Recursive Depth First Search

Graph Input: Adjacency Matrix

```
#include<stdio.h>
#include<stdlib.h>
struct graph {
   int a[20][20];
   int n;
};
void dfs(struct graph *g, int visited[], int v){
   int i;
      visited[v]=1;
   for(i=0;i<g->n;i++){
      if(g->a[v][i] && !visited[i]){
         printf("  -- >  %d",i);
         dfs(g,visited,i);
         printf(" b--> %d ", v);
      }
   }
}
int main(){
   struct graph G;
   int visited[20];
   int count=0;
   int i,j;
   printf("\n Enter the number of vertices in the Graph: ");
   scanf("%d", &G.n);
   for(i=0;i<G.n;i++){
      visited[i]=0;
      for(j=0;j<G.n;j++)
         G.a[i][j]=0;
   }
   printf("\n Enter the adjacenecy matrix\n ");
   for(i=0;i<G.n;i++)
      for(j=0;j<G.n;j++)
         scanf("%d", &G.a[i][j]);
   printf("\n 0");
   dfs(&G,visited,0);
}
```

Iterative Depth First Search
Graph Input: Adjacency Matrix

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

struct graph {
    int a[20][20];
    int n;
};
struct stack{
    int st[40];
    int top;
};
struct stack push(struct stack s, int item){
    //printf("%d",s.top);
    s.top++;
    s.st[s.top]=item;
    //printf("%d",s.st[s.top]);
    return s;
}
struct stack pop(struct stack s){
    s.top--;
    return s;
}
int isstackempty(struct stack s){
    if(s.top==-1){
        return 1;
    }
    else
    {
        return 0;
    }
  }
void dfs(struct graph *g, int visited[], int v, struct stack s ){
    int i,j;
      s=push(s,v);
    visited[v]=1;
    printf("%d --> ", v);
    //printf("%d ",v);
    while(!isstackempty(s)){
```

```
        v=s.st[s.top];
        //s=pop(s);
        for(i=0;i<g->n;i++){
            if(g->a[v][i] && !visited[i]){
                s=push(s,i);
                visited[i]=1;
                printf("%d --> ",i);
                break;
            }
        }
        if(i==g->n){
            s=pop(s);
            j=s.st[s.top];
            printf("%d --> ",j);
        }
    }
}

int main(){
    struct graph G;
    int visited[20];
    int count=0;
    int i,j;
    struct stack s;
    s.top=-1;
    printf("\n Enter the number of vertices in the Graph: ");
    scanf("%d", &G.n);
    for(i=0;i<G.n;i++){
        visited[i]=0;
        for(j=0;j<G.n;j++)
            G.a[i][j]=0;
    }
    printf("\n Enter the adjacenecy matrix\n ");
    for(i=0;i<G.n;i++)
        for(j=0;j<G.n;j++)
            scanf("%d", &G.a[i][j]);
    //printf("\n 0");
    dfs(&G,visited,0,s);
}
```

Recursive Depth First Search
Graph Input: Adjacency List

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int node_no;
   struct node *next;
};
struct graph{
   int n;
   struct node **nodelist;
};
struct node *insertrear(struct node *nodelist, struct node *newnode){
   struct node *temp;
   if(nodelist==NULL){
      nodelist=newnode;
   }
   else{
      temp=nodelist;
      while(temp->next!=NULL)
         temp=temp->next;
      temp->next=newnode;
   }
   return nodelist;
}
void display(struct graph g){
struct node *temp;
int i;

for(i=0;i<g.n;i++){
   temp=g.nodelist[i];
      printf(" %d:-->",i);
      while(temp!=NULL){
         printf("%d -->",temp->node_no);
         temp=temp->next;
      }
      printf(" NULL\n ");
}
}
```

```
void dfs(struct graph *g, int *visited, int v){
    int i;
    struct node *temp;
    visited[v]=1;
    temp=g->nodelist[v];

    while(temp!=NULL){
        if(!visited[temp->node_no]){
            printf("%d-->",temp->node_no);
            dfs(g,visited,temp->node_no);
        }
        else{
        temp=temp->next;
        }
    }

}

int main(){
    int i, adjvertex,j,k;
    struct graph g;
    struct node *newnode;
    int *visited;
    printf("\n Enter the number of nodes in the graph: ");
    scanf("%d",&g.n);
    g.nodelist=(struct node **)malloc(g.n*sizeof(struct node *));
    visited=(int *)malloc(sizeof(g.n*sizeof(int)));
    for(i=0;i<g.n;i++)
        visited[i]=0;
    for(i=0;i<g.n;i++)
        g.nodelist[i]=NULL;
    for(i=0;i<g.n;i++){
        printf("\n Enter the number of nodes adjacent to %d: ",i);
        scanf("%d", &j);
        for(k=0;k<j;k++){
            printf("\n Enter the verticex adjacent to vertex %d.:  ",i);
            scanf("%d",&adjvertex);
            struct node *newnode=(struct node *)malloc(sizeof(struct node));
            newnode->node_no=adjvertex;
```

```
            newnode->next=NULL;
            g.nodelist[i]=insertrear(g.nodelist[i],newnode);

        }
    }
    display(g);
    printf("\n 0 --> ");
    dfs(&g, visited,0);

}
```