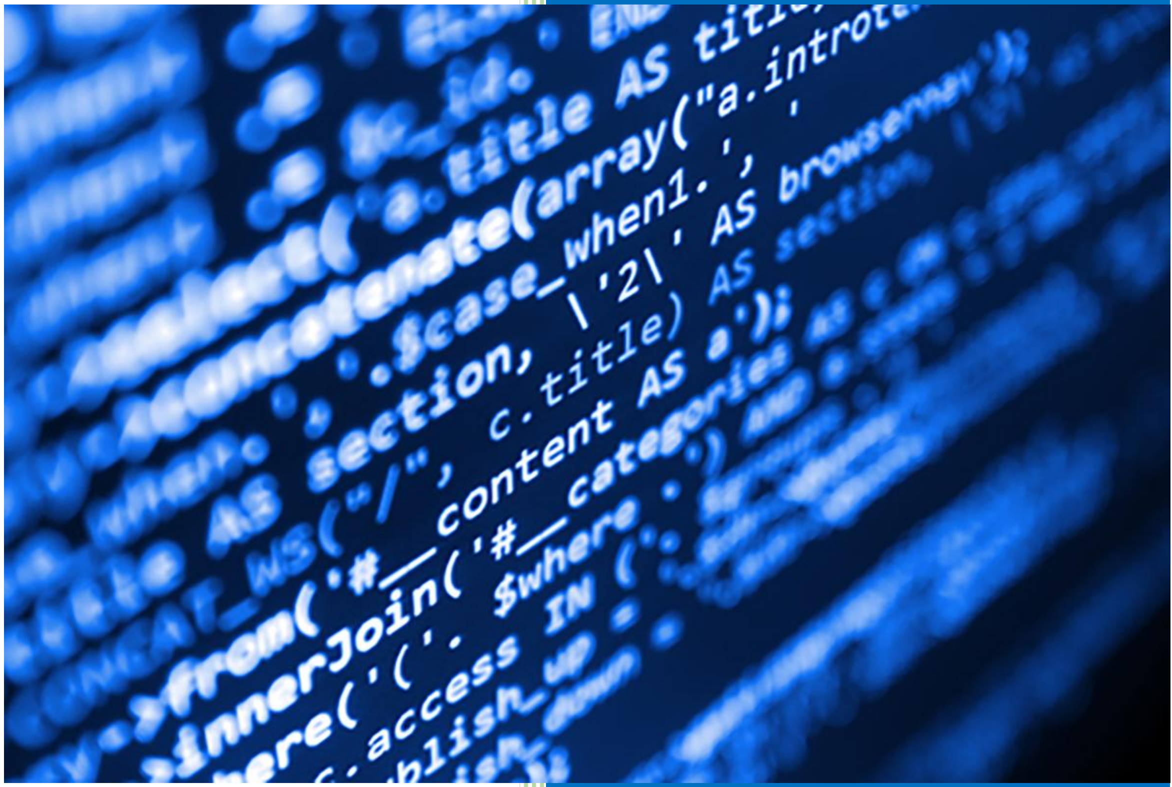
FAKULTI SAINS KOMPUTER DAN MATEMATIK

2025/2026

DATA STRUCTURE & ALGORITHM



Lab 8: Sorting-Part 1

Name: Arisya Syahirah binti Azizi
Matric Number: S76119
Lab: MP2
Date: 15/12/2025

VERSION 1

TABLE OF CONTENTS

INSTRUCTIONS

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Fakulti Teknologi Kejuruteraan Kelautan dan Informatik, Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual.

This laboratory manual is for use by the students of the Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu (UMT) only. It is not permissible to print and distribute this manual without the official authorisation of the author.

Please follow step by step as described in the manual.

TASK 1: Testing Sorting Algorithms

OBJECTIVE

This lab will cover implementation and timing of various sorting algorithms.

TASK DESCRIPTION

In this lab, students need to test the implementation of various algorithms of sorting. There are a few algorithms of sorting such as

- Insertion-sort
- Selection sort
- Heap sort
- Merge-sort
- Quick-sort
- Bucket-sort
- Radix-sort

Besides that, students need to generate the unsorted integers using Math.random method.

ESTIMATED TIME

[90 minutes] – Lab 8 -Sorting Part 1

STEPS:

Sorting is an activity to arrange a sequence of data in a particular order, for example arrange 100 integers in ascending order.

There are a few algorithms that will be tested in this lab.

