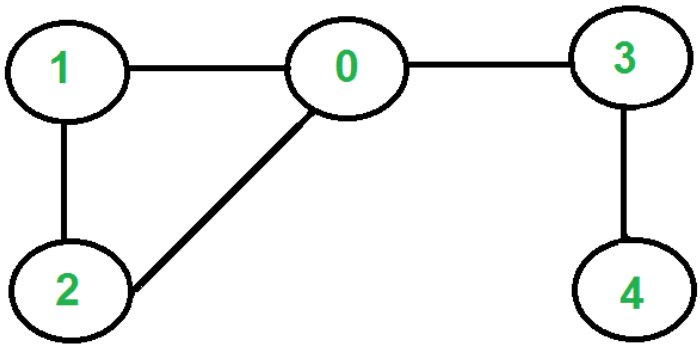
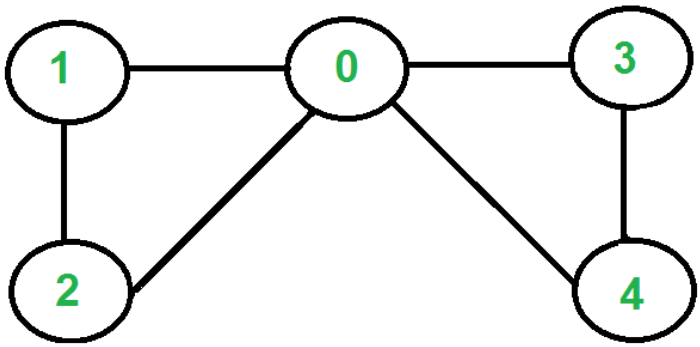
Eulerian path and circuit for undirected graph

Eulerian Path is a path in graph that visits every edge exactly once. Eulerian Circuit is an Eulerian Path which starts and ends on the same vertex.



The graph has Eulerian Paths, for example "4 3 0 1 2 0", but no Eulerian Cycle. Note that there are two vertices with odd degree (4 and 0)



The graph has Eulerian Cycles, for example "2 1 0 3 4 0 2" Note that all vertices have even degree

How to find whether a given graph is Eulerian or not?

The problem is same as following question. “Is it possible to draw a given graph without lifting pencil from the paper and without tracing any of the edges more than once”.

A graph is called Eulerian if it has an Eulerian Cycle and called Semi-Eulerian if it has an Eulerian Path. The problem seems similar to Hamiltonian Path which is NP complete problem for a general graph. Fortunately, we can find whether a given graph has a Eulerian Path or not in polynomial time. In fact, we can find it in O(V+E) time.

Following are some interesting properties of undirected graphs with an Eulerian path and cycle. We can use these properties to find whether a graph is Eulerian or not.

Eulerian Cycle

An undirected graph has Eulerian cycle if following two conditions are true.

….a) All vertices with non-zero degree are connected. We don’t care about vertices with zero degree because they don’t belong to Eulerian Cycle or Path (we only consider all edges).

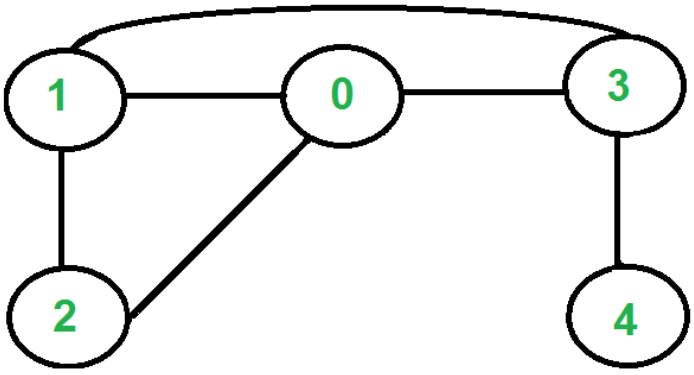
….b) All vertices have even degree.

Eulerian Path

An undirected graph has Eulerian Path if following two conditions are true.

….a) Same as condition (a) for Eulerian Cycle

….b) If zero or two vertices have odd degree and all other vertices have even degree. Note that only one vertex



The graph is not Eulerian. Note that there are four vertices with odd degree (0, 1, 3 and 4)

with odd degree is not possible in an undirected graph (sum of all degrees is always even in an undirected graph)

Note that a graph with no edges is considered Eulerian because there are no edges to traverse.

