HPC-1A(BFS)

#include <iostream>

#include <vector>

#include <queue>

#include <omp.h>

Using namespace std;

// Parallel BFS function

Void parallelBFS(const vector<vector<int>>& graph, int start) {

    Int n = graph.size();

    Vector<bool> visited(n, false);

    Queue<int> q;

    Visited[start] = true;

    q.push(start);

    while (!q.empty()) {

        int levelSize = q.size();

        vector<int> currentLevel;

        // Collect current level nodes

        For (int I = 0; I < levelSize; i++) {

            Int node = q.front();

            q.pop();

            currentLevel.push\_back(node);

            cout << “Visited: “ << node << endl;

        }

        // Process neighbors in parallel

        #pragma omp parallel for

        For (int I = 0; I < currentLevel.size(); i++) {

            Int node = currentLevel[i];

            For (int neighbor : graph[node]) {

                If (!visited[neighbor]) {

                    #pragma omp critical

                    {

                        If (!visited[neighbor]) {

                            Visited[neighbor] = true;

                            q.push(neighbor);

                        }

                    }

                }

            }

        }

    }

}

Int main() {

    Int nodes, edges;

    Cout << “Enter number of nodes: “;

    Cin >> nodes;

    Vector<vector<int>> graph(nodes);

    Cout << “Enter number of edges: “;

    Cin >> edges;

    Cout << “Enter edges (u v):” << endl;

    For (int I = 0; I < edges; i++) {

        Int u, v;

        Cin >> u >> v;

        Graph[u].push\_back(v);

        Graph[v].push\_back(u); // For undirected graph

    }

    Int start;

    Cout << “Enter start node: “;

    Cin >> start;

    Cout << “\nParallel BFS traversal starting from node “ << start << “:\n”;

    parallelBFS(graph, start);

    return 0;

}