Data Structure Lab Program - 2016

Lab exercise – 10

Note: All programs must be written in C following coding rules as mentioned on the course website. Vertex number starts from 1 for the given input example. You can use any label for vertex.

1. Print discovered nodes by BFS for an undirected, unweighted graph G with *n* vertices and source *s*.
   Input: (n, s, G)
   ```
   5
   2
   0 1 1 0 0
   1 0 0 1 0
   1 0 0 1 1
   0 1 1 0 1
   0 0 1 1 0
   ```
   Output:
   ```
   2 1 4 3 5
   ```

2. Print discovered nodes by DFS for an undirected, unweighted graph G with *n* vertices and source *s*.
   Input: (n, s, G)
   ```
   5
   2
   0 1 1 0 0
   1 0 0 1 0
   1 0 0 1 1
   0 1 1 0 1
   0 0 1 1 0
   ```
   Output:
   ```
   2 1 3 5 4
   ```

3. For an undirected, unweighted graph G with *n* vertices if two vertices v_i, v_j are connected.
   Input: (n, Vi, Vj, G)
   ```
   5
   2 4
   0 1 1 0 0
   1 0 1 0 0
   1 1 0 0 0
   0 0 0 0 1
   0 0 0 1 0
   ```
   Output:
   ```
   No
   ```

4. Given an undirected, unweighted graph G with *n* vertices, print all connected pairs.
   Input: (n , G)
   ```
   5
   0 1 1 0 0
   1 0 1 0 0
   1 1 0 0 0
   0 0 0 0 1
   0 0 0 1 0
   ```

Output:

```
1 2
1 3
2 3
4 5
```

5. Print the number of connected components (sub-graphs) for an undirected unweighted graph G with *n* vertices. Vertices from different connected component have no path connecting them, while any two vertices from the same are connected by at least one path.

   Input: (n, G)

   ```
   5
   0 1 1 0 0
   1 0 1 0 0
   1 1 0 0 0
   0 0 0 0 1
   0 0 0 1 0
   ```

   Output:

   2

6. The diameter of a graph G = (V, E) is defined as the largest of the shortest path distances of the tree. Given an undirected unweighted graph, find its diameter.

   Input: (n, G)

   ```
   5
   0 1 1 0 0
   1 0 1 1 0
   1 1 0 1 0
   0 1 1 0 1
   0 0 0 1 0
   ```

   Output:

   ```
   3
   ```

7. Let G = (V, E) be a connected, undirected graph. An articulation point of G is a vertex whose removal disconnects G (ie braks it into two or more connected components). Find all the articulation points.

   Input: (n, G)

   ```
   5
   0 1 1 0 0
   1 0 1 0 0
   1 1 0 1 1
   0 0 1 0 1
   0 0 1 1 0
   ```

   Output:

   ```
   3
   ```