Matthew Horowitz

A20377155

CS 430- Section 5 (Online)

Girish Rajani-Bathija

A20503736

CS 430- Section 4 (Live)

Bhavesh Rajesh Talreja

A20516822

CS 430-Section 4 (Live)

Muhammad Sheheryar Qureshi

A20517229

CS 430-Section 4 (Live)

Homework One

This report details the processes used to determine the asymptotic time complexity of the following recurrence:

$$T(n+1) = T(n)\frac{n^2 + \ 4n}{n^2 + \ 4n + 3}^{n}\frac{(n+4)}{(n+1)}$$

This report concludes the recurrence is of a Big Theta (Θ) $\log_2 n$ time complexity [Θ($\log_2$ n)]—logarithmic time. The subsequent sections will detail the numeric and analytic methodologies employed to come to this conclusion.

Question ONE:

Calculate present and analyze the numerical results of the subsequent steps of this recurrence:

*Running the Code, Recording Results*

As part of the initial investigation into the complexity of the recurrence, Java code was written to help visualize the general pattern of complexities of $T(n+1)$ outputs generated for each n recursive call.

Below is a copy of the recursive method, written in Java. (Note: a full version of the .java text file has been included with the submission of this report).

```java
//Method used for finding the complexity for a specific T(n+1) value recursively--
since double is used to get precise values, Math.pow is used

	public static double asymptoticCheck(double n) {

		if(n==0) {
			return 4;
		}
		else {

			//To simplify the ((N^2 +4N)/(N^2 + 4N +3)) is calculated in
method calculate base

			return asymptoticCheck(n-1)*(Math.pow(Homework1.calculateBase(n),
n))*((n+4)/(n+1));

		}
	}

	public static double calculateBase(double n) {
		return ((Math.pow(n,2)+4*n)/(Math.pow(n, 2)+4*n+3));
	}
```

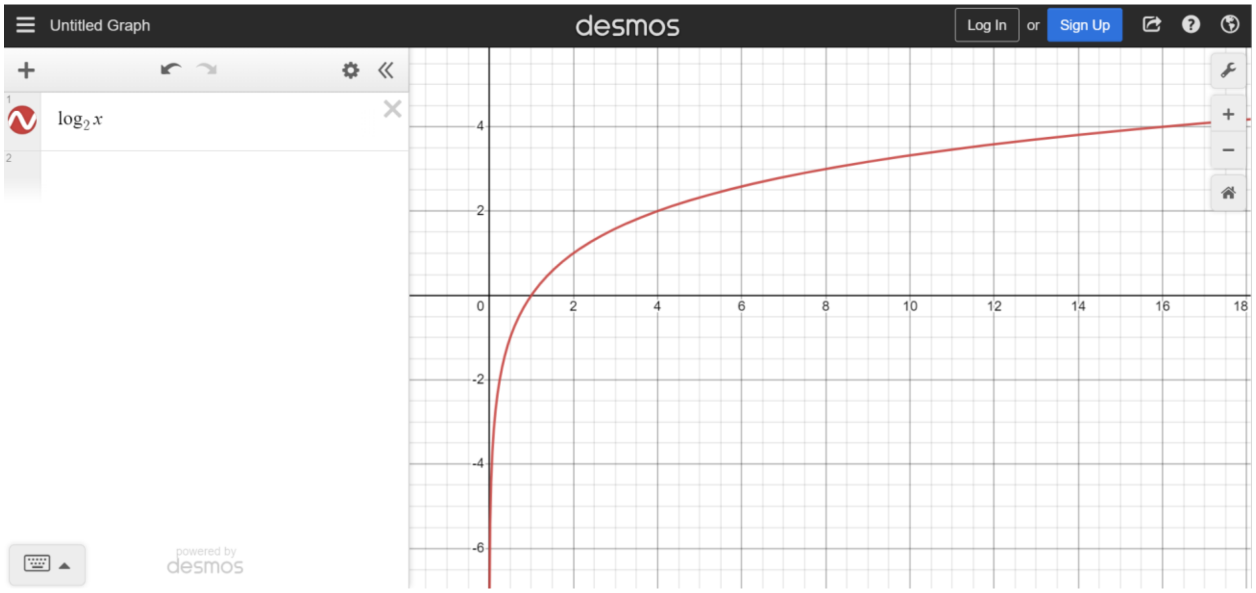
Below is a table depicting the $T(n+1)$ complexity found for every size n call.

