Bioinformatics III

First Assignment

Name1 (MatrNr1)
Name2 (MatrNr2)

April 15, 2021

Exercise 1.1: The random network

(a) Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Listing 1 shows source code.

(b) Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Exercise 1.2: Some other exercise

(a) Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

(b) Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Listing 1: Example listing of source code.

```
class Node:
    def __init__(self, identifier):
        """
        Creates a Node-object with the given identifier.
        :param identifier: node ID
        """
        self.identifier = identifier
        # contains the identifiers of other nodes connected to this node
        self.neighbour_nodes = set()

    def __str__(self):
        """
        :return: string representation of the node identifier
```

```
14          """
15          return '{0}: {1}'.format(self.identifier, sorted(self.neighbour_nodes))
16
17      def __eq__(self, node):
18          """
19          :param node: Node-object
20          :return: True if the other node has the same identifier, False otherwise
21          """
22          # TODO
23                  raise NotImplementedError
24
25      def has_edge_to(self, node):
26          """
27          :param node: Node-object
28          :return: True if this node has an edge to the other node, False otherwise
29          """
30          # TODO
31                  raise NotImplementedError
32
33      def add_edge(self, node):
34          """
35          Adds an edge to the other node by adding it to the neighbour-nodes.
36          :param node: Node-object
37          """
38          # TODO
39                  raise NotImplementedError
40
41      def remove_edge(self, node):
42          """
43          Removes the edge to the other node, if that edge exists, by removing the
                ↪ other node from the neighbour nodes.
44          :param node: Node-object
45          """
46          # TODO
47                  raise NotImplementedError
48
49      def degree(self):
50          """
51          :return: the degree of this node (= number of neighbouring nodes)
52          """
53          # TODO
54                  raise NotImplementedError
```