Exercise Sheet 3

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>∑</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

You are allowed to work in groups of two. Only one submission per pair is necessary, but please indicate who you are working together with. Submit your results as a zip-file to jpreusse@uni-koblenz.de until 14.05.2012, 11:59 pm.

1 Breadth-First Search  (5 points)

Write a function that performs a BFS on an undirected network \( G = (V, E) \) with \( |V| = n \) from a particular start node \( s \) and returns a node vector \( v \in \mathbb{N}^n \) with \( v(i) \) defined as distance in the BFS tree from the start node to node \( i \). The signature of your function should look like this:

```matlab
function bfsVector = BFS(fileName, startNode)
```

and the function should be called in the command line e.g. with

```matlab
bfsVector = BFS('Undirected1.txt', 1)
```

where '1' is the nodeID of the first node in the node list and 'Undirected1.txt' is a sample network data that contains a sparse adjacency matrix of an undirected network:

```plaintext
% undirected network 'Undirected1.txt'
1,2
1,4
2,3
2,4
3,4
3,6
4,5
4,7
4,9
5,6
```

Thus, above’s function call should return

```matlab
bfsVector = (0,1,2,1,2,3,2,Inf,2)
```

where '0' indicates the start node, '1' indicates that the node is at the first level of the BFS tree, '2' at the second level of the BFS tree, and 'Inf' indicates that the node is not connected to the start node.

Instead of implementing the ‘standard’ BFS you should implement the following idea:

Let \( A \) be the adjacency matrix of size \( n \times n \) of the Network \( G \). Now, instead of using \( A \) for this algorithm you should use the modified matrix \( \tilde{A} := A + E \), where \( E \) is the unit matrix of size of \( n \times n \). Assume node \( s \in \{1, \ldots, n\} \) is the start node of the BFS. We define the vector \( v_0 = (0, \ldots, 0, 1, 0, \ldots, 0)^T \in \mathbb{R}^n \) with entry 1 at position \( s \). Now we define \( v_1 \) by

\[
v_1 = \tilde{A} \cdot v_0 \in \{0, 1\}^n.
\]
Make your self clear that $v_1(i) = 1$ means that there is an edge between node $i$ and node $s$, with $i = 1, \ldots, n$ with $i \neq s$. In the next step we define vector $v_2$ by

$$v_2(i) = \begin{cases} 1, & \text{if } \tilde{v}_2(i) > 0, \\ 0, & \text{if } \tilde{v}_2(i) = 0 \end{cases}$$

where

$$\tilde{v}_2 = \tilde{A} \cdot v_1 \in \mathbb{N}^n_0.$$ 

Make yourself clear that for every node $i$ with $v_2(i) = 1$ and $v_1(i) = 0$ there exists at least one edge to a node $j$ with $v_1(j) = 1$. This algorithm can be generalized to $v_c$ in iteration $c$ as

$$v_c(i) = \begin{cases} 1, & \text{if node } i \text{ can be reached in at most } c \text{ steps from } s, \\ 0, & \text{else.} \end{cases}$$

This implementation is rewarded with 3 points. Further two points can be obtained by answering the following two questions.

- Explain why $\tilde{A} = A + E$ should be used! (Hint: Consider a connected graph with two nodes) (1 point)
- What is the stopping criterion for this algorithm and why? (1 point)

2 Connected Network (2 points)

Write a function that returns whether a given undirected network is connected, i.e. that there is a path between every node pair in the network. The signature of your function should look like this:

```matlab
function isConnected = isConnected(fileName)
```

The function should return '0' if the network is not connected and '1' if it is. You can test your function on the hamster network from the previous exercise sheet.

3 Node Eccentricity (3 points)

3.1 Eccentricity (1 points)

Write a function that returns the eccentricity for a given nodeID. Recall that the eccentricity of a node is defined as the longest shortest pairwise distance from the node to any other nodes in the network. We define the eccentricity as $\infty$ if the network is not connected. The signature of your function should look like this:

```matlab
function eccentricity = eccentricityOfNode(fileName, nodeId)
```

You can test your function on the double house of the Nicolaus network from the previous exercise sheet.
3.2 Radius and Diameter (2 point)

Write a function that returns the radius and diameter of an undirected network. (1 point)

Radius and diameter of a network are defined as the smallest or respectively biggest node eccentricity in the network. The signature of your function should look like this:

```matlab
function [radius, diameter] = radiusAndDiameter(fileName)
```

Hence, calling this function for 'Undirected.txt' should return:

```matlab
radius = Inf
diameter = Inf
```

Does this definition of radius and diameter make sense for real networks? In theory, how could it be changed, so that it makes sense as a network characteristic? (1 point)