| n size | T(n+1) size | T(n+1) output |
|---|---|---|
| 0 | T(1) | 4.000000000 |
| 1 | T(2) | 6.250000000 |
| 2 | T(3) | 8.000000000 |

| | | |
|---|---|---|
| 3 | T(4) | 9.378906250 |
| 4 | T(5) | 10.48576000 |
| 5 | T(6) | 11.39062500 |
| 6 | T(7) | 12.14265678 |
| 7 | T(8) | 12.77678495 |
| 8 | T(9) | 13.31829497 |
| 9 | T(10) | 13.78584918 |
| 10 | T(11) | 14.19347887 |
| 11 | T(12) | 14.55191522 |
| … | … | … |
| 5000 | T(5001) | 20.06747890 |
| 9000 | T(9001) | 20.07550000 |

*Interpreting the Findings of Numeric Analysis*

From the table above, it appears that the function fitting the time complexity of T(n+1) undergoes growth but appears to grow at a rate that indicates diminishing returns. Such a function can be said to be monotonically increasing, but with exponential decay. Such a function would fit the general pattern expected from the $\log_2 n$ function (as illustrated in the below graph generated on desmos.com, where the variable x is substituted in for n):



In the above graph, one can see the similarity to T(n+1) recurrence, the $\log_2 n$ function experiences initial growth followed by a decay, where the rate of change always remains positive, but the rate of growth continuously decreases. As such, at this point, it appears to that the time complexity of the given recurrence follows a $\log_2 n$ complexity.

In the subsequent sections, this hypothesis will be tested and verified by exploring the limit of the given recursion, both graphically and analytically.

*Exploring the Recurrence, Testing Limits, Discussing Implication*

So far, the numeric analysis points to the asymptotic behavior of the recurrence as fitting a $\log_2 n$ distribution. This section will justify why this belief holds true.

As with all recurrences, a recurrent function can best be seen as being comprised of two distinct parts: the recurrence itself (containing the value of the previous returned output) and the augmented section, which is responsible for either increasing or decreasing the previous returned value going forward for the next input term. Below is an illustration of this, utilizing the given recurrent equation:

$$T(n+1) = T(n)\frac{n^2+\ 4n}{n^2+\ 4n+3}^{\,n}\frac{(n+4)}{(n+1)}$$

where the term highlighted in yellow is the recurrent aspect, and the terms highlighted in green comprise the augmented section.

To gain better insight into the given recurrence, it would be advantageous to examine the limit of the augmented terms, as n approaches infinity. Doing so would help to determine how with each n input, the rate of change—or augmentation—to the overall recurrence changes.

Illustration of Possible Outcomes:

Given that T(1)=4, the following statements can be made about the limit of the augmented terms of the given recurrence T(n=1).

A. If, as n approaches infinity, this augmented term is divergent towards infinity, the overall recurrence should also grow exponentially since the unbounded growth of the augmented section should result in an unbounded growth in the overall given recurrence T(n+1). This is because the with each n input, the augmented term grows, and when multiplied by the previous value would always return an unbounded growth.
B. If, as n approaches infinity, the limit of the augmented terms goes to zero, the overall recurrence should shrink in size, as the augmented aspect would eventually result in a decimal, which when multiplied by the previous recurrent term, would shrink the overall recurrence T(n+1). This result should not be expected because if this result occurred, it would mean that total complexity would shrink even as the problem size, n, got larger.
C. If, as n approaches infinity, the limit of the augmented term is negative infinity, then the expected value of the overall recurrence should grow without bounds, but whose sign (+ or -) should alternate given that that the augmented terms would always result in a negative output, and the recurrent term would either be positive or negative depending on the n input, resulting in the product fitting this alternating pattern. This result should not be expected, as it would mean that as the size of the problem, n, increases, the time could go negative.

