Bioinformatics III

First Assignment

Alexander Flohr (MatrNr1)
Andrea Kupitz (2550260)

April 16, 2018

Exercise 1.1: The random network

(a) Listing **??** shows source code.

Listing 1: Example Listing of source code

```
class Node:
    def __init__(self, identifier):
        """
        Sets node id and initialize empty node list that references its connected nodes
        """

    def hasLinkTo(self, node):
        """
        Returns True if this node is connected to node asked for,
        False otherwise
        """

    def addLinkTo(self, node):
        """
        Adds link from this node to parameter node (only if there is no link connection already
        does not automatically care for a link from parameter node to this node
        """

    def degree(self):
        """
        Returns degree of this node
        """

    def __str__(self):
        """
        Returns id of node as string
        """
```

(b) Listing **??** shows source code.

Listing 2: Example Listing of source code

```
from Node import Node

class AbstractNetwork:
    """Abstract network definition, can not be instantiated"""

    def __init__(self, amount_nodes, amount_links):
        """
        Creates empty nodelist and call createNetwork of the extending class
        """
        self.nodes = {}
        self.__createNetwork__(amount_nodes, amount_links)

    def __createNetwork__(self, amount_nodes, amount_links):
```

```
        """
        Method overwritten by subclasses, nothing to do here
        """
        raise NotImplementedError

    def appendNode(self, node):
        """
        Appends node to network
        """

    def maxDegree(self):
        """
        Returns the maximum degree in this network
        """

    def size(self):
        """
        Returns network size (here: number of nodes)
        """

    def __str__(self):
        '''
        Any string-representation of the network (something simply is enough)
        '''

    def getNode(self, identifier):
        """
        Returns node according to key
        """
```

(c) Listing **??** shows source code.

Listing 3: Example Listing of source code

```
from AbstractNetwork import AbstractNetwork
from Node import Node
import random # you will need it :-)

class RandomNetwork(AbstractNetwork):
    """Random network implementation of AbstractNetwork"""

    def __createNetwork__(self, amount_nodes, amount_links): # remaining methods are taken from
        """
        Creates a random network
        1. Build a list of n nodes
        2. For i=#links steps, add a connection between for two randomly chosen nodes that are
        """
        random.seed()
```

Exercise 1.2: Degree Distribution

(a) Listing **??** shows source code.

Listing 4: Example Listing of source code

```
class Node:
    def __init__(self, identifier):
        """
        Sets node id and initialize empty node list that references its connected nodes
        """

    def hasLinkTo(self, node):
        """
        Returns True if this node is connected to node asked for,
        False otherwise
        """
```

```
    def addLinkTo(self, node):
        """
        Adds link from this node to parameter node (only if there is no link connection already
        does not automatically care for a link from parameter node to this node
        """

    def degree(self):
        """
        Returns degree of this node
        """

    def __str__(self):
        """
        Returns id of node as string
        """
```

(b) Listing **??** shows source code.

Listing 5: Example Listing of source code

```
class Node:
    def __init__(self, identifier):
        """
        Sets node id and initialize empty node list that references its connected nodes
        """

    def hasLinkTo(self, node):
        """
        Returns True if this node is connected to node asked for,
        False otherwise
        """

    def addLinkTo(self, node):
        """
        Adds link from this node to parameter node (only if there is no link connection already
        does not automatically care for a link from parameter node to this node
        """

    def degree(self):
        """
        Returns degree of this node
        """

    def __str__(self):
        """
        Returns id of node as string
        """
```

(c) Listing **??** shows source code.

Listing 6: Example Listing of source code

```
class Node:
    def __init__(self, identifier):
        """
        Sets node id and initialize empty node list that references its connected nodes
        """

    def hasLinkTo(self, node):
        """
        Returns True if this node is connected to node asked for,
        False otherwise
        """

    def addLinkTo(self, node):
        """
        Adds link from this node to parameter node (only if there is no link connection already
```

```
15          does not automatically care for a link from parameter node to this node
            """

        def degree(self):
            """
20          Returns degree of this node
            """

        def __str__(self):
            """
25          Returns id of node as string
            """
```