Coursework: Divide and Conquer – Report

March 5, 2014

Latex Examples

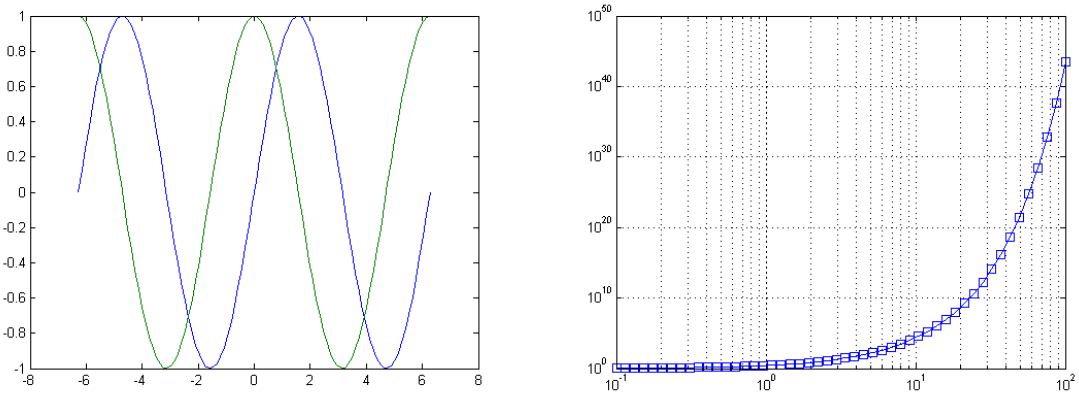

Figure 1: Example for embedding plots in Latex.

1 Recurrences

1.1 Substitution Method

Recursion trees allow you to analyse recurrences to obtain a guess for the substitution method. A closely related method is to expand out the recurrence a few times, until a pattern emerges. We call this the expansion method. An example of how to use the expansion method is given below.

Use the expansion method to guess an upper bound and the substitution method to verify your guess:

- Example: $T(n) = 2T(n/2) + n$

  Introduce constants c and expand the recurrence:

$$\begin{aligned} T(n) &\leq 2T(n/2) + cn \\ &\leq 2[2T(n/4) + cn/2] + cn = 4T(n/4) + 2cn \\ &\leq 4[2T(n/8) + cn/4] + 2cn = 8T(n/8) + 3cn \\ &\leq 8[2T(n/16) + cn/8] + 3cn = 16T(n/16) + 4cn \\ &\vdots \end{aligned}$$

  A pattern is emerging. The general term is $T(n) \leq 2^k T(n/2^k) + kcn$. Plugging in $k = \lg n$ (the height of the recursion tree), we get $T(n) \leq nT(1) + cn \lg n = O(n \lg n)$.

Now we verify $O(n \lg n)$ using the substitution method:

Prove that

$$T(n) \leq c(n \lg n)$$

Assume that

$$T(\frac{n}{2}) \leq c(\frac{n}{2} \lg \frac{n}{2})$$

By substitution

$$\begin{aligned} T(n) &\leq 2(c(\frac{n}{2} \lg \frac{n}{2})) + n \\ &= cn \lg \frac{n}{2} + n \\ &= cn \lg n - cn \lg 2 + n \\ &\leq cn \lg n \quad \text{holds for } c \geq 1 \end{aligned}$$

- $T(n) = 3T(n/2) + n$
  Insert your solution here
- $T(n) = T(n/2) + n^2$
  Insert your solution here
- $T(n) = 4T(n/2 + 2) + n$
  Insert your solution here
- $T(n) = 2T(n - 1) + 1$
  Insert your solution here
- $T(n) = T(n - 1) + T(n/2) + n$
  Insert your solution here

1.2 Master Method

Use the master method to give tight asymptotic Θ bounds:

- Example: $T(n) = 2T(n/2) + n$
  Cost at leaves: $\Theta(n^{\log_b a}) : n^{\log_2 2} = n = \Theta(n)$
  Cost per depth: $f(n) = n$
  Case 2: $f(n) = \Theta(n^{\log_b a})$: $f(n) = \Theta(n)$
  Solution: $\Theta(n^{\log_b a} \lg n) = \Theta(n \lg n)$
- $T(n) = 2T(n/4) + 1$
  Insert your solution here
- $T(n) = 2T(n/4) + n$
  Insert your solution here
- $T(n) = 2T(n/2 + 17) + n$
  Insert your solution here
- $T(n) = 4T(n/2) + 2^n$
  Insert your solution here
- $T(n) = T(3n/4) + \sqrt{n}$
  Insert your solution here

