Assignment 1

TPK4186

Torstein Heltne Hovde
Lars Magnus Johnsen
Simen Eger Heggelund

Intro:
In this assignment we are exploring Markov chains. We have here designed a program that manages discrete-time Markov chains, or DTMC for short. A DTMC is a sequence of random variables, which we define as a stochastic process. In this stochastic process the value of the next state depends only on the value of the current state and not the variables in the past. The probabilities of transitions is given through a probability distribution that defines the likeliness of transition between states, in our assignment between the calm, moderate and rough state.

We have made a runTasks() function that will run the whole program and all its associated functions.

Task 1:
We started by creating a DTMC class with the parameters: [name, states, transitions]. For each parameter we created getters, and setters functions.

Task 2:
A probability distribution class was created with the parameters: [name, initial_state, states_probabilities]. Also for this class, get and set functions were created for each parameter.

Task 3:
We created the class “Manage”. This manages the dtmc´s and probability distributions. A more suitable name would possibly be MarkovChain, but since it only contains management functions we thought it was appropriate with the name “Manage”.

Task 4:
In this task we have created test-data, or a benchmark of DTMC and probability distributions according to the format in the exercise text. The data given as input is used to test our program.

Task 5:
In this task we implemented a function writeToFile(filename, dtmc, probability_dist) that prints a list of DTMC and a list of probability distributions into a file by using the python open command. We have used try and except to catch if something went wrong when writing the file. States and initial_state are taken from the dtmc and probability_dist objects that are given as parameters.

Task 6:

Here we implemented a Token class that encodes tokens, with associated management functions(getters and setters). The class is used to represent a token, which is the smallest individual unit in a python program. In the code the attributes and management functions are well documented.

Task 7:
We implemented the function readFileCreateTokens(filename) which reads a text file and returns a list of tokens that consists in the file. A triple for loop is used to keep track of row number, column number and also be able to classify each token with its corresponding type. All different types of keywords, operators, delimiters, separators and identifiers are also manually defined in the function. In the aftermath we could have implemented an approach using regex, which would have been much easier to read, and implement since it provides functionality directly for what was needed in this task.

Task 8:
Created the help functions parseTokens2DTMC and parseTokens2ProbabilityDistribution to assist a main parser function. This function is supposed to create DTMC's and probability distributions by parsing a list of tokens. This parser function was not completed.

Task 9:
For this task we created the method "checkMarkovChain". This method checks if the DTMC and the sea condition probability input are correct. The method returns a message if the inputs are correct.

Task 10:
Here we created the method "calcProbDist". The method return a text that states how many iterations that are needed for the probability distribution to converge as well as the result as a list. The first element in the list is the probability for the sea condition to be "CALM", the second "MODERATE" and the third "ROUGH".

Task 11:
Here we created the method "calcProbDistContinous". As opposed to method "calcProbDist" in task 10 the user can now choose the number of iteration. The method returns the probability distribution after every step as a list.

Task 12:
For this task we created the method "timeseries". The user can choose the number of iterations he/she wants as well as the initial state, and the method generates the time series and returns it. The next state is based on the probability found in seaCondition.txt.

Task 13:
We created a function createDTMC that given a time series, creates a DTMC by using the frequency of transitions in the time series from one state to another. To get the ratios correct it counts each appearance of the start state through a dictionary. It then takes the length of the sequence and divides it by the count of appearances of that specific state.

Task 14:

An experimental study was performed where we determined the length of a timeseries to get an offset of <= 1%. We could have automated this process by creating a while loop that iterated as long as the offset is >1%, but we wanted to perform a more time-efficient and calculation-efficient method, so we manually tested with different increases of iterations until the result was sufficient. This could also have been implemented through a for loop with large increments for each iteration.