Homework 5: Changes to hw4

I made several strategic changes to the hw4 code to optimize performance and memory usage for the large-scale professor paths assignment:

Optimization of Graph Implementation

- **Disabled `checkRep()` for Performance**: The most significant change was ensuring the `CHECK_REP_ENABLED` flag in `Graph.java` was set to `false`. This optimization eliminated the expensive representation invariant checks during performance-critical operations, substantially improving both memory usage and execution time when working with large datasets.
- **Retained Core Functionality**: The core Graph ADT implementation and API remained unchanged, as the existing functionality provided all the necessary features:
  - Adding nodes (professors) with string identifiers
  - Adding directed edges with string labels (courses)
  - Retrieving children with their edge labels (needed for BFS)
  - Support for multiple edges between the same pair of nodes (allowing professors to teach multiple shared courses)

BFS Implementation Optimizations

- **Lexicographical Path Finding**: Improved the breadth-first search algorithm by separating professor node sorting from course sorting. This ensured that paths are truly lexicographically ordered while minimizing performance impact.
- **Memory-Efficient Collection Usage**: Used efficient data structures for tracking visited nodes, maintaining the queue, and storing paths during BFS traversal. These optimizations were critical for processing large networks within the 256MB memory constraint.
- **Efficient Path Construction**: Implemented an efficient path construction mechanism that builds paths incrementally during traversal rather than reconstructing them afterward, reducing both time and memory overhead.

Test Dataset Generation

- Created Python scripts (`generate_medium.py` and `generate_larger.py`) for generating test datasets of different sizes to properly validate performance and memory usage.
- Generated a large test network with 5,000 professors and 1,000 courses that creates approximately 1.5 million potential edges, providing a rigorous test for the implementation.

These changes enabled the ProfessorPaths implementation to efficiently process large networks within the required memory constraints while maintaining correct functionality and performance requirements.