

Extract all possible Hamiltonian circuits and edge disjoint Hamiltonian circuits if graph is Hamiltonian else extract Hamiltonian paths.

Part-1

Anshul Agarwal (IWM2017008), Vardhan Malik (IWM2017007), Aman Gupta (IWM2017006)

Abstract—This paper demonstrates the extraction of all possible Hamiltonian circuits, edge disjoint Hamiltonian circuits, and Hamiltonian paths. It is also provided with the validation for the directed circuits.

I. KEYWORDS

Circuit, Path, Hamiltonian Path, Hamiltonian Circuit, Matrix, Incidence Matrix, Adjacency Matrix, Disjoint Edges, Node, Vertex.

II. INTRODUCTION

IN Graph theory, Circuit is a path which ends at the vertex it begins (so a loop is an circuit of length one). Circuit may have repeated edges but no vertex repetition is allowed. Circuit is a closed trail.

Path is a trail in which neither vertices nor edges are repeated i.e. if we traverse a graph such that we do not repeat a vertex and nor we repeat an edge. As path is also a trail, thus it is also an open walk.

Hamiltonian path is a path in an undirected or directed graph that visits each vertex exactly once. A graph with n vertices has a Hamiltonian path if, for every non-adjacent vertex pairs the sum of their degrees and their shortest path length is greater than n . A graph that contains a Hamiltonian path is called a traceable graph. A graph is Hamiltonian-connected if for every pair of vertices there is a Hamiltonian path between the two vertices.

A Hamiltonian circuit is a circuit that visits every vertex once with no repeats. Being a circuit, it must start and end at the same vertex. A Hamiltonian path also visits every vertex once with no repeats, but does not have to start and end at the same vertex. A Hamiltonian cycle is a Hamiltonian path that is a cycle. A graph that contains a Hamiltonian cycle is called a Hamiltonian graph.

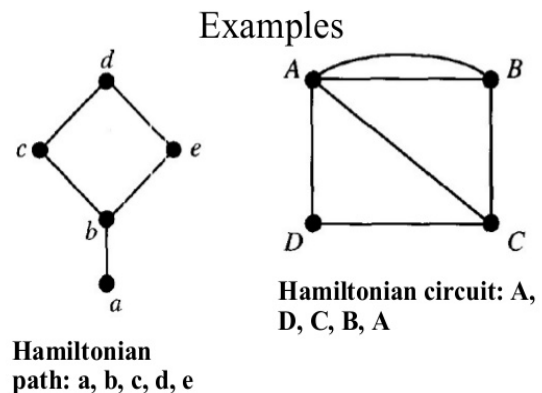
Two paths are said edge disjoint if they don't share any edge. Hence, two Hamiltonian circuits are said to be edge disjoint if they have no edge in common.

There are certain criteria which rule out the existence of a Hamiltonian circuit in a graph, such as- if there is a vertex of degree one in a graph then it is impossible for it to have

a Hamiltonian circuit. There are certain theorems which give sufficient but not necessary conditions for the existence of Hamiltonian graphs.

Dirac's Theorem- If G is a simple graph with n vertices with $n \geq 3$ such that the degree of every vertex in G is at least $n/2$, then G has a Hamiltonian circuit.

Ore's Theorem- If G is a simple graph with n vertices with $n \geq 3$ such that $\deg(u) + \deg(v) \geq n$ for every pair of non-adjacent vertices u and v in G , then G has a Hamiltonian circuit.



III. ALGORITHM DESIGN AND EXPLANATION

Our approach to given problem for Adjacency Matrix input includes the following steps:

1. We are taking the number of vertices and number of edges as an input and constructing the graph.
2. Then we are checking Hamiltonian Paths and Cycles for each individual node one by one.
3. Before checking for paths and cycles we are recursively pushing the nodes in the path vector and marked them as visited so that each node is visited only once.
4. Later they will be marked as unvisited and are removed from the path vector.

5. If there are no hamiltonian cycles then we are printing the hamiltonian paths.

6. Otherwise if there are hamiltonian circuits which are stored in string format to access the edges, then we will first print the hamiltonian circuits and we will be looking for disjoint edges in the circuits.

7. For each cycle we are creating an unordered map to store all its edges.

8. Further, we are looking for every combination of that cycle to check whether a particular edges is repeating or not. Non-repeating edges are pushed into the resultant vector.

