Algorithm to find the Hamiltonian cycle if closure of a graph is complete

Pradeep Gangwar
IHM2016501

Rahul Chanderiya
IIM2016002

Aditi Agrawal
IIM2016001

October 15, 2019

Abstract

The paper details the algorithm to find the Hamiltonian cycle if closure of a graph is complete.

Keywords :- *Hamiltonian Cycle, Complete Graph, Closure, Depth First Search, Directed Graph, Undirected Graph, Mixed Graph, Adjacency Matrix*

1 Introduction

Graphs are a collection of nodes/ vertices and edges that represent relationships. There are two types of graphs- directed and undirected graphs. While directed graphs contain an ordered pair of vertices, undirected graphs contain an unordered pair of vertices. A Hamiltonian Path visits each vertex of the graph exactly once. A Hamiltonian cycle (also known as Hamiltonian circuit) is a Hamiltonian Path such that there is an edge (in the graph) from the last vertex to the first vertex of the Hamiltonian Path. Closure of a directed graph is a set of vertices with no outgoing edges. Our aim is to find the Hamiltonian Cycle of a graph if its closure is complete. Following are the important notations and definitions that we will use frequently in this paper.

1.1 Basic Definitions

1. *Graph:* Graph is a pair G=(V,E) where,

   - V is a set of vertices (or nodes), and
   - E $\subseteq$ (VxV) is a set of edges.

   A graph may be directed or undirected, if graph is undirected, then adjacency relation defined by edges is symmetric.

2. *Directed and Undirected Graph:* A directed graph is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are directed from one vertex to another. A directed graph is sometimes called a digraph or a directed network. In contrast, a graph where the edges are bidirectional is called an undirected graph.

3. *Hamiltonian Cycle:* A Hamiltonian Path visits each vertex of the graph exactly once. A Hamiltonian cycle is a Hamiltonian Path such that there is an edge in the graph from the last vertex to the first vertex of the Hamiltonian Path. In other words, Hamiltonian cycle is a path traveling from a point back to itself, visiting every node en route.

4. *Closure of a graph:* Closure of a directed graph is a set of vertices with no outgoing edges. That is, the graph should have no edges that start within the closure and end outside the closure.

1.2 Objective

Here we will propose algorithms to find the hamiltonian cycle of a graph given that its closure is complete.

2 Algorithm Design

Here we are designing an algorithm, that would find whether the given directed/undirected/mixed graph contains any Hamiltonian Cycle or not. If the graph contains the Hamiltonian cycle, then it will return any one of the Hamiltonian cycle. Graphs in the form of adjacency matrix would be given as input, which would be analysed for the presence of Hamiltonian Cycle. In the end the output would be the list of vertices in the order they form the Hamiltonian Cycle, or an empty list representing that the graph do not contain any Hamiltonian Cycle.

2.1 Algorithm Implementation

(i) Firstly, we will fix any of the vertix to start with (Preferably the 0th vertex).

(ii) Now we will perform the Depth-First-Traversal

(DFS) from this fixed vertix, until we complete the cycle and revisit the current vertex.

(iii) For performing DFS we would keep a visited vertex count and a stack to set the order in which vertices are visited.

(iv) We would check the neighbouring vertices of the current vertex, if the vertex is not visited or is not the vertex from which our current vertex is redirected, then we would recursively perform DFS on this neighbouring vertex.

(v) Whenever the current vertex being visited in DFS becomes equal to the target vertex, i.e the first vertex (to detect the loop) and the total number of vertices in the stack equals to the total number of vertices in the graph, then we return true, i.e we have found the Hamiltonian cycle.

(vi) If the current vertex and target vertex are same, but the total number of vertices in the stack are not equal to the total number of vertices in the graph. Then it implies that we have detected the cycle in the graph, but this cycle do not form Hamiltonian Cycle. Therefore we return false.

(vii) Finally if the output of DFS call turns out to be true, then we pop-out the vertices from the stack and print them, which result in the Hamiltonian Cycle we have found. Or else is the DFS call return false, then the graph do not contain any Hamiltonian cycle.

