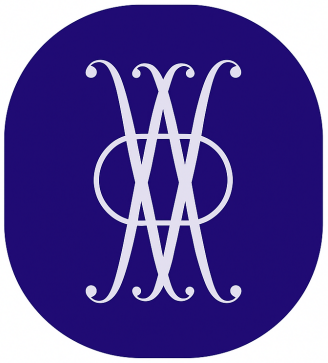
Graph Theory

(Graph Isomorphisms Game)



Sandro Rodríguez Muñoz

February 16, 2026

YouTube Channel: Sandrodmun

Interactive Animation: Graph Isomorphisms Game

Contents

1 Introduction: What is a Graph?

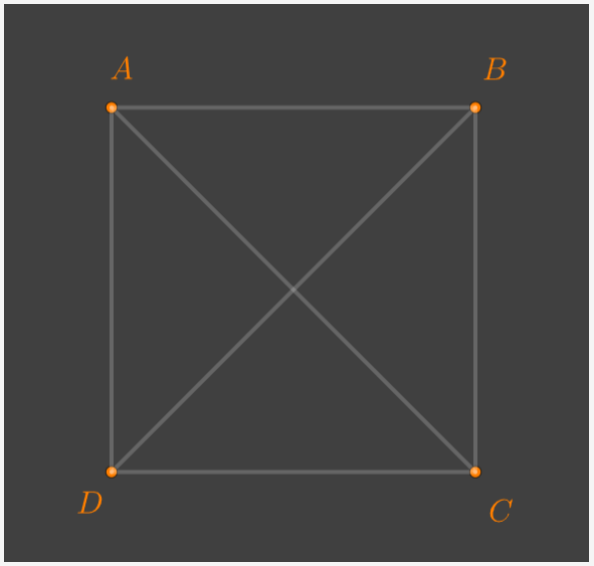
In mathematics, a **graph** is a way of representing relationships between different objects. A graph G consists of two main components:

- **Vertices (or Nodes):** These are the points representing the objects themselves. We usually denote the set of vertices as V.
- **Edges:** These are the lines that connect two vertices, representing a relationship between them. We denote the set of edges as E.

We represent a graph as $G = (V, E)$. For the purpose of this study, we will focus on **undirected graphs without loops**, meaning that an edge between point A and point B is the same as an edge between B and A, and no point is connected to itself.

Example 1.1. Square Graph

Let's consider the following graph:



The set of vertices is $V = \{A, B, C, D\}$, and the set of edges is

$$E = \Big\{\{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \{C, D\}\Big\}. \tag{1}$$

Notice that an edge is represented as a **set containing the two vertices that are connected**, and as such, **the order doesn't matter**, namely, the edge joining A and B can be written either as $\{A, B\}$ or as $\{B, A\}$. ■

2 Graph Isomorphism

One of the most interesting aspects of graph theory is that the **way we draw a graph does not change what the graph is**. As long as the connections (edges) between the points (vertices) remain the same, the graph is mathematically identical.

We say that two graphs G_1 and G_2 are **isomorphic** if they are "the same" in terms of their structure. Formally, two graphs are isomorphic if there exists a **bijective function** f that maps the vertices of G_1 to the vertices of G_2 such that:

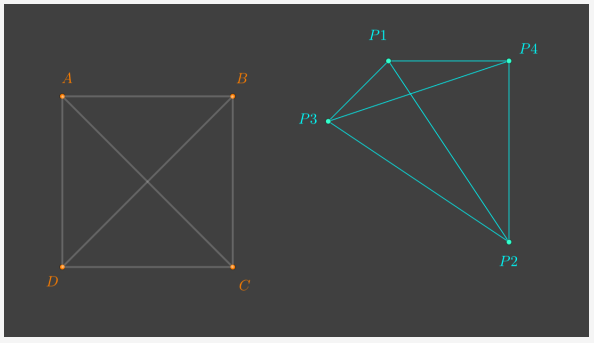
$$\{u, v\} \in E_1 \iff \{f(u), f(v)\} \in E_2. \tag{2}$$

In simpler terms, if two vertices are connected in the first graph, their corresponding images must be connected in the second graph, and vice versa. This condition is called **edge preservation**.

Such a function is called an **isomorphism**.

