

Assignment -11

Name: Ujjwal Kumar Jha

Registration No. 20194196

Group: C₂

Q 1. WAP to construct a Graph using adjacency matrix and implement the following:

- a. Depth first search.**
- b. Breadth first search.**

a.

```
#include <stdio.h>
#include <stdlib.h>
/*          ADJACENCY MATRIX          */
int source,V,E,time,visited[20],G[20][20];
void DFS(int i)
{
    int j;
    visited[i]=1;
    printf(" %d->",i+1);
    for(j=0;j<V;j++)
    {
        if(G[i][j]==1&&visited[j]==0)
            DFS(j);
    }
}
int main()
{
    int i,j,v1,v2;
    printf("\t\t\t\tGraphs\n");
    printf("Enter the no of edges:");
    scanf("%d",&E);
    printf("Enter the no of vertices:");
    scanf("%d",&V);
    for(i=0;i<V;i++)
    {
        for(j=0;j<V;j++)
```

```

        G[i][j]=0;
    }
    /*    creating edges :P    */
    for(i=0;i<E;i++)
    {
        printf("Enter the edges (format: V1 V2) : ");
        scanf("%d%d",&v1,&v2);
        G[v1-1][v2-1]=1;

    }

    for(i=0;i<V;i++)
    {
        for(j=0;j<V;j++)
            printf(" %d ",G[i][j]);
        printf("\n");
    }
    printf("Enter the source: ");
    scanf("%d",&source);
    DFS(source-1);
    return 0;
}

```

The screenshot shows a Visual Studio Code window with a terminal running a C program. The terminal output is as follows:

```

PS C:\Users\inspiron\Desktop\oopGui> cd DS11
PS C:\Users\inspiron\Desktop\oopGui\DS11> gcc 1.c
PS C:\Users\inspiron\Desktop\oopGui\DS11> ./a.exe
Graphs
Enter the no of edges:10
Enter the no of vertices:8
Enter the edges (format: V1 V2) : 0 1
Enter the edges (format: V1 V2) : 0 2
Enter the edges (format: V1 V2) : 1 3
Enter the edges (format: V1 V2) : 1 4
Enter the edges (format: V1 V2) : 2 5
Enter the edges (format: V1 V2) : 2 6
Enter the edges (format: V1 V2) : 2 7
Enter the edges (format: V1 V2) : 4 7
Enter the edges (format: V1 V2) : 5 7
Enter the edges (format: V1 V2) : 6 7
0 0 1 1 0 0 0 0
0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Enter the source: 4
4-> 7->
PS C:\Users\inspiron\Desktop\oopGui\DS11>

```

The Visual Studio Code interface shows the file explorer on the left with '1.c' selected. The terminal window at the bottom displays the command prompt and the program's execution. The status bar at the bottom indicates the current line and column (Ln 40, Col 25) and the file encoding (UTF-8, CRLF).

b.

```
#include<stdio.h>
#include<conio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
void bfs(int v) {
    for (i=1;i<=n;i++)
        if(a[v][i] && !visited[i])
            q[++r]=i;
    if(f<=r) {
        visited[q[f]]=1;
        bfs(q[f++]);
    }
}
void main() {
    int v;
    // clrscr();
    printf("\n Enter the number of vertices:");
    scanf("%d",&n);
    for (i=1;i<=n;i++) {
        q[i]=0;
        visited[i]=0;
    }
    printf("\n Enter graph data in matrix form:\n");
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    printf("\n Enter the starting vertex:");
    scanf("%d",&v);
    bfs(v);
    printf("\n The node which are reachable are:\n");
    for (i=1;i<=n;i++)
        if(visited[i])
            printf("%d\t",i); else
            printf("\n Bfs is not possible");
    getch();
}
```

```
File Edit Selection View Go Run Terminal Help
1b.c - oopGui - Visual Studio Code

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL
1:a

Enter the number of vertices:7

Enter graph data in matrix form:
0 0 0 1 1 0 0
0 0 0 0 0 2 0
0 2 3 0 0 0 0
0 0 1 0 0 0 0
1 5 0 0 1 0 0
1 2 3 0 0 0 0
2 9 0 0 0 0 0

Enter the starting vertex:4

The node which are reachable are:
1 2 3 4 5 6
Bfs is not possible
```

