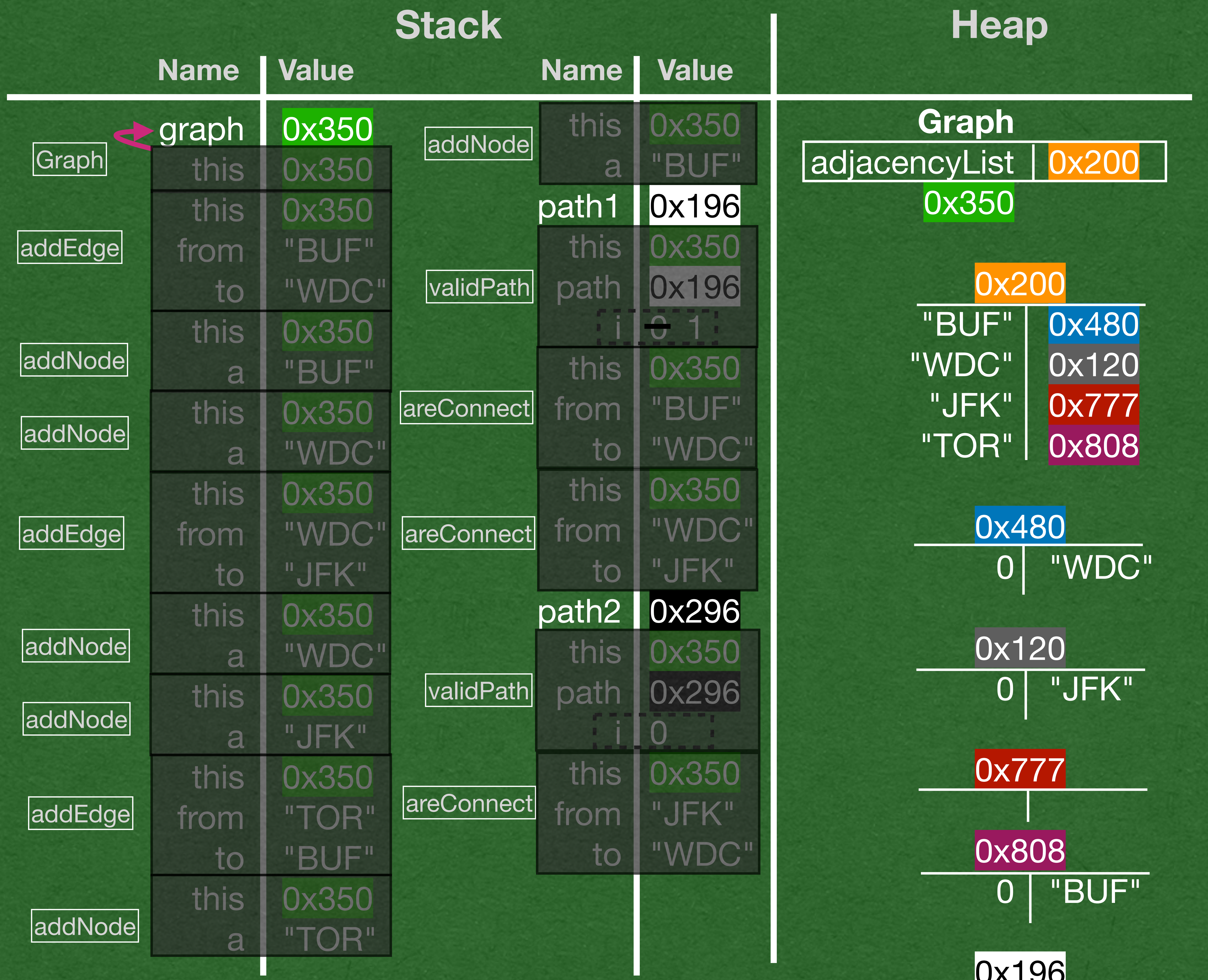
0	"JFK"
1	"WDC"
2	"BUF"

- We reach a return statement
- The entire stack frame ends and false is returned

in/out

true

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"
	path2	0x296
validPath	this	0x350
	path	0x296
	i	0
areConnect	this	0x350
	from	"JFK"
	to	"WDC"

Heap

Graph (0x350)

adjacencyList	0x200

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"

