Department of
Computer Science and Engineering

Title: Implement Bread-First Search Traversal

Data Structures Lab

CSE 106

Green University of Bangladesh

1 Objective(s)

- To understand how to represent a graph using adjacency matrix.
- To understand how Bread-First Search (BFS) works.

2 Problem analysis

Every graph is a set of points referred to as vertices or nodes which are connected using lines called edges. The vertices represent entities in a graph. Edges, on the other hand, express relationships between entities. Hence, while nodes model entities, edges model relationships in a network graph. A graph G with a set of V vertices together with a set of E edges is represented as G= (V, E). Both vertices and edges can have additional attributes that are used to describe the entities and relationships. Figure 1 depicts a simple graph with five nodes and seven edges.

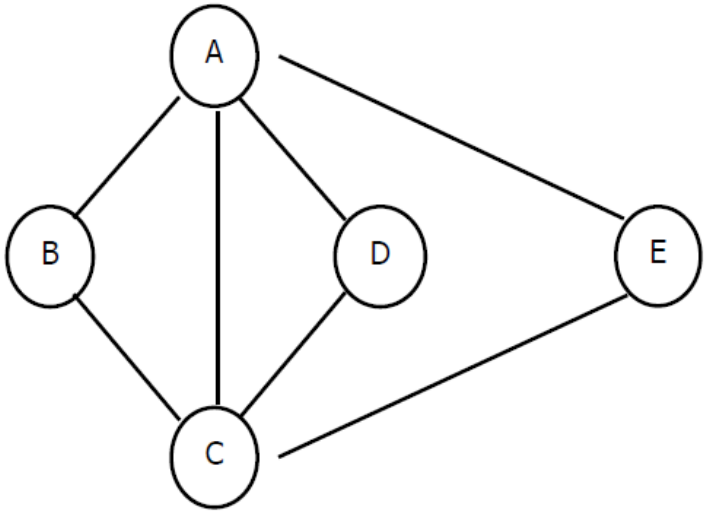


Figure 1: A simple graph

Adjacency Matrix:
Vertices are labelled (or re-labelled) with integers from 0 to V (G) - 1. A two-dimensional array "matrix" with dimensions V (G) * V (G) contains a 1 at matrix [j] [k] if there is an edge from the vertex labelled j to the vertex labelled k, and a 0 otherwise.Table:1 represents the graph of figure:1;

