COMPETITIVE PROGRAMMING - BFS

#include<iostream>

#include <list>

using namespace std;

class DiGraph

{

int V;

list<int> \*adjList;

public:

DiGraph(int V);

void addEdge(int v, int w);

void BFS(int s);

};

DiGraph::DiGraph(int V)

{

this->V = V;

adjList = new list<int>[V];

}

void DiGraph::addEdge(int v, int w)

{

adjList[v].push\_back(w);

}

void DiGraph::BFS(int s)

{

bool \*visited = new bool[V];

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

visited[i] = false;

list<int> queue;

visited[s] = true;

queue.push\_back(s);

list<int>::iterator i;

while(!queue.empty())

{

s = queue.front();

cout << s << " ";

queue.pop\_front();

for (i = adjList[s].begin(); i != adjList[s].end(); ++i)

{

if (!visited[\*i])

{

visited[\*i] = true;

queue.push\_back(\*i);

}

}

}

}

int main()

{

DiGraph dg(5);

dg.addEdge(0, 1);

dg.addEdge(0, 2);

dg.addEdge(0, 3);

dg.addEdge(1, 2);

dg.addEdge(2, 4);

dg.addEdge(3, 3);

dg.addEdge(4, 4);

cout << "Breadth First Traversal for given graph (with 0 as starting node): "<<endl;

dg.BFS(0);

return 0;

}

OUTPUT