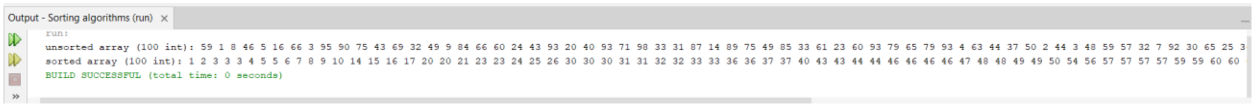
1. **Insertion sort.** Insertion sort is a brute-force sorting algorithm that is based on a simple method that people often use to arrange hands of playing cards: Consider the cards one at a time and insert each into its proper place among those already considered (keeping them sorted). The following code mimics this process in a Java method that sorts strings in an array.

```
public void Selectionsort (int array[])
{
   int a =array.length;
   for (int i=0;i < a-1; i++)
   {
   int minimum_index = i;
     for ( int j = i+1; j< a; j++)
     if (array[j] < array [minimum_index])
         minimum_index = j;

     int temp = array[minimum_index];
     array[minimum_index] = array[i];
     array[i]= temp;
}
```

Code:

```
package sorting.algorithms;
public class InsertionSort {
    void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i){
            int key = arr[i];
            int j = i -1;

            //move elements of arr[0..i-1],
            //that are greater than key,
            //to one position ahead of their current position
            while (j >= 0 && arr[j]>key){
                arr[j + 1] = arr[j];
                j=j-1;
            }
            arr[j + 1] = key;
        }
    }

    //print array of size n
    static void printArray(int arr[]){
        int n = arr.length;
        for (int i = 0; i <n; ++i)
```

```
            System.out.print(arr[i] + " ");
        System.out.println();
    }
}

import java.util.Random;
public class TestSort {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int n = 100;
        int[] arr = new int[n];

        // Create a Random object
        Random random = new Random();
        InsertionSort ob = new InsertionSort();
        // Assign random values to the array
        for (int i = 0; i < n; i++) {

            // Generate a random integer
            // between 0 and 99
            arr[i] = random.nextInt(100);
        }
        System.out.print("unsorted array (100 int): ");
        ob.printArray(arr);
        ob.sort(arr);
        System.out.print("sorted array (100 int): ");
        ob.printArray(arr);
    }

}
```



Output - Sorting algorithms (run)

```
run:
unsorted array (100 int): 59 1 8 46 5 16 66 3 95 90 75 43 69 32 49 9 84 66 60 24 43 93 20 40 93 71 98 33 31 87 14 89 75 49 85 33 61 23 60 93 79 65 79 93 4 63 44 37 50 2 44 3 48 59 57 32 7 92 30 65 25 3
sorted array (100 int): 1 2 3 3 3 4 5 5 6 7 8 9 10 14 15 16 17 20 20 21 23 23 24 25 26 30 30 30 31 31 32 32 33 33 36 36 37 37 40 43 43 44 44 46 46 46 46 47 48 48 49 49 50 54 56 57 57 57 57 59 59 60 60
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
output :
run:
unsorted array (100 int): 59 1 8 46 5 16 66 3 95 90 75 43 69
32 49 9 84 66 60 24 43 93 20 40 93 71 98 33 31 87 14 89 75 49
85 33 61 23 60 93 79 65 79 93 4 63 44 37 50 2 44 3 48 59 57 32
```

```
7 92 30 65 25 36 56 54 84 82 77 62 17 20 71 5 37 6 92 46 30 47
85 95 91 57 90 23 57 30 67 48 46 10 21 36 15 71 57 26 3 31 66
46

sorted array (100 int): 1 2 3 3 3 4 5 5 6 7 8 9 10 14 15 16 17
20 20 21 23 23 24 25 26 30 30 30 31 31 32 32 33 33 36 36 37 37
40 43 43 44 44 46 46 46 46 47 48 48 49 49 50 54 56 57 57 57 57
59 59 60 60 61 62 63 65 65 66 66 66 67 69 71 71 71 75 75 77 79
79 82 84 84 85 85 87 89 90 90 91 92 92 93 93 93 93 95 95 98
BUILD SUCCESSFUL (total time: 0 seconds)
```

TASK 2 GENERATE RANDOM NUMBERS

ESTIMATED TIME

[90 minutes] – Lab 8 -Sorting Part 1

STEPS:

Generate three set of random integers in the range of 1 to 100,000 which have different quantities.

Set 1 – 100 integers

Set 2 – 1000 integers

Set 3 – 10,000 integers

QUESTIONS:

1) Record the run time (in milliseconds) for each sorting techniques using the three sets of integers and record the findings in Table 1. You may use `System.currentTimeMillis()` for identifying the current run time.

Table 1 – The time taken for each of sorting techniques using different set of data

Techniques	Set 1 – 100	Set 2 – 1000	Set 3 – 10,000
Insertion sort			

2) Based on the findings, highlight which sorting technique perform the best, and which perform the worst. Why?

