East West University
Department of Computer Science and Engineering

CSE 110: LAB 03 (Handout)
Course Instructor: Dr. Mohammad Rezwanul Huq

Sorting, Searching and Multi-dimensional Arrays

Lab Objective

Familiarize students with selection sort algorithm, binary search and multi-dimensional array manipulations.

Lab Outcomes

After completing this lab successfully, students will be able to:

1. **Understand and Code** selection sort and binary search algorithms.
2. **Solve, write** and **execute** programs related to multi-dimensional arrays in Java.

Psychomotor Learning Levels

This lab involves activities that encompass the following learning levels in psychomotor domain.

Level	Category	Meaning	Keywords
P1	Imitation	Copy action of another; observe and replicate.	Relate, Repeat, Choose, Copy, Follow, Show, Identify, Isolate.
P2	Manipulation	Reproduce activity from instruction or memory	Copy, response, trace, Show, Start, Perform, Execute, Recreate.

Lab Activities

A. Sorting Algorithms

- Sorting is a very useful operation to solve several computational problems. Sorting means to arrange values in a particular order, either ascending or descending.
- Bubble sort is a well known sorting algorithm. You will find the pseudocode of Bubble sort below. Let A be the array and n is the size of the array

```
for i = 0 to n-2
   for j = n-1 downto i+1
      if A[j] < A[j-1]
     exchange A[j] with A[j-1]
```

- Selection sort is another algorithm which is more efficient than the Bubble sort algorithm.
- The working procedure of Selection sort will be discussed in the class. The pseudocode of Selection sort algorithm is given below. Complexity of both Bubble and Selection sort is $O(n^2)$

```
for i = 0 to n-2
   minIndex = i
   minValue = A[i]
   for j = i+1 to n-1
       if minValue > A[j]
           minValue = A[j]
           minIndex = j
   if i ≠ minIndex
      exchange A[i] with A[minIndex]
```

- **Write a Java program that implements both Bubble sort and Selection Sort algorithm and compare their execution time. Use *System.currentTimeMillis()* method to compute execution time. An example is given below.**

```
long startTime = System.currentTimeMillis();
bubbleSort(A);
long endTime = System.currentTimeMillis();
long executionTime = endTime-startTime;
```

B. Searching Algorithm

- Searching algorithm finds a given value in an array.
- Linear search is the most basic searching algorithm that may take upto n iterations where n is the size of the array. Complexity: O(n)
- Pseudocode of Linear Search

  Let A be the array, n be the size of the array and V be the given value

```
for i = 0 to n-1
   if A[i] = V
      return i;
return -1;
```

- Binary search is more efficient than linear search. However, the given array must be sorted. Complexity: O (log n)
- Details of Binary Search will be discussed in the class.
- Pseudocode of Binary Search

  Let A be the array, n be the size of the array and V be the given value

```
low = 0
high = n-1
mid = (low+high)/2
while low <= high
   if A[mid] = V
      return mid;
   else if A[mid] < V
      low = mid + 1
   else
     high = mid - 1
return -1;
```

- **Write a Java program that implements both Linear Search and Binary Search algorithm. Compare their execution time.**

C. Multi-dimensional Arrays

- A variable for two-dimensional arrays can be declared using the syntax:
  `elementType[][] arrayVar`
- A two-dimensional array can be created using the syntax:
  `new elementType [ROW_SIZE][COLUMN_SIZE]`
- Each element in a two-dimensional array is represented using the syntax:
  `arrayVar[rowIndex][columnIndex]`
- Details on multi-dimensional arrays will be discussed in the class.
- **Write a Java program that takes a two-dimensional array of size m×n and prints the following:**
  - **Row-wise sum**
  - **Column-wise sum**
  - **Sum of the main diagonals**
- Extra Topic: Matrix Multiplication
  - Matrix Multiplication is a problem that needs to be solved using Multi-dimensional Arrays.
  - Details in the class.

This lab sheet must be preserved till the end of the semester.

Student ID		**Number of Problems Solved**	
Student Name			
Section			
Date			

Lab 03: Homework Problems

Lab03_Problem01: Suppose the weekly hours for all employees are stored in a two-dimensional array. Each row records an employee's seven-day work hours with seven columns. For example, the following array stores the work hours for eight employees. Write a program that displays employees and their total hours in decreasing order of the total hours.

	Su	M	T	W	Th	F	Sa
Employee 0	2	4	3	4	5	8	8
Employee 1	7	3	4	3	3	4	4
Employee 2	3	3	4	3	3	2	2
Employee 3	9	3	4	7	3	4	1
Employee 4	3	5	4	3	6	3	8
Employee 5	3	4	4	6	3	4	4
Employee 6	3	7	4	8	3	8	4
Employee 7	6	3	5	9	2	7	9

Lab03_Problem02: Write a program that randomly fills in 0s and 1s into a 4-by-4 matrix, prints the matrix, and finds the first row and column with the most 1s. Here is a sample run of the program:

```
0011
0011
1101
1010
The largest row index: 2
The largest column index: 2
```

Lab03_Problem03: Write the following method that returns the location of the largest element in a two-dimensional array.

```
public static int[] locateLargest(double[][] a)
```

The return value is a one-dimensional array that contains two elements. These two elements indicate the row and column indices of the largest element in the two-dimensional array. Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array.
Here is a sample run:

```
Enter the number of rows and columns of the array: 3 4 Enter
Enter the array:
23.5 35 2 10 Enter
4.5 3 45 3.5 Enter
35 44 5.5 9.6 Enter
The location of the largest element is at (1, 2)
```

Lab03_Problem04: Write a method to sort a two-dimensional array using the following header:

public static void sort(**int** m[][])

The method performs a primary sort on rows and a secondary sort on columns.
For example, the following array {{4, 2},{1, 7},{4, 5},{1, 2},{1, 1},{4, 1}}
will be sorted to {{1, 1},{1, 2},{1, 7},{4, 1},{4, 2},{4, 5}}.

Lab03_Problem05: Write a method that solves the following 2 * 2 system of linear equations:

$$a_{00}x + a_{01}y = b_0$$
$$a_{10}x + a_{11}y = b_1$$

$$x = \frac{b_0a_{11} - b_1a_{01}}{a_{00}a_{11} - a_{01}a_{10}} \qquad y = \frac{b_1a_{00} - b_0a_{10}}{a_{00}a_{11} - a_{01}a_{10}}$$

The method header is **public static double**[] linearEquation(**double**[][] a, **double**[] b)

The method returns **null** if $a_{00}a_{11} - a_{01}a_{10}$ is **0**. Write a test program that prompts the user to enter a_{00}, a_{01}, a_{10}, a_{11}, b_0, and b_1, and displays the result. If $a_{00}a_{11} - a_{01}a_{10}$ is **0**, report that "The equation has no solution."

Programming Assignment # 01

Using the program that you have implemented in Lab Activities A (Sorting) and Lab Activities B (Searching), Complete the following table by writing the execution time for different size of arrays.

	Sorting Algorithm (execution time in milliseconds)		Searching Algorithm (execution time in milliseconds)	
Array Size	Bubble Sort	Selection Sort	Linear Search	Binary Search
100000				
200000				
300000				
400000				
500000				
600000				
700000				
800000				
900000				
1000000				

You should also prepare an appropriate bar chart to visualize the obtained result.

You must submit a report (hardcopy) with a cover page that describes the following:

- Your code
- Your Result Table
- Your Chart
- Discussion about the result