IFN 505 – Analysis of Programs

Assignment 2 – Project

(a.k.a. Theoretical and empirical comparison of the two algorithms)

Due date: Friday, October 12th (11.59pm)

Summary

In this assignment you will analyse two algorithms, both theoretically and experimentally, in terms of both efficiency and correctness. You will be implementing them as executable programs, measuring their run-time characteristics, and comparing the results with theoretical predictions for these algorithms.
The aim is to directly compare two different algorithms that both perform the same function but have different efficiencies. In particular, you must ensure that both programs are analysed under exactly the same conditions, to ensure that the results are meaningfully comparable. You will be required to count the number of basic operations performed, measure execution times, and produce a clear report describing your findings.

Tasks

To complete this assignment you must submit a written report, and accompanying evidence, describing the results of your experiments to compare the efficiency of two given algorithms. The steps you must perform, and the corresponding summaries required in your written report, are as follows.

1. You must ensure you understand the algorithms to be analysed.
   - Your report must briefly describe the two algorithms and the task that they address.

2. You must establish the correctness of the algorithms.
   - Your report must include a proof of the partial correctness of each algorithm.
   - Your report must also include a proof that both algorithms terminate (total correctness).

3. You must define a common basis for meaningful comparison of the two algorithms, in terms of basic operations and the size of inputs.
   - Your report must describe your choice of '***basic operation***' applicable to both algorithms.
   - Your report must describe your choice of '***problem size***' applicable to both algorithms.
   - Your report must summarise the predicted (theoretical) time efficiency of the two algorithms. As part of this, you need to argue whether the best-case, average-case and worst-case are different. If they are, you are only request to reflect on the ***average-case efficiency***. Of course, you may still reflect on the best-case and worst-case efficiencies if that can help in your process (as we have seen in some examples).
     You must present the arguments underpinning these predictions, either from first principles or referenced from a text. It is ***NOT*** sufficient merely to state the efficiency class with an accompanying reference.

4. You must describe the methodology you used for performing the experiments.
   - Your report must describe your choice of programming language and computing environment. Alignment of programming language, environment, and where appropriate, choice of data structures, are extremely important in providing a balanced comparison between the algorithms.

- Your report must describe your programming language implementation of the given algorithms. You must ensure that the correspondence between features of the algorithms and the corresponding program code is clear, to confirm the validity of the experiments. The program code should replicate the structure of the algorithms as faithfully as possible.
- Your report must explain how you showed that your programs work correctly. Your approach should involve thorough testing. You must provide a clear explanation of how the testing corresponds to any input classes – groups of similar inputs, boundary cases and so on – identified during your formal analysis of the algorithms.
- Your report must explain clearly how you produced test data for the experiments and chose cases to test, as appropriate. Usually this will involve generating random data for different sizes of input. In particular, it is important that both algorithms are tested using the same data, to ensure that the comparative results are meaningful.
- Your report must explain clearly how you counted basic operations, e.g., by showing the relevant program code for maintaining 'counter' variables. In particular, it should be easy to see that the method used is accurate with respect to the original algorithms.
- Your report must explain clearly how you measured execution times, e.g., by showing the relevant program code augmented with calls to 'clock' or 'time' procedures. Since it is often the case that small program fragments execute too quickly to measure accurately, you may need to time a large number of identical tests and divide the total time by the number of tests to get useful results.

5. You must describe the results of your experiments.
   - Building on the methodology for correctness considered above, your report must summarise the range of testing undertaken and note its coverage.
   - Your report must show in detail the results of comparing the number of basic operations performed by the two algorithms for different sizes of inputs. The results should be plotted as one or more graphs which make it easy to see how the two algorithms compare. You must produce enough data points to show a clear 'trend' in the outcomes. It must be clear how many individual data points contributed to the lines shown and how many individual tests contributed to each data point. You must briefly explain what the results reveal about the comparative efficiency of the two algorithms.
   - Your report must show in detail the results of comparing the execution times of the two programs for different sizes of inputs. The results should be plotted as one or more graphs which make it easy to see how the two algorithms compare. You must produce enough data points to show a clear 'trend' in the outcomes. It must be clear how many individual data points contributed to the lines shown and how many individual tests contributed to each data point. You must briefly explain what the results reveal about the comparative efficiency of the two programming language implementations of the algorithms.