9. Number of disjoint edges depends on the size of resultant vector.

The pseudo-code for this algorithm is:-

Algorithm 1 Defining global variables

```
vector < string > HamiltotianCycle    ▷ t ← t+1
vector < string > HamiltotianPath    ▷ t ← t+1
MinimumHamiltonianPath ← INTMAX    ▷ t ← t+1
```

Algorithm 2 Input

```
Nodes ← Input    ▷ t ← t+1
Edges ← Input    ▷ t ← t+1
while i < edges do    ▷ t ← t+1
    InputVertex(u)    ▷ t ← t+1
    InputVertex(v)    ▷ t ← t+1
    graph[u][v] = 1    ▷ t ← t+1
    graph[v][u] = 1    ▷ t ← t+1
end while
```

IV. ALGORITHM ANALYSIS

The above algorithm is designed to reduce the time complexity for this question as much as possible. The problem is an NP complete problem and uses most of the concepts of graph theory. In case of Adjacency matrix, the problem uses recursion, backtracking, and Depth first search as the algorithmic concepts while in case of incidence matrix, it uses dynamic programming, Bit masking and topological sorting for the efficiency of the algorithm. However, if both time and space complexity are considered, then incidence matrix produces better output but not optimal while adjacency matrix produces optimal but with some more time complexities. Since the problem is oriented towards the accuracy of the algorithm with optimality, thus the adjacency matrix representation produces the desired output.

Algorithm 3 Main()

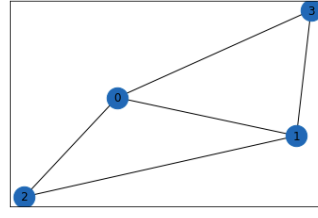
```
vector < int > path    ▷ t ← t+1
while i < nodes do    ▷ t ← t+1
    Path ← pushnode    ▷ t ← t+1
    Mark node visited ← True    ▷ t ← t+1
    Call Hamiltonian_Paths_and_Cycles    ▷ t ← t+1
    Mark node visited ← False    ▷ t ← t+1
    Path ← popnode    ▷ t ← t+1
end while
if Hamiltonian cycle size == 0 then    ▷ t ← t+1
    No Hamiltonian Circuit Exists    ▷ t ← t+1
    Print Hamiltonian Paths    ▷ t ← t+1
    while i < nodes do    ▷ t ← t+1
        print node    ▷ t ← t+1
    end while
else
    Hamiltonian Circuit Exists    ▷ t ← t+1
    Print Hamiltonian Circuits    ▷ t ← t+1
    while i < length HamiltonianCycle do    ▷ t ← t+1
        print node    ▷ t ← t+1
    end while
    Unordered map to store edge mappings    ▷ t ← t+1
    while i < length HamiltonianCycle do    ▷ t ← t+1
        String ← Each cycle    ▷ t ← t+1
        while j < length of ith cycle do    ▷ t ← t+1
            Mark map for edge ← 1    ▷ t ← t+1
        end while
        j ← i + 1    ▷ t ← t+1
        while j < length HamiltonianCycle do    ▷ t ← t+1
            String ← Eachcycle    ▷ t ← t+1
            while k < length of jth cycle - 1 do    ▷ t ← t+1
                String ← Eachcycle    ▷ t ← t+1
                flag ← 0    ▷ t ← t+1
                if map is marked then    ▷ t ← t+1
                    flag ← 1    ▷ t ← t+1
                    Break    ▷ t ← t+1
                end if
            end while
            if flag == 0 then    ▷ t ← t+1
                answer ← push cycles having disjoint edges.    ▷ t ← t+1
            end if
        end while
    end while
    if ans size == 0 then    ▷ t ← t+1
        No Edge Disjoint Hamiltonian Circuit exists    ▷ t ← t+1
        Print edge Disjoint Hamiltonian Circuits    ▷ t ← t+1
    while i < size ans do    ▷ t ← t+1
        Print edge    ▷ t ← t+1
    end while
end if
```

Algorithm 4 Hamiltonian_Paths_and_Cycles

```

if Path size == N then                                ▷ t ← t+1
    string k ← ""                                       ▷ t ← t+1
    while i < nodes - 1 do                             ▷ t ← t+1
        k ← k + tostring(path[i])                     ▷ t ← t+1
    end while
    k ← k + tostring(path[N - 1])                     ▷ t ← t+1
    if graph[path[0]][path[N - 1]] = 0 then             ▷ t ← t+1
        k ← k + tostring(path[0])                     ▷ t ← t+1
        path ← push ith node                           ▷ t ← t+1
        HamiltonianCycle ← push k                      ▷ t ← t+1
    else
        HamiltonianPath ← push k                       ▷ t ← t+1
    end if
end if
while i < nodes do                                    ▷ t ← t+1
    Path ← pushnode                                    ▷ t ← t+1
    Mark node visited ← True                           ▷ t ← t+1
    Call Hamiltonian_Paths_and_Cycles                  ▷ t ← t+1
    Mark node visited ← False                           ▷ t ← t+1
    Path ← popnode                                     ▷ t ← t+1
end while
=0

