**Practical 12**

**Name: Shantanu Sethi**

**Roll no. 163**

**Aim: To implement different graph traversal technique**

**Objectives:**

1. To implement Depth First Search (DFS)
2. To implement Breadth First Search (BFS)

**Theory:**

**DFS:** Beginning at a starting node depth-first search (DFS) recursively visits the first yet undiscovered direct neighbor of every reached node before continuing in the same manner with the remaining neighbors at this node

**Algorithm:**

Algorithm DFS (G <graph>)

Pre: G is graph structure;Post: Graph traversed in Depth First

1. for (all v in G)
   1. visited[v] = false
2. for (all v in G)
   1. if (not visited[v])
      1. traverse(v)

Algo traverse(v <vertex of graph>)

1. visited[v] = true
2. process v
3. for (all w adjacent to v )
   1. if (not visited[w])
      1. traverse(w)

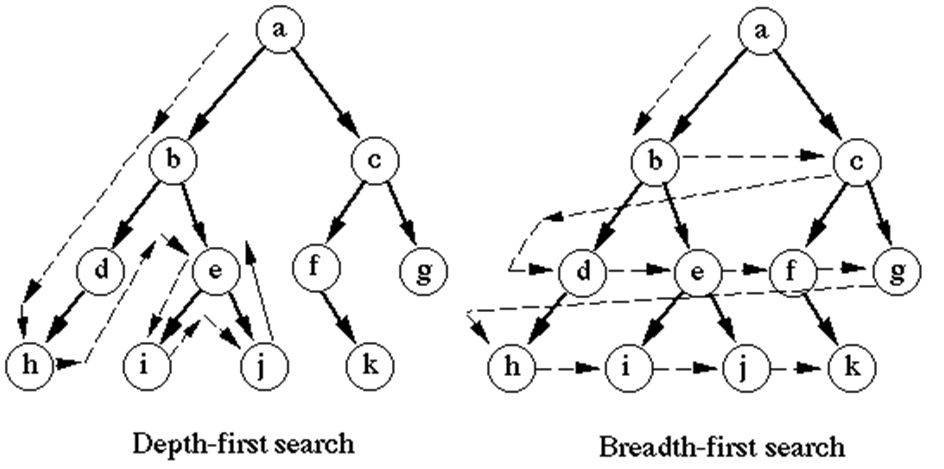
• **BFS:** Beginning at a starting node breadth-first search visits all of its direct neighbors (having distance 1) before it in turn uses these already visited nodes as new starting nodes to continue with their direct neighbors. That way, it visits all reachable nodes at distance k to the original starting node before those at distance k+1.

Algorithm BFS (G <graph>)

Pre: G is graph structure;Post: Graph traversed in Breadth First

1. for (all v in G)
   1. visited[v] = false
2. for (all v in G)
   1. if (not visited[v])
      1. add(v, queue)
      2. while (queue not empty)
         1. v = delete (queue)
         2. if (not visited[v])
            1. visited[v]=true
            2. process v
         3. for (all w adjacent to v)
            1. if (not visited[w])

1. add(w, queue)



**Program:**

**BFS:**

package mod2;

import java.util.Scanner;

public class BFS {

int[] reach = new int[20];

int[][] a = new int[20][20];

int[] q = new int[20];

int n, f, r, index;

BFS() {

f = r = 0;

index = 1;

}

void get() {

Scanner sc = new Scanner(System.in) ;

System.out.println("Enter the number of vertices:");

n = sc.nextInt();

System.out.println("Enter the adjacency matrix:");

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

a[i][j] = sc.nextInt();

}

reach[i] = 0;

}

}

void bfs() {

reach[1] = 1;

f++;

r++;

q[r] = index;

System.out.println("BFS is:");

while (f <= r) {

index = q[f];

f++;

System.out.print(index + "\t");

for (int j = 1; j <= n; j++) {

if (a[index][j] == 1 && reach[j] == 0) {

reach[j] = 1;

r++;

q[r] = j;

}

}

}

}

public static void main(String[] args) {

System.out.println("Abhinav Singh-184");

BFS b = new BFS();

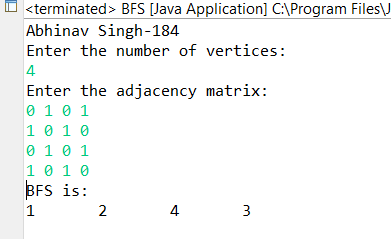
b.get();

b.bfs();

}

}

**OutPut:**

****

**DFS:**

package mod2;

import java.util.Scanner;

public class DFS {

Scanner sc = new Scanner(System.in);

int[][] a = new int[20][20];

int[] visited = new int[20];

int n;

void dfstraverse(int v) {

visited[v] = 1;

System.out.print((v + 1) + "\t");

for (int k = 0; k < n; k++) {

if (a[v][k] == 1 && visited[k] == 0) {

dfstraverse(k);

}

}

}

void get() {

System.out.println("Enter the number of nodes : ");

n = sc.nextInt();

for (int i = 0; i < n; i++) {

visited[i] = 0;

}

System.out.println("Enter the adjacency matrix:");

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

a[i][j] = sc.nextInt();

}

}

System.out.println("DFS traversal starting from node 1:");

dfstraverse(0);

}

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("Abhinav Singh-184");

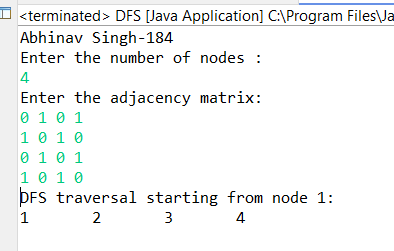
DFS d = new DFS();

d.get();

}

}

**OutPut:**

****

**Conclusion**: Successfully implemented graph traversal; techniques DFS using stack data structure and BFS using queue data structure