Department of
Computer Science and Engineering

Title: Implement Depth-First Search Traversal

Algorithms Lab

CSE 206

Green University of Bangladesh

1 Objective(s)

- To understand how to represent a graph using adjacency list.
- To understand how Depth-First Search (DFS) works.

2 Problem analysis

Two of the most popular tree traversal algorithms are breadth-first search (BFS) and depth-first search (DFS). Both methods visit all vertices and edges of a graph; however, they are different in the way in which they perform the traversal. This difference determines which of the two algorithms is better suited for a specific purpose.

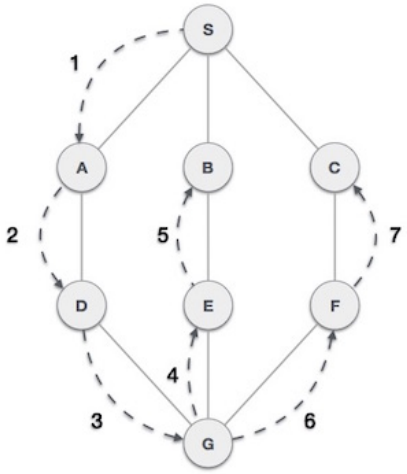


Figure 1: A simple graph

Adjacency List:
Vertices are labelled (or re-labelled) from 0 to V(G)- 1. Corresponding to each vertex is a list (either an array or linked list) of its neighbours. Table: 1 represents the adjacency list of figure1.

A to	D, S
B to	E, S
C to	F, S
D to	A, G
E to	B, G
F to	C, G
G to	D, E, F
S to	A, B, C

Table:1

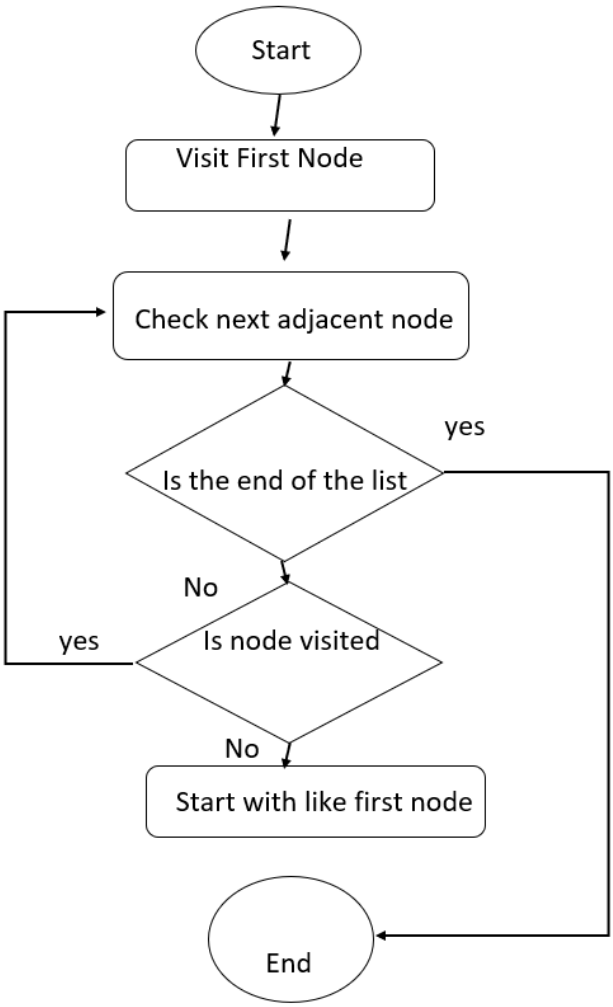
DFS:
Depth-first Search or Depth-first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph. Figure 1 shows the DFS graph traversal.As in the example given above, the DFS algorithm traverses from S to A to D to G to E to B first, then to F, and lastly to C. It employs the following rules.

3 Algorithm (DFS)

A standard DFS 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.
The DFS algorithm works as follows:
Step 1. Start by putting any one of the graph's vertices on top of a stack.
Step 2. Take the top item of the stack and add it to the visited list.
Step 3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
Step 4. Keep repeating steps 2 and 3 until the stack is empty.

4 Flowchart



5 Implementation in Java

```
1  import java.util.*;
2  public class DFS {
3    static char[] c={'A','B','C','D','E','F','G','S'};
4    static int e[]={2,2,2,2,2,2,3,3};
5    static int list[][]={{3,7},{4,7},{5,7},{0,6},{1,6},{2,6},{3,4,5},{0,1,2}};//
       adjacency list of graph figure 1.
6    static int[] checked=new int[20];
7    static int[] stk=new int[20];
```

```
  static int top=0;

    public static void main(String[] args) {
        int i,n,f=0;//f is a flag that tells when to pop
push(7);//inserting the first node from where the traversal starts.
while(top!=0){//loop will execute till the stack is not empty.
n=stk[top-1];
    for(i=0;i<e[n];i++){
        f=0;
       if(notChecked(list[n][i])==1){
              push(list[n][i]);
              f=1;
             break;
       }

      }
    if(f==0)
     pop();
    }
    }

    static int notChecked(int n){// this method checks the node visited or not
if(checked[n]==1)
    return 0;
return 1;
}


    static int pop(){// this method is used to pop a node from stack
//System.out.print(c[stk[top-1]]+" ");//print popping sequence
top--;
return stk[top];
}

  static void push(int n){//this method is used to push a node from stack
checked[n]=1;
System.out.print(c[n]+" ");
stk[top]=n;
top++;
}

}
```

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

Output:
S A D G E B F C

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

1. Write a program to perform DFS traversal on a dynamic graph from user.
2. Write a program to find the path from source to destination using DFS.

8 Lab Exercise (Submit as a report)

- Write a program to perform topological search using BFS.

9 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.