D. If, as n approaches infinity, the limit of the augmented term is 1, then the expected value of the overall recurrence as n approaches infinity should grow, but at ever decreasing rates. This would be due to the fact that as n approaches infinity, the recurrent term would produce an output just greater than one, which when multiplied by the previous term would result in a product that is only slightly greater than the previous term. This is because as n approaches infinity, the augmented term in this case should produce an output just greater than one, and, as such, an number multiplied by a term slightly greater than one will return the principal of that number, and slightly grow.

Given the findings from the numeric analysis, presented in an earlier section of this report, it appears that the limit as n approaches infinity of the augmented term of the recurrence of interest should approach 1.

Below is the calculation of the limit for the augmented term:

$$\lim_{n->\infty}(\frac{n^2+4n}{n^2+4n+3})^n\frac{n+4}{n+1}$$

$$\lim_{n->\infty}(\frac{n^2+4n}{n^2+4n+3})^n * \lim_{n->\infty}\frac{n+4}{n+1}$$

Left Hand Manipulation

$$\lim_{n->\infty}(\frac{n^2+4n}{n^2+4n+3})^n$$

Results in an indeterminate form

To Fix raise to the power of $e^{ln}$

$$e^{ln((n^2+4n)/(n^2+4n+3))^n}$$

Power reduce

$$e^{n(ln((n^2+4n)/(n^2+4n+3))}$$

Apply Differentiation Using the Product Rule

$$d/dn\; e^{n(ln((n^2+4n)/(n^2+4n+3))}$$

Results in

$$\frac{ln(\frac{n^2+4n}{n^2+4n+3)})}{\frac{1}{n}}$$

Applying L'Hôpital's Rule, as the above is in an indeterminant form, along with the resultant quotient rule and chain rule on the previous result yields,

$$\frac{(\frac{n^2+4n+3}{n^2+4n})(\frac{(2n+4)(n^2+4n)-(2n+4)(n^2+4n+3)}{(n^2+4n+3)^2}}{-n^{-2}}$$

Simplification

$$(\frac{n^2+4n+3)}{n^2+4n})(\frac{(2n+4)(n^2+4n-(n^2+4n+3)}{(n^2+4n+3)^2}) * n^2$$

The above result is no longer an indeterminate form.

Given that the denominator is of a higher power compared to the numerator, the result of the limit as n approaches infinity is:

0

Inserting this limit into the limit base e,

$\lim_{n->\infty} e^0$=1

Right Hand Side, apply L'Hôpital's Rule, results in:

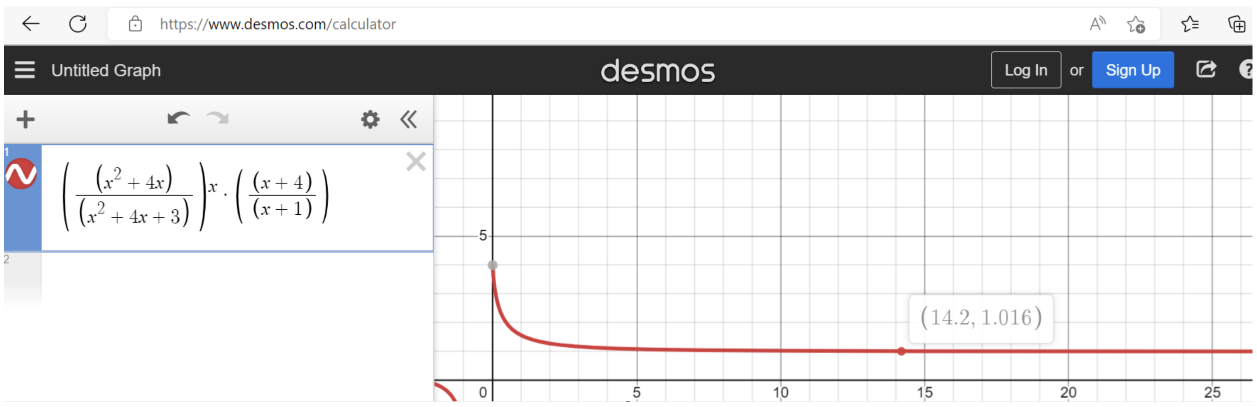
$$d/dn\frac{n+4}{n+1}=\frac{1}{1}$$

$\lim_{n->\infty}$ 1=1

Combining both sides of the augmented terms limits results in

1*1=1.

