Course Schedule

This problem is equivalent to detecting if a directed graph has a cycle:
- Courses = nodes.
- Prerequisites = directed edges ($b_i \to a_i$)
- If there's no cycle, we can finish all courses (topological ordering exists).
- If there's a cycle, it's impossible to finish all courses.

Approach: DFS cycle detection

Steps:
1. Build graph: adjacency list from prerequisites.
2. Use DFS to detect cycles:
   - Maintain three states for each node:
     - 0: unvisited
     - 1: visiting (currently in recursion stack)
     - 2: visited (processed)
   - If we encounter a node marked 1 during DFS, we found a cycle → return false.
3. If no cycles found → return true.

Complexity:

Time: - building graph: $O(V+E)$
- DFS traversal: $O(V+E)$
- Total: $O(V+E)$ where V = numCourses, E = prerequisites.length

Space: - $O(V+E)$ for adjacency list and recursion stack.