|  |  |
| --- | --- |
| 1b | Design and implement Parallel Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for DFS |

#include <iostream>

#include <vector>

#include <stack>

#include <omp.h>

using namespace std;

const int MAX = 100000;

vector<int> graph[MAX];

bool visited[MAX];

void dfs(int start\_node) {

stack<int> s;

s.push(start\_node);

while (!s.empty()) {

int curr\_node = s.top();

s.pop();

if (!visited[curr\_node]) {

visited[curr\_node] = true;

cout << curr\_node << " "; // Output visited nodes

#pragma omp parallel for

for (int i = 0; i < graph[curr\_node].size(); i++) {

int adj\_node = graph[curr\_node][i];

if (!visited[adj\_node]) {

s.push(adj\_node);

}

}

}

}

}

int main() {

int n, m, start\_node;

cout << "Enter the number of nodes, edges, and start node: ";

cin >> n >> m >> start\_node;

cout << "Enter pairs of edges:" << endl;

for (int i = 0; i < m; i++) {

int u, v;

cin >> u >> v;

graph[u].push\_back(v);

graph[v].push\_back(u);

}

#pragma omp parallel for

for (int i = 0; i < n; i++) {

visited[i] = false; // Initialize visited array

}

cout << "DFS Output: ";

dfs(start\_node);

return 0;

}