CSE 2003: Lab Assignment #13

Due on Thursday, April 13, 2017

Prof. Shaik Naseera 2:00pm

Jacob John

Contents

Problem 1

Write a C program to Implement Depth First Search using stack

Listing 1: Depth first Search Program in C

```c
/*Program for traversing a directed graph through DFS,
visiting only vertices reachable from start vertex*/

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

#define MAX 100

#define initial 1
#define visited 2

int n;    /* Number of nodes in the graph */
int adj[MAX][MAX]; /*Adjacency Matrix*/
int state[MAX]; /*Can be initial or visited */

void DF_Traversal();
void DFS(int v);
void create_graph();

int stack[MAX];
int top = -1;
void push(int v);
int pop();
int isEmpty_stack();

int main()
{
     create_graph();
     DF_Traversal();
}/*End of main()*/

void DF_Traversal()
{
     int v;

     for(v=0; v<n; v++)
          state[v]=initial;

     printf("Enter starting node for Depth First Search : ");
     scanf("%d",&v);
     DFS(v);
     printf("\n");
}/*End of DF_Traversal( )*/

void DFS(int v)
{
     int i;
     push(v);
```

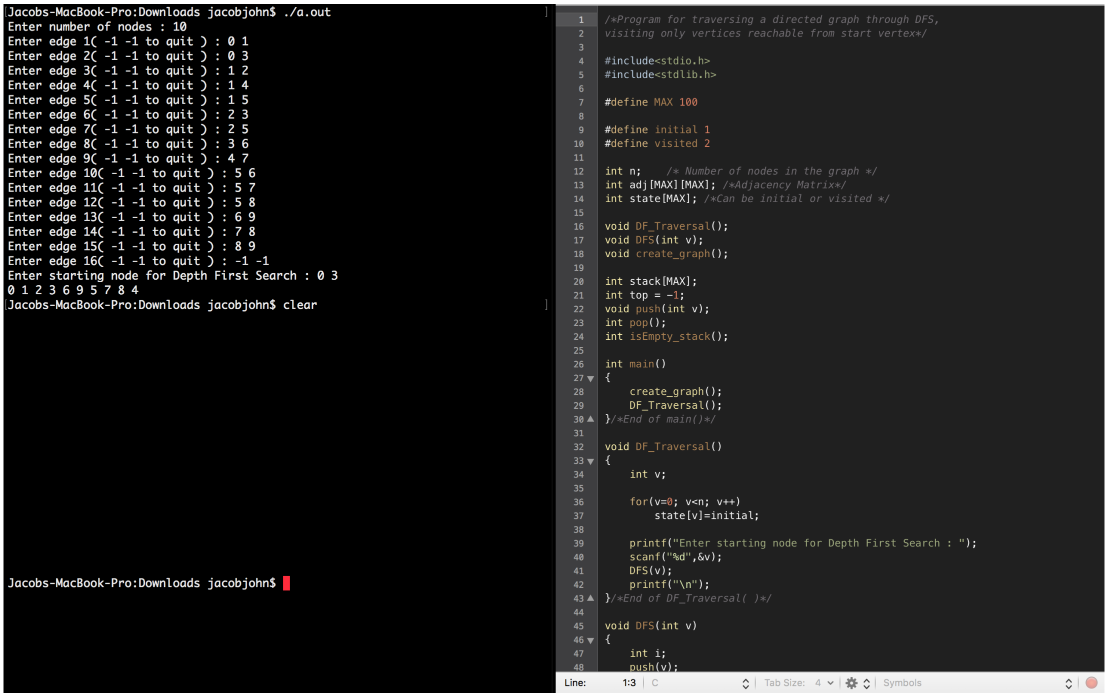
Problem 1 continued on next page. . .

```
     while(!isEmpty_stack())
     {
          v = pop();
          if(state[v]==initial)
          {
               printf("%d ",v);
               state[v]=visited;
          }
          for(i=n-1; i>=0; i--)
          {
               if(adj[v][i]==1 && state[i]==initial)
                    push(i);
          }
     }
}/*End of DFS( )*/

void push(int v)
{
     if(top == (MAX-1))
     {
          printf("Stack Overflow\n");
          return;
     }
     top=top+1;
     stack[top] = v;

}/*End of push()*/

int pop()
{
     int v;
     if(top == -1)
     {
          printf("Stack Underflow\n");
          exit(1);
     }
     else
     {
          v = stack[top];
          top=top-1;
          return v;
     }
}/*End of pop()*/

int isEmpty_stack( )
{
  if(top == -1)
       return 1;
  else
       return 0;
}/*End if isEmpty_stack()*/

void create_graph()
```

Problem 1 continued on next page. . .

```
{
    int i,max_edges,origin,destin;

    printf("Enter number of nodes : ");
    scanf("%d",&n);
    max_edges=n*(n-1);

    for(i=1;i<=max_edges;i++)
    {
        printf("Enter edge %d( -1 -1 to quit ) : ",i);
        scanf("%d %d",&origin,&destin);

        if( (origin == -1) && (destin == -1) )
            break;

        if( origin >= n || destin >= n || origin<0 || destin<0)
        {
            printf("Invalid edge!\n");
            i--;
        }
        else
        {
            adj[origin][destin] = 1;
        }
    }
}
```

Output:



```
Jacobs-MacBook-Pro:Downloads jacobjohn$ ./a.out
Enter number of nodes : 10
Enter edge 1( -1 -1 to quit ) : 0 1
Enter edge 2( -1 -1 to quit ) : 0 3
Enter edge 3( -1 -1 to quit ) : 1 2
Enter edge 4( -1 -1 to quit ) : 1 4
Enter edge 5( -1 -1 to quit ) : 1 5
Enter edge 6( -1 -1 to quit ) : 2 3
Enter edge 7( -1 -1 to quit ) : 2 5
Enter edge 8( -1 -1 to quit ) : 3 6
Enter edge 9( -1 -1 to quit ) : 4 7
Enter edge 10( -1 -1 to quit ) : 5 6
Enter edge 11( -1 -1 to quit ) : 5 7
Enter edge 12( -1 -1 to quit ) : 5 8
Enter edge 13( -1 -1 to quit ) : 6 9
Enter edge 14( -1 -1 to quit ) : 7 8
Enter edge 15( -1 -1 to quit ) : 8 9
Enter edge 16( -1 -1 to quit ) : -1 -1
Enter starting node for Depth First Search : 0 3
0 1 2 3 6 9 5 7 8 4
Jacobs-MacBook-Pro:Downloads jacobjohn$ clear
```

```
Jacobs-MacBook-Pro:Downloads jacobjohn$
```

```
/*Program for traversing a directed graph through DFS,
visiting only vertices reachable from start vertex*/

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

#define MAX 100

#define initial 1
#define visited 2

int n;    /* Number of nodes in the graph */
int adj[MAX][MAX]; /*Adjacency Matrix*/
int state[MAX]; /*Can be initial or visited */

void DF_Traversal();
void DFS(int v);
void create_graph();

int stack[MAX];
int top = -1;
void push(int v);
int pop();
int isEmpty_stack();

int main()
{
    create_graph();
    DF_Traversal();
}/*End of main()*/

void DF_Traversal()
{
    int v;

    for(v=0; v<n; v++)
        state[v]=initial;

    printf("Enter starting node for Depth First Search : ");
    scanf("%d",&v);
    DFS(v);
    printf("\n");
}/*End of DF_Traversal( )*/

void DFS(int v)
{
    int i;
    push(v);
```