DSA mini Project

Mohith B
1RN22CS092

February 26, 2024

1 Problem Statement

C program to Implement graph using Linked list

2 Approach Used

We provide the Adjacency matrix and do Breadth First Search.

3 Functions Used

3.1 Read Adjacency Matrix

```
void read_adj_matrix(int a[20][20], int n)
{
  int i, j;
  printf("Enter the adjacency matrix\n");
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
}
```

3.2 Print Adjacency Matrix

```
void print_adj_matrix(int a[20][20], int n)
{
  int i, j;
  printf("The adjacency matrix is\n");
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
    {
      printf("%d ", a[i][j]);
    }
    printf("\n");
  }
}
```

3.3 Breadth First Search

```
NODE BFS(int a[20][20], int n, int source)
{
  int i, j, visited[20], q[20], f = 0, r = -1;
  NODE first = NULL, last = NULL, cur;
  for (i = 0; i < n; i++)
  {
    visited[i] = 0;
  }
  visited[source] = 1;
  q[++r] = source;
  while (f <= r)
  {
    i = q[f++];
    for (j = 0; j < n; j++)
    {
      if (a[i][j] == 1 && visited[j] == 0)
      {
        q[++r] = j;
        visited[j] = 1;
        cur = getnode();
        cur->info = j;
        cur->link = NULL;
        if (first == NULL)
        {
          first = cur;
          last = cur;
        }
        else
        {
          last->link = cur;
          last = cur;
        }
      }
    }
  }
  return first;
}
```

4 Final Code

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

void read_adj_matrix(int a[20][20], int n)
{
  int i, j;
  printf("Enter the adjacency matrix\n");
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
    {
      scanf("%d", &a[i][j]);
```

```
    }
  }
}

void print_adj_matrix(int a[20][20], int n)
{
  int i, j;
  printf("The adjacency matrix is\n");
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
    {
      printf("%d ", a[i][j]);
    }
    printf("\n");
  }
}

struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;

NODE getnode()
{
  NODE temp;
  temp = (NODE)malloc(sizeof(struct node));
  if (temp == NULL)
  {
    printf("memory not allocated");
    exit(0);
  }
  return temp;
}

NODE BFS(int a[20][20], int n, int source)
{
  int i, j, visited[20], q[20], f = 0, r = -1;
  NODE first = NULL, last = NULL, cur;
  for (i = 0; i < n; i++)
  {
    visited[i] = 0;
  }
  visited[source] = 1;
  q[++r] = source;
  while (f <= r)
  {
    i = q[f++];
    for (j = 0; j < n; j++)
    {
      if (a[i][j] == 1 && visited[j] == 0)
      {
        q[++r] = j;
        visited[j] = 1;
```

```
        cur = getnode();
        cur->info = j;
        cur->link = NULL;
        if (first == NULL)
        {
          first = cur;
          last = cur;
        }
        else
        {
          last->link = cur;
          last = cur;
        }
      }
    }
  }
  return first;
}

int main()
{
  int a[20][20], n, source;
  NODE first;
  printf("Enter the number of vertices\n");
  scanf("%d", &n);
  read_adj_matrix(a, n);
  print_adj_matrix(a, n);
  for (int i = 0; i < n; i++)
  {
    source = i;
    first = BFS(a, n, source);
    printf("\nThe BFS traversal for source %d is :", source);
    while (first != NULL)
    {
      printf("%d ", first->info);
      first = first->link;
    }
  }
  return 0;
}
```

5 Output Screenshots

```
Enter the number of vertices
5
Enter the adjacency matrix
0 0 0 1 0
0 0 0 0 1
1 1 0 1 1
0 0 0 0 0
0 0 0 0 0
The adjacency matrix is
0 0 0 1 0
0 0 0 0 1
1 1 0 1 1
0 0 0 0 0
0 0 0 0 0

The BFS traversal for source 0 is :3
The BFS traversal for source 1 is :4
The BFS traversal for source 2 is :0 1 3 4
The BFS traversal for source 3 is :
The BFS traversal for source 4 is :
```

Figure 1: Output for Test Case