Design and implement **Parallel** **Depth First Search** based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS

#include <iostream>

#include <vector>

#include <stack>

#include <omp.h>

#include <bits/stdc++.h>

using namespace std;

void dfs(int start, int\* arr, int n, int visited[]) {

//#pragma omp parallel for ordered

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

//#pragma omp ordered

if( (\*(arr + (start\*n) + i) == 1) && (!visited[i]) )

{

visited[i] = 1;

cout<<i<<" ";

dfs(i, (int\*)arr, n, visited);

}

}

}

int main()

{

// freopen("input.txt","r",stdin);

// freopen("output.txt","w",stdout);

cout<<"Enter the number of vertices:";

int n;

cin>>n;

int arr[n][n] = {0};

cout<<"Enter the number of edges:";

int edges;

cin>>edges;

for(int j=0; j<edges; j++)

{

int a, b;

cout<<"Enter the two edges:"<<endl;

cin>>a>>b;

arr[a][b] = 1;

arr[b][a] = 1;

}

int visited[n] = {0};

cout<<"Enter the start vertex: ";

int start;

cin>>start;

clock\_t strt = clock();

cout<<start<<" ";

visited[start] = 1;

dfs(start, (int \*)arr, n, visited);

clock\_t stop = clock();

cout<<"\nTime required : "<<(double)(stop-strt)<<" ms"<<endl;

return 0;

}

/\*

"Parallel Execution"

PS D:\C++> g++ -fopenmp parallel\_dfs.cpp

PS D:\C++> ./a out

Enter the number of vertices:7

Enter the number of edges:6

Enter the two edges:

0 1

Enter the two edges:

0 2

Enter the two edges:

1 3

Enter the two edges:

1 4

Enter the two edges:

2 5

Enter the two edges:

2 6

Enter the start vertex: 0

0 1 3 4 2 5 6

Time required : 9 m

"Serial Execution"

PS D:\C++> g++ parallel\_dfs.cpp

PS D:\C++> ./a out

Enter the number of vertices:7

Enter the number of edges:6

Enter the two edges:

0 1

Enter the two edges:

0 2

Enter the two edges:

1 3

Enter the two edges:

1 4

Enter the two edges:

2 5

Enter the two edges:

2 6

Enter the start vertex: 0

0 1 3 4 2 5 6

Time required : 12 ms

\*/