How does this work?

In Eulerian path, each time we visit a vertex v, we walk through two unvisited edges with one end point as v. Therefore, all middle vertices in Eulerian Path must have even degree. For Eulerian Cycle, any vertex can be middle vertex, therefore all vertices must have even degree.

```
// A C++ program to check if a given graph is Eulerian or not
#include<iostream>
#include <list>
using namespace std;

// A class that represents an undirected graph
class Graph
{
    int V;    // No. of vertices
    list<int> *adj;    // A dynamic array of adjacency lists
public:
    // Constructor and destructor
    Graph(int V)   {this->V = V; adj = new list<int>[V]; }
    ~Graph() { delete [] adj; } // To avoid memory leak

     // function to add an edge to graph
    void addEdge(int v, int w);

    // Method to check if this graph is Eulerian or not
    int isEulerian();

    // Method to check if all non-zero degree vertices are connected
    bool isConnected();

    // Function to do DFS starting from v. Used in isConnected();
    void DFSUtil(int v, bool visited[]);
};

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
    adj[w].push_back(v);  // Note: the graph is undirected
}

void Graph::DFSUtil(int v, bool visited[])
{
    // Mark the current node as visited and print it
    visited[v] = true;

    // Recur for all the vertices adjacent to this vertex
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            DFSUtil(*i, visited);
```

```
}

// Method to check if all non-zero degree vertices are connected.
// It mainly does DFS traversal starting from
bool Graph::isConnected()
{
    // Mark all the vertices as not visited
    bool visited[V];
    int i;
    for (i = 0; i < V; i++)
        visited[i] = false;

    // Find a vertex with non-zero degree
    for (i = 0; i < V; i++)
        if (adj[i].size() != 0)
            break;

    // If there are no edges in the graph, return true
    if (i == V)
        return true;

    // Start DFS traversal from a vertex with non-zero degree
    DFSUtil(i, visited);

    // Check if all non-zero degree vertices are visited
    for (i = 0; i < V; i++)
       if (visited[i] == false && adj[i].size() > 0)
            return false;

    return true;
}

/* The function returns one of the following values
   0 --> If grpah is not Eulerian
   1 --> If graph has an Euler path (Semi-Eulerian)
   2 --> If graph has an Euler Circuit (Eulerian)  */
int Graph::isEulerian()
{
    // Check if all non-zero degree vertices are connected
    if (isConnected() == false)
        return 0;

    // Count vertices with odd degree
    int odd = 0;
    for (int i = 0; i < V; i++)
        if (adj[i].size() & 1)
            odd++;

    // If count is more than 2, then graph is not Eulerian
    if (odd > 2)
        return 0;

    // If odd count is 2, then semi-eulerian.
    // If odd count is 0, then eulerian
    // Note that odd count can never be 1 for undirected graph
    return (odd)? 1 : 2;
}

// Function to run test cases
void test(Graph &g)
{
    int res = g.isEulerian();
    if (res == 0)
        cout << "Graph is not Eulerian\n";
    else if (res == 1)
```

```
        cout << "Graph has a Euler path\n";
    else
        cout << "Graph has a Euler cycle\n";
}

// Driver program to test above function
int main()
{
    // Let us create and test graphs shown in above figures
    Graph g1(5);
    g1.addEdge(1, 0);
    g1.addEdge(0, 2);
    g1.addEdge(2, 1);
    g1.addEdge(0, 3);
    g1.addEdge(3, 4);
    test(g1);

    Graph g2(5);
    g2.addEdge(1, 0);
    g2.addEdge(0, 2);
    g2.addEdge(2, 1);
    g2.addEdge(0, 3);
    g2.addEdge(3, 4);
    g2.addEdge(4, 0);
    test(g2);

    Graph g3(5);
    g3.addEdge(1, 0);
    g3.addEdge(0, 2);
    g3.addEdge(2, 1);
    g3.addEdge(0, 3);
    g3.addEdge(3, 4);
    g3.addEdge(1, 3);
    test(g3);

    // Let us create a graph with 3 vertices
    // connected in the form of cycle
    Graph g4(3);
    g4.addEdge(0, 1);
    g4.addEdge(1, 2);
    g4.addEdge(2, 0);
    test(g4);

    // Let us create a graph with all veritces
    // with zero degree
    Graph g5(3);
    test(g5);

    return 0;
}
```

Run on IDE

Output:

```
Graph has a Euler path
Graph has a Euler cycle
Graph is not Eulerian
Graph has a Euler cycle
Graph has a Euler cycle
```

Time Complexity: O(V+E)

We will soon be covering following topics on Eulerian Path and Circuit
1) Eulerian Path and Circuit for a Directed Graphs.
2) How to print a Eulerian Path or Circuit?

