Apply the concept of graph to solve the Konigsberg Bridge Problem using technological tool C++

Introduction Konigsberg Bridge Problem

The Konigsberg Bridge Problem is a famous mathematical puzzle that originated in the city of Konigsberg (now Kaliningrad, Russia) in the 18th century. The city was situated on the Pregel River and consisted of four land masses connected by seven bridges. The challenge was to find a path that would allow a person to cross each bridge exactly once and return to the starting point.

The problem attracted the attention of renowned mathematician Leonhard Euler, who analyzed and solved it in 1736. Euler approached the problem abstractly by representing the land masses as points and the bridges as lines connecting those points. He realized that the key to solving the problem lay in understanding the connections between the land masses and the bridges.

Euler introduced the concept of a graph, a mathematical structure that represents a set of objects (in this case, the land masses) and the relationships between them (the bridges). He proved that in order for a path to exist that crosses each bridge exactly once, there must be either zero or two land masses with an odd number of bridges connected to them.

In the case of the Konigsberg bridges, all four land masses had an odd number of bridges, violating Euler's criterion. Therefore, Euler concluded that it was impossible to find a path that satisfied the given conditions.

This problem and Euler's solution laid the foundation for graph theory, a branch of mathematics with applications in various fields, including computer science, network analysis, and optimization. The Konigsberg Bridge Problem remains an important historical puzzle that illustrates the power of abstract mathematical thinking and the development of graph theory.

Konigsberg Bridge Problem solved:

The Konigsberg Bridge Problem can be solved using various tools and programming languages., the problem is solved using C++, a popular programming language. The code utilizes basic data structures, such as vectors and matrices, to represent the graph and its connections.

The code employs a graph representation with an adjacency matrix to model the bridges and land masses. It defines a Graph class that allows the addition of edges (bridges) between vertices (land masses). The i**sEulerianCircuitPossible** function checks whether an **Eulerian** circuit is possible by verifying if all vertices have an even degree.

Code:

```
#include <iostream>
```

```
#include <vector>

using namespace std;

// Graph class
class Graph {
    int numVertices;
    vector<vector<int>> adjMatrix;

public:
    Graph(int vertices) : numVertices(vertices), adjMatrix(vertices, vector<int>(vertices, 0)) {}

    // Function to add an edge between two vertices
    void addEdge(int src, int dest) {
        adjMatrix[src][dest] = 1;
        adjMatrix[dest][src] = 1;
    }

    // Function to check if an Eulerian circuit exists
    bool isEulerianCircuitPossible() {
        // Check if all vertices have even degree
        for (int i = 0; i < numVertices; i++) {
            int degree = 0;
            for (int j = 0; j < numVertices; j++) {
                if (adjMatrix[i][j] == 1)
```

```
                degree++;
            }
            if (degree % 2 != 0)
                return false;
        }

        return true;
    }
};

int main() {
    int numVertices = 4;  // Number of land masses in Konigsberg

    // Create a graph
    Graph graph(numVertices);

    // Add the bridges as edges in the graph
    graph.addEdge(0, 1);
    graph.addEdge(0, 2);
    graph.addEdge(1, 2);
    graph.addEdge(2, 3);

    // Check if an Eulerian circuit is possible
    if (graph.isEulerianCircuitPossible())
        cout << "An Eulerian circuit is possible in the Konigsberg Bridge Problem." << endl;
```

```
    else
        cout << "No Eulerian circuit is possible in the Konigsberg Bridge Problem." << endl;

    return 0;
}
```