In addition, below is a graphical representation of the augmented term, created in desmos.com, illustrating that the limit of this term does in fact approach 1:


https://www.desmos.com/calculator
Untitled Graph
desmos
Log In
or
Sign Up
$\left(\frac{(x^2+4x)}{(x^2+4x+3)}\right)^x \cdot \left(\frac{(x+4)}{(x+1)}\right)$
(14.2, 1.016)
5
0
5
10
15
20
25

As expected, the limit of the augmented term does in fact go to 1. It is for this reason that the recurrence T(n+1) shows initial rates of growth that rapidly begins to level off as n becomes larger. The only function that fits this pattern is the $\log_2 n$ time complexity.

<u>Explain your methodology. How can you investigate the asymptotic behavior of the recurrence based on numerical results. Justify based on the definitions of Big Omega, Big Theta, or Big O</u>

With a strong intuition that this recurrence fits the $\log_2 n$ complexity, it now becomes imperative to solidify this belief by examining the definition of each of the three asymptotic time complexities, with emphasis on Big Theta—the tight bound case--.

Below are the brief definitions of each of the time complexities described above:

1. Big O: often described as the worst-case, Big O can be defined as:
   $O(g(n))=\{f(n)$: provided there exists a positive constant c and a value of $n \geq n_0$, such that $0 \leq f(n) \leq cg(n)\}$. Effectively, if a c constant can be found for a bound of interest that ensures that the bound is always greater than the recurrence of interest—an upper bound, proved n values greater than $n_0$, then the worst case of the recurrence has been found.

2. Big Omega: often described as the best case, Big Omega can be defined as:
   $\Omega(g(n))=\{f(n)$: provided there exists a positive constant c and a value of $n \geq n_0$, such that $0 \leq cg(n) \leq f(n)\}$. Effectively, if a c constant can be found for a bound of interest, that ensures that the bound is always less than the recurrence of interest—a lower bound-- proved n values greater than $n_0$, then the best case of the recurrence has been found.

3. Big Theta : often described as the tight bound, Big Theta can be defined as: $\Theta(g(n)=\{f(n)$: provided there exists positive constants, $c_1$ and $c_2$ and a value of $n \geq n_0$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$. Effectively, if $c_1$ and $c_2$ can be found, such that the recurrence of interest f(x) does not cross either a lower bound $c_1 g(n)$ or upper bound $c_2 g(n)$—that is of the same function g(n), but with different constants-- for all values of n greater than $n_0$, a tight bound has been found that effectively describes the both the best and worst case time complexities of the recurrent function.

   (CLRS, 2009)

Based on the graphical and numeric analysis presented earlier in this report, the recurrence T(n+1) appears to tightly approximate the logarithmic time complexity—$\log_2 n$. As such, using the above definitions of time complexity, and the desire to depict the tightest bounds possible, the remainder of this report will investigate and justify the conclusion that this recurrence is of time complexity $\Theta(\log_2 n)$.

