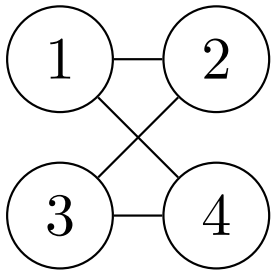
Graph Isomorphism

Jacob Urisman

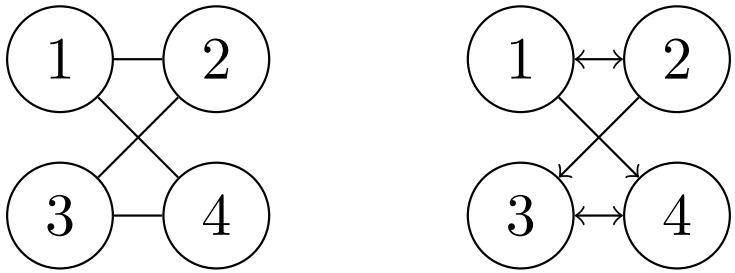
November 20, 2025

Overview

Graph Basics



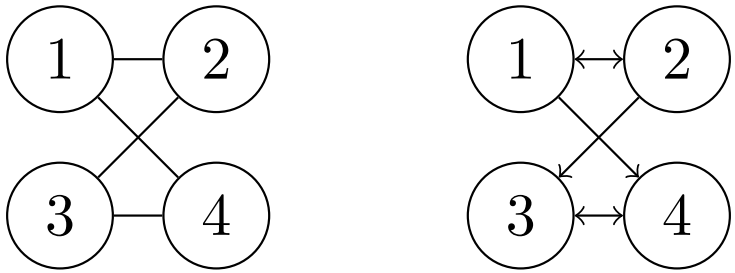
Graph Basics



Definition

A graph G is defined by its vertex (or node) set $V = \{1, \ldots, n\}$ and edge set $E \subseteq V \times V$.

Graph Basics



Definition

A graph G is defined by its vertex (or node) set $V = \{1, \ldots, n\}$ and edge set $E \subseteq V \times V$.

Elements in E look like (u, v) where $u, v \in V$. If G is directed, this means the edge goes from u to v. If G is undirected, then $(u, v) \in E \iff (v, u) \in E$.

Why Do We Care About Graphs?

Graphs are very useful for modeling networks. For example, think of social media networks. Let V be the set of all users of a social media app and let E be the follower connections, i.e. if $(u, v) \in E$, then u follows v.

Graph Isomorphism

One natural question that arises when studying graphs is:

Question

When are two graphs structurally the same? In fact, what do we mean by "structurally the same?"

Graph Isomorphism

One natural question that arises when studying graphs is:

Question

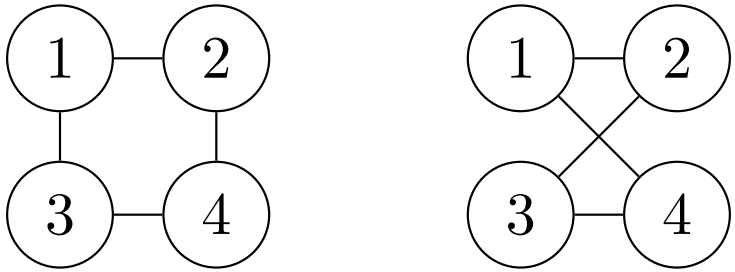
When are two graphs structurally the same? In fact, what do we mean by "structurally the same?"

Definition (Isomorphism)

Two graphs G, H are isomorphic if $\exists f : V_G \to V_H$ such that $\forall (u, v) \in E_G : (f(u), f(v)) \in E_H$. We call f the isomorphism between G and H.

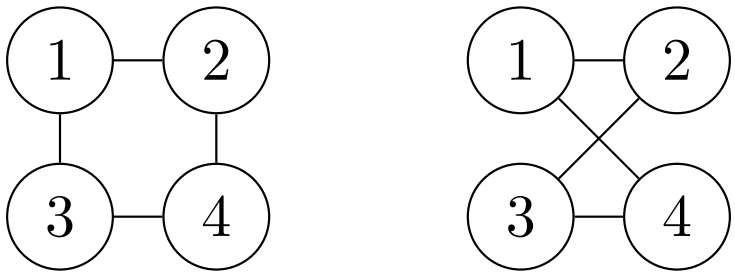
Graph Isomorphism II

Consider the following two undirected graphs. Are they isomorphic? If so, what is an example f?



Graph Isomorphism II

Consider the following two undirected graphs. Are they isomorphic? If so, what is an example f?



They are isomorphic! Ex. $f(1) = 1, f(2) = 2, f(3) = 4, f(4) = 3$.

Note that two graphs G, H, where $|V_G| \neq |V_H|$ or $|E_G| \neq |E_H|$ cannot be isomorphic.

Why Should We Care About Graph Isomorphism?

Graph isomorphism is one of the oldest problems in complexity theory and theoretical computer science in general. It was talked about in both Cook's and Levin's papers on what eventually became the Cook-Levin Theorem and also in Karp's 1972 paper giving the earliest reductions between many of the standard NP-complete problems.

Why Should We Care About Graph Isomorphism?

Graph isomorphism is one of the oldest problems in complexity theory and theoretical computer science in general. It was talked about in both Cook's and Levin's papers on what eventually became the Cook-Levin Theorem and also in Karp's 1972 paper giving the earliest reductions between many of the standard NP-complete problems.

Despite that, it is one of just a few problems that both has no known polynomial time algorithm (i.e. not known to be in P) AND it is not known to be NP-complete.

In fact, this has remained true since those earliest complexity theory papers.

Are things hopeless?

