Practical-9

1.Graphs (Bfs).

Aim:

To traverse a graph using Breadth-First Search method.

Theory:

Traversed graph using linked list also maintained node list and edge list using linked list data structure and in this it traverse from the source and explores the graph level by level.

Code:

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
 int data;
 struct node * next;
};

struct edge
{
 int src,dest;
 struct edge * next;
};

struct node * vertex=NULL;
struct edge * edgelist=NULL;
int visited[100];

struct node * createNode(int data)
```

```
{
 struct node * temp=(struct node *)malloc(sizeof(struct node));
 temp->data=data;
 temp->next=NULL;
 return temp;
}

struct edge * createEdge(int s,int d)
{
 struct edge * temp=(struct edge *)malloc(sizeof(struct edge));
 temp->src=s;
 temp->dest=d;
 temp->next=NULL;
 return temp;
}

void addVertex(int data)
{
 struct node * temp=createNode(data);
 if(vertex==NULL) vertex=temp;
 else
 {
  struct node * p=vertex;
  while(p->next!=NULL) p=p->next;
  p->next=temp;
 }
}

void addEdge(int s,int d)
```

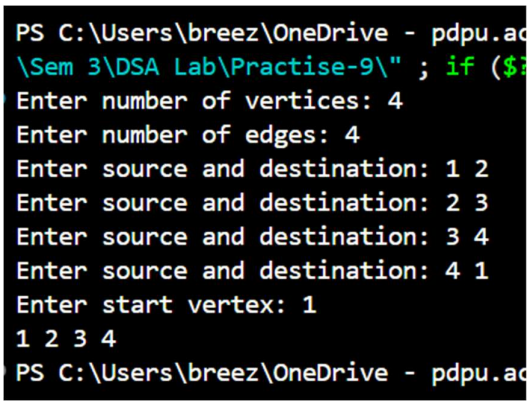
```
{
  struct edge * temp=createEdge(s,d);
  if(edgelist==NULL) edgelist=temp;
  else
  {
    struct edge * p=edgelist;
    while(p->next!=NULL) p=p->next;
    p->next=temp;
  }
}

void bfs(int start)
{
  int q[100],front=0,rear=0;
  q[rear++]=start;
  visited[start]=1;
  while(front<rear)
  {
    int v=q[front++];
    printf("%d ",v);
    struct edge * e=edgelist;
    while(e!=NULL)
    {
      if(e->src==v && visited[e->dest]==0)
      {
        q[rear++]=e->dest;
        visited[e->dest]=1;
      }
      e=e->next;
```

```
   }
  }
}

int main()
{
  int n,e,i,s,d,start;
  printf("Enter number of vertices: ");
  scanf("%d",&n);
  for(i=1;i<=n;i++) addVertex(i);
  printf("Enter number of edges: ");
  scanf("%d",&e);
  for(i=0;i<e;i++)
  {
    printf("Enter source and destination: ");
    scanf("%d%d",&s,&d);
    addEdge(s,d);
  }
  printf("Enter start vertex: ");
  scanf("%d",&start);
  bfs(start);
  return 0;
}
```

Output:



2.Graphs (Dfs).

Aim:

To traverse a graph using Depth-First Search method.

Theory:

Traversed graph using linked list also maintained node list and edge list using linked list data structure and in this it explores all the path from the source before coming back.

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

 int data;

 struct node * next;

};


struct edge

{

 int src,dest;

 struct edge * next;

};


struct node * vertex=NULL;
```

```
struct edge * edgelist=NULL;
int visited[100];

struct node * createNode(int data)
{
 struct node * temp=(struct node *)malloc(sizeof(struct node));
 temp->data=data;
 temp->next=NULL;
 return temp;
}

struct edge * createEdge(int s,int d)
{
 struct edge * temp=(struct edge *)malloc(sizeof(struct edge));
 temp->src=s;
 temp->dest=d;
 temp->next=NULL;
 return temp;
}

void addVertex(int data)
{
 struct node * temp=createNode(data);
 if(vertex==NULL) vertex=temp;
 else
 {
  struct node * p=vertex;
  while(p->next!=NULL) p=p->next;
  p->next=temp;
```

```
 }
}

void addEdge(int s,int d)
{
 struct edge * temp=createEdge(s,d);
 if(edgelist==NULL) edgelist=temp;
 else
 {
  struct edge * p=edgelist;
  while(p->next!=NULL) p=p->next;
  p->next=temp;
 }
}

void dfs(int v)
{
 printf("%d ",v);
 visited[v]=1;
 struct edge * e=edgelist;
 while(e!=NULL)
 {
  if(e->src==v && visited[e->dest]==0)
    dfs(e->dest);
  e=e->next;
 }
}

int main()
```

```
{
 int n,e,i,s,d,start;
 printf("Enter number of vertices: ");
 scanf("%d",&n);
 for(i=1;i<=n;i++) addVertex(i);
 printf("Enter number of edges: ");
 scanf("%d",&e);
 for(i=0;i<e;i++)
 {
  printf("Enter source and destination: ");
  scanf("%d%d",&s,&d);
  addEdge(s,d);
 }
 printf("Enter start vertex: ");
 scanf("%d",&start);
 dfs(start);
 return 0;
}
```

Output:

```
PS C:\Users\breez\OneDrive - pdpu.a
\Sem 3\DSA Lab\Practise-9\" ; if ($
Enter number of vertices: 4
Enter number of edges: 4
Enter source and destination: 1 2
Enter source and destination: 2 3
Enter source and destination: 3 4
Enter source and destination: 4 1
Enter start vertex: 2
2 3 4 1
PS C:\Users\breez\OneDrive - pdpu.a
```

Link for all codes:

https://github.com/PanavPatel06/DSA-Lab/tree/main/Practise-9