Kristianstad University Sweden

Kristianstad University
SE-291 88 Kristianstad
Sweden
+46 44 250 30 00
www.hkr.se

DA256G: Algorithms and Data Structures

Daniel Einarson

Seminar 1

Goals

The goal of this seminar is to prepare the students to fulfill the learning outcomes

- give an account of quantitative methods applicable to the analysis of algorithms and data structures (2)
- practically apply algorithm theory and data structures (3)
- find, collect, and critically interpret relevant information to formulate responses to well-formulated problems within algorithm theory (4)
- give an account of and discuss their knowledge with various groups (5).
- advance their reflections, reasonings and arguments within algorithm theory and data structures (6).

Before the Seminar –

The seminar is mandatory. A student who did not upload the report and the programming tasks before the seminar is not allowed to participate in the seminar.

All the tasks must be solved, and they should be solved individually.

It is not sufficient to only solve the tasks; the student must also be able to explain the source code, explain how each algorithm and method works, discuss a method of evaluation the algorithm, as well as present the results in a scientific way in a written report, otherwise the seminar will not be approved.

Note that hints of how some of the different algorithms work or how to implement them can be found in the course literature [1]. Make sure that for all other solutions that you find on the Internet, used as inspiration, you have to refer to the source. Make sure to understand the algorithms before implementing them.

Remember, in this course, you have an opportunity to test different scenarios and learn how the algorithms behave!

The challenge is to find the best solution, i.e., the fastest algorithm ☺

Task 1

You should implement 2 sorting algorithms using both Iteration and Recursion.

	Method	**Recursive**	**Iterative**
1	**quickSort**(…)		
1.1	median-of-three pivot		
1.2	random pivot		
1.3	pivot = array[0]		
2	**insertionSort**(…)		

Read the instructions below carefully!

Each method shall use the algorithm to sort a given array of numbers. Note that the algorithm must be the one described in a book (see [1]). Each method must be implemented as:

a) an **iterative** method and

b) a **recursive** method.

Note! If is an iterative method – you are not allowed to use recursion!!!

After completing the implementation of each algorithm, the complexity/running time should be measured, i.e., how long it takes for each method to sort the arrays containing different inputs, e.g., 100, 10000, and 1000000 numbers.

The arrays shall be filled with random numbers read from the file (Seminar1 → File with random numbers).

You have to explain how you chose the input.

You have to analyze the results as well as present the results in a scientific way.

The output (for each task) should be

- presented in a table, e.g.

Method \ Input	100	1 000	10 000	100 000	1 000 000
quickSort(…) method, **recursive**					
median-of-three pivot					
random pivot					
pivot = array[0]					
quickSort(…) method, **iterative**					
etc.					

- plotted on a diagram, e.g., using Excell or similar plotting tool. Note: one diagram for each task
- and analyzed by comparing to the theory (by doing calculations)!

The requirements:

Do not change the input inside the program, i.e., read the input from the keyboard to easily test other inputs!

Or: implement a little menu !!!! no hardcoding!

Write a **script**/code/API that runs all methods with all inputs. Run your script and take a FIKA while the computer will do the job ☺ After fika you should see all results.

What can you do more with the scripts? Write one that repeats your tests many times and measures the average time. It gives you more trustworthy results.

By the way, this is a typical scenario for performance testing.

In the analysis part the comparison between the recursive versus iterative method is expected, as well as the analysis of **WHY** the results look different (for each specific input). Present the theoretical analysis of the code using Big Oh notation and **compare** the **theoretical results** to your **practical experiment**, i.e., to the output you provided in the table, as in the plot.

However, some of the analysis part should be done during the seminar, when you will discuss your results in mixed groups.

Present one more diagram for the outputs from all the methods, but only for the input 1000000, and provide the analysis.

Task 2:

You should implement 1 searching algorithm, e.g. **BinarySearch**(…) method.

The method shall use the Binary Search algorithm to find out if a number sent as a parameter to the method exists in an array that is also sent to the method as a parameter. The **BinarySearch** method must be implemented as a **recursive** method and shall return *true* if the number is found in the array and *false* if it is not found.

You should also do the measurement and presentation of the results in a similar way to Task 1.

Upload before Seminar:

Before the seminar, you should upload **two files**:

1) a **PDF**-file with your report
2) a **ZIP**-file with the codes

Note that the report should follow HKR degree thesis template. You can find it on the HKR home page.

You should name the file as **Seminar1-NameSecondname.pdf**

During the Seminar –

The seminar is for learning purposes. It is the chance to exchange the experiences gathered during the preparations. Each student should therefore explain the algorithm, explain the code, discuss the complexity of the code/algorithm in relation to the different inputs, etc.

References

1. Weiss, Mark. A. (2012), Data structures and algorithm analysis in Java. 3rd edition Harlow, Essex : Pearson. (632 p).