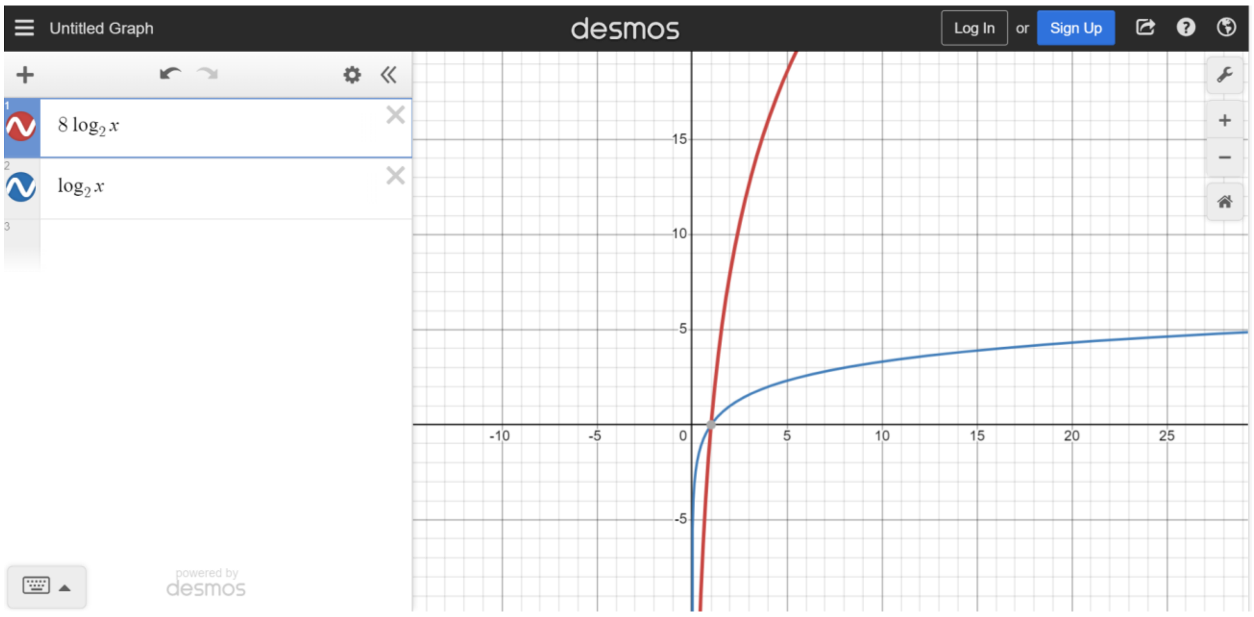
Before delving into the analytical analysis of the time complexity for the recurrence, it is imperative to justify this belief that T(n+1) fits the $\Theta(\log_2 n)$ complexity. To do this, this report will revisit the general graph of $y= \log_2 n$ to determine if there are positive constants $C_1$ and $C_2$ that when multiplied by $\log_2 n$, always produce values less than and greater than (respectively) the outputs from the recurrence T(n+1), values which are found in the table in section one of this report.

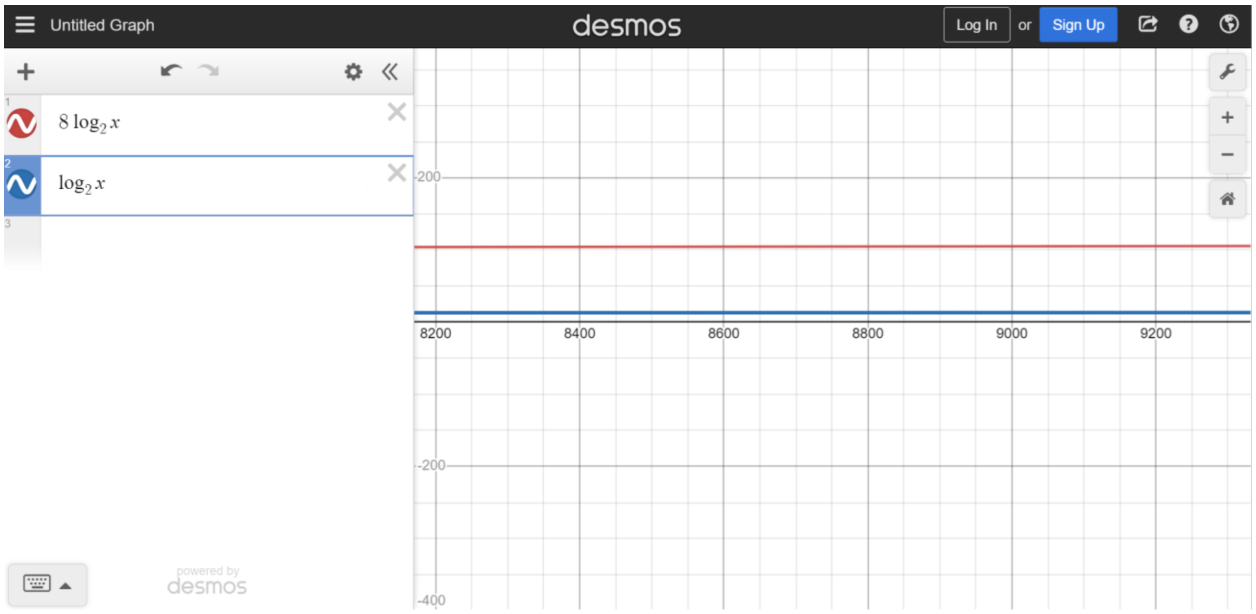
To do this, curve sketching was undertaken by visiting desmos.com. Iteratively, different $C_1$ and $C_2$ values were tested to see if an upper and lower bound could be approximated.