No, things are not hopeless. In fact, there have been very large strides forward quite recently. First, we need to establish the fundamentals.

The Weisfeiler-Lehman Algorithm

The most helpful algorithm for research purposes is the Weisfeiler-Lehman algorithm (also often spelled Leman). This algorithm works by iteratively coloring nodes.

The Weisfeiler-Lehman Algorithm

The most helpful algorithm for research purposes is the Weisfeiler-Lehman algorithm (also often spelled Leman). This algorithm works by iteratively coloring nodes.

Let χ be a fully injective coloring function from multisets of colored nodes or multisets of multisets of colored nodes to colors. Fully injective means that if two input multisets are different, χ will output two different colors.

The Weisfeiler-Lehman Algorithm II

For simplicity, let us work with undirected graphs.

The Weisfeiler-Lehman Algorithm II

For simplicity, let us work with undirected graphs.

Steps:

1. Color each node by it's number of neighbors (or equivalently its degree).

The Weisfeiler-Lehman Algorithm II

For simplicity, let us work with undirected graphs.

Steps:

1. Color each node by it's number of neighbors (or equivalently its degree).
2. For each node u, take the multiset of its color and the colors of all of u's neighbors and feed that into χ. Color u by the output of χ.

The Weisfeiler-Lehman Algorithm II

For simplicity, let us work with undirected graphs.

Steps:

1. Color each node by it's number of neighbors (or equivalently its degree).
2. For each node u, take the multiset of its color and the colors of all of u's neighbors and feed that into χ. Color u by the output of χ.
3. Repeat step two until the colors stabilize.

The Weisfeiler-Lehman Algorithm II

For simplicity, let us work with undirected graphs.

Steps:

1. Color each node by it's number of neighbors (or equivalently its degree).
2. For each node u, take the multiset of its color and the colors of all of u's neighbors and feed that into χ. Color u by the output of χ.
3. Repeat step two until the colors stabilize.

If two graphs stabilize to different colorings after running Weisfeiler-Lehman on both, they are distinguished by the algorithm. This is known as the 1-dimensional Weisfeiler-Lehman algorithm.

The k-dimensional Weisfeiler-Lehman Algorithm

Very similar to 1-dimensional, but instead of coloring a node based on its color and its neighbors colors, it colors k-tuples of nodes based on the multiset of multisets of neighbors for each node in the k-tuple.

CFI Graphs

In 1992, Cai, Fürer and Immerman (Emeritus at UMass) proved that there exist pairs of graphs which can be distinguished by at least k-dimensional Weisfeiler-Lehman for any $k \geq 0$ and gave a construction for such graphs for every k.

CFI Graphs

In 1992, Cai, Fürer and Immerman (Emeritus at UMass) proved that there exist pairs of graphs which can be distinguished by at least k-dimensional Weisfeiler-Lehman for any $k \geq 0$ and gave a construction for such graphs for every k.

In that same paper, they prove that two graphs which need r rounds of k-dimensional Weisfeiler-Lehman to be distinguished can be distinguished by a first order logical sentence with $k+1$ quantifiers and r variables.

Babai's Quasipolynomial Algorithm

In 2017, Laszlo Babai gave an algorithm for distinguishing non-isomorphic graphs in $O(2^{(\log n)^c})$ for n-node graphs and some constant c. The algorithm was based on the Weisfeiler-Lehman algorithm (although many, many speedups were added).

Babai's Quasipolynomial Algorithm

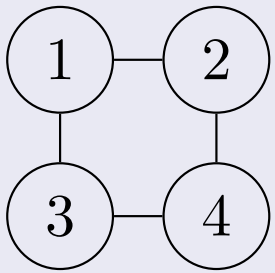
In 2017, Laszlo Babai gave an algorithm for distinguishing non-isomorphic graphs in $O(2^{(\log n)^c})$ for n-node graphs and some constant c. The algorithm was based on the Weisfeiler-Lehman algorithm (although many, many speedups were added).

Harald Helfgott then showed that $c = 3$, so the full running time is $O(2^{(\log n)^3})$.

Adjacency Matrices

For a graph Γ, you can represent it using a matrix as your data structure. If there is an edge from node i to node j in Γ, then the cell in row i and column j is a 1. If there is no edge, then the cell is a 0.

Example



0	1	1	0
1	0	0	1
1	0	0	1
0	1	1	0

Note that the matrix is symmetric about the diagonal because the graph is undirected. There are also no 1's on the diagonal because there are no self-loops.

Polynomials

Let x_{ij} be a variable representing the cell at row i and column j. This variable takes on the value 0 or 1 when evaluated at a specific graph.

Question

Are there polynomials which distinguish graphs?

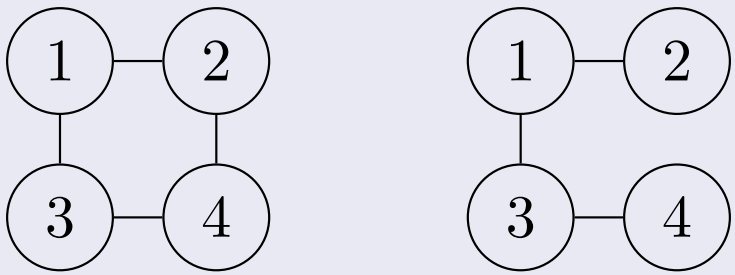
Polynomials

Let x_{ij} be a variable representing the cell at row i and column j. This variable takes on the value 0 or 1 when evaluated at a specific graph.

Question

Are there polynomials which distinguish graphs?

Example



The two graphs are distinguished by the following polynomial:

$$\left(\sum_{1 \leq i < j \leq n} x_{ij} \right) - 4$$

Thank you!