**Detect a Cycle in an Undirected Graph**

You are given an undirected graph represented by an adjacency list. Your task is to determine if the graph contains any cycle.

A cycle is formed if you can traverse through a sequence of edges that starts and ends at the same vertex, with at least one edge in between.

**Input:**

* An integer V representing the number of vertices in the graph.
* A list of edges, where each edge connects two vertices of the graph.

**Output:**

* Return true if the graph contains a cycle, otherwise return false.

**Examples:**

* Example 1  
  Input: V = 5, Edges = [[0, 1], [1, 2], [2, 3], [3, 4], [4, 0]]

Output: true  
Explanation: The edges form a cycle: 0 → 1 → 2 → 3 → 4 → 0.

**Constraints:**

* 1 ≤ V ≤ 10⁴
* 0 ≤ E ≤ 10⁴
* The graph is undirected.
* The graph may contain multiple edges between the same pair of nodes.
* There are no self-loops.

**Test Cases:**

1. Input: V = 5, Edges = [[0, 1], [1, 2], [2, 3], [3, 4], [4, 0]]

Output: true

1. Input: V = 3, Edges = [[0, 1], [1, 2]]

Output: false

1. Input: V = 4, Edges = [[0, 1], [1, 2], [2, 0]]

Output: true

**Edge Cases:**

1. If the graph has no edges, return false.
2. If the graph is a tree (V nodes, V-1 edges, and connected), return false since trees have no cycles.
3. If there are parallel edges between two nodes, return true since they form a cycle of length 2.