Example 2.1. Isomorphic Squares

Let's consider the square graph from the previous example G_{Square}, and the following graph G_2, drawn in blue:



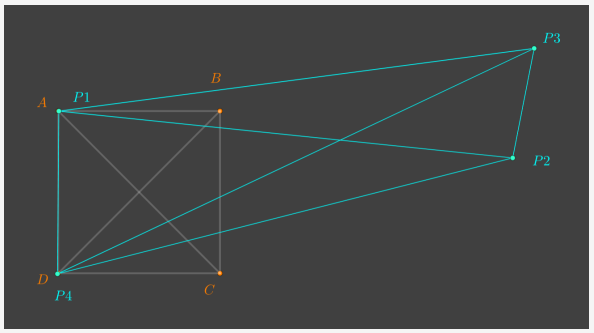
Although they look different, they are, in essence, the same graph, namely, they are **isomorphic**. Indeed, we can define a function $f : G_{Square} \to G_2$ which is **bijective** and which **preserves the edges** as follows:

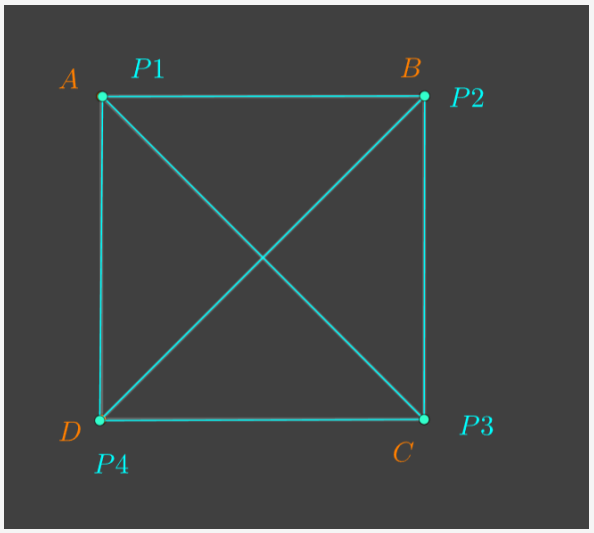
$$\begin{aligned} f(A) &= P1 \\ f(B) &= P2 \\ f(C) &= P3 \\ f(D) &= P4 \end{aligned} \tag{3}$$

This definition tells us that f is bijective, and it can be easily checked that f preserves the edges. For instance, the edge $\{C, D\} \in E_{Square}$ gets mapped to the edge $\{f(C), f(D)\} = \{P3, P4\}$, which does indeed exist, because $P3$ and $P4$ ae connected. We can do this for every edge and conclude that f is indeed edge preserving.

This means that $G_{Square} \cong G_2$, namely, both graphs are **isomorphic**.

To check that both graphs are the same more visually, we can drag the vertices of G_2 and try to make the graph coincide exactly with G_{Square}





Notice that f is **not the only isomorphism between** G_{Square} **and** G_2. For instance, the function $g : G_{Square} \to G_2$ defined as follows is also an isomorphism:

$$\begin{aligned} g(A) &= P2 \\ g(B) &= P3 \\ g(C) &= P4 \\ g(D) &= P1 \end{aligned} \tag{4}$$

■

3 Guide for Using the Applet

Now that we have seen what it means for graphs to be isomorphic, we will explain how we can play around with this concept using the following app: App:Graph Isomorphisms Game.

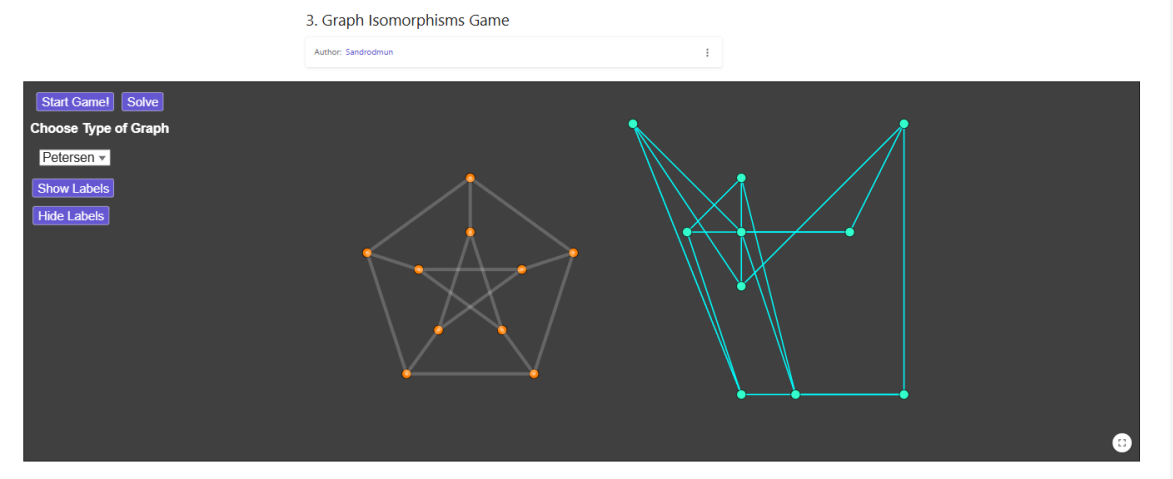
The goal of this applet is to help develop an intuition for graph structures. In the game, we have two graphs:

