Q2:

3:

Use a BFS-based approach to check for bipartiteness. We are given an adjacency list of tuples.

Initialize a map to keep track of the specie (A, B, or null) of each butterfly.
Initialize a map to keep track of the edges in our graph (convert the input list)
For each node in our graph:
    If the specie already exists, skip it.
    Else, start BFS:
        For each layer, assign opposite specie to each ‘null’ node, and validate that each non null node is the opposite.
        If validation fails, return False
Return True

Q3:

1:

Reverse the graph, conduct topological sort, then return part of this result for each node.

For each edge in the adjacency list:
    Add the reverse to a map.
For each node:
    Conduct DFS on the reversed graph to find its descendants.
    Answer[i] will contain these nodes.

2:

The time complexity of reversing the graph is $O(m)$ where m is the number of edges.

Building the answer list takes $O(n^2)$ in the worst case.

Total: $O(n^2)$

3:

- A node u is an ancestor of node v if there exists a directed path from u to v in the original DAG.
- In a reversed graph, there must instead be a directed path from v to u.
- By conducting DFS on node v, all pairs of u, v will be found because u will be reachable from v in the reversed graph.

4:

- We should compute topological sort on the reversed graph, and build the answer list in this order.
- In a topological sort on reversed graph, any nodes u, v where u is an ancestor of v will have v before u.
- Thus, we are guaranteed that ancestors of a previously computed node u will be v's ancestors as well.
- The time complexity is reduced to $O(n + m)$, assuming that array copy is $O(1)$.