Q 2. WAP to construct a Graph using adjacency list representation and implement the following:

- a. Create an edge between two nodes.**
- b. Remove an edge between two nodes.**
- c. Degree of a particular node.**
- d. Create a new node.**
- e. Remove an existing node.**

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int v,weight;
    struct node *next;
}node;
typedef struct vertex{
    int v,degree;
    node *head;
}vertex;
vertex adj[100001];
int V = 0;
```

```

//utility function to return the index(in adj array) of a given vertex whose value is v
int findIndex(int v)
{
    int i;
    for(i=0;i<V;i++)
    {
        if(adj[i].v==v)
            break;
    }
    return i;
}

//function to add a node in the array of adj
void createNode()
{
    printf("Enter the value of vertex : ");
    int v;
    scanf("%d",&v);
    adj[V].v = v;
    adj[V].degree = 0;
    adj[V++].head = NULL;
    printf("Vertex with value %d is successfully added to the graph\n",v);
}

//function to create an edge (bidirectional) between two adj
void addEdge(int vertex1, int vertex2)
{
    //first finding the indices corresponding to the vertex values vertex1,vertex2
    int v1 = findIndex(vertex1);
    int v2 = findIndex(vertex2);
    if(v1==V || v2==V){
        printf("One or both of the adj with values %d and %d may not exist\n",vertex1,vertex2);
        return;
    }
    //adding v2 to adjacency list of v1
    printf("Enter the weight of the edge between %d and %d : ",v1,v2);
    int w;
    scanf("%d",&w);
    node * newNode1 = (node*)malloc(sizeof(node));
    newNode1->v = adj[v2].v;
    newNode1->weight = w;
    newNode1->next = adj[v1].head;
    adj[v1].head = newNode1;
    adj[v1].degree += 1;
}

```

```

    //adding v1 to adjacency list of v2;
    node * newNode2 = (node*)malloc(sizeof(node));
    newNode2->v = adj[v1].v;
    newNode2->weight = w;
    newNode2->next = adj[v2].head;
    adj[v2].head = newNode2;
    adj[v2].degree += 1;
    printf("Edge between the vertices %d and %d is added\n",vertex1,vertex2);
}
//utility function to delete a node in the adjacency list of a vertex
void deletion(int v1,int v2)
{
    node *p = adj[v1].head,*q=NULL;
    if(p!=NULL&& p->v==adj[v2].v)
    {
        adj[v1].head = p->next;
        free(p);
        adj[v1].degree -= 1;
        return;
    }
    while(p!=NULL&& p->v!=adj[v2].v)
    {
        q = p;
        p = p->next;
    }
    if(p==NULL){
        // printf("Such an edge doesn't exist\n");
        return;
    }
    q->next = p->next;
    free(p);

    adj[v1].degree -= 1;
}
//function to remove an edge between two nodes
void removeEdge(int vertex1,int vertex2)
{
    //first finding the indices corresponding to the vertex values vertex1,vertex
    2
    int v1 = findIndex(vertex1);
    int v2 = findIndex(vertex2);
    if(v1==V || v2==V){
        printf("One or both of the adj with values %d and %d may not exist\n",ver
tex1,vertex2);
        return;
    }

```

