Ex 12: BFS AND DFS

Name: Rowin.U

Reg No.:231801141

BREATH FIRST SEARCH:

PROGRAM:

#include <stdio.h>

#include <stdlib.h>

struct node {

int vertex;

struct node\* next;

};

struct adj\_list {

struct node\* head;

};

struct graph {

int num\_vertices;

struct adj\_list\* adj\_lists;

int\* visited;

};

struct node\* new\_node(int vertex) {

struct node\* new\_node = (struct node\*)malloc(sizeof(struct node));

new\_node->vertex = vertex;

new\_node->next = NULL;

return new\_node;

}

struct graph\* create\_graph(int n) {

struct graph\* graph = (struct graph\*)malloc(sizeof(struct graph));

graph->num\_vertices = n;

graph->adj\_lists = (struct adj\_list\*)malloc(n \* sizeof(struct adj\_list));

graph->visited = (int\*)malloc(n \* sizeof(int));

int i;

for (i = 0; i< n; i++) {

graph->adj\_lists[i].head = NULL;

graph->visited[i] = 0;

}

return graph;

}

void add\_edge(struct graph\* graph, int src, int dest) {

struct node\* new\_node1 = new\_node(dest);

new\_node1->next = graph->adj\_lists[src].head;

graph->adj\_lists[src].head = new\_node1;

struct node\* new\_node2 = new\_node(src);

new\_node2->next = graph->adj\_lists[dest].head;

graph->adj\_lists[dest].head = new\_node2;

}

void bfs(struct graph\* graph, int v) {

int queue[1000];

int front = -1;

int rear = -1;

graph->visited[v] = 1;

queue[++rear] = v;

while (front != rear) {

int current\_vertex = queue[++front];

printf("%d ", current\_vertex);

struct node\* temp = graph->adj\_lists[current\_vertex].head;

while (temp != NULL) {

int adj\_vertex = temp->vertex;

if (graph->visited[adj\_vertex] == 0) {

graph->visited[adj\_vertex] = 1;

queue[++rear] = adj\_vertex;

}

temp = temp->next;

}

}

}

int main() {

struct graph\* graph = create\_graph(6);

add\_edge(graph, 0, 1);

add\_edge(graph, 0, 2);

add\_edge(graph, 1, 3);

add\_edge(graph, 1, 4);

add\_edge(graph, 2, 4);

add\_edge(graph, 3, 4);

add\_edge(graph, 3, 5);

add\_edge(graph, 4,5);

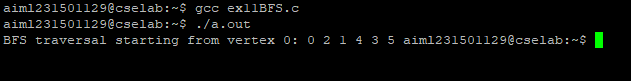
printf("BFS traversal starting from vertex 0: ");

bfs(graph, 0);

return 0;

}

OUTPUT:



DEPTH FIRST SEARCH:

PROGRAM:

#include <stdio.h>

#include <stdlib.h>

int vis[100];

struct Graph {

int V;

int E;

int\*\* Adj;

};

struct Graph\* adjMatrix()

{

struct Graph\* G = (struct Graph\*)

malloc(sizeof(struct Graph));

if (!G) {

printf("Memory Error\n");

return NULL;

}

G->V = 7;

G->E = 7;

G->Adj = (int\*\*)malloc((G->V) \* sizeof(int\*));

for (int k = 0; k < G->V; k++) {

G->Adj[k] = (int\*)malloc((G->V) \* sizeof(int));

}

for (int u = 0; u < G->V; u++) {

for (int v = 0; v < G->V; v++) {

G->Adj[u][v] = 0;

}

}

G->Adj[0][1] = G->Adj[1][0] = 1;

G->Adj[0][2] = G->Adj[2][0] = 1;

G->Adj[1][3] = G->Adj[3][1] = 1;

G->Adj[1][4] = G->Adj[4][1] = 1;

G->Adj[1][5] = G->Adj[5][1] = 1;

G->Adj[1][6] = G->Adj[6][1] = 1;

G->Adj[6][2] = G->Adj[2][6] = 1;

return G;

}

void DFS(struct Graph\* G, int u)

{

vis[u] = 1;

printf("%d ", u);

for (int v = 0; v < G->V; v++) {

if (!vis[v] && G->Adj[u][v]) {

DFS(G, v);

}

}

}

void DFStraversal(struct Graph\* G)

{

for (int i = 0; i < 100; i++) {

vis[i] = 0;

}

for (int i = 0; i < G->V; i++) {

if (!vis[i]) {

DFS(G, i);

}

}

}

void main()

{

struct Graph\* G;

G = adjMatrix();

DFStraversal(G);

}

OUTPUT:

