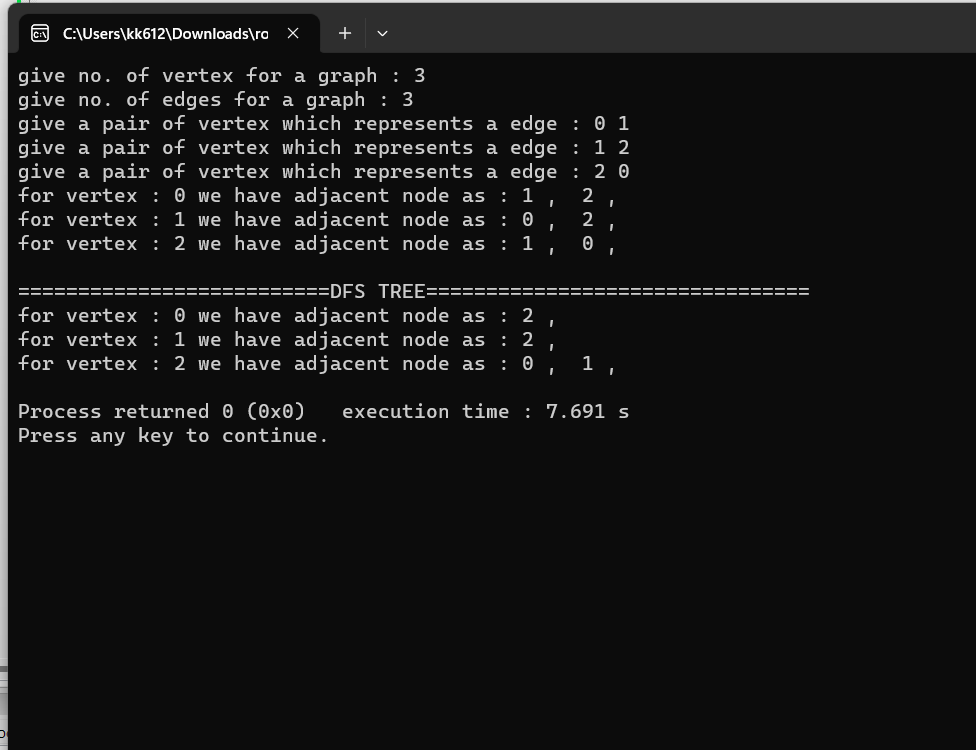
**8.DFS**



#include <iostream>

#include <stack>

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 DFS(LinkedList \*vertex\_arr,int size,int s){

stack<int> stk;//

stk.push(s); //0

int \*explored = new int[size];

int \*parent = new int[size];

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

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

explored[i]=0;

}

int previous\_explored = s; //0

while(!stk.empty()){ //0

int u = stk.top();//0 ,1, 3 , 2

if(explored[u]==0){

explored[u]=1; //0, 1 , 3 , 2

dfs\_tree[u].set\_head(new node(u)); //0 , 1 , 3 , 2

if(previous\_explored != u){ // 1 , 3 , 2

parent[u] = previous\_explored; //

dfs\_tree[previous\_explored].set\_next(new node(u));

dfs\_tree[u].set\_next(new node(previous\_explored));

previous\_explored = u; //1 , 3 , 2

}

node \*temp = vertex\_arr[u].head; //0 1 3 2

previous\_explored = u;//0 1 3 2

temp = temp->next; //

int count=0,county=0;

while(temp !=0){

county++;

if(explored[temp->get\_vertex()]==0){

stk.push(temp->get\_vertex());// [stk :

count++;

}

temp = temp->next;

}

if(count ==0){ // 0 != 1

//dead end back track

previous\_explored = parent[u];

stk.pop();

}

}else{

previous\_explored = parent[u];//2

stk.pop();

}

}

print(dfs\_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<<"==========================DFS TREE================================"<<endl;

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

return 0;

}