# Extract the least cost Hamiltonian circuit (or Hamiltonian path) in weighted graph. Part-2

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

***Abstract*—This paper demonstrates the extraction of least cost Hamiltonian circuits (or Hamiltonian paths) for minimum cost. It is also provided with the validation for the directed circuits.**

## I. Keywords

Circuit, Path, Hamiltonian Path, Hamiltonian Circuit, Matrix, Incidence Matrix, Adjacency Matrix, Weighted edges, Node, Vertex, Least Cost.

## 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.

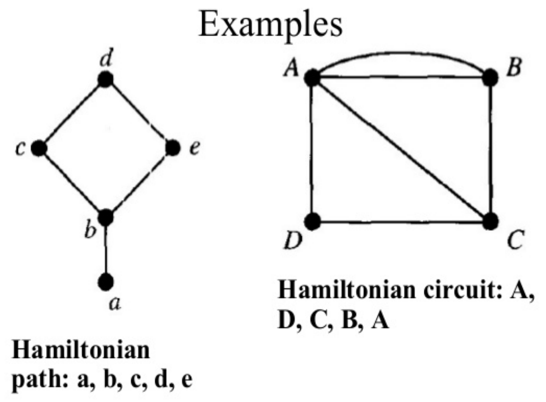
Hamiltonian path is a path in an undirected or directed graph that visits each vertex exactly once. A graph with n vertices has a Hamiltonian path if, for every non-adjacent vertex pairs the sum of their degrees and their shortest path length is greater than n. A graph that contains a Hamiltonian path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

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>=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>= 3 such that deg(u) + deg(v) >= n for every pair of non-adjacent vertices u and v in G, then G has a Hamiltonian circuit.


Examples
d
c
e
b
a
Hamiltonian path: a, b, c, d, e
A
B
D
C
Hamiltonian circuit: A, D, C, B, A

## 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 with weight of each edge.

2. Then we are checking Hamiltonian Paths for each individual node one by one.

3. Before checking if the path exists or not 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 .As soon as the paths are found , calculate the path length using weighted edges and each time a path is found compare the path lengths.The path with shortest length is printed.

**The pseudo-code for this algorithm is:-**

**Algorithm 1** Defining global variables

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

**Algorithm 2** Input

```
$Nodes \leftarrow Input$                            ▷ t ← t+1
$Edges \leftarrow Input$                            ▷ 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
```

**Algorithm 3** Main()

```
if $ans! = INTMAX$ then                             ▷ t ← t+1
    $Print\ shortest\ Hamiltonian$                  ▷ t ← t+1
    $q\ string$                                     ▷ t ← t+1
end if
```

**Algorithm 4** Hamiltonian_Shortest_Path

```
if $Path\ size == N$ then                           ▷ t ← t+1
    $string\ k \leftarrow ""$                       ▷ t ← t+1
    $ll\ sum \leftarrow 0$                          ▷ t ← t+1
    while $i < nodes - 1$ do                        ▷ t ← t+1
        $sum \leftarrow sum + graph[path[i]][path[i+1]]$   ▷ t ← t+1
        $k \leftarrow k + tostring(path[i])$        ▷ t ← t+1
    end while
    $k \leftarrow k + tostring(path[N-1])$          ▷ t ← t+1
    if $graph[path[0]][path[N-1]]! = 0$ then        ▷ t ← t+1
        $k \leftarrow k + tostring(path[0])$        ▷ t ← t+1
        $sum \leftarrow sum + graph[path[i]][path[i+1]]$   ▷ t ← t+1
        $path \leftarrow push\ ith\ node$           ▷ t ← t+1
    end if
    if $sum < ans$ then                             ▷ t ← t+1
        $ans \leftarrow sum$                        ▷ t ← t+1
        $q \leftarrow k$                            ▷ t ← t+1
    end if
end if
```

## 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 is uses most of the concepts of graph theory. In case of Adjacency matrix, the problem

**Algorithm 5** Incidence_Matrix_pseudo-code

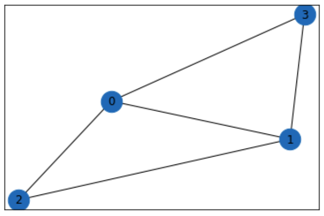
```
$vector < vector < int >> dp$                       ▷ t ← t+1
while $i < N$ do                                    ▷ t ← t+1
    $dp[i][1 << i] \leftarrow true$                 ▷ t ← t+1
end while
while $i < 1 << N$ do                               ▷ t ← t+1
    while $j < N$ do                                ▷ t ← t+1
        if $i\ and\ (1 << j)$ then                  ▷ t ← t+1
            while $k < N$ do                        ▷ t ← t+1
                if $i(1 << k)\ and\ adj[k][j]\ and\ k! = j dp[k][i^{(}1 << j)]$ then   ▷ t ← t+1
                    $dp[i][j] \leftarrow dp[k][i^{(}1 << j)] + 1$   ▷ t ← t+1
                end if
            end while
        end if
    end while
end while
```

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.

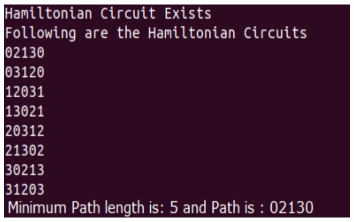
## V. Experimental Study

### 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) .



fig(1): Input Graph



```
Hamiltonian Circuit Exists
Following are the Hamiltonian Circuits
02130
03120
12031
13021
20312
21302
30213
31203
 Minimum Path length is: 5 and Path is : 02130
```

fig(2): Output

### 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 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 ”Introduction to Algorithm 3rd Edition By Thomas H.Cormen”

[7] Online ”http://www.mathcs.emory.edu/ cheung/Courses/171/Syllabus/11-Graph/weighted.html”

[8] Online ”https://www.overleaf.com”