HPC -1\_DFS\_openmp :-

#include <iostream>

#include <vector>

#include <omp.h>

using namespace std;

const int MAXN = 1e5;

vector<int> adj[MAXN + 5]; // adjacency list

bool visited[MAXN + 5]; // mark visited nodes

void dfs(int node)

{

visited[node] = true;

cout << node << " "; // Print the visited node here

#pragma omp parallel for

for (int i = 0; i < adj[node].size(); i++)

{

int next\_node = adj[node][i];

if (!visited[next\_node])

{

dfs(next\_node);

}

}

}

int main()

{

cout << "Please enter nodes and edges: ";

int n, m; // number of nodes and edges

cin >> n >> m;

for (int i = 1; i <= m; i++)

{

int u, v; // edge between u and v

cin >> u >> v;

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

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

}

int start\_node; // start node of DFS

cout << "Enter the start node for DFS: ";

cin >> start\_node;

dfs(start\_node);

cout << endl; // Print a newline after DFS traversal

return 0;

}

OUTPUT:-

~/HPC$ g++ -o main -fopenmp main.cpp

~/HPC$ ./main

Please enter nodes and edges: 7 6

1 2

1 3

2 4

2 5

3 6

3 7

Enter the start node for DFS: 1

1 2 4 5 3 6 7

~/HPC$

HPC-1\_openmp\_BFS :-

#include <iostream>

#include <queue>

#include <vector>

#include <omp.h>

using namespace std;

int main() {

int num\_vertices, num\_edges, source;

cout << "Enter number of vertices, edges, and source node: ";

cin >> num\_vertices >> num\_edges >> source;

// Input validation

if (source < 1 || source > num\_vertices) {

cout << "Invalid source node!" << endl;

return 1;

}

vector<vector<int>> adj\_list(num\_vertices + 1);

for (int i = 0; i < num\_edges; i++) {

int u, v;

cin >> u >> v;

// Input validation for edges

if (u < 1 || u > num\_vertices || v < 1 || v > num\_vertices) {

cout << "Invalid edge: " << u << " " << v << endl;

return 1;

}

adj\_list[u].push\_back(v);

adj\_list[v].push\_back(u);

}

queue<int> q;

vector<bool> visited(num\_vertices + 1, false);

q.push(source);

visited[source] = true;

while (!q.empty()) {

int curr\_vertex = q.front();

q.pop();

cout << curr\_vertex << " ";

// Parallel loop for neighbors

#pragma omp parallel for

for (int i = 0; i < adj\_list[curr\_vertex].size(); i++) {

int neighbour = adj\_list[curr\_vertex][i];

if (!visited[neighbour]) {

visited[neighbour] = true;

q.push(neighbour);

}

}

}

cout << endl;

return 0;

}

OUTPUT:-

~/HPC$ g++ -o bfs -fopenmp bfs.cpp

~/HPC$ ./bfs

Enter number of vertices, edges, and source node: 7 6 1

1 2

1 3

2 4

2 5

3 6

3 7

1 2 3 4 5 7 6

~/HPC$