UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

Analysis of Advanced Algorithms
Laboratory 2

Semester II, 2019

1 Aims

The aim of this lab is to give you more experience in developing some of the implementations of the sorting algorithms studied in class. You are required to perform an empirical analysis of your implementations and compare them to your theoretical analysis of the same algorithms.

2 Submission

You are required to implement your algorithms in *Java*. Generate the graphs using the graphing tool `AAAgraphGenerator.py` on Moodle. The results generated from the experiments need be to a csv file in the following format:

```
input,time
0,0
1,1
2,2
3,3
4,4
5,5
10,10
```

where the first line is the column header separated by a comma (do not change the column header names). The following lines will be the increasing n values against the time (in milliseconds) separated by a comma.

Run the graphing tool in the terminal with additional arguments n, $ndatafiles$ and $title$ where n is the number of data files, $ndatafiles$ is a list of n data files and $title$ is the question number. For example, if there are 3 data files: `data_1.csv, data_2.csv` and `data_3.csv`, generate the multi-plot by running

```
python AAAgraphGenerator.py 3 data_1.csv data_2.csv data_3.csv Lab2Experiement1
```

The output of the graph will be saved in pdf format with the file name $Lab1Experiment1.pdf$. **Submit the pdf and csv files to Moodle. Zip up and submit the associated Java source code to Moodle as well.**

3 Code required

Implement

1. bubble sort with no escape clause
2. bubble sort with an escape clause
3. merge sort

4 Experiments

1. Test the performance of the bubble sort algorithm (with no escape clause) on lists of different sizes (i.e. values of n). Use key comparison as the basic operation. You should use the programs that you developed in Lab 1 to create the data required for these tests. Note that you are likely to have to use lists of integers which are quite big (i.e. large n) in order to get reasonable running times. Experiment to find suitable values of n.

2. Test the performance of the bubble sort (with no escape clause) using swop operation as your basic operation. Plot graphs to verify the advantage of using swop operation against using key comparisons as a basic operation.

3. Test the performance of the bubble sort (with escape clause) for best case situations. Use key comparison as your basic operation. Plot graphs to verify the advantage of escape clause against previous version of bubble sort (with no escape clause) for different values of n.

4. Test the time performance of the merge sort algorithm for lists of different sizes (i.e. values of n). Plot graphs to verify the growth rate of the algorithm.