Lab 2

Alessandro Artale, Werner Nutt,
Ognjen Savković, Francesco Sportelli

2. Measurements with the Class ArrayUtility and Matrix Multiplication[1]

1. Implementation of Insertion Sort and Bubble Sort

In the lecture, you have seen three sorting algorithms. Implement the two algorithms Insertion Sort and Bubble Sort as methods in your ArrayUtility class.

2. Analysis of the running time of Insertion Sort and Bubble Sort

In this task, you will experiment with the two algorithms Insertion Sort and Bubble Sort, measuring their running times for varying input and analysing the times. For each version, proceed following the guidelines below.
Automate the tests as fas as possible by writing code in Java.

1. Measure the running time for random arrays of size 10^i where $i = 1, 2, \ldots, m$. Start with $i = 1$ and keep increasing the size until the running time of the algorithm exceeds 5 seconds, which gives you the value of m.

2. Let m be the number from Part 1 and let $n = 100$. For each i, where $1 \leq i \leq m$, we want to measure and record the running time for n random arrays of size 10^i. To do so, create a 2-dimensional $m \times n$ array T. Then, for every $i = 1, \ldots, m$, and $j = 1, \ldots n$, generate a random array $A^{(i,j)}$ of size 10^i, measure the running time $t_{i,j}$ of the algorithm for the input $A^{(i,j)}$, and store $t_{i,j}$ in $T[i, j]$. The result of this will be a matrix that looks like the one below

3. For each size 10^i (i.e., each row of the Matrix), compute the average, the variance, and the median values of the corresponding running times.

   - The *average* of the running times in row i is

     $$\bar{t_i} = \frac{\sum_{j=1}^{n} t_{i,j}}{n}.$$

	1	2	$\cdots$	n
10	$t_{1,1}$	$t_{1,2}$	$\cdots$	$t_{1,n}$
10^2	$t_{2,1}$	$t_{2,2}$	$\cdots$	$t_{2,n}$
.	.	.		
.	.	.		
.	.	.		
10^m	$t_{m,1}$	$t_{m,2}$	$\cdots$	$t_{m,n}$

This is also known as the arithmetic mean.

- The *variance* of the running times in row i is

$$v_i = \frac{\sum_{j=1}^{n}(t_{i,j} - \bar{t_i})^2}{n}.$$

The variance indicates how much the values $t_{i,j}$ vary from the average $\bar{t}$.

- The *median* m_i of the running times in row i is the middle number in the sorted list of running times.

For example, the median of the list $\{1, 2, 2, 3, 7, 8, 8\}$ is 3. If the length of the list is even, we define the median as the average of the two middle values.

4. For each algorithm, output both the resulting matrix and the average, variance and median values in a simple structured way.

5. What do you deduce from the results?

Hints: (i) The variance indicates if the running times tend to be grouped around the average time or to be dispersed (many best and worst case scenarios). (ii) In the presence of extreme values, the median is more robust than the average as a measure of the centrality of the data (the "average" performance of the algorithm).

3. Multiplication of matrices

As an additional method of ArrayUtility, implement a method that multiplies two integer matrices A and B and returns a matrix that contains the result of the multiplication. The method should have the signature

```
public static int[][] MatMult(int[][] A, int[][] B).
```

Note that two matrices A and B can only be multiplied if the number of columns of A equals the number of rows of B. Assuming that the dimension of A is $n \times m$ and the dimension of B is $m \times p$, the resulting Matrix $C = A \cdot B$ is of size $n \times p$ and its entries are computed as follows:

$$C[i][j] = \sum_{k=1}^{m} A[i][k] \cdot B[k][j].$$

This is an example of a matrix multiplication:

$$\begin{aligned} & \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \cdot \begin{pmatrix} 6 & -1 \\ 3 & 2 \\ 0 & -3 \end{pmatrix} \\ = & \begin{pmatrix} 1 \cdot 6 + 2 \cdot 3 + 3 \cdot 0 & 1 \cdot (-1) + 2 \cdot 2 + 3 \cdot (-3) \\ 4 \cdot 6 + 5 \cdot 3 + 6 \cdot 0 & 4 \cdot (-1) + 5 \cdot 2 + 6 \cdot (-3) \end{pmatrix} \\ = & \begin{pmatrix} 12 & -6 \\ 39 & -12 \end{pmatrix} \end{aligned}$$