6. You must produce a brief written report describing all of the above steps.
   - Your report should be prepared to a professional standard and must not include errors in spelling, grammar or typography.
   - You are free to consult any legitimate reference materials such as textbooks, web pages, etc, that can help you complete the assignment. However, you must appropriately acknowledge all such materials used either via citations to a list of references, or using footnotes. (Copying your assignment from one of your classmates is not a legitimate process to follow, however. Note the comments below concerning plagiarism.)
   - Your report must be organised so that it is easy to read and understand. The main body of the report should summarise what you did and what results you achieved as clearly and succinctly as possible. Supporting evidence, such as program listings or execution traces, should be relegated to appendices so that they do not interrupt the 'flow' of your overall argument.
   - There should be enough information in your report to allow another competent programmer to fully understand your results. This does not mean that you should include voluminous listings of

programs, execution traces, etc. However, you should include the important parts of the code, and brief, precise descriptions of your experimental methodology.

7. You must provide all of the essential information needed for someone else to duplicate your experiments in electronic form, including complete copies of program code (because the written report will normally contain code extracts only). As a minimum this data must include:
   - An electronic copy of your report.
   - The complete source code for your implementations of the algorithms.
   - The complete source code for the test procedures used in your experiments.
   - Optionally you may also include electronic versions of the data produced by the experiments.
   - In all cases, choose descriptive folder and file names.

As for assignment 1, submission will be through Blackboard.

Submission Process

There will be two submission links. The first one is for your report, which must be a ***Word or PDF file***. The second link is for an archive that contains: your code, the data used for testing, and your experimental results. For convenience, please put these items in separate folders, and compress these into a single archive. Please compress as a ***ZIP archive***. In past offerings of the unit, we have often had problem with RAR files being considered invalid.
Submissions will be done through Blackboard.

Late Penalties and Academic Dishonesty

All assignments are subject to QUT's stringent late submission policy and academic dishonesty guidelines. Please see the Faculty student services pages for details. Please note that this is an ***individual*** assignment.
Note: You may be asked to come to a meeting and explain your results (see below).

Assessment Criteria

The specific criteria by which your assignment will be assessed are detailed in the Marking Schema and Feedback Sheet for this assignment.
However, if there is any concern about the originality of your work or the quality of your experimental results, you may be asked to give a practical demonstration of your program and/or an oral explanation of your proofs. If you are not able to demonstrate and explain your results, you will receive a 0 for the corresponding part of the assignment and will be deferred to the University academic misconduct committee.

Algorithms

Algorithm1 *MaxDistance*($A[0..n-1]$)
// Input: Array $A[0..n-1]$ of numbers
// Output: Maximum distance between two of its elements
$dmax \leftarrow 0$
for $i \leftarrow 0$ **to** $n-1$ **do**
    for $j \leftarrow 0$ **to** $n-1$ **do**
        if $i \neq j$ **and** $|A[i] - A[j]| > dmax$
            $dmax \leftarrow |A[i] - A[j]|$
return $dmax$

Algorithm2 *MaxDistance2*($A[0..n-1]$)
// Input: Array $A[0..n-1]$ of numbers
// Output: Maximum distance between two of its elements
$dmax \leftarrow 0$
for $i \leftarrow 0$ **to** $n-2$ **do**
    for $j \leftarrow i+1$ **to** $n-1$ **do**
        $temp \leftarrow |A[i] - A[j]|$
        if $temp > dmax$
            $dmax \leftarrow temp$
return $dmax$