0x296

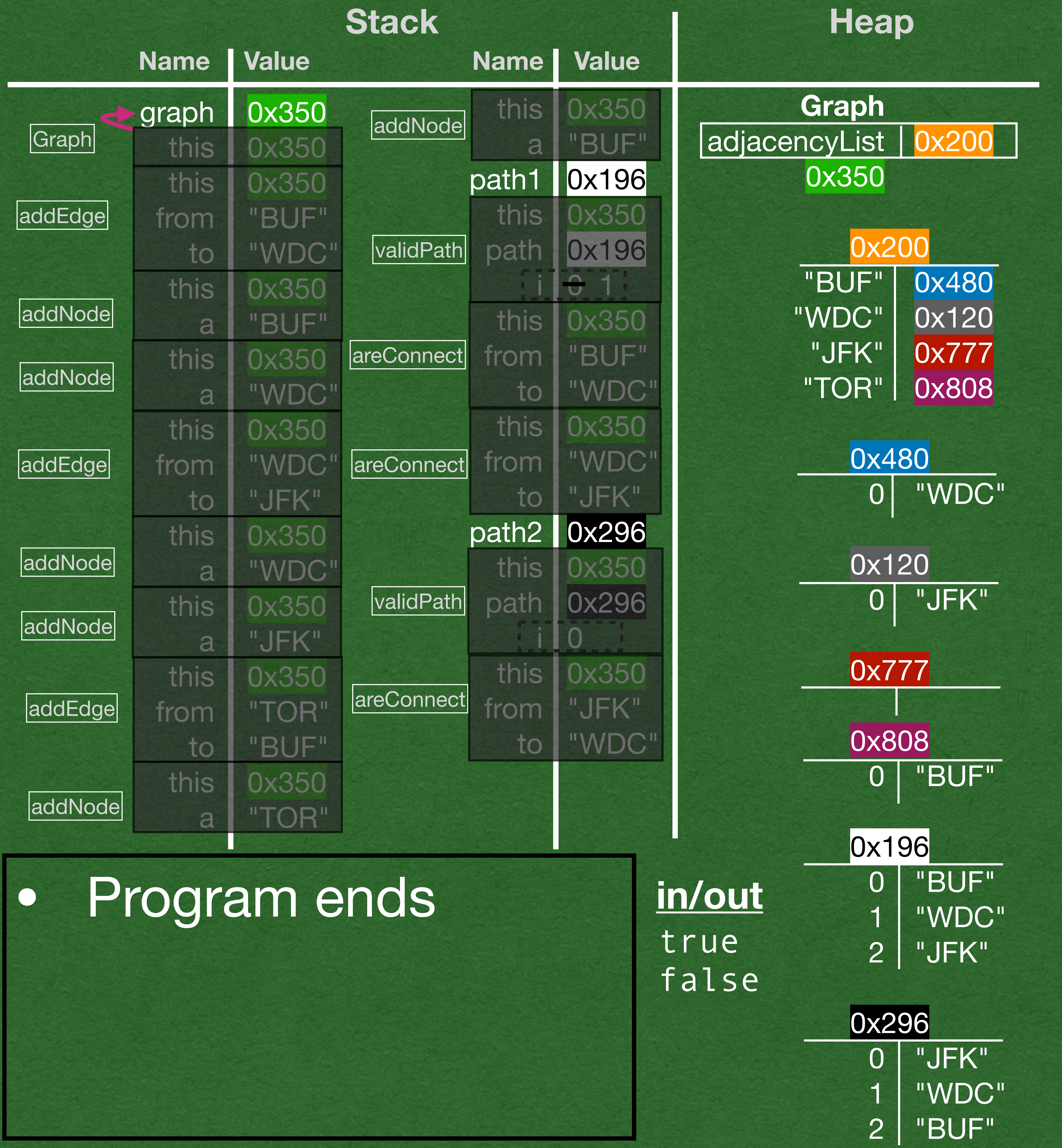
0	"JFK"
1	"WDC"
2	"BUF"

- Print false to the screen

in/out

```
true
false
```

```java
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

	Name	Value
	graph	0x350
Graph	this	0x350
addEdge	this	0x350
	from	"BUF"
	to	"WDC"
addNode	this	0x350
	a	"BUF"
addNode	this	0x350
	a	"WDC"
addEdge	this	0x350
	from	"WDC"
	to	"JFK"
addNode	this	0x350
	a	"WDC"
addNode	this	0x350
	a	"JFK"
addEdge	this	0x350
	from	"TOR"
	to	"BUF"
addNode	this	0x350
	a	"TOR"

	Name	Value
addNode	this	0x350
	a	"BUF"
	path1	0x196
validPath	this	0x350
	path	0x196
	i	~~0~~ 1
areConnect	this	0x350
	from	"BUF"
	to	"WDC"
areConnect	this	0x350
	from	"WDC"
	to	"JFK"
	path2	0x296
validPath	this	0x350
	path	0x296
	i	0
areConnect	this	0x350
	from	"JFK"
	to	"WDC"

Heap

Graph

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"

0x296

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

- Program ends

in/out

```
true
false
```