# Problem 1059: All Paths from Source Lead to Destination

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 37.02%
**Paid Only:** Yes
**Tags:** Graph, Topological Sort

## Problem Description

Given the `edges` of a directed graph where `edges[i] = [ai, bi]` indicates there is an edge between nodes `ai` and `bi`, and two nodes `source` and `destination` of this graph, determine whether or not all paths starting from `source` eventually, end at `destination`, that is:

* At least one path exists from the `source` node to the `destination` node * If a path exists from the `source` node to a node with no outgoing edges, then that node is equal to `destination`. * The number of possible paths from `source` to `destination` is a finite number.

Return `true` if and only if all roads from `source` lead to `destination`.

**Example 1:**

![](https://assets.leetcode.com/uploads/2019/03/16/485_example_1.png)

**Input:** n = 3, edges = [[0,1],[0,2]], source = 0, destination = 2 **Output:** false **Explanation:** It is possible to reach and get stuck on both node 1 and node 2.

**Example 2:**

![](https://assets.leetcode.com/uploads/2019/03/16/485_example_2.png)

**Input:** n = 4, edges = [[0,1],[0,3],[1,2],[2,1]], source = 0, destination = 3 **Output:** false **Explanation:** We have two possibilities: to end at node 3, or to loop over node 1 and node 2 indefinitely.

**Example 3:**

![](https://assets.leetcode.com/uploads/2019/03/16/485_example_3.png)

**Input:** n = 4, edges = [[0,1],[0,2],[1,3],[2,3]], source = 0, destination = 3 **Output:** true

**Constraints:**

* `1 <= n <= 104` * `0 <= edges.length <= 104` * `edges.length == 2` * `0 <= ai, bi <= n - 1` * `0 <= source <= n - 1` * `0 <= destination <= n - 1` * The given graph may have self-loops and parallel edges.

## Code Snippets

**C++:**

```
class Solution {
public:
bool leadsToDestination(int n, vector<vector<int>>& edges, int source, int
destination) {

}
};
```

**Java:**

```
class Solution {
public boolean leadsToDestination(int n, int[][] edges, int source, int
destination) {

}
}
```

**Python3:**

```
class Solution:
def leadsToDestination(self, n: int, edges: List[List[int]], source: int,
destination: int) -> bool:
```