To find minimum number of Edge-disjoint circuits that can covers an entire graph.

Ankit Petkar - IRM2014002
irm2014002@iiita.ac.in

Surendra Pal Uikey - IWM2014006
iwm2014006@iiita.ac.in

Varun Kumar - ICM2014008
icm2014008@iiita.ac.in

***Abstract*- This document describes the methods to find minimum number of edge-disjoint circuits that can cover an entire graph. It begins with explaining edge-disjoint circuits, further we apply DFS algorithm for traversal and counting the number of edge-disjoint circuits. Complexities and analysis of the implementation is explained.**

MOTIVATION

Closed walk can be a circuit in a graph when it can have no edges and vertex repeated except the starting point. An edge-disjoint circuit between any two circuit when they don't have any common edges. To find minimum number of such edge-disjoint circuits which covers the whole graph where a single vertex cannot be traversed more than one except the starting one. For calculating the required number of minimum circuit in the graph it should have the even degree for every vertex in case of undirected and for the directed one every vertex should have the equal number of incoming and outgoing edges.

INTRODUCTION

Circuit:

A circuit [1] is a closed walk in which no vertex (except for first and last vertex) appears more than once is called a circuit. For example, as shown in Figure 1 two parallel edges between two vertices is a circuit and self-loop on a vertex is also a circuit.

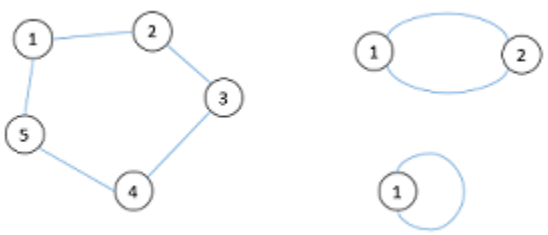


Figure 1: Three different circuits [1].

Edge-disjoint circuit:

An edge-disjoint circuit [1] is a closed walk in which no edge appears more than once but a vertex can be traversed more than once. Figure 2 shows an example of having circuits as 1-1(self-loop), 1-2-4-1, 2-3-4-2 etc.

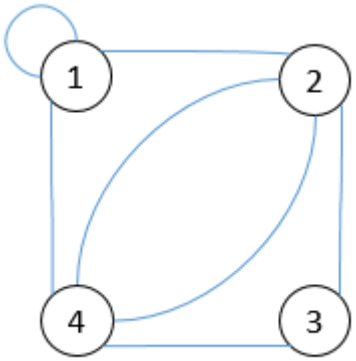
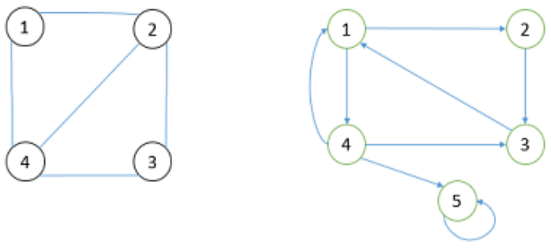


Figure 2: Graph showing edge-disjoint circuits as 1-1(self-loop), 1-2-4-1, 2-3-4-2 etc.

As we can see in Figure 3 that though in part (a) the undirected graph has circuit: 1-2-3-4-1 or 1-2-4-1 etc are possible we still can't cover all the edges , similarly in part (b) the directed graph has circuits: 1-4-1, 1-2-3-1, 5-5, etc. but still the edge from vertex 4 to 5 is not traversed.



(a) (b)

Figure 3 : (a) Undirected graph , (b) Directed Graph.

ALGORITHM

1. First check whether graph can be decompose in the edge-disjoint circuit or not. For decomposition all vertex should have the even degree.
2. V is the vertex in the graph
3. Loop 1 to V
   3.1. For undirected
   3.2. If degree(v) is odd //for undirected
      3.2.1. Circuit not possible.
      3.2.2. Break.
      3.2.3. Goto 11.
   3.3. For directed
   3.4. If incoming Degree(v) != outgoing Degree(v)
      3.4.1. Circuit not possible.
      3.4.2. Break.
      3.4.3. Goto 11.
4. End loop
5. Otherwise count the circuit in the graph. Goto 6.
6. Counter = #number of self-loops;
   (Because self-loop itself create the circuit.)
7. root = anyone(v1, v2, , vn); (which has some degree.)
8. if(DFS(root))
   8.1. Counter = Counter + 1;
9. Goto 7 until all the edges in the graph traversed.
10. finish.

