**TOPIC 07 - TRACTABILITY AND APPROXIMATION ALGORITHM**

**7.1**

**Aim**

To implement a program that verifies whether a given decision problem belongs to class P or NP. Specifically, we use the Hamiltonian Path problem.

* If in P → A polynomial-time algorithm exists.
* If in NP → We can verify a proposed solution in polynomial time.

**Algorithm (Hamiltonian Path Verification)**

1. Input:
   * A graph G(V,E)G(V, E)G(V,E).
   * A candidate path PPP.
2. Steps:
   * Check if the path contains all vertices exactly once.
   * For every consecutive pair in the path, check if an edge exists in EEE.
   * If both conditions are satisfied → It is a Hamiltonian Path.
3. Output:
   * True (and display the path) if it’s Hamiltonian.
   * False otherwise.

**Code:**

def is\_hamiltonian\_path(graph, vertices, path):

if set(path) != set(vertices):

return False

for i in range(len(path) - 1):

if (path[i], path[i+1]) not in graph and (path[i+1], path[i]) not in graph:

return False

return True

vertices = ["A", "B", "C", "D"]

edges = {("A", "B"), ("B", "C"), ("C", "D"), ("D", "A")}

graph = edges

candidate\_path = ["A", "B", "C", "D"]

result = is\_hamiltonian\_path(graph, vertices, candidate\_path)

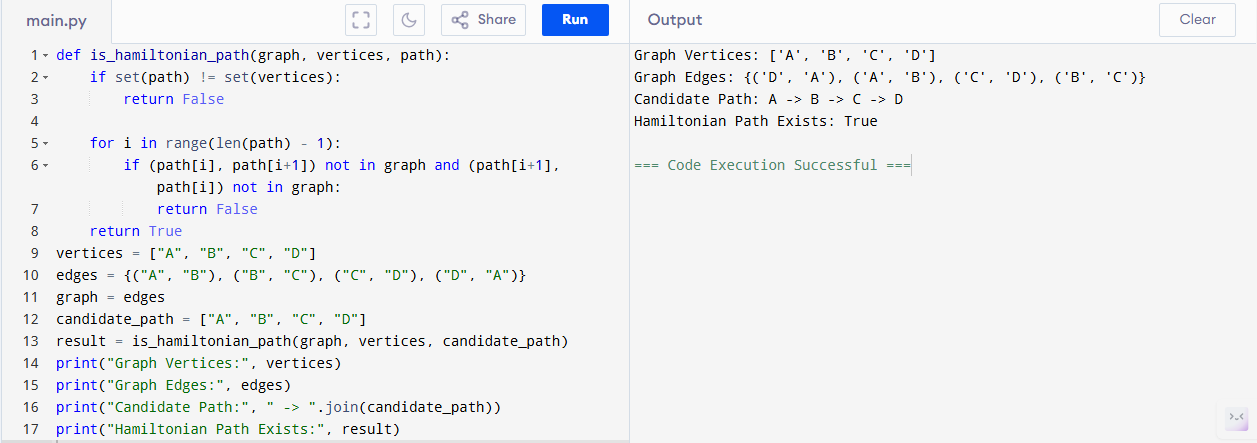
print("Graph Vertices:", vertices)

print("Graph Edges:", edges)

print("Candidate Path:", " -> ".join(candidate\_path))

print("Hamiltonian Path Exists:", result)

**Output screenshot:**



**Sample Input & Output:**

**Input:**

Vertices = {A, B, C, D}

Edges = {(A, B), (B, C), (C, D), (D, A)}

Candidate Path = A -> B -> C -> D

**Output:**

Graph Vertices: ['A', 'B', 'C', 'D']

Graph Edges: {('A', 'B'), ('B', 'C'), ('C', 'D'), ('D', 'A')}

Candidate Path: A -> B -> C -> D

Hamiltonian Path Exists: True

**Result:**

The given class P and NP problem has been executed and verified successfully.

**Performance Analysis**

Time complexity : O(V)