Assignment 6 Problem 10 Xinyanliu

1 Core Steps of the Approach

1.1 Using BFS for Bipartiteness Detection

We begin a **Breadth-First Search (BFS)** from any unvisited node and assign **layer indices**:

- The first node is assigned **layer 0**, its child nodes are assigned **layer 1**, its child nodes' child nodes are assigned **layer 0**, and so on.
- If we encounter a node that has already been visited:
  - **If the colors (layers) are different**, it is normal and we continue.
  - **If the colors (layers) are the same**, then we have found an **odd cycle**.

1.2 Backtracking to Find the Lowest Common Ancestor (LCA)

Upon detecting an **odd cycle**, we trace the two nodes who share the same layer to find their **least common ancestor**. The shortest odd cycle is then constructed as:

Path from LCA to the first node + Path from LCA to the second node + Direct edge between the two nodes

which forms an **odd-length cycle**. The length of this cycle (number of edges) is recorded.

1.3 Continuing BFS

- **Remove all parent nodes in the detected cycle** and restart BFS from the current node, marking it as **layer 0**.
- If another odd cycle is found, repeat the process and record all the odd cycles detected.

1.4 Selecting the Shortest Odd Cycle

Among all recorded odd cycles, we select the **one with the shortest length** as the final answer.

2 Complexity Analysis

- **BFS Complexity**:
  $$O(n+m)$$
  since each node and edge are processed at most once and have all been visited.
- **Backtracking to Find an Odd Cycle**: If we use some data structure (such as dictionary) to record the parents node for each node, then it will be faster to do the backtracking, therefore the time for all the backtracking is
  $$O(m)$$
  since each edge is backtracked only once, as they will be deleted after knowing they are odd cycles, the total cost does not exceed
  $$O(m).$$
- **Final Time Complexity**: The total complexity remains
  $$O(n+m)+O(m)=O(n+m),$$
  as BFS ensures that already processed parts are not redundantly visited.

3 Proof of Correctness

The key question is whether this method guarantees finding the **shortest odd cycle**, ensuring that we **"cannot fail to really find some shortest odd cycle."**

- **No Odd Cycle is Missed**:
  - Since BFS explores the entire graph and records every detected odd cycle, no odd cycle is overlooked.
- **Ensuring the Shortest Odd Cycle is Found**:
  - The **odd cycle detected in BFS by finding their nearest ancestor is always the shortest** within that BFS traversal, as BFS expands layer by layer.
  - The method identifies the **closest odd cycle to the each BFS root**, ensuring that it is the shortest.
- **No Shorter Odd Cycle is Missed**:
  - Since BFS continues searching for new odd cycles, even if multiple odd cycles overlap, this method will detect all possible odd cycles and ultimately choose the shortest one.