3 Random Surfer - Markov Chain Monte Carlo

This exercise largely corresponds to the "Random Surfer" slides from the lectures.

Your task is to complement the skeleton code from http://lec.inf.ethz.ch/baug/informatik2/2015/ex/ex03/skeleton/Main.java, eventually implementing the random surfer through both simulation and iterative computation. Note that the code compiles even without filling any gaps. You can test individual parts separately and submit partial solutions to the judge to get the individual parts evaluated. Note that there may be dependencies: please process the tasks in the provided order.

We provide sample input data to the test program `Main` at the end of the source code. Use them in order to test your code locally before submitting to the judge. The judge uses different input data for evaluating your code.

3.1 Matrix Output

Implement method

```
// pre: non-empty matrix m
// post: matrix printout on System.out
public static void printMatrix(double[][] m)
```

for the output of a matrix. The output should provide the number of rows r and columns c as two whitespace separated integers followed by $r \cdot c$ floating point matrix entries. Insert a new-line character for each finished row. Example output for a 2×3 matrix:

```
2 3
0.10000 0.20000 0.30000
1.10000 1.20000 1.30000
```

Use the following instruction for output of each matrix entry of type `double`:

```
System.out.printf("%7.5f ", e); // replace "e" by matrix entry
```

Validate your solution here: https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB1505

3.2 Matrix Input

Implement method

```
// pre: matrix data provided via scanner
// post: returns matrix data
public static double[][] readMatrix(java.util.Scanner scanner)
```

in order to read a matrix from input via scanner. The format of the input data corresponds to matrix output from above: the number r of rows followed by the number c of columns, both integers, followed by a sequence of $r \cdot c$ doubles. Numbers are separated by whitespaces.

Validate your solution here: https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB1506

3.3 Vector-Matrix-Multiplication

Implement the method

```
// pre: non-empty input vector of length N, matrix of size N times M
// post: return vector-matrix product v * m in a vector of size M
public static double[] multiplyVectorMatrix(double[] v, double[][] m)
```

computing the product of a vector of length N and a matrix of size $N \times M$, $N, M > 0$. Return a new vector with the result. Do not modify input data.

Validate your solution here: https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB1507

3.4 Checking Convergence

The stationary distribution μ of a Markov chain with transition matrix P fulfils $\mu P = \mu$. In order to check if a vector v is sufficiently close to μ, we check if $vP \approx v$ by comparing vP element-wise with v. Implement method

```
// non-empty vector v and probability matrix m, precision > 0
// returns if v * m is close at v
public static boolean checkConvergence(double[] v, double[][] P, double preci
```

returning if absolute values of all entries of $vP - v$ are smaller than `precision`.

Validate your solution here: https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB1508

3.5 Simulation

Implement method

```
// pre: non-empty probability matrix P
// post: return relative frequencies of visits in each point
//       after MCMC simulation until convergence reached
public static double[] simulate(double[][] P, double precision)
```

simulating the behavior of the random surfer. The random surfer starts in state 0 and makes steps according to the transition matrix P. For simulation steps the already existing method `simulateLine` should be used, just as we did in the lectures. But in contrast to the version provided in class you should not do a fixed number of iterations but loop until either one of the following two conditions is met:

1. The stationary point is sufficiently well approximated (using the convergence check from above)
2. 10000000 iterations have passed.

Validate your solution here: https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB1509

3.6 Direct calculation without simulation

Implement method

```
// pre: non-empty vector v and n times n probability matrix
// post: vector v close at stationary distribution of m
//       returns number of vector-matrix multiplications required
public static int calculateDirect(double[] v, double[][] m,
                                                    double precision)
```

using formula $v_{n+1} = v_n \cdot P$ as presented in the course. As in the previous task you should perform this operation until the convergence criterion is met, or 10000000 iterations have passed.

Note the difference to previous exercises: here it is explicitly required that input parameter v changes its values.

Validate your solution here: https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB1510

Run the main routine of the RandomSurfer routine and observe the different run times of the two approaches. As a little theoretical exercise, compute and compare the number of multiplications required for the different approaches. What is your conclusion?