1. **The Target Graph:** A static graph drawn in its "standard" or most recognizable form (like a Petersen graph or the square graph of the example we have just seen).
2. **The Random Graph:** A graph that is **isomorphic** to the target, but its vertices have been placed at random positions.

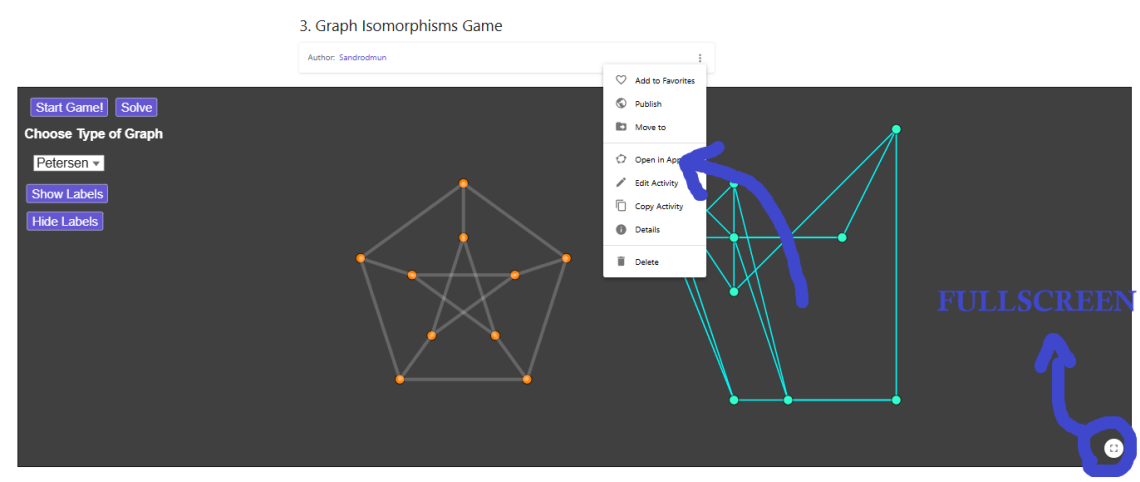
Our task is to **drag the vertices** of the random graph and place them exactly on top of the corresponding vertices of the target graph until the shapes match perfectly. There might be many ways of doing this, or they might only be one way of doing it. As the graphs start having more vertices and more edges the difficulty increases rapidly.

Guide 3.1. Choosing a Function

1. Once we click on the link, we will be taken to the following window:



2. There are 2 options:
   - Click the **Fullscreen button**:
     - We see the app in fullscreen
     - We will **not** be able to edit anything.
   - Click the **3 dots** next to the **author label**, and select **Open in App**:
     - You will be able to edit anything you want
     - You can change the sizes of the windows
     - You can see it fullscreen too
     - You can hide, delete, add, etc., anythnig you want



I recommend choosing **Open in App** always, as it is much more flexible.

■

Guide 3.2. Starting the Game

1. **Select a Graph Type:** Use the menu to choose between famous graphs:
   - **Square:** THis is the graph we saw in the example.
   - **Petersen Graph:** A famous graph with 10 vertices and 15 edges.
   - **Heawood Graph:** A highly symmetric graph with 14 vertices.
   - **Coxeter Graph:** A more complex symmetric graph with 28 vertices.
2. **Custom Mode:** If you want a unique challenge, select the **Random** option. Here, you can use the sliders to choose the number of **points** and **edges** you want the graph to have.
3. **Generate:** Click the **Start Game** button to generate a new scrambled version of your chosen graph.

■

Guide 3.3. Playing and Tools

- **Dragging Vertices:** Click and hold any vertex P_i in the random graph to move it.
- **Show/Hide Labels:** If you are stuck, you can toggle the labels $(P_1, P_2, \dots)$.

This helps you identify which vertex in the scrambled graph corresponds to the underlying structure.

- **The Solve Button:** If the puzzle is too difficult, clicking **Solve** will automatically move all vertices of the random graph to their correct positions on top of the target graph.

■