References:
http://en.wikipedia.org/wiki/Eulerian_path

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

19 Comments Category: Graph Tags: Graph

Related Questions:

- Karger's algorithm for Minimum Cut | Set 2 (Analysis and Applications)
- Steiner Tree Problem
- Karger's algorithm for Minimum Cut | Set 1 (Introduction and Implementation)
- Greedy Algorithms | Set 9 (Boruvka's algorithm)
- Assign directions to edges so that the directed graph remains acyclic
- K Centers Problem | Set 1 (Greedy Approximate Algorithm)
- Find the minimum cost to reach destination using a train
- Applications of Breadth First Traversal

Mr. Lazy • 4 months ago
http://www.graph-magics.com/ar...
• Reply • Share ›

black_bird • 7 months ago
Why this code is not working for this input??
5
1 2
2 3

3 4
4 5
5 6

• Reply • Share ›

see_language ➚ black_bird • a month ago

May be because you are inserting an edge from 5th vertex to 6th vertex in a graph with only 5 vertices. -_-

• Reply • Share ›

Guest • 8 months ago

pls correct the typo non-zero to zero
// Find a vertex with non-zero degree

• Reply • Share ›

Anurag Singh ➚ Guest • 8 months ago

It's correct. We are looking for vertex with non-zero degree.

• Reply • Share ›

disqus_UH0j6NNdcp • a year ago

there is an error in ((test(g1);))

and also in ((bool visited[V];)) - array with unknown size !

it didn't work when i tried to run it

• Reply • Share ›

Jon Snow ➚ disqus_UH0j6NNdcp • a year ago

What kind of error? When I run it on my machine it runs without any errors.

• Reply • Share ›

nitesh78 • a year ago

in alogo its written eulerian path can exist if their are 0 or 2 odd vertices but eulerian path can not exist if their are 0 odd vertices!!

1 • Reply • Share ›

GOPI GOPINATH ➚ nitesh78 • a year ago

Eulerian path + first vertex is nothing but Eulerian cycle. Hence if we have 0 odd vertices then its Eulerian cycle (as well as Eulerian path) and if 2 odd vertices Eulerian path alone.

1 • Reply • Share ›

rohit • 2 years ago

the last test result --> How a graph can be eulerian if there are no edges?? For last test

result that is g5, your result says that it has eulerian cycle. Please clarify.

• Reply • Share ›

Sudipta Sen • 2 years ago

Please can anybody explain me this line.

~Graph() { delete [] adj; } // To avoid memory leak

• Reply • Share ›

GOPI GOPINATH ➚ Sudipta Sen • a year ago

Its a destructor.

• Reply • Share ›

Castle Age ➚ Sudipta Sen • 2 years ago

In C++, "delete []" handles memory deallocation of dynamically allocated arrays. It is an undefined behavior if you call "delete" on a dynamically allocated array.

1 • Reply • Share ›

coder • 2 years ago

good work

• Reply • Share ›

JK • 2 years ago

Awesome Work!

• Reply • Share ›

GeeksforGeeks • 2 years ago

Thanks for pointing this out. We have corrected it.

• Reply • Share ›

Alexander Korobeynikov • 2 years ago

"Eulerian Path a path in graph that visits every vertex exactly once" - nope. It&#039s Hamiltonian path, not Eulerian.
Eulerian Path visits every _edge_ exactly once.

• Reply • Share ›

CareTaker • 2 years ago

ERROR:Line 1:Eulerian path is a path in which every EDGE is visited exactly once and not every vertex(which is a Hamiltonian Path).

• Reply • Share ›

GeeksforGeeks ➚ CareTaker • 2 years ago

Thanks for pointing this out. We have corrected it.