7.1 CLASS P OR NP VERIFICATION (HAMILTONIAN PATH)

Question:
Implement a program to verify if a given problem is in class P or NP. Choose a specific decision problem (e.g., Hamiltonian Path) and implement a polynomial-time algorithm (if in P) or a non-deterministic polynomial-time verification algorithm (if in NP).

Input:
• Graph G with vertices V = {A, B, C, D} and edges E = {(A, B), (B, C), (C, D), (D, A)}

Output:
• Hamiltonian Path Exists: True (Path: A -> B -> C -> D)

AIM
To implement a Python program to verify whether a Hamiltonian Path exists in a graph, illustrating the concept of NP problems where a solution can be verified in polynomial time.

ALGORITHM
1. Input the graph with vertices and edges.
2. Represent the graph using adjacency list or adjacency matrix.
3. Use backtracking to try all possible paths starting from each vertex.
4. At each step, move to an adjacent unvisited vertex and continue building the path.
5. If a path visits all vertices exactly once, return True.
6. Otherwise, after exploring all possibilities, return False.
7. This algorithm is exponential in nature, but any given solution (a candidate path) can be verified in polynomial time, hence Hamiltonian Path is in NP.

PROGRAM

```
from itertools import permutations

def hamiltonian_path(vertices, edges):
    edge_set = set((min(u, v), max(u, v)) for u, v in edges)
    for perm in permutations(vertices):
        valid = True
        for i in range(len(perm) - 1):
            if (min(perm[i], perm[i+1]), max(perm[i], perm[i+1])) not in edge_set:
                valid = False
                break
        if valid:
            return True, perm
    return False, []

n = int(input("Enter number of vertices: "))
vertices = input("Vertices: ").split()
m = int(input("Enter number of edges: "))
edges = []
for _ in range(m):
    u, v = input("Edge: ").split()
    edges.append((u, v))

exists, path = hamiltonian_path(vertices, edges)
print("Hamiltonian Path Exists:", exists)
if exists:
    print("Path:", " -> ".join(path))
```

Input:

Graph G with vertices V = {A, B, C, D} and edges E = {(A, B), (B, C), (C, D), (D, A)}

Output:

```
Enter number of vertices: 4
Vertices: A B C D
Enter number of edges: 4
Edge: A B
Edge: B C
Edge: C D
Edge: D A
Hamiltonian Path Exists: True
Path: A -> B -> C -> D
>>>
```

RESULT:

Thus, the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

Time Complexity: O(n!) in the worst case due to checking all permutations of vertices.

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