8. Hamiltonian cycle

Time complexity: O(n!)

Algorithm:

1. Add vertex V to the path.

2. If path length is equal to no of vertices in G:

- If there is an edge from the last vertex to the starting vertex, return Ham cycle.

Eg, a → b, b → d, d → c, c → a.
Traverse -
* Start from ver A
* Move to ver B
*   "   "   " C
*   "   "   " D
*   "   "   " A

* From the above fig, there exists an edge from D to A.
* return Ham cycle.

- else, return null.

3. For each unvisited neighbour u of vertex V in G:
- call the fun find_Ham_cycle_from_vertex(u, G, path)
- If not found, return it.

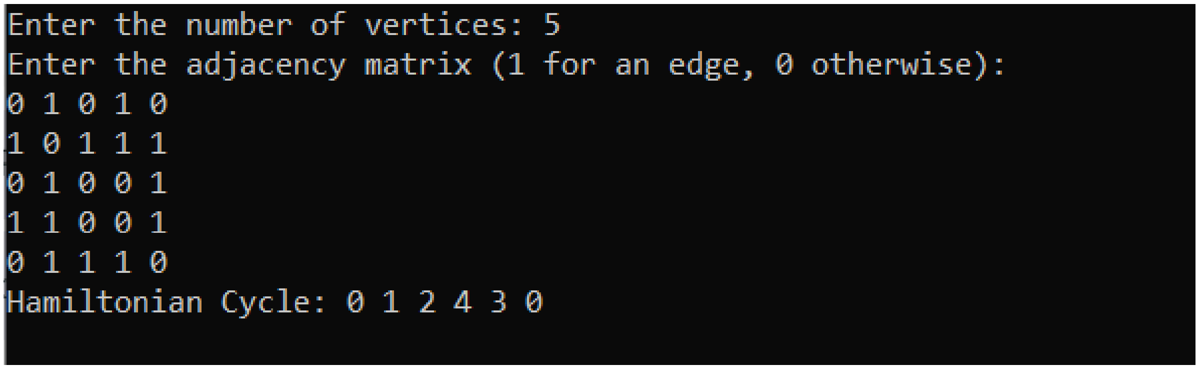
4. Remove vertex V, from the path.
5. If the path is empty, return null.

Code:

```
#include <stdio.h>
#include <stdbool.h>

#define MAX_VERTICES 10

bool isSafe(int vertex, bool adjacencyMatrix[MAX_VERTICES][MAX_VERTICES], int path[], int
position, int numVertices)
{
    if (adjacencyMatrix[path[position - 1]][vertex] == 0)
        return false;

    for (int i = 0; i < position; i++)
```

```
        if (path[i] == vertex)
            return false;

    return true;
}

bool hamiltonianCycleUtil(bool adjacencyMatrix[MAX_VERTICES][MAX_VERTICES], int path[],
int position, int numVertices)
{
    if (position == numVertices)
            {
        if (adjacencyMatrix[path[position - 1]][path[0]] == 1)
            return true;
        else
            return false;
    }

    for (int vertex = 1; vertex < numVertices; vertex++)
            {
        if (isSafe(vertex, adjacencyMatrix, path, position, numVertices))
                        {
            path[position] = vertex;
            if (hamiltonianCycleUtil(adjacencyMatrix, path, position + 1, numVertices) == true)
                return true;
            path[position] = -1;
        }
    }

    return false;
}

void hamiltonianCycle(bool adjacencyMatrix[MAX_VERTICES][MAX_VERTICES], int
numVertices)
{
    int path[MAX_VERTICES];
    for (int i = 0; i < numVertices; i++)
        path[i] = -1;

    path[0] = 0;
    if (hamiltonianCycleUtil(adjacencyMatrix, path, 1, numVertices) == false)
            {
        printf("Hamiltonian Cycle does not exist\n");
        return;
    }
    printf("Hamiltonian Cycle: ");
    for (int i = 0; i < numVertices; i++)
        printf("%d ", path[i]);
    printf("%d\n", path[0]);
}

int main()
{
    int numVertices;
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    bool adjacencyMatrix[MAX_VERTICES][MAX_VERTICES];
    printf("Enter the adjacency matrix (1 for an edge, 0 otherwise):\n");
    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            scanf("%d", &adjacencyMatrix[i][j]);
        }
    }

    hamiltonianCycle(adjacencyMatrix, numVertices);

    return 0;
}
```

input /output:



```
Enter the number of vertices: 5
Enter the adjacency matrix (1 for an edge, 0 otherwise):
0 1 0 1 0
1 0 1 1 1
0 1 0 0 1
1 1 0 0 1
0 1 1 1 0
Hamiltonian Cycle: 0 1 2 4 3 0
```