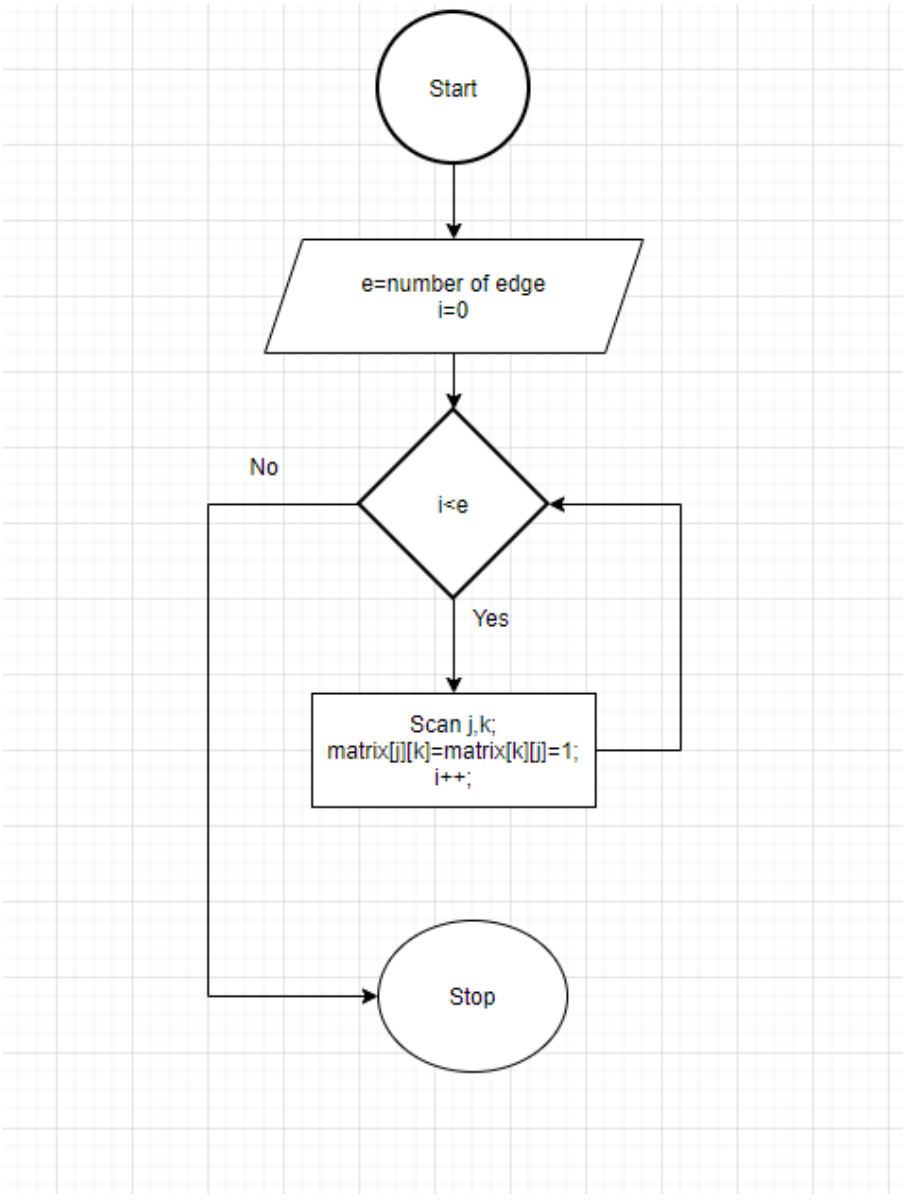
	A	B	C	D	E
A	0	1	1	1	1
B	1	0	1	0	0
C	1	1	0	1	1
D	1	0	1	0	0
E	1	0	1	0	0

Table: 1

3 Algorithm (Adjacency Matrix)

Step 1. Set i=0, e = Number of edges.
Step 2. e (number of edge) < i (Decision). • if no - continue with the step 7.
Step 3. Take the values of edge by giving the adjacency nodes [j], [k] (A, B, C, D, E=0,1,2,3,4).
Step 4. matrix[j][k] =matrix[k][j] =1.
Step 5. Increment i (i++).
Step 6. continue with the step 2.
Step 7. Stop.

4 Flowchart



5 Implementation in C

```
1
2  #include <stdio.h>
3
4  int main()
```

```
{
    int edge,node,a,b,i,j;
    printf("Enter Number of Vertices: ");
    scanf("%d",&node);
    int matrix [node][node];
    for(i=0;i<node;i++)
    {
        for(j=0;j<node;j++)
        {
           matrix[i][j]=0;

        }

    }
    printf("Enter Number of Edges: ");
    scanf("%d",&edge);
    printf("Enter Your Edges: ");
    for(i=0;i<edge;i++)
    {
        scanf("%d%d",&a,&b);
        matrix[a][b]=1;
        matrix[b][a]=1;
    }

    printf("Here is your graph representation :\n");
    for(i=0;i<node;i++)
    {
        for(j=0;j<node;j++)
        {
            printf("%d ",matrix[i][j]);

        }
        printf("\n");

    }
    return 0;
}
```

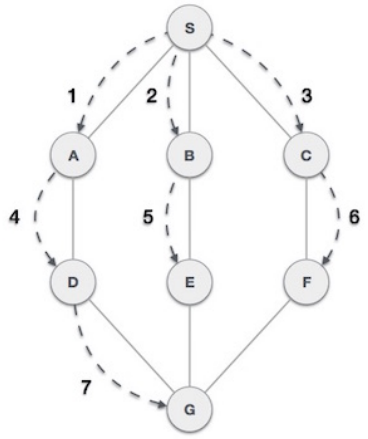
6 Sample Input/Output (Compilation, Debugging & Testing)

Input:
Enter Number of Vertices-
5
Enter Number of Edges-
7
Enter The Edges:
0 1
0 2
0 3
0 4
1 2
2 3
2 4
Output:
0 1 1 1 1
1 0 1 0 0
1 1 0 1 1

1 0 1 0 0
1 0 1 0 0

7 Bread First Search

Breadth First Search (BFS) algorithm traverses a graph in a breadth Ward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.



