CAB301 Algorithms and Complexity

Nathan Perkins

April 3, 2016

Contents

1 Executive Summary

2 Bubble Sort

2.1 Algorithm

Briefly Describe Algorithm

2.2 Average Case Efficiency

Explain average case efficiency and order of growth for the basic algorithm, sourced online, for comparison later

3 Better Bubble Sort

3.1 Algorithm

Briefly describe algorithm and how it differs from basic bubble sort

3.2 Basic Operation

You must explain clearly the choice of basic operation for the particular algorithm of interest.

3.3 Average Case Efficiency

Your report must summarise the expected time efficiency of the algorithm with respect to the size of its input(s). This should be expressed as the algorithms predicted average-case efficiency and/or order of growth. You must explain as clearly as possible how these predictions were calculated or justified. (In some cases you will find an appropriate analysis in the literature. In other cases you may need to calculate the algorithms efficiency yourself.)

4 Methodology

4.1 Computing Environment

Explain the use of linux, eclipse as a C++ IDE and the usage of python as data analysis

4.2 Test Data Generation

Explain the generate array, reverse array for worst case etc

5 Implementation

5.1 Program

5.2 Testing

6 Experimental Results

6.1 Operational Efficiency

Your report must explain clearly how you counted basic operations, e.g., by highlighting the relevant statements inserted into the program. In particular, it should be easy to see that the method used is accurate with respect to the original algorithm.
You must perform enough experiments to produce a clear trend in the outcomes. Your report must explain how you produced test data. Depending on the kind of algorithm involved, you may need to produce sets of random values (so that you can produce average- case results for a particular size of input), or an ordered sequence of test values (so that you can show how the algorithm grows with respect to the inputs size). In either case you may choose to create test data manually (which may be very tedious) or automatically (which may require some programming).
You must present your experimental results as a graph. NB: You must state clearly how many data points contribute to the line(s) on the graph and what each data point represents. If possible, you should use a graph drawing tool that displays each data point as a distinct symbol.
You must state whether or not the experimental results matched the predicted number of operations. If they do not match then you must offer some explanation for the discrepancy. (Normally we would expect that counting basic operations produces results that closely match the theoretical predictions, but it is possible that there is some peculiarity of your experimental set-up that skews the results, or even that the theoretical predictions are wrong.)

6.2 Time Efficiency

Your report must explain clearly how you measured execution times, e.g., by showing the relevant test program. (Alternatively, you may even choose to time your program with a stopwatch, although this is unlikely to produce accurate results.) It is often the case that small program fragments execute too quickly to time accurately. Therefore, 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.
You must perform sufficient experiments to produce a clear trend in the outcomes. Your report must make clear how you produced test data (as per the discussion above on counting basic operations).
You must present your experimental results as a graph. NB: You must state clearly how many data points contribute to the results on the graph and what

each data point represents. If possible, you should use a graph drawing tool that displays each data point as a distinct symbol.
You must state whether or not the experimental results matched the predicted order of growth. It is possible that your measured execution times may not match the prediction due to factors other than the algorithms behaviour, and you should point this out if this is the case in your experiments. For instance, an algorithm with an anticipated linear growth may produce a slightly convex scatterplot due to operating system and memory management overheads on your computer that are not allowed for in the theoretical analysis. (However, a concave or totally random scatterplot is more likely to be due to errors in your experimental methodology in this case!)