Checking Isomorphism Between Two Graphs

Anshul Anand
IIIT Allahabad
Email: icm2014501@iiita.ac.in

Akanshu Gupta
IIIT Allahabad
Email: iwm2014501@iiita.ac.in

Amit Khanna
IIIT Allahabad
Email: iim2014004@iiita.ac.in

***Abstract*—Consider two graphs G and H with given Adjacency Matrices A_G and A_H , in this work we propose an algorithm to find whether the given graphs are isomorphic or not . Further we evaluate the proposed algorithm's time complexity by the help of a plot of Time vs Number of nodes.**

***Index Terms*—Graph Theory, Isomorphism,Adjacency Matrix,Time Complexity**

I. Introduction

An interconnection of points is known as Graphs , or in a more formal parlance , A graph ***G*** is a set of ***E*** and ***V*** , where ***E*** denotes the set of edges and ***V*** denotes the set of Vertices . Here a vertice v ∈ ***V*** represents a point or node and an edge e ∈ ***E*** is a connection between two points vertices. If an edge e$_{ij}$ connects the vertices v$_i$ and v$_j$ then v$_i$ and v$_j$ are known as **endpoints** of that edge.

A. Adjacency Matrix

The adjacency matrix, sometimes also called the connection matrix, of a simple labeled graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether $v_i and v_j$ are adjacent or not. For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal. For an undirected graph, the adjacency matrix is symmetric.

B. Graph Isomorphism

Two graphs which contain the same number of graph vertices connected in the same way are said to be isomorphic. Formally, two graphs G and H with graph vertices $V_n = 1, 2, ..., n$ are said to be isomorphic if there is a permutation p of V_n such that u,v is in the set of graph edges E(G) iff p(u),p(v) is in the set of graph edges E(H).

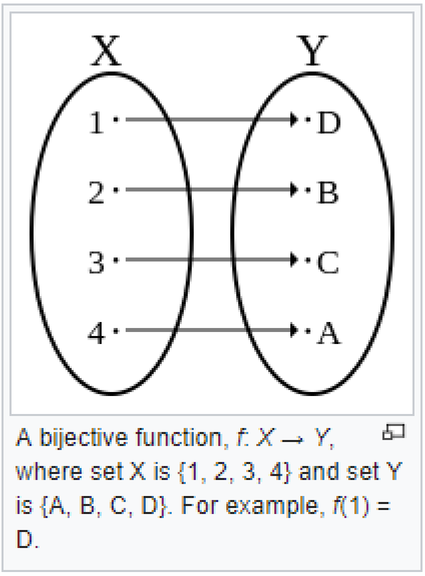
In a more formal way , in graph theory, an isomorphism of graphs G and H is a bijection between the vertex sets of G and H $f : V(G) \to V(H)$ such that any two vertices u and v of G are adjacent in G if and only if f(u) and f(v) are adjacent in H. This kind of bijection is commonly described as "edge-preserving bijection", in accordance with the general notion of isomorphism being a structure-preserving bijection.

If an isomorphism exists between two graphs, then the graphs are called isomorphic and denoted as $G \cong H$

The isomorphism problem consists of finding a mapping from the vertices of G to H such that they are identical. Such a mapping is called an isomorphism.

Fig. 1 shows two isomorphic graphs and the corresponding isomorphic function that exists between them .

1) Bijection: In mathematics, a bijection, bijective function or one-to-one correspondence is a function between the elements of two sets, where each element of one set is paired with exactly one element of the other set, and each element of the other set is paired with exactly one element of the first set. There are no unpaired elements. In mathematical terms, a bijective function f: X → Y is a one-to-one (injective) and onto (surjective) mapping of a set X to a set Y.



A bijective function, *f*: *X* → *Y*, where set X is {1, 2, 3, 4} and set Y is {A, B, C, D}. For example, *f*(1) = D.

Fig. 1. A bijective function

II. Motivation

The formal notion of "isomorphism", e.g., of "graph isomorphism", captures the informal notion that some objects have "the same structure" if one ignores individual distinctions of "atomic" components of objects in question. Whenever individuality of "atomic" components (vertices and edges, for graphs) is important for correct representation of whatever is modeled by graphs, the model is refined by imposing additional restrictions on the structure, and other mathematical objects are used: digraphs, labeled graphs, colored graphs, rooted trees and so on. The isomorphism relation may also be defined for all these generalizations of graphs: the isomorphism bijection must preserve the elements of structure which define the object type in question: arcs, labels, vertex/edge colors, the root of the rooted tree, etc.