As in the figure given above, from source S BFS algorithm traverses its child A, B, C first and will put in a queue. In the next phases child of A, B and C nodes which are D, E and F respectively will be visited and inserted in the queue. Finally G node can be visited from node D.

8 Algorithm of BFS

A standard BFS implementation puts each vertex of the graph into one of two categories:
1. Visited
2. Not Visited
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

Algorithm 1: Breadth-First Search

```
   Data: graph[][], visited[], level[]
 1 for each vertex u ∈ V − {s} do
 2 |   visited[u] = WHITE
 3 |   level[u] = inf
 4 end
 5 visited[s] = GRAY
 6 level[s] = 0
 7 Q = empty
 8 ENQUEUE(Q, s)
 9 while Q not empty do
10 |   u = DEQUEUE(Q)
11 |   for each v ∈ adj[u] do
12 |   |   if visited[v] == WHITE then
13 |   |   |   visited[v] = GREY
14 |   |   |   level[v] = level[u] + 1
15 |   |   |   Enqueue(Q, v)
16 |   |   end
17 |   end
18 |   visited[u] = BLACK
19 end
```

9 Implementation in C

```c
#include<stdio.h>
void enqueue (int a);
int dequeue ();
int q[20],r,f;


int main()
{
int node,edge,a,b,s,u,v,des,count=0,S[20],x;
int q[20];

    printf("Enter your node number :\n");
    scanf("%d",&node);
    int matrix[node][node];
    int c[node];
    int p[node];
    int d[node];
    int i,j;

     for(i=0;i<node;i++)
    {
        for(j=0;j<node;j++)
        {
           matrix[i][j]=0;

        }

    }

    printf("Enter your edge number :\n");
    scanf("%d",&edge);
    printf("Enter your edges :\n");


    for(i=0;i<edge;i++)
    {
        scanf("%d%d",&a,&b);

        matrix[a][b]=1;
        matrix[b][a]=1;


    }


    printf("Here is your graph representation :\n");

    for(i=0;i<node;i++)
    {
        for(j=0;j<node;j++)
        {
            printf("%d ",matrix[i][j]);

        }
        printf("\n");

```

```
57      }
58
59      printf("Enter your source : \n");
60      scanf("%d",&s);
61      for(u=0;u<node;u++)
62      {
63          c[u]=1;
64          p[u]=-1;
65          d[u]=-1;
66      }
67      c[s]=2;
68      d[s]=0;
69      p[s]=-1;
70      f=0;
71      r=0;
72      enqueue(s);
73      while(f!=r)
74      {
75          u=dequeue();
76          for(v=0;v<node;v++)
77          {
78              if(matrix[u][v]==1 && c[v]==1 )
79              {
80                  c[v]=2;
81                  d[v]=d[u]+1;
82                  p[v]=u;
83                  enqueue(v);
84              }
85          }
86          c[u]=3;
87      }
88
89      printf("Enter your Destination node :\n");
90      scanf("%d",&des);
91      for(i=0;x!=s;i++)
92      {
93          x=des;
94          S[i]=x;
95          des=p[x];
96          count++;
97
98      }
99
100     for(i=count-1;i>=0;i--)
101     {
102         printf("%d ",S[i]);
103
104     }
105
106
107     return 0;
108 }
109
110 void enqueue (int a)
111 {
112     q[r++]=a;
113 }
114
```

```
115 int dequeue ()
116 {
117     int a;
118     a=q[f++];
119     return a;
120 }
```

10 Sample Input/Output (Compilation, Debugging & Testing)

Output
Enter your node number :
4
Enter your edge number :
4
Enter your edges :
0 1
1 2
2 3
1 3
Here is your graph representation :
0 1 0 0
1 0 1 1
0 1 0 1
0 1 1 0
Enter your source :
0
Enter your Destination node :
2
0 1 2

11 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a program to find the parent of each node using BFS.

12 Lab Exercise (Submit as a report)

- Write a program to detect the cycle in a graph using BFS.

13 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.