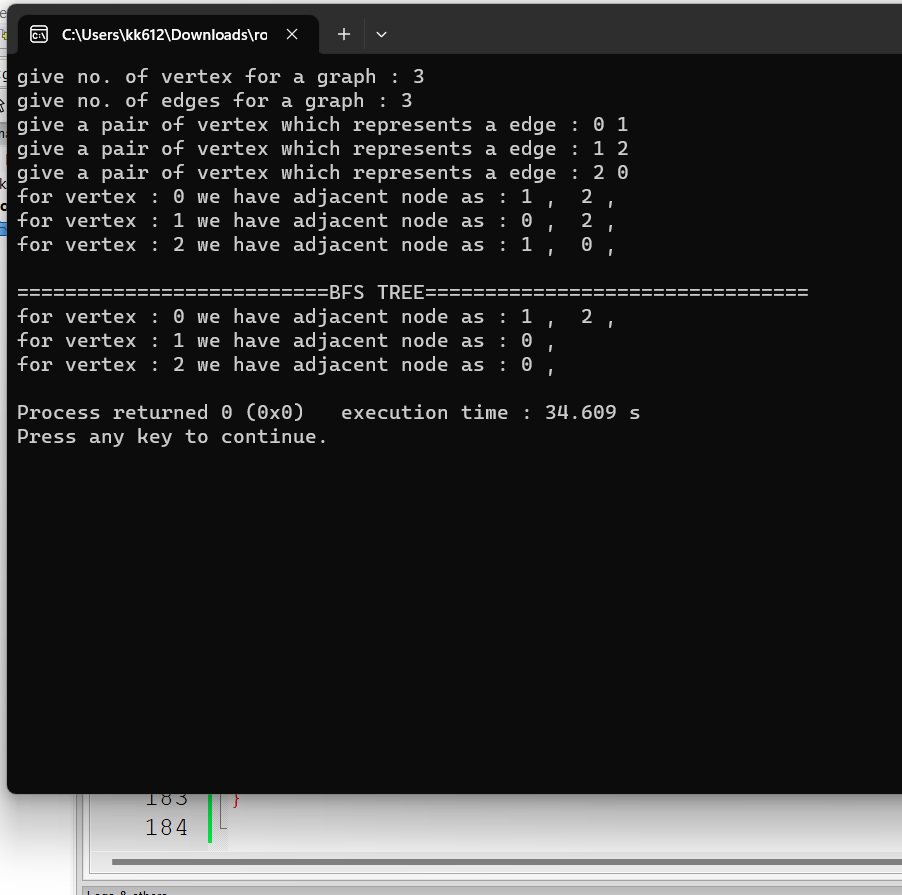
**7. BFS**



#include <iostream>

#include <queue>

using namespace std;

class node{

int vertex;

friend class LinkedList;

public:

node\* next;

node(){

next=0;

}

node(int v){

vertex = v;

next=0;

}

int get\_vertex(){

return vertex;

}

};

//linkedlist

class LinkedList{

friend class node;

public:

node \*head;

node \*last;

LinkedList(){

head=0;

last=0;

}

void set\_head(node \*ptr){

head =ptr;

last = ptr;

}

void set\_next(node \*ptr){

if(head !=0){

last->next = ptr;

}

last = ptr;

}

};

void print(LinkedList \*vertex\_arr,int size){

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

cout<<"for vertex : "<<i<<" we have adjacent node as : ";

node \*temp = vertex\_arr[i].head;

temp = temp->next;

while(temp !=0){

cout<<temp->get\_vertex();

cout<<" , ";

temp = temp->next;

}

cout<<endl;

}

}

void BFS(LinkedList \*vertex\_arr,int size,int src){

int \*discovered = new int[size];

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

discovered[i] =0;

} //setting discovered for every vertex as false

discovered[src] = 1; // for s discovered is true

int dis=1;

int layer\_count=0;

//current bfs tree = null

LinkedList \*bfs\_tree = new LinkedList[size];

queue<int> q;

q.push(src);

bfs\_tree[src].set\_head(new node(src));

while(!q.empty()){

int u = q.front();

q.pop();

node \*v= vertex\_arr[u].head;

v = v->next;

while(v !=0){ //breaks this loop will add adjacent vertices of node

if(discovered[v->get\_vertex()]==0){ //if not discovered

if(bfs\_tree[v->get\_vertex()].head == 0){

bfs\_tree[v->get\_vertex()].set\_head(new node(v->get\_vertex()));

}

discovered[v->get\_vertex()]=1;

dis++;

bfs\_tree[u].set\_next(new node(v->get\_vertex()));

bfs\_tree[v->get\_vertex()].set\_next(new node(u));

q.push(v->get\_vertex());

}

v = v->next;

}

layer\_count++;

}

print(bfs\_tree,size);

}

int main()

{

cout<<"give no. of vertex for a graph : ";

int no\_vertex=0;

cin>>no\_vertex;

//initalization of the array of list , each has a vertex has head

LinkedList \*vertex\_arr = new LinkedList[no\_vertex];

//giving values to each head vertex

for(int i=0;i<no\_vertex;i++){

vertex\_arr[i].set\_head(new node(i));

}

cout<<"give no. of edges for a graph : ";

int no\_edges=0;

cin>>no\_edges;

//building a Graph :: Adjancent list

for(int i=0;i<no\_edges;i++){

cout<<"give a pair of vertex which represents a edge : ";

int u=0,v=0;

cin>>u>>v;

vertex\_arr[u].set\_next(new node(v));

vertex\_arr[v].set\_next(new node(u));

}

print(vertex\_arr,no\_vertex);

cout<<endl;

cout<<"==========================BFS TREE================================"<<endl;

BFS(vertex\_arr,no\_vertex,0);

return 0;

}