CMPS 101: Spring 2016: HW 4 Analysis

Authors: Tarum Fraz - 1349796
Andres Segundo - 1408968

Q1:

In order to figure out the time complexity of *matmult*, we will look at the running time of the if statements and for statements. The if statement $if(xlen! = n)$: has a constant running time of $O(1)$. We also have a nested for loops. Both of which will iterate $0(n)$ times each. Since we know both of these running times, we can sum them up to get the worst case running time for our algorithm. *Therefore, the worst case running time for matmult is* $T(n) = O(n^2)$.

Q3:

An efficient algorithm which uses $Equation(2)$ that is faster than the brute force matrix multiplication will definitely have to use the Divide and Conquer technique. In order to successfully multiply a $2^k x 2^k$ matrix with an arbitrary 2^k dimensional, we have to denote the vector into two different vectors both of length $\frac{n}{2}$, the first vector is x_1 and consists of the first $\frac{n}{2}$ coordinates,and the second vector,x_1 consists of the remaining vectors. We will use the fact that

$$(H_n, x)_1 = H_{n-1}x_1 + H_{n-1}x_2 = H_{n-1}(x_2 + x_2)$$
$$(H_n, x)_2 = H_{n-1}x_1 - H_{n-1}x_2 = H_{n-1}(x_2 - x_2)$$

We then can use recursion to calculate the actual value of $x_1 + x_2$ and $x_1 - x_2$, and then compute $H_{n-1}(x_1 + x_2)$ and $H_{n-1}(x_1 + x_2)$. From there, the combine part is simple and we can append the values. Our divide and conquer algorithm leads to the recurrence relation of $T(n \leq 2T(\frac{n}{2}) + n$
in order to prove this recurrence relation, we will use the master theorem

Theorem (Master Theorem)

Let $T(n)$ be a monotonically increasing function that satisfies

$$T(n) = aT(\tfrac{n}{b}) + f(n)$$
$$T(1) = c$$

where $a \geq 1, b \geq 2, c > 0$. If $f(n) \in \Theta(n^d)$ where $d \geq 0$, then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

We see a = 2, b = 2, and c = 1
From the Master Theorem, we see that $2 = 2^1$, $2 = 2$
This matches case three in the theorem, meaning our time complexity is $T(n) = \theta(n^d \log n)$
.
The running time or worst case time complexity of *hadmult* is $T(n) = \theta(n \log n)$
.

Q5:

When analyzing the plot, we can clearly see that *matmult* takes longer than *hadmult*. This makes a lot of sense, because the algorithm we used in *matmult* was the brute force algorithm, and as we analyzed in $Q1$, the algorithm has a running time of $O(n^2)$; *matmult* is shown in blue. The red line, which is *hadmult* is a lot faster than *matmult*, and the line is consistent with that of $O(n \log_2 n)$. This matches the time complexity analysis we completed in $Q5$. Overall, we can say that the Divide and Conquer algorithm is much faster than the brute force algorithm.