2.2 Pseudo Code

Algorithm 1 public:cls(vector <vecotr > > graph)

```
$graph \leftarrow this-> graph$
que.push_back(0)
bool x = dfs(0,0,0)
if x == false then
    print "No Hamiltonian cycle exists"
else
    print_cycle()
end if
```

Algorithm 2 bool_dfs(int cur, int par, int tar)

```
if cur != tar then
    setx.insert(cur)
end if
for i=0 to v do
    if graph[cur][i]==0 or i==par then
        continue
    end if
    if setx.find(i) != setx.end() then
        continue
    end if
    que.push_back(i)
    if i==tar and que.size()==v+1 then
        return true;
    end if
    bool x = dfs(i,cur,tar)
    if x == false then
        que.pop_back()
        setx.erase(i)
        continue
    else
        return true
    end if
end for
return false
```

3 Analysis

3.1 Time Complexity Analysis

The time complexity of the given algorithm is quadratic for all the cases with respect to total number of vertices present in the graph. This is evident from the following facts :-

(i) For all the cases, we have to perform Depth-First-Traversal (DFS) from the starting vertex (0th vertex) exactly once to detect Hamiltonian Cycle.

(ii) As the complexity of performing DFS is $O(E)$, where E represent the total number of edges in the graph.

(iii) Since the total number of edges in the graph could be V^2 in the worst case. Therefore the complexity for DFS is $O(V^2)$ in the worst case, and this DFS has to be performed exactly once.

(v) Therefore the final complexity of the algorithm turns out to be $O(V^2)$ for all the cases.

<u>All Cases:</u>

$time_{best} = time_{average} = time_{worst} = O(V^2)$

where V is the total no. of vertices in the graph.

3.2 Space Complexity Analysis

Our algorithm requires an auxiliary space to implement stack data structure, which stores the order in which the vertices of the graph appears to form the cycle. Therefore the space complexity of our algorithm is linear in nature.

$$space_{complexity} = O(V)$$

where V is the total no. of vertices in the graph.

4 Experimental Study

4.1 Profiling

The best way to study an algorithm is by graphs and profiling.

4.1.1 Time Complexity

We have seen that

$$time_{best} = time_{average} = time_{worst} = O(V^2)$$

So,the graphs for all the cases remains the same.

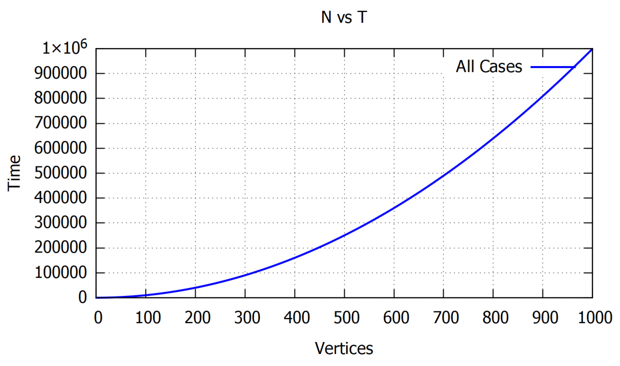

Figure 1: V vs T graph (All Cases)

4.2 Sample Outputs

As the best way to understand something is through the working examples. Therefore this section consist of some examples to understand our algorithm better.

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Output: 0 1 2 3 4 0

Output prints the hamiltonian cycle for the graph depicting the nodes through which the cycle passes.

5 Discussions

5.1 Applications

The concepts of graph theory are used widely to study and model various real time applications in different fields. Several algorithms are used to find optimal graphical solutions. The concepts of Hamiltonian Path and Hamiltonian Cycles are used in mixed graphical problems, maps and travelling routes, image processing, etc.

6 Conclusion

The main aim of this paper was to find the Hamiltonian Cycle of a graph if its closure is complete. The main challenge was to is to check if the given graphs are isomorphic. The main challenge is to determine the complete closure of the graph and check whether it has Hamiltonian cycles present in it. For this we used Depth First Search approach to reach to our solution. The time-complexity as well as space complexity of the algorithm has also been shown in this paper.

References

[1] Keijo Ruohonen, Graph Theory, 2010,PP. 27-35

[2] Thomas H. Cormen is the co-author of Introduction to Algorithms

[3] Jonathan L. Gross and Jay Yellen, Graph Theory and its applications, Columbia University, New York, USA, 2005

[4] A. Gibbons, Algorithmic Graph Theory, Cambridge University Press, 2003.

[5] C. Godsil and G.Royle, Algorithmic Graph Theory, Cambridge University Press, 2001