DAA Lab - Session 8 - DFS and BFS

Decrease-and-Conquer: Implementation of DFS and BFS algorithms

**Problem Definition:**

Find the number of components in the given undirected graph using **DFS and then BFS algorithms.**

**Input:** The input begins with the number t of test cases in a single line (t <= 50). Each test case begins with the number n of the order of the adjacency matrix of the undirected graph (n <= 100) followed by the adjacency matrix. An adjacency matrix is represented in n lines having n integers (0s or 1s) separated by a space in each line.

**Output:** For every test case print the number of the components the graph has in a new line. Print the total time taken in milliseconds.

**Sample Input:**

7

1

1

2

0 0

0 0

2

0 1

1 0

2

1 1

1 1

3

0 0 0

0 0 0

0 0 0

3

1 1 1

1 0 0

1 0 0

3

0 1 0

1 0 0

0 0 0

**Sample Output:**

1

2

1

1

3

1

1

2

0.000000 sec.

**Algorithms:**

Algorithm DFS\_MAIN( G(V, E) )

Mark each vertex in v with 0

for each vertex v in V

if (v is marked with 0)

dfs\_recurse(v)

Procedure dfs\_recurse(v)

Mark v with 1

for each vertex w in V adjacent to v

if (w is marked with 0)

dfs\_recurse(w)

Algorithm BFS\_main( G(V, E) )

Mark each vertex in v with 0

for each vertex v in V

if(v is marked with 0)

bfs\_node(v)

Procedure bfs\_node(v)

Mark v with 1 and insert v into the Queue

while the Queue is not empty

v ← remove a vertex from the Queue

for each vertex w in V adjacent to v

if(w is marked with 0)

Mark w with 1

Add w to the Queue

**Practice-Problems:** Simulate a maze search using DFS technique. Represent a 2-dimentional maze by grid of nodes, where a node in the grid can potentially connect to four other nodes in the grid using pointers. If the maze has an opening for the particular adjacent node, the respective pointer in the current node points to the adjacent node, otherwise it will be a null pointer. One can use a two-dimensional array of “struct”s in C for a node with four pointers in each struct.