**EULER TOUR**

**Satheesh Ganapathy**

**Objective**

Given a graph as an input, write an efficient algorithm to find if a Euler tour/path exists and print the same. If there is no such tour/path, inform the user that the graph is not Eulerian.

**Hypothesis**

We implement Hierholzer’s algorithm to accomplish the given objective. Using this algorithm, we will be able to achieve the output in O(E) time.

**Algorithm**

* If the given graph has a Euler tour, start from any vertex and traverse the graph until we form a cycle and we cannot go any further i.e. the Vertex in which the cycle ended has used all its edges.
* At this point, we choose any Vertex which is already part of the tour and start repeating the previous step. When this vertex forms a cycle and stops at itself, we merge the current tour into the existing tour at the location where the start Vertex for the current tour was.
* Repeat these two steps until all the edges are traversed and we will form a Euler Tour
* If the graph has an Euler path, choose an odd degree vertex and use the same algorithm.

**Implementation**

To implement the algorithm efficiently, we modify the Graph, Vertex and Edge class to incorporate a few indices. The basic idea is to create a Doubly Linked List with indices which will give reference to the nodes in which every element is stored. This in turn will give way to constant time Merging and deletion.

**Implementation Details**

We use three indexed doubly linked lists for the implementation.

* **List of vertices part of the tour with unused edges**
  + This list has to be indexed because while traversing a local tour, if a vertex runs out of edges we remove it in constant time
* **List of edges in Euler Tour order**
  + This indexed list paves way for constant merging.
  + Each vertex has an index of one of its edges which is already part of the euler tour.
  + So, based on the start vertex for the cycle, the merge of the local tour into the main euler tour can be performed easily
* **List of unused edges for each vertex**
  + This indexed list paves way for constant time removal of edges which are being processed as part of the euler tour.
  + The index for this list is stored in each Edge which stores both the index positions in both the vertices’ unused edges list.
  + Each Vertex has its list of unused edges. When a tour is being processed, if an edge is seen, using this index we remove the edge from list of both vertices.

**Running Time Analysis**

The algorithm performs efficiently as desired with time taken being 2.5 seconds to process and give the output for the big input case. For smaller inputs, the program is able to finish its execution within 20 milliseconds in the worst case.

**Conclusion**

Based on the analysis and the implementation, Doubly Linked Circular Indexed lists are the best way to implement Hierholzer’s algorithm.