Techniques	Set 1 – 100	Set 2 – 1000	Set 3 – 10,000
Insertion sort	23ms	64ms	256ms
Selection Sort	56ms	234ms	1447ms
Bubble Sort	64ms	238ms	1412ms

- Insertion Sort performed the best overall.
- Bubble Sort and Selection Sort performed similarly for smaller sets, but Selection Sort was slightly slower for larger sets and became the worst performer for the largest set.

- Insertion Sort wins for small or nearly sorted data due to its adaptive nature and low overhead.

- Selection Sort is usually the worst among these three in practice because it never benefits from existing order and always does the same number of comparisons.

- Bubble Sort is typically slow due to many swaps, but can be okay if data is nearly sorted and early exit is used.

INSTRUCTION FOR SUBMISSION

In this lab, you should submit

1. This lab module with necessary answers for the questions had been asked.
2. InsertionSorting.java
3. TestSorting.java

Note: You may also have another java file for generating random numbers. But it is not a must.

Source code:

insertion sort

```
import java.util.Random;


public class InsertionSort {

   public InsertionSort() {

   }


   void sort(int[] arr) {

      int n = arr.length;


      for(int i = 1; i < n; ++i) {

         int key = arr[i];


         int j;

         for(j = i - 1; j >= 0 && arr[j] > key; --j) {

            arr[j + 1] = arr[j];

         }
```

```
            arr[j + 1] = key;
        }

    }

    static void printArray(int[] arr) {
        int n = arr.length;

        for(int i = 0; i < n; ++i) {
            System.out.print(arr[i] + " ");
        }

        System.out.println();
    }

    public static void main(String[] args) {
        long start = System.currentTimeMillis();
        int n = 100;
        int[] arr = new int[n];
        Random random = new Random();
        InsertionSort IS = new InsertionSort();

        for(int i = 0; i < n; ++i) {
            arr[i] = random.nextInt(100);
        }
        System.out.print("unsorted array (100 int): ");
        printArray(arr);
        IS.sort(arr);
        System.out.print("sorted array (100 int): ");
        printArray(arr);
        long end = System.currentTimeMillis();
        System.out.println("Sorting takes " + (end - start) +
"ms");
    }
}
```

Bubble sort

```
import java.util.Random;

public class bubblesort {
        static void bubbleSort(int arr[], int n){
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++) {
            swapped = false;
            for (j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {

                    // Swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            // If no two elements were
            // swapped by inner loop, then break
            if (swapped == false)
                break;
        }
    }

    // Function to print an array
    static void printArray(int arr[], int size){
        int i;
        for (i = 0; i < size; i++)
            System.out.print(arr[i] + " ");
        System.out.println();
    }



    public static void main(String[] args) {
      long start = System.currentTimeMillis();
      int n = 10000;
      int[] arr = new int[n];
      Random random = new Random();
      bubblesort SS = new bubblesort();

      for(int i = 0; i < n; ++i) {
         arr[i] = random.nextInt(10000);
      }

      System.out.print("unsorted array (10000 int): ");
      printArray(arr, n);
      bubbleSort(arr, n);
      System.out.print("sorted array (10000 int): ");
      printArray(arr, n);
      long end = System.currentTimeMillis();
      System.out.println("Sorting takes " + (end - start) +
"ms");
   }
}
```

Selection sort

```
// Source code is decompiled from a .class file using FernFlower
decompiler (from Intellij IDEA).
import java.util.Random;

public class selectionsort {
   public selectionsort() {
   }

   static void selectionSort(int[] arr) {
      int n = arr.length;

      for(int i = 0; i < n - 1; ++i) {
         int min_indx = i;

         int j;
         for(j = i + 1; j < n; ++j) {
            if (arr[j] < arr[min_indx]) {
               min_indx = j;
            }
         }

         j = arr[i];
         arr[i] = arr[min_indx];
         arr[min_indx] = j;
      }

   }

   static void printArray(int[] arr) {
      int[] var1 = arr;
      int var2 = arr.length;
      for(int var3 = 0; var3 < var2; ++var3) {
         int val = var1[var3];
         System.out.print("" + val + " ");
      }

      System.out.println();
   }

   public static void main(String[] args) {
      long start = System.currentTimeMillis();
      int n = 10000;
      int[] arr = new int[n];
      Random random = new Random();
      selectionsort SS = new selectionsort();

      for(int i = 0; i < n; ++i) {
         arr[i] = random.nextInt(10000);
      }

      System.out.print("unsorted array (10000 int): ");
      printArray(arr);
      selectionSort(arr);
      System.out.print("sorted array (10000 int): ");
      printArray(arr);
```

```
        long end = System.currentTimeMillis();

        System.out.println("Sorting takes " + (end - start) +
"ms");

    }

}
```