```



fig(1): Input Graph

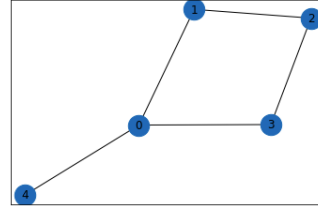
```

Hamiltonian Circuit Exists
Following are the Hamiltonian Circuits
02130
03120
12031
13021
20312
21302
30213
31203
No Edge Disjoint Hamiltonian Circuit exists

```

fig(2): Output

Taken the input graph fig(3) and applying our algorithm on the graph and we got that it does not contains any hamiltonian circuit in the graph and we got only the hamiltonian path present in the graph. fig(4)



fig(3): Input Graph

```

No Hamiltonian Circuit Exists
Following are the Hamiltonian Paths
12304
32104
40123
40321

```

fig(4): Output

Algorithm 5 Incidence_Matrix_pseudo-code

```

vector < vector < bool >> dp                                ▷ t ← t+1
while i < N do                                           ▷ t ← t+1
    dp[i][1 << i] ← true                                   ▷ t ← t+1
end while
while i < 1 << N do                                       ▷ t ← t+1
    while j < N do                                         ▷ t ← t+1
        if i and (1 << j) then                             ▷ t ← t+1
            while k < N do                                 ▷ t ← t+1
                if i(1 << k) and adj[k][j] and k! =      ▷ t ← t+1
                    jdp[k][i(1 << j)] then                 ▷ t ← t+1
                        dp[i][j] ← true                     ▷ t ← t+1
                end if
            end while
        end if
    end while
end while

```

B. Time Complexity

The time complexity analysis of an Algorithm is quite necessary to learn about it's efficiency and optimization.

The time complexity of an algorithm(1) for adjacency matrix is-

$$T = O(N * N!).$$

Where N is number of nodes and N! is factorial of N

The time complexity of an algorithm(2) for incidence matrix is-

$$T = O(2^N * N^3).$$

where N is the number of node in graph.

C. Space Complexity

The Space Complexity of algorithm(1) for adjacency matrix is-

$$O(N^2).$$

where N is number of nodes.

The Space Complexity of algorithm(2) for incidence matrix is-

$$O(max(N * 2^N + N * E)).$$

where N is number of nodes and E is number of Edges.

VI. CONCLUSION

In this paper, we deduced an algorithm to find all possible Hamiltonian circuits and Hamiltonian Paths with Minimum

V. EXPERIMENTAL STUDY**A. Experimental Result**

Taken the input graph fig(1) and applying our algorithm on the graph and we got all the hamiltonian circuit present in the graph and it does not contains any edge disjoint circuits. fig(2) .

Hamiltonian Path and all Edge Disjoint Hamiltonian Circuits. The proposed algorithm uses Recursion, Backtracking, Dynamic Programming, BIT Masking as the algorithm paradigms. The order of time complexity is $O(N * N!)$ and $O(2^N * N^3)$ respectively, where n is the number of edges(nodes). The pruning in the code has been performed quite efficiently in case of fully connected as well as disjoint graphs.

REFERENCES

- [1] Online "https://en.wikipedia.org/wiki/Hamiltonian_path"
- [2] Online "<https://www.gatevidyalay.com/hamiltonian-graphs-hamiltonian-path-hamiltonian-circuit/>"
- [3] Online "https://en.wikipedia.org/wiki/Hamiltonian_path"
- [4] Online "<https://www.gatevidyalay.com/hamiltonian-graphs-hamiltonian-path-hamiltonian-circuit/>"
- [5] Online "<https://www.britannica.com/topic/graph-theory>"
- [6] Online "<https://www.geeksforgeeks.org/find-edge-disjoint-paths-two-vertices/>"
- [7] Online "Introduction to Algorithm 3rd Edition By Thomas H.Cormen"
- [8] Online "<https://www.overleaf.com>"