```
#include<iostream>
#include<bits/stdc++.h>
#include<omp.h>
using namespace std;
vector<bool> v;
vector<vector<int>> g;
void bfsTraversal(int b)
{
 queue<int> q; //Declare a queue to store all the nodes connected to b
 q.push(b); //Insert b to queue
 v[b]=true; //mark b as visited
 cout<<"\nThe BFS Traversal is: ";
 double start=omp_get_wtime();
 while(!q.empty())
 {
 int a = q.front();
 q.pop(); //delete the first element form queue
 #pragma omp parallel
 for(auto j=g[a].begin();j!=g[a].end();j++)
 {
 if (!v[*j])
 {
 v[*j] = true;
q.push(*j);
 }
 }
 cout<<a<<" ";
 }
 double end=omp_get_wtime();
 double time=end-start;
 cout<<"\n\nTime taken => "<<time<<endl;
}
void makeEdge(int a, int b)
{
 g[a].push_back(b); //an edge from a to b (directed graph)
}
int main()
{
 omp_set_num_threads(4);
 int n,e;
 cout<<"Consider first vertex => 0"<<endl;
 cout<<"\nEnter the number of vertices: ";
 cin >> n;
 cout<<"\nEnter the number of edges: ";
```

```
 cin>>e;
 v.assign(n, false);
 g.assign(n, vector<int>());
 int a, b, i;
 cout << "\nEnter the edges with source and target vetex: "<<endl;
 for(i=0;i<e;i++)
 {
 cin>>a>>b;
makeEdge(a, b);
 }
 for (i=0;i<n;i++)
 {
 if (!v[i]) //if the node i is unvisited
 {
 bfsTraversal(i);
 }
 }
 return 0;
}
```