6.12 HAMILTONIAN CYCLE DETECTION

Question:

You are given an undirected graph represented by a list of edges and the number of vertices n. Your task is to determine if there exists a Hamiltonian cycle in the graph. A Hamiltonian cycle is a cycle that visits each vertex exactly once and returns to the starting vertex.

Write a function that takes the list of edges and the number of vertices as input and returns true if there exists a Hamiltonian cycle in the graph, otherwise return false.

Example: Given edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2), (2, 4), (4, 0)] and n = 5

AIM

To implement a Python program to determine whether a Hamiltonian cycle exists in a given undirected graph using backtracking.

ALGORITHM

1. Input the number of vertices n and the list of edges.
2. Represent the graph using an adjacency matrix.
3. Choose a starting vertex (vertex 0) and mark it as visited.
4. Recursively attempt to visit all vertices exactly once:
5. - Move to an adjacent unvisited vertex.
6. - Add it to the current path.
7. If all vertices are included and there is an edge back to the starting vertex, a Hamiltonian cycle exists.
8. Otherwise, return False if no path satisfies the condition.

PROGRAM

```
def has_hamiltonian_cycle(n, edges):
    from collections import defaultdict
    graph = defaultdict(list)
    for u, v in edges:
        graph[u].append(v)
        graph[v].append(u)

    path = []

    def backtrack(v, visited):
        path.append(v)
        if len(path) == n:
            return path[0] in graph[v]
        for u in graph[v]:
            if u not in visited:
                visited.add(u)
                if backtrack(u, visited):
                    return True
                visited.remove(u)
        path.pop()
        return False

    for start in range(n):
        if backtrack(start, {start}):
            return True
    return False

n = int(input("Enter number of vertices: "))
m = int(input("Enter number of edges: "))
edges = []
for _ in range(m):
    u, v = map(int, input("Edge: ").split())
    edges.append((u, v))
print("Hamiltonian Cycle Exists:", has_hamiltonian_cycle(n, edges))
```

Input:

Number of vertices: n = 5

Edges: [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2), (2, 4), (4, 0)

Output:

```
Enter number of vertices: 5
Enter number of edges: 7
Edge: 0 1
Edge: 1 2
Edge: 2 3
Edge: 3 0
Edge: 0 2
Edge: 2 4
Edge: 4 0
Hamiltonian Cycle Exists: True
>>>
```

RESULT:

Thus, the program is successfully executed and verified to detect the existence of a Hamiltonian cycle in the given graph.

PERFORMANCE ANALYSIS:

Time Complexity: O(n!) in the worst case since all permutations of vertices may be checked.

Space Complexity: O(n) for recursion depth and path storage.