Національный Технічний Університет України
«Київський Політехнічний Інститут»

Лабораторна работа №4

Виконав:
студент группи IO-02
Варщук Богдан.

Київ, 2011

```java
import java.util.*;

public class Test {
	private static double[] getTablePropabilities(int[] numbers,
			double[] propabilities) {
		double[] ret = new double[numbers[numbers.length - 1]
				* numbers[numbers.length - 1]];
		for (int i = 0; i < numbers.length; i++) {
			for (int j = 0; j < numbers.length; j++) {
				ret[numbers[i] * numbers[j] - 1] += propabilities[i]
						* propabilities[j];
			}
		}
		return ret;
	}

	private static int[] getSequence(int[] numbers, double[] propabilities,
			int size) {
		double[] prop = new double[propabilities.length];
		prop[0] = propabilities[0];
		for (int i = 1; i < propabilities.length; i++) {
			prop[i] = prop[i - 1] + propabilities[i];
		}
		int[] ret = new int[size];
		Random random = new Random();
		double temp;
		int j;
		for (int i = 0; i < size; i++) {
			temp = random.nextDouble();
			j = 0;
			while (temp > prop[j]) {
				j++;
			}
			ret[i] = numbers[j];
			temp = random.nextDouble();
			j = 0;
			while (temp > prop[j]) {
				j++;
			}
			ret[i] *= numbers[j];
		}
		return ret;
	}

	private static int[] getIntervals(int[] sequence, int numberOfIntervals) {
		int[] ret = new int[numberOfIntervals + 1];
		int sizeOfIntervals = sequence.length / numberOfIntervals;
		int j;
		int k;
		for (int i = 1; i < numberOfIntervals; i++) {
			j = sizeOfIntervals * i;
			while ((sequence[j] == sequence[j - 1]) && j < sequence.length) {
				j++;
			}
			k = sizeOfIntervals * i;
			while ((sequence[k] == sequence[k - 1]) && k > 1) {
				k--;
			}
			if (j - sizeOfIntervals * i < sizeOfIntervals * i - k) {
				ret[i] = j;
			} else {
```

```
                if (ret[i - 1] == k) {
                    ret[i] = j;
                } else {
                    ret[i] = k;
                }
            }
        }
        ret[numberOfIntervals] = sequence.length;
        return ret;
    }

    private static double getPropability(int[] sequence,
            double[] tablePropabilities) {
        HashMap<Integer, Double[]> map = new HashMap<Integer, Double[]>();
        Double[] two = { 0.02, 0.04, 0.103, 0.211, 0.446, 0.713, 1.386, 2.41,
                3.22, 4.6, 5.99 };
        map.put(2, two);
        Double[] nine = { 2.09, 2.53, 3.32, 4.17, 5.38, 6.39, 8.34, 10.66,
                12.24, 14.68, 16.92 };
        map.put(9, nine);
        double[] g = { 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9,
                0.95 };
        int numberOfIntervals;
        if (sequence.length <= 20) {
            numberOfIntervals = 3;
        } else {
            numberOfIntervals = 10;
        }
        int[] intervals = getIntervals(sequence, numberOfIntervals);
        int[] sizeOfIntervals = new int[intervals.length - 1];
        for (int i = 0; i < sizeOfIntervals.length; i++) {
            sizeOfIntervals[i] = intervals[i + 1] - intervals[i];
        }
        double[] propobilitiesOfIntervals = new double[intervals.length - 1];
        for (int i = 0; i < sequence[intervals[1]] - 1; i++) {
            propobilitiesOfIntervals[0] += tablePropabilities[i];
        }
        for (int i = 1; i < propobilitiesOfIntervals.length - 1; i++) {
            for (int j = sequence[intervals[i]] - 1; j < sequence[intervals[i + 1]] - 1; j++) {
                propobilitiesOfIntervals[i] += tablePropabilities[j];
            }
        }
        for (int i = sequence[intervals[propobilitiesOfIntervals.length - 1]] - 1; i <
tablePropabilities.length; i++) {
            propobilitiesOfIntervals[propobilitiesOfIntervals.length - 1] += tablePropabilities[i];
        }

        double xi = 0;
        for (int i = 0; i < sizeOfIntervals.length; i++) {
            xi += Math.pow((sizeOfIntervals[i] - sequence.length
                    * propobilitiesOfIntervals[i]), 2)
                    / (propobilitiesOfIntervals[i] * sequence.length);
        }
        Double[] tableXi = map.get(numberOfIntervals - 1);
        int j = 0;
        for (int i = 0; i < 11; i++) {
            j = 0;
            while ((xi > tableXi[j]) && j != 10) {
                j++;
            }
        }
        double ret=1-g[j];
```

```java
		return ret;
	}

	public static void main(String[] args) {
		double border=0.7;
		int size = 1000;
		int[] numbers = { 1, 2, 3, 4, 5 };
		double[] propabilities = { 0.1, 0.2, 0.25, 0.3, 0.15 };
		double[] tablePropabilities = getTablePropabilities(numbers,
				propabilities);
		int[] sequence = getSequence(numbers, propabilities, size);
		Arrays.sort(sequence);
		System.out.println("Propability - "+getPropability(sequence, tablePropabilities));
	}
}
```