2 Sorting

2.1 Plots

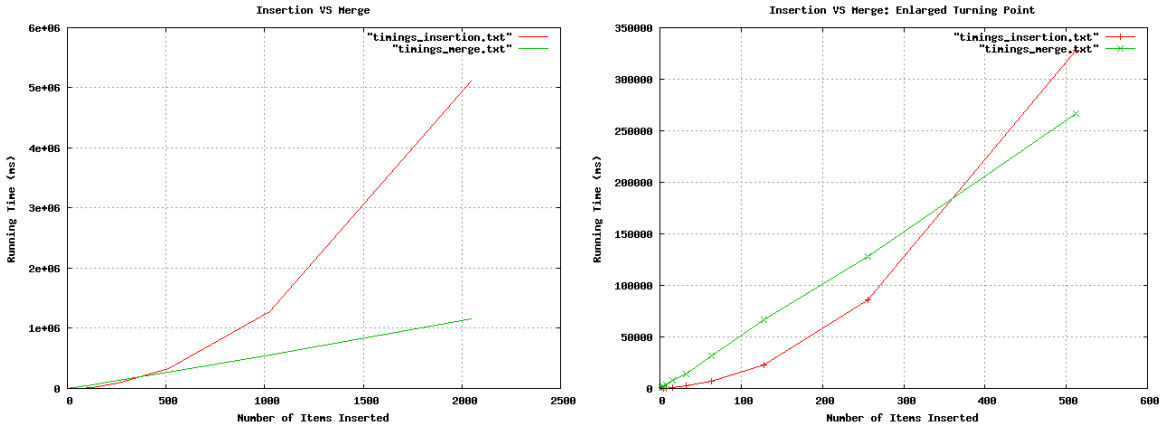

Figure 2: Graphs used to approximate the turning point.

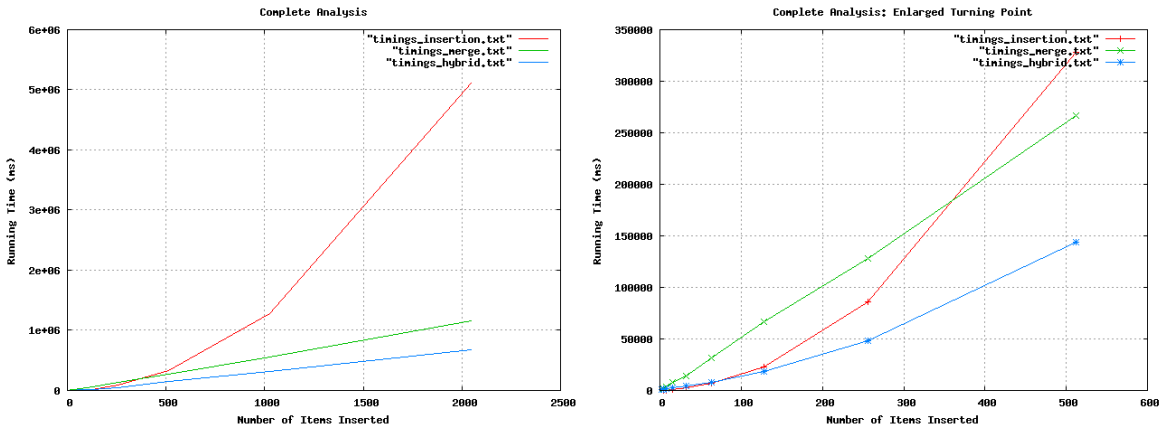

Figure 3: Final analysis of time complexities for the 3 sorting algorithms.

2.2 Discussion

The graphs in figure 2 are of insertion sort and merge sort. Both algorithms were tested with various input sizes (powers of two 1, 2, 3, ... 11) and the time taken to sort the input was recorded (in milliseconds). One can clearly see the exponential complexity of insertion sort however the nlog(n) complexity of merge sort appears almost linear in comparison. The graph on the right is of the same data however only up to an input size of 512 so that the point at which insertion sort becomes more efficient than merge sort for smaller input sizes is clearly visible (in this case 360).

The graphs in figure 3 are the same as those in figure 2 only now hybrid sort has also been plotted for the same input sizes. One can clearly see in the left graph that it is much more efficient than both other algorithms on larger data sizes. If we look at the graph on the right it also demonstrates an improvement in efficiency for input sizes smaller than 360 however at an input size of about 60 it seems to be slightly less efficient than insertion sort but still outperforms merge sort.

3 Integer Multiplication

3.1 Plots

Insert your plots here

3.2 Discussion

Insert your discussion here

3.3 Recurrences

3.3.1 Recursive Multiplication

Insert your solution here

3.3.2 Karatsuba Multiplication

Insert your solution here