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

#include<iostream>

#include<vector>

#include<omp.h>

#include<queue>

#include<bits/stdc++.h>

using namespace std;

queue<int>q;

void bfs(int start, int\* arr, int n, int visit[])

{

#pragma omp parallel for ordered

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

{

#pragma omp ordered

if( ( \*(arr + (n\*start) + i) == 1 ) && (visit[i] == 0) )

{

cout<<i<<" ";

q.push(i);

visit[i] = 1;

}

}

q.pop();

if(!q.empty()) bfs(q.front(), (int\*)arr, n, visit);

}

int main()

{

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

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

//cout<<"BFS 0 1 2 3 4 5 6"<<endl;

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 visit[n] = {0};

cout<<"Enter the start vertex: ";

int start;

cin>>start;

clock\_t strt = clock();

visit[start] = 1;

cout<<start<<" ";

q.push(start);

bfs(start, (int\*)arr, n, visit);

clock\_t stop = clock();

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

return 0;

}

/\*

"Parallel Execution"

PS D:\C++> g++ -fopenmp parallel\_bfs.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 2 3 4 5 6

Time required : 3 ms

"Serial Execution"

PS D:\C++> g++ parallel\_bfs.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 2 3 4 5 6

Time required : 11 ms \*/