LP0: Euler Tour

Abstract:

A graph G is called Eulerian if it is connected and the degree of every vertex is an even number. It is known that such graphs always have a tour (a cycle that may not be simple) that goes through every edge of the graph exactly once. Such a tour (sometimes called a circuit) is called an Euler tour. If the graph is connected, and it has exactly 2 nodes of odd degree, then it has an Euler Path connecting these two nodes that includes all the edges of the graph exactly once. In this project, we implemented the Hierholzer's Algorithm to find an Euler tour or an Euler Path in a given graph.

Problem Statement:

a. Implement the Hierholzer's algorithm to find the Euler tour or Trail in the given graph such that the runtime of algorithm is O(|E|).
b. Implement the verify Euler Tour function such that the runtime of the algorithm is O(|E|)

Methodology:

Given Problem can be split into following sub problems:

1. Determine given graph is Euler graph or not.
    a. Graph is connected?
    b. Number of odd degree Vertices
2. If it is an Euler Graph with 0 odd degree Vertices find the Euler tour.
3. If it is an Euler Graph with 2 odd degree vertices find the Euler trail or path.
4. Verify the Euler Tour for the given graph

Determining given Graph is Euler Graph or Not

Graph is connected or not:

BFS Algorithm is used to find the whether the graph is connected or not. If BFS covers all the vertices in the graph then it is connected.

Graph is Euler or Not if it is connected

Check the size of the adjacency list of the vertices to determine whether the vertex has odd degree or not. Then if the graph has 0 odd degree vertex then it has a Euler Tour, 2 odd degree vertex then the graph has a Euler Trail or Path, else it is not a Euler Graph

Euler Tour

If the given graph is a Euler graph then we can determine the Euler tour. We have implemented the HierHolzer's Algorithm to find the Euler tour.

Finding unused edges exiting each vertex, finding a new starting vertex for a tour, and connecting two tours that share a vertex are done in constant time. So that the algorithm itself will run in O(|E|) time.

Vertex class modified to add the following fields:

1. DoublyLinkedList<Edge>.Entry<Edge> index : index to the First edge containing this vertex in the tour to achieve O(1) Run Time for Merging the tours
2. List<Edge> unUsedEdges : storing the unused edges during Euler tour

Data Structures used for HierHolzer's Algorithm:

DoublyLinkedList<Edge> currentTour: - to store the current tour in the linked list

DoublyLinkedList<Edge> fullTour: - to Store the complete tour in the linked list

LinkedList<Vertex> vertexWithUnusedEdgeList: to Store the vertex with unused edges

Note:

Doubly Linked List is implemented to merge the tour in the O(1) Time.

1. `void mergeListBefore(DoublyLinkedList<T>.Entry<T> index,DoublyLinkedList<T> secList) – Does the merging of the second List before the given index.`

   `Eg) fullTour.mergeListBefore(index,currentTour);`

2. `DoublyLinkedList<T>.Entry<T> addAndGetIndex(T x) – Add the given edge in the linked List and return the index(reference) in the linked list which in turn will be stored in the Vertex.index field for merging the tour.`

HierHolzer PseudoCode:

currentVertex=startVertex

currentTour={}

fullTour={}

while(currentVertex is not null)

1. Remove the unused Edge from the current vertex.
2. Add the edge to the current Tour and get the index (reference) of the edge in the tour.
3. If the current vertex have unused Edges and it is not in the VertexWithUnusedEdgeList(by checking the index field in the current vertex for null), then update the index field of the current vertex and add the current vertex into the VertexWithUnusedEdgeList.
4. Get the other end vertices of the Edge.
5. Remove the current edge from the other end vertex.
6. CurrentVertex <- Other End Vertex
7. If CurrentVertex == StartVertex Then //Found a circuit
    a. Join the current Tour with the Full Tour at the startVertex using the index in the StartVertex - O(1)
    b. StartVertex <- get the Next Vertex with Unused Edges.
    c. CurrentVertex <- StartVertex

Euler Trail or Path

Finding an Euler Trail is a challenge using the Hierholzer algorithm. As we cannot run the algorithm on a graph with Euler Path.

Following changes done to find Euler Graph

1. Added an fake edge between Odd Degree vertices to create an Euler Circuit – O(1)
2. Find the Euler Tour starting with smaller numbered vertices – O(|E|)
3. Construct an Euler Path from the Euler Tour by breaking the circuit at fake edge. Then rearranging the list – O (E).
4. Remove the fake edge form the given graph – O(1)

By using above procedure we are able to find the Euler Trail from the given graph in O(|E|).

Constructing Euler Path From the Euler Tour

For Constructing Euler path from the Euler Tour. We need to break the circuit at the fake edge.

Euler Tour is broken into List 1 -> fake Edge -> List 2.

List1-> contains the edge from the head of the euler tour to the fake edge.

List2-> contains the edge from the fake edge to the the tail of the tour.

fakeEdge -> edge added between the odd vertices.

We need to join the List 1 and List 2 to get the final Euler path. There are two cases which needs to be handled.

Let A and B be the odd degree Vertices. A be the small numbered Vertices

Case 1:

List 1: starts with A and ends with A

List 2: starts with B and ends with A

Input (Euler Tour):

List1(A1>....->A1)->fakeEdge(A1,B2)-> List2(B2->....->A2)

Operation:

List1 + Reversed (List2)

Output (Euler Path):

Euler Path A1->....->A1 + A2->.....->B2

Case 2:

List 1: starts with A and ends with B

List 2: starts with A and ends with A

Input (Euler Tour):

List1(A1>....->B1)->fakeEdge(B1,A2)-> List2(A2->....->A2)

Operation:

List2 + List1

Output (Euler Path):

Euler Path A2->....->A2 + A1->.....->B1

Verify Tour

It is used to used determine given a Euler tour or path for a given graph. Whether the path is valid or not in O (|E|).

Edge Class:

Traversed Boolean field is added to the edge class to determine whether the edge is traversed or not.

Conditions to be checked:

a. All the edges in the Euler Tour should be adjacent to each other.
b. All the Edges in the graph should be visited exactly once.

Development platform

Operating System: Windows 8

Language: Java 8

Processor: Core I5

RAM: 4 GB

Running Time

Euler Tour or Euler Path for graph with 500000 vertices and 5249924 edges: ~ **30 Seconds**

Verify Euler Tour for the same Graph: ~ 350 ms.

Conclusion

We can observe from the result that HierHolzer's Algorithm runs in linear time O(E) using doubly linked list for finding unused edges exiting each vertex, finding a new starting vertex for a tour, and connecting two tours that share a vertex are done in constant time. The challenge was finding the Euler path from the Euler circuit.