Hierholzer's Algorithm to find Eulerian Circuit

Biki Chaudhary
Email:
ism2014001@iiita.ac.in
Tara Prasad Tripathy
Email: ihm2014003@iiita.ac.in

Index Terms—Graphs, Directed Graphs, Undirected Graphs, Adjacency List, strongly connected component, Euler circuit

I. Introduction

A Euler path is a path that uses every edge of a graph exactly once. A Euler circuit is a circuit that uses every edge of a graph exactly once. A Euler path starts and ends at different vertices whereas a Euler circuit starts and ends at the same vertex. Examples of a Euler circuit are shown in figures 1 for given graph.

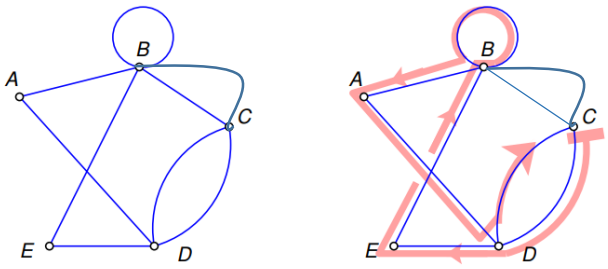


Figure 1 A Euler circuit: CDCBBADEBC

It is possible to determine whether a graph has a Euler circuit or not. The criteria is that all vertices with non-zero degree should be connected and every vertex must have an even degree if the graph is undirected and the graph should be strongly connected and the in-degree should be equal to the out-degree for each vertex if the graph is directed.

We can allow zero degree vertices to be disconnected because they don't have any edge and thus are not part of the Euler circuit. To understand the degree requirement let's suppose that a graph G has a Euler circuit C. For every vertex v in G, each edge having v as an endpoint shows up exactly once in C. The circuit C enters v the same number of times that it leaves v (say s times), so v has degree 2s. That is, v must be an even vertex. The requirement for an even degree is because we should be able to leave a vertex when we enter it. For the same reason there should equal number of edges going in and going out from a vertex in case of a directed graph.

II. Motivation

There are many useful applications of Euler circuits that motivates us to explore the solution to this particular task. With the knowledge about Euler paths and circuits we can solve many difficult problems, like the Konigsberg Bridge problem. They can also be used by mail carriers who want to have a route where they don't retrace any of their previous steps. This is because a mailman has to visit every road where there is mail but it's a waste to visit the same road again as all the mails would have delivered the first time. With the same logic Euler circuits can also be useful to painters, garbage collectors, airplane pilots and world navigators or tour planners.

III. Algorithms

A. Algorithm 1: Eulerian Circuit for Directed Graph

Input: Adjacency List with valid graph.
Output: Eulerian Circuit

1) Let us consider stack-current_path and vector-circuit which store the current path and required Eulerian circuit respectively.
2) Initially, insert any vertex having nonzero degree in the stack current_path which is our starting vertex.
3) Perform line 4-6 until stack is not empty.
4) Let, Vi = top of the stack and find if it is connected to any other vertices Vj using adjacency list.
5) If there is edge between vertex Vi to Vj then remove the edge Vi to Vj and push the vertex Vj in the current_path.
6) Else insert the vertex Vi in the vector circuit and remove vertex Vi from current_path.
7) Finally, print the elements in the vector circuit in reverse order which is our required Eulerian circuit.

B. Algorithm 2: Eulerian Circuit for Undirected Graph

Input: Adjacency List with valid graph.

Output: Eulerian Circuit

1) Let us consider stack-current_path and vector-circuit which store the current path and required Eulerian circuit respectively.
2) Initially, insert any vertex having nonzero degree in the stack current_path which is our starting vertex.

3) Perform line 4-6 until stack is not empty.
4) Let, Vi = top of the stack and find if it is connected to any other vertices Vj using adjacency list.
5) If there is edge between vertex Vi to Vj then remove the edge Vi to Vj and Vj to Vi and push the vertex Vj in the current_path.
6) Else insert the vertex Vi in the vector circuit and remove vertex Vi from current_path.
7) Finally, print the elements in the vector Circuit in reverse order which is our required Eulerian Circuit.

IV. Implementation and Results

The algorithms were implemented in C++. And the time vs. number of edges graph was plotted using GNUPLOT. The plot shown in Fig 3 and 4 was drawn by taking the number of edges in the x axis and the time in milliseconds in the y axis. For the purposes of plotting we used a fully connected complete graph without the self loops to obtain the running time for different vertex count.

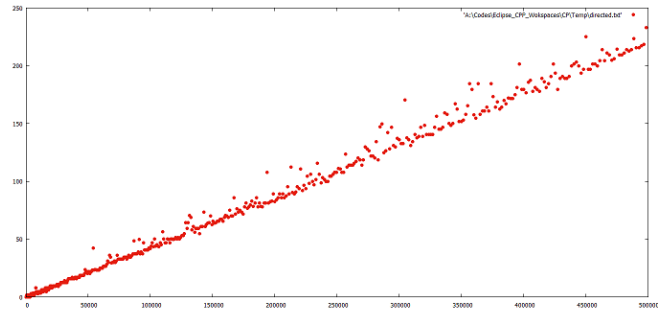


Figure 2 Plot of time vs. number of edges for a directed graph. From the curve it's easy to observe that the complexity of the algorithm presented is O (E).

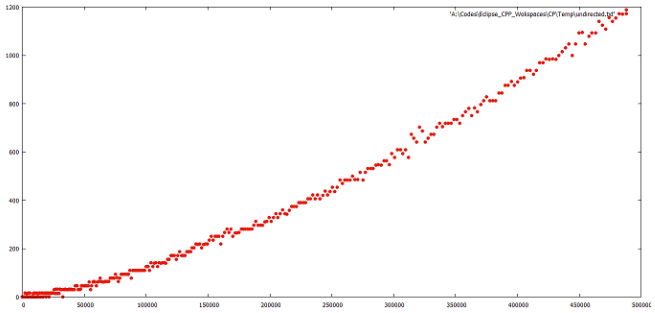


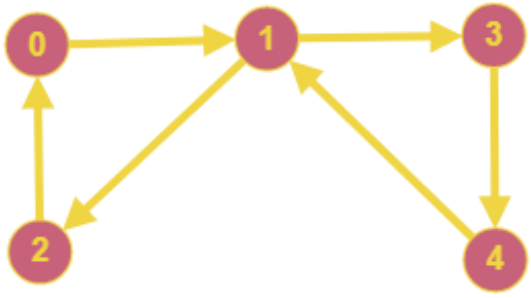
Figure 3 Plot of time vs. number of edges for an undirected graph.

The graph was implemented using adjacency list and we only find the Eulerian circuit in case such a circuit exist. As mentioned in Introduction the following conditions should be satisfied for directed graph to be valid to have Eulerian circuit: -

- All vertices with nonzero degree should be single strongly connected component.
- Every vertex in graph should have equal number of incoming and outgoing edges.

And, for undirected graph to have Eulerian cycle:

- All vertices with nonzero degree should be connected.
- All vertices should have even degree.



Here, in this graph the Eulerian Circuit is :

0=>1=>3=>4=>1=>2=>0

In this valid graph we start from 0 vertex and traverses all the edges exactly once in the graph and reach back to the starting vertex "0" which defines the eulerian circuit.

The complexity of both the algorithms is linear w.r.t. the number of edges i.e. O (E). The algorithm was implemented using stack to store the vertices where we have reached till now and array/vector to store the required Eulerian circuit. Initially, any vertex was inserted into stack and following tasks were done until stack becomes empty. We take the top of the stack and see if we can reach to any other vertices. If we find any reachable vertex then we insert that vertex into stack and remove the connecting edge between them from graph permanently. In case of undirected graph we remove both the outgoing and incoming edges.

Else if we get stuck to the vertex i.e. we cannot go to any other vertices then we take that vertex out from the stack and add to the array that stores our Eulerian cycle and similarly for undirected graph. Once the stack becomes empty, the final elements stored in our array gives our Eulerian cycle, taken in reverse order. This explains the O (E) total time complexity for this algorithm.

V. Conclusion

So, we discussed the Hierholzer's solution for the problem of finding a Euler circuit in both directed and undirected graph. The complexity is O(E), the best that could be achieved since we at least have to visit all possible edges to get the Euler circuit. This was further proven by the plot drawn from the running time observed for various number of vertices of a complete graph.