```

    }
    //finding v2 in the adjacency list of v1 and then deleting that entire node
    deletion(v1,v2);
    //finding v1 in the adjacency list of v2 and then deleting that entire node
    deletion(v2,v1);
    printf("Edge between the vertices %d and %d is removed\n",vertex1,vertex2);
}
//utility function to perform swapping of adj in order to delete one of them
void swap(vertex *v1,vertex *v2)
{
    vertex temp = *v1;
    *v1 = *v2;
    *v2 = temp;
}
//function to remove a vertex given its value
void removeVertex(int v)
{
    //we will swap the vertex with the last vertex
    int i = findIndex(v),j;
    swap(&adj[i],&adj[V-1]);
    //now go through the adjacency list of all the vertices and delete the node i
    (which will now be at index V-1)
    for(j=0;j<V-1;j++)
        deletion(j,V-1);
    //delete all the nodes in the adjacency list of V-1
    while(adj[V-1].head!=NULL){
        node *p = adj[V-1].head;
        adj[V-1].head = (adj[V-1].head)->next;
        free(p);
    }
    //now simply decrement the count of V because one vertex is deleted
    V--;
    printf("Vertex with value %d is removed successfully\n",v);
}
void printGraph()
{
    int i;
    for(i=0;i<V;i++)
    {
        node *p = adj[i].head;
        printf("%d -> ",adj[i].v);
        while(p!=NULL)
        {
            printf("%d - ",p->v);
            p = p->next;
        }
    }
}

```

```

    }
    printf("\n");
}
}
int main()
{
    int choice;
    int v1,v2;
    printf("Choices are : \n");
    printf("1. Create a new node : \n");
    printf("2. Create an edge between two nodes : \n");
    printf("3. Remove an edge between two nodes : \n");
    printf("4. Find the degree of a node : \n");
    printf("5. Remove an exisiting node : \n");
    printf("6. Print graph : \n");
    printf("7. Exit :\n");
    while(1){
        printf("\nEnter your choice : ") ;
        scanf("%d",&choice);
        switch(choice){
            case 1:
                createNode();
                break;
            case 2:
                printf("Enter the adj you want to connect with an edge : ");
                scanf("%d%d",&v1,&v2);
                addEdge(v1,v2);
                break;
            case 3:
                printf("Enter the adj whose edge you want to remove : ");
                scanf("%d%d",&v1,&v2);
                removeEdge(v1,v2);
                break;
            case 4:
                printf("Enter the vertex whose degree you want to find : ");
                scanf("%d",&v1);
                int i = findIndex(v1);
                printf("The degree of node with value %d is %d\n",adj[i].v,adj[i]
.degree);
                break;
            case 5:
                printf("Enter the vertex you want to remove : ");
                scanf("%d",&v1);
                removeVertex(v1);
                break;

```



```

        case 6:
            printGraph();
            break;
        case 7:
            exit(0);
        default :
            printf("Invalid choice\n");
            break;
    }
}
}

```

The screenshot shows the Visual Studio Code interface with the following components:

- File Explorer:** Shows the project structure with files like `2.c` and `a.exe`.
- Terminal:** Displays the execution of the program. The output shows the menu choices and the user's interactions:


```

PS C:\Users\inspiron\Desktop\oopGui\DS11> gcc 2.c
PS C:\Users\inspiron\Desktop\oopGui\DS11> ./a.exe
Choices are :
1. Create a new node :
2. Create an edge between two nodes :
3. Remove an edge between two nodes :
4. Find the degree of a node :
5. Remove an existing node :
6. Print graph :
7. Exit :

Enter your choice : 1
Enter the value of vertex : 8
Vertex with value 8 is successfully added to the graph

Enter your choice : 6
8 ->

Enter your choice : 1
Enter the value of vertex : 9
Vertex with value 9 is successfully added to the graph

Enter your choice : 2
Enter the adj you want to connect with an edge : 3
3 1
One or both of the adj with values 3 and 3 may not exist

Enter your choice : Enter the value of vertex : 4
Vertex with value 4 is successfully added to the graph

Enter your choice : 6
8 ->
9 ->
4 ->

Enter your choice : 
      
```
- Bottom Bar:** Shows the status bar with information like "Ln 182, Col 40", "Spaces: 4", "UTF-8", "CRLF", "C", "Go Live", "Win32", and the system clock showing "8:45 PM 11/18/2020".