# GRAPH THEORY

## MINI-PROJECT REPORT

ALGORITHM FOR ISOMORPHISM

**Name : JAGADEESH R**

**Year : 2023**

# ISOMORPHISM

## AIM:

The aim of the program is to determine whether two given graphs are isomorphic or not. Graph isomorphism refers to a one-to-one correspondence between the vertices of two graphs in such a way that edges are preserved.

## ALGORITHM:

1. Check Vertex and Edge Counts:
    - If the number of vertices in both graphs is different, they are not isomorphic.
    - If the number of edges in both graphs is different, they are not isomorphic.
2. Degree Sequence Check:
    - For each vertex in both graphs, compare their degrees (number of adjacent vertices).
    - If the degree sequences are different, the graphs are not isomorphic.
3. Graph Isomorphism Testing:
    - Choose a starting vertex in the first graph.
    - For each vertex in the second graph, attempt to match it with the starting vertex from the first graph.
    - Recursively check if the remaining subgraphs are isomorphic.
    - If a valid mapping is found for all vertices, the graphs are isomorphic.
4. Backtracking:
    - If at any point during the recursive isomorphism testing, an invalid mapping is detected, backtrack and try a different mapping.
5. Output:
    - Displays the output as the given graph is isomorphic or not.

## IMPLEMENTATION TOOLS:

- Programming language C++
- G++ compiler

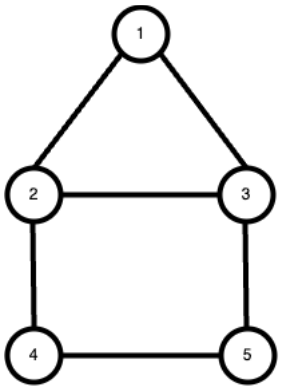
## SOURCE CODE:

```
#include <iostream>
#include <unordered_map>
#include <unordered_set>
#include <vector>

using namespace std;
bool isomorphic(const unordered_map<int, vector<int>>& graph1, const
unordered_map<int, vector<int>>& graph2) {
    // Check if the number of vertices is the same
    if (graph1.size() != graph2.size()) {
        return false;
    }

    // Check if the degree sequence is the same
    unordered_set<int> degrees1, degrees2;

    for (const auto& entry : graph1) {
        degrees1.insert(entry.second.size());
```

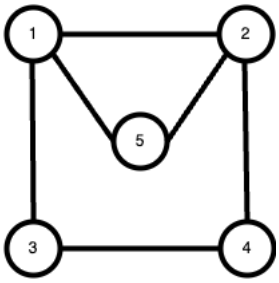
```
    }

    for (const auto& entry : graph2) {
        degrees2.insert(entry.second.size());

    }

    if (degrees1 != degrees2) {
        return false;

    }

    return true;
}

unordered_map<int, vector<int>> inputGraph() {
    unordered_map<int, vector<int>> graph;

    int numVertices, numEdges;
    cout << "Enter the number of vertices: ";
    cin >> numVertices;

    cout << "Enter the number of edges: ";
    cin >> numEdges;

    cout << "Enter the edges (vertex1 vertex2):" << endl;
    for (int i = 0; i < numEdges; ++i) {
        int vertex1, vertex2;
        cin >> vertex1 >> vertex2;

        // Add edges to the graph
        graph[vertex1].push_back(vertex2);
        graph[vertex2].push_back(vertex1);
    }

    return graph;
}

int main() {
    cout << "Enter details for the first graph:" << endl;
    unordered_map<int, vector<int>> graph1 = inputGraph();

    cout << "Enter details for the second graph:" << endl;
    unordered_map<int, vector<int>> graph2 = inputGraph();

    if (isomorphic(graph1, graph2)) {
        cout << "The graphs are isomorphic." << endl;
    } else {
        cout << "The graphs are not isomorphic." << endl;
    }

    return 0;
}
```

## SAMPLE INPUT GRAPHS:

Input Graph 1:

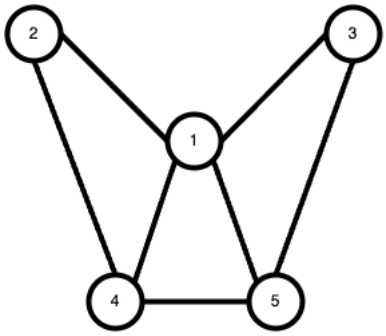



Sample Graph 1

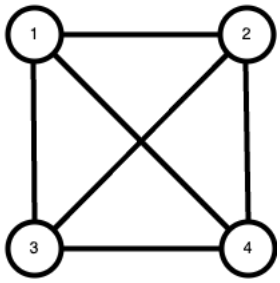



Sample Graph 2

Input Graph 2:



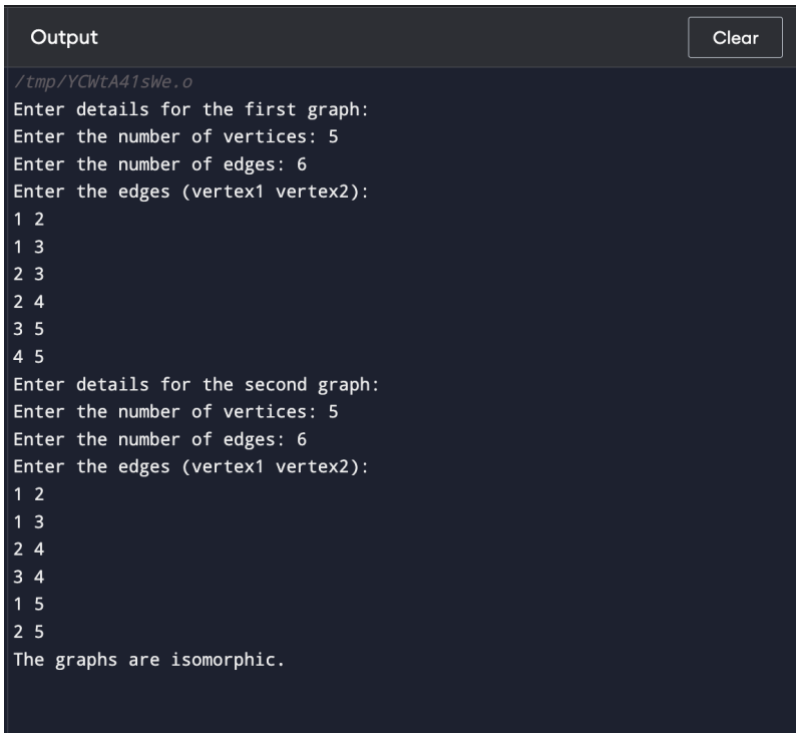

Sample Graph 3




Sample Graph 4

## OUTPUT:

Output for Input Graph 1:




Output for Input Graph 2: