CSE 2003: Lab Assignment #12

Due on Thursday, April 13, 2017

Prof. Shaik Naseera 2:00pm

Jacob John

Contents

Problem 1

Write a C program to Implement Breadth First Search using queue

Listing 1: Breadth first Search Program in C

```c
/* Program for traversing a directed graph through BFS,
visiting only those vertices that are reachable from start vertex */

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

#define MAX 100

#define initial 1
#define waiting 2
#define visited 3

int n;    /*Number of vertices in the graph*/
int adj[MAX][MAX]; /*Adjacency Matrix*/
int state[MAX]; /*can be initial, waiting or visited*/

void create_graph();
void BF_Traversal();
void BFS(int v);

int queue[MAX], front = -1,rear = -1;
void insert_queue(int vertex);
int delete_queue();
int isEmpty_queue();

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

void BF_Traversal()
{
     int v;

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

     printf("Enter starting vertex for Breadth First Search : ");
     scanf("%d", &v);
     BFS(v);
}/*End of BF_Traversal()*/

void BFS(int v)
{
     int i;

     insert_queue(v);
```

Problem 1 continued on next page...

```
     state[v] = waiting;

     while(!isEmpty_queue())
     {
          v = delete_queue( );
          printf("%d ",v);
          state[v] = visited;

          for(i=0; i<n; i++)
          {
               /*Check for adjacent unvisited vertices */
               if(adj[v][i] == 1 && state[i] == initial)
               {
                    insert_queue(i);
                    state[i] = waiting;
               }
          }
     }
     printf("\n");
}/*End of BFS()*/

void insert_queue(int vertex)
{
     if(rear == MAX-1)
          printf("Queue Overflow\n");
     else
     {
          if(front == -1)  /*If queue is initially empty */
               front = 0;
          rear = rear+1;
          queue[rear] = vertex ;
     }
}/*End of insert_queue()*/

int isEmpty_queue()
{
     if(front == -1 || front > rear)
          return 1;
     else
          return 0;
}/*End of isEmpty_queue()*/

int delete_queue()
{
     int del_item;
     if(front == -1 || front > rear)
     {
          printf("Queue Underflow\n");
          exit(1);
     }

     del_item = queue[front];
     front = front+1;
```

```
     return del_item;
}/*End of delete_queue() */

void create_graph()
{
     int i,max_edges,origin,destin;

     printf("Enter number of vertices : ");
     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;
          }
     }/*End of for*/
}/*End of create_graph()*/
```

Output:

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

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

```c
/* Program for traversing a directed graph through BFS,
visiting only those vertices that are reachable from start vertex */

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

#define MAX 100

#define initial 1
#define waiting 2
#define visited 3

int n;    /*Number of vertices in the graph*/
int adj[MAX][MAX]; /*Adjacency Matrix*/
int state[MAX]; /*can be initial, waiting or visited*/

void create_graph();
void BF_Traversal();
void BFS(int v);

int queue[MAX], front = -1,rear = -1;
void insert_queue(int vertex);
int delete_queue();
int isEmpty_queue();

main()
{
    create_graph();
    BF_Traversal();
}/*End of main()*/

void BF_Traversal()
{
    int v;

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

    printf("Enter starting vertex for Breadth First Search : ");
    scanf("%d", &v);
    BFS(v);
}/*End of BF_Traversal()*/

void BFS(int v)
{
    int i;

    insert_queue(v);
```

Line: 131:27 | C | Tab Size: 4 | create_graph