Roll No. - 41251

Code :

#include<iostream>

#include<omp.h>

using namespace std;

int q[100];

int visited[7];

int local\_q;

void bfs(int adj\_matrix[7][7], int first, int last, int q[], int n\_nodes) {

if(first==last)

return;

int cur\_node = q[first++];

cout<<" "<<cur\_node;

omp\_set\_num\_threads(3);

#pragma omp parallel for shared(visited)

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

if(adj\_matrix[cur\_node][i] == 1 && visited[i] == 0){

q[last++] = i;

visited[i] = 1;

}

}

bfs(adj\_matrix, first, last, q, n\_nodes);

}

int main() {

int first = -1;

int last = 0;

int n\_nodes = 7;

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

visited[i] = 0;

}

int adj\_matrix[7][7] = {

{0, 1, 1, 0, 0, 0, 0},

{1, 0, 1, 1, 0, 0, 0},

{1, 1, 0, 0, 1, 0, 0},

{0, 1, 0, 0, 1, 0, 0},

{0, 0, 1, 1, 0, 1, 0},

{0, 0, 0, 0, 1, 0, 1},

{0, 0, 0, 0, 0, 1, 0}

};

int start\_node = 3;

q[last++] = start\_node;

first++;

visited[start\_node] = 1;

bfs(adj\_matrix, first, last, q, n\_nodes);

return 0;

}

Output :

