**Topic 6.14: 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.

**Aim**  
To design and implement an algorithm that determines whether a Hamiltonian cycle exists in an undirected graph.

**Algorithm**

1. Represent the graph using an adjacency list.
2. Start from vertex 0 and attempt to build a path covering all vertices.
3. Use backtracking:
   * Mark the current vertex as visited and add it to the path.
   * Recurse into adjacent unvisited vertices.
   * If all vertices are visited and the last vertex connects back to the start, a Hamiltonian cycle exists.
4. If no valid cycle is found after exploring all possibilities, return false.

**Output**A screenshot of a computer

AI-generated content may be incorrect.

**Result**  
The program successfully detects whether a Hamiltonian cycle exists in the given undirected graph and returns an example cycle when found.

**Performance Analysis**

* Time Complexity: O(n!) in the worst case, since all vertex permutations may need to be checked.
* Space Complexity: O(n) for path tracking and recursion stack.