The notion of ”graph isomorphism” allows us to distinguish graph properties inherent to the structures of graphs themselves from properties associated with graph representations: graph drawings, data structures for graphs, graph labelings, etc. For example, if a graph has exactly one cycle, then all graphs in its isomorphism class also have exactly one cycle. On the other hand, in the common case when the vertices of a graph are (represented by) the integers 1, 2,... N, then the expression

$$\sum_{v \in V(G)} v \cdot \deg v \text{ may be different for two isomorphic graphs.}$$

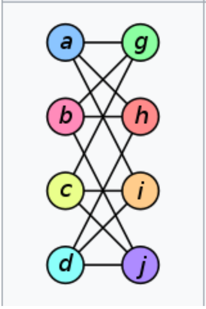
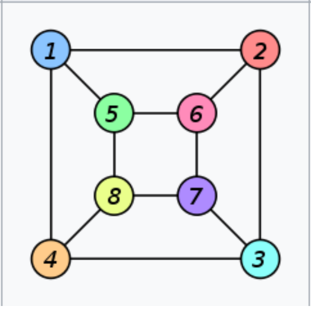
Graph G	Graph H	An isomorphism between G and H
		f(a) = 1 f(b) = 6 f(c) = 8 f(d) = 3 f(g) = 5 f(h) = 2 f(i) = 4 f(j) = 7

Fig. 2. Two isomorphic functions and the isomorphic map between them

III. Algorithm Proposed

A. Algorithm : CheckIsomorphism

IV. Results

The following are the plot for the given algorithm for graphs of the size from 1 to 10.

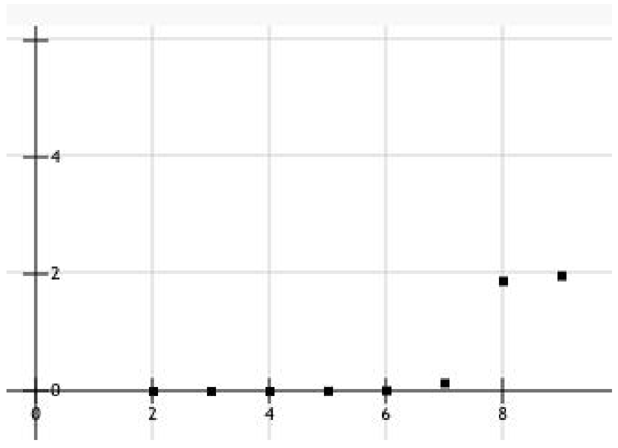


Fig. 3. n = 1 to 9

V. Conclusion

For a graph with N number of nodes, time complexity to find parallel edges is O(N!*N^3).

Algorithm 1 Check Isomorphism

```
if Number of vertices are different then
    Not Isomorphic
else if Number of edges are different then
    Not Isomorphic
else
    for all vertex in graph G do
        Calculate degree in $dG$
    end for
    for all vertex in graph H do
        Calculate degree in $dH$
    end for
    sort $dG$ and $dH$
    if $dG$ and $dH$ are different then
        Not Isomorphic
    else
        initialize $ar$ with 0 to n-1
        for all permutations of ar do
            generate $per$ such that all entries are 0 except $per[i][j]$=1 where $ar[i]=j$
            generate transpose of $per$ in $perT$
            if $per * H * perT = G$ then
                Isomorphic
                print permutation matrix
                return
            end if
        end for
    end if
    not isomorphic
end if
```

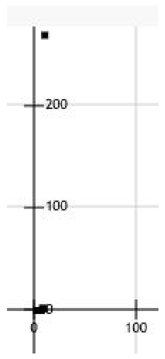


Fig. 4. n = 1 to 10

References

[1] www.uow.edu.au/ bmaloney/wuct121/GraphsWeek10Lecture2.pdf
[2] Jonathan L. Gross, Jay Yellen, *Handbook of Graph Theory*.
[3] www.wikipedia.org/wiki/Bijection
[4] http://mathworld.wolfram.com/IsomorphicGraphs.html