UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

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

Advanced Analysis of Algorithms
Laboratory 1

Sheng Yan Lim

Semester II, 2019

1 Aim

The aim of this lab is to implement some of the basic algorithms, including the ones studied in the lecture, to give you more experience at the experimental side of Computer Science. For each implementation, you will need to plot a graph which shows a relation between the time complexity of the algorithm with increasing values of n (the number of elements in the list).

Note that you will be allowed to work in small groups (2 students per group) for your assignments so you are encouraged to start finding your team mate, with whom you would like to (or are prepared to) work with by doing the lab work collaboratively. It would also be a good practice to think about code reuse when you design the code for each lab session, in order to make it easy to import code for your assignment.

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 wordings). The following lines will 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 the csv files to Moodle. Zip up and submit the associated Java source code to Moodle as well.

3 Code required

1. Develop code to produce an array (or other data structure if you prefer) of size n, where n is a user specified number, of distinct random integers. In this case you may interpret *random* as being pseudo-random integers within some range where there is no discernible order in which the numbers appear in the array. In this case, *distinct* should be taken to mean that if a number appears in the array, it does so once only.

2. Develop code to produce an array (or other data structure if you prefer) of size n, where n is a user specified number, of distinct integers which are arranged in increasing order. Again *distinct* should be taken to mean that if a number appears in the array, it does so once only.

3. Develop code which will allow a user specified number, the *key*, to be inserted into a *particular* position in an array of n integers (created as in 1 or 2 above). Note that this number should not appear anywhere else in the array of integers.

4. Implement linear search algorithm studied in class.

5. Develop code which will allow you to measure the running time of the algorithms implemented above.

4 Experiments

1. Test the performance of the linear search algorithm in the best and worst case for lists of different sizes (i.e. values of n).

   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 for the worst case performance of the algorithm. Experiment to find suitable values of n.

   Plot graphs to verify that the growth rate of the algorithm is in fact $O(n)$.

2. Using the same list sizes as above, test for average case performance of the linear search by placing the *key* into many different positions in your input array and measuring the running time for each such position.

   Plot graphs to verify that the growth rate of the algorithm on average is, in fact, in $O(n)$ but with a smaller constant factor than in the worst case.