CO322 - Lab 01

Comparison of simple sorting algorithms

January 19th, 2017

1 Aim

At the end of this laboratory class, a student should be able to;

1. implement the bubble, selection and insertions sort algorithms,
2. anlyse the theoretical time complexity of the above algorithms,
3. using proper method(s) measure the time taken to sort a given array, and
4. critically analyse the relationship between the theoretical complexity and the time measurement.

2 Introduction

You have learned that in programming, whatever algorithm you construct in order to perform some task is unlikely to be unique. There could be a number of other algorithms which could perform the same task.

For you to decide which algorithm is best suited for the nature of your task, you must analyze and compare each of these possible approaches. Although there are numerous ways algorithms can be compared, in this lab you will consider two measurements – asymptotic time complexity of the algorithm and its execution time.

So for this, we need some algorithms and their implementations. In this lab class we select three simple sorting algorithms – bubble sort, selection sort and insertion sort.

Task 1 Your first task is to implement the above three algorithms using a suitable programming language. Here you have the power to select – however remember with power comes responsibility. Keep in mind that you will need to measure the time of these implementations.

Task 2 Once you have implemented the algorithms, next is to check if the implementation is correct. For this you should decide on suitable test cases and test the implementation. Make sure you consider about the possible corner cases that might break your implementation.

Task 3 Next, you want to compare the performance of the three algorithms. For this you should generate some random data of size N and feed that same data to all implementations and measure the time. Recall that algorithm's performance would depend on the arrangement of data; in other words they would have different performances for the same array size N depending on how the data is arranged. Change the array size N and see what happens to the performance and the comparison.

Task 4 Your last task is to compare the empirical – experimental data that you gathered above – with the theoretical run time complexity.

It is recommended that you use Python for the implementation and measurements above. You can find number of python resources from the web and some links we found useful are given in the moodle.

3 What to submit

You are required to submit the following:

- Python implementations of the simple sorting algorithms, the test cases and the code used for testing and the time measurement code.
- A report explaining the performance variation of the three algorithms when the size is changed. Your report should provide the asymptotic runtime of the three algorithms. If the asymptotic run times are the same does that mean the algorithms should have the same execution time for a given data set? Your report should clearly discuss this question and justify the answer with the results you got. Report should not be longer than 5 pages and should not contain *cut-paste text* from random cites in the Internet.

3.1 Plagiarism

We will check the code and the report for Plagiarism. If we find copies which matches over 60%, zero marks will be offered for both reports. The same holds for code but at a similarty of 80% for the sorting algorithm implementations.