CS 5200 Homework - 4

Instructor Avah Banerjee
Due: November 15, 11:59PM

Introduction

The graph isomorphism problem is a fundamental question in computer science, which asks whether two graphs can be considered structurally the same, even if their representations differ. This problem is not just of theoretical interest; it has practical applications in fields such as chemistry, where molecules can be represented as graphs, and in social network analysis, where the patterns of connections may reveal similar structures despite different contexts.

The challenge of graph isomorphism lies in its computational complexity. The problem is one of the few in the complexity class NP (nondeterministic polynomial time) that is neither known to be solvable in polynomial time (P) nor known to be NP-complete. The most efficient algorithm known to date for general graph isomorphism runs in quasi-polynomial time, which is faster than exponential but not quite polynomial. This algorithm, due to Babai (2015), has a running time of $\exp\left((\log n)^{O(1)}\right)$, where n is the number of vertices in the graphs. Understanding and improving graph isomorphism algorithms is important not only for the problem itself but also because it could shed light on the broader P vs NP question, one of the seven Millennium Prize Problems in mathematics.

In this assignment, we will explore a signature-based method for graph isomorphism testing, which offers an innovative angle on this classic problem. By analyzing the limitations and performance of this method, we can gain insights into the complexities of graph isomorphism.

Graph Isomorphism

Graph isomorphism is a type of equivalence relation between two graphs. Two graphs, G and H, are said to be isomorphic if there exists a bijection $f : V(G) \to V(H)$ such that any two vertices u and v are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. This bijection is known as an isomorphism.

To illustrate, consider the following two simple graphs:

In the example given in Figure 1, one can demonstrate an isomorphism between the two graphs by the mapping $f(1) = A$, $f(2) = B$, $f(3) = C$, and $f(4) = D$. It is easy to verify that for every edge that exists between two vertices in graph G, there is a corresponding edge between the two vertices in graph H to which they map.

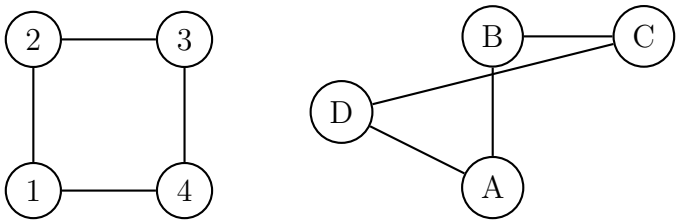

Figure 1: Two isomorphic graphs, G and H.

Tasks

1. Write a function to generate all shortest path trees (SPTs) from each vertex of a given graph.
2. Define and implement the signature of a graph using the collection of its SPTs.
3. Create an algorithm that compares two signatures to test for isomorphism.
4. Analyze the time complexity of your algorithm.

(Extra Credit) Counterexample Construction

1. Construct two non-isomorphic graphs, G and H, each with 6 vertices, that have matching SPT signatures according to your algorithm from Part 2.
2. Provide a detailed explanation of why these graphs are not isomorphic, despite having matching signatures.

Deliverables

1. Document the process of implementing the algorithm.
2. Present your analysis on the time complexity.
3. **Code**: A well-documented codebase containing the implementation of the algorithm (in Python). [You may use NetworkX to check whether two trees are isomorphic (see here), but you may not use NetworkX or any other external code to directly check if two graphs are isomorphic.]
4. Discuss the constructed counterexample and the implications for the SPT signature method (optional).
5. You may use NetworkX (see here) to generate random graphs for testing your algorithm. Choose $n = 10, 50, 100$ and $p = 1/\log n$.
6. GTAs will provide further instructions about specific code formatting, input and test requirements and other submission instructions.

Evaluation Criteria

1. Correctness and efficiency of the algorithm.
2. Quality of the code and documentation.
3. Accuracy in the time complexity analysis.