Course Scheduling II

This problem is asking for a topological ordering of courses based on prerequisites:
- Courses = nodes
- Prerequisites = directed edges ($b_i \rightarrow a_i$)
- If a valid topological ordering exists (graph has no cycles), return it.
- If there is a cycle, return an empty array.

Approach: BFS (Kahn's algorithm)

Steps:
1. Build an adjacency list for the graph.
2. Compute in-degrees (number of prerequisites) for each course.
3. Initialize a queue with nodes having in-degree 0 (no prerequisites).
4. While queue is not empty:
   - Pop a node, add it to result.
   - Reduce in-degree of its neighbors.
   - If any neighbor's in-degree becomes 0, push it to queue.
5. If result size == numCourses → return result, else return empty array (cycle detected).

Complexity:
Time: - building graph: $O(V+E)$
- BFS traversal: $O(V+E)$
- total: $O(V+E)$ where V = numCourses and E = prerequisites.length

Space: - $O(V+E)$ for adjacency list and in-degree array.