FUNCTIONS

1. bool dfsUtil(int root, bool vis[], int parent)
   1.1. vis[root] = true;
   1.2. Loop 1 to V
      if i is not visited and there should be an edge
         Remove edges between root and the visiting vertex.
         if(dfsUtil(i, vis, root))
         return true;
      else if it already visited and it form circuit but it should not be the parent of the root
         Remove edges b/w root and the visiting vertex.
         return true;
      End Loop
      return false;

2. bool DFS(int root)
   2.1. Boolean array vis[V].
   2.2. Initially set to false.
   2.3. If dfsUtil(root, vis, -1)
      (return true that means there is circuit.)
         return true;
   2.4. return false;

IMPLEMENTATION

Algorithm for Minimum number of circuit in the undirected graph:

1. As we know that self-loop it-self create the circuit so if take it as count is equal to the number of self-loops and remove self-loops from the graph.
2. Now we need to check whether the given graph can give the edge-disjoint circuit or not. For this we have to find that all the vertex should contains the even number of the degree. If all the vertex contains the even degree then it contains the edge-disjoint circuit otherwise not.
3. Now we need to apply DFS taking root as any of the vertex but in our case we have chosen the starting vertex should have the maximum degree. While applying the DFS on the graph we will discarded the edges that we have traversed and whenever we find any node that is already visited in the graph then we will count it as one circuit. And then again run the DFS for another circuit this way we keep on doing this until the all the edges have been traversed.
4. Total number of minimum circuit possible in the graph is total self-loops and the circuit we get from the DFS.
5. Complexity would be :
   V - number of vertex.
   E – number of edges.
   Time :- O(V*V + V + E).
   Space :- O(V*V).

Algorithm for Minimum number of circuit in the directed graph:

1. As we know that self-loop it-self create the circuit so if take it as count is equal to the number of self-loops and remove self-loops from the graph.
2. Now we need to check whether the given graph can give the edge-disjoint circuit or not. For this we have to find that all the vertex should contains equal number of out degree and in degree. If all the vertex contains the equal number of in and out degree then the graph contains the edge-disjoint circuit otherwise not.
3. Now we need to apply DFS taking root as any of the vertex but in our case we have chosen the starting vertex should have the maximum degree. While applying the DFS on the graph we will discarded the edges that we have traversed and whenever we find any node that is already visited in the graph then we will count it as one circuit. And then again run the DFS for another circuit this way we keep on doing this until the all the edges have been traversed.
4. Total number of minimum circuit possible in the graph is total self-loops and the circuit we get from the DFS.
5. Complexity would be :
   V - number of vertex.
   E - number of Edges.
   Time :- O(V*V + V + E).
   Space :- O(V*V).

RESULT AND ANALYSIS

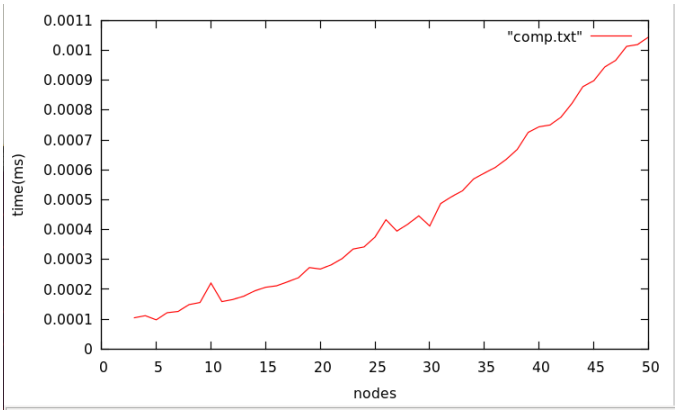


Figure 4: Analysis of time taken vs number of nodes/vertices.

As shown in Figure 4 as the number of nodes or vertices increases the execution time also increases. Then the complexities of the applied algorithm are discussed above.

CONCLUSION

Thus applying the above algorithms we get the minimum number of edge-disjoint circuits of a given graph. Complexities of both the algorithms are thus calculated.

Analyzing the time taken vs the number of vertices can be viewed in Figure 4 for undirected graph. The graph shows that conclude that as the number of vertices increases the execution time increases.

REFERENCES

[1] N. Deo, Graph theory with applications to engineering and computer science