Since the recurrence being explored is defined as T(n+1), the input of n=0 results in T(1), which is undefined as it would result in $\log_2 0$, while an input of n=1 results in T(2), which equals zero as $\log_2 1= 0$, attempts were made to find upper and lower bounds for $n>n_0$, where $n_0=2$, which corresponds with T(3).

Utilizing Desmos, raw value coefficients were input, so that for every value greater than $n_0 = 2$, a $C_1\log_2 n$ and a $C_2\log_2 n$ were found. Through iteration and testing, checking values up to n=9000, it appears that setting $C_1$ equal to 1, and $C_2$ equal to 8, yielding $\log_2 n$ for a lower bound and $8\log_2 n$ for an upper bound, satisfies the definition of Big Theta, where $\log_2 n$ produced values less than and $8\log_2 n$ produced values greater than those found for the recurrence T(n+1) through numeric analysis.

Below are outputs from desmos.com depicting the outcome of this iterative process:





<u>Question TWO:</u>

<u>Can you find an analytical solution to the recurrence. Feel free to use any methodology. Compare the asymptotic complexity of T(n) to the numeric one found in item (1). Are they the same?</u>

As of this point, numeric analysis and curve sketching has indicated that the given recurrence T(n+1) fits an Θ($\log_2 n$) time complexity. However, to establish certainty, analytic analysis was conducted. For this report, the following 2 methods were utilized:

A. A recursion tree used to see the pattern of T(n+1) values
B. The substitution method, starting with:
    a. Testing the base case
    b. Creating an inequality—of upper and lower bounds in accordance with the definition of Big Theta-- through substitution
    c. Testing if the inequality holds by examining the limit of the augmented part of the recursion

with the results being compared to findings found through numeric analysis, in order to establish with strong certainty that the recurrence T(n+1) fits the Θ($\log_2 n$) time complexity.

*Recursion Tree*:

The first step taken to analyze the time complexity was a recursion tree. This was done to help visualize each piece of the recursion for every n input. Below is a representation of the tree created to help visualize the complexity:

Lower Bound ≤ Recursion ≤ Upper Bound

| | | |
|---|---|---|
| $C_1\log_2(n+1)$ | $T(n+1) = T(n)(\frac{n^2+4n}{n^2+4n+3})^n \frac{n+4}{n+1}$ | $C_2 \log_2(n+1)$ |
| $C_1\log_2(n)$ | $T(n) = T(n-1)(\frac{(n-1)^2+4(n-1)}{(n-1)^2+4(n-1)+3})^{n-1}\frac{n+3}{n}$ | $C_2 \log_2(n)$ |
| … | … | … |
| $C_1\log_2(0)$ | $T(1) = 4$ | $C_2 \log_2(0)$ |

What becomes apparent from above is that both the base case n=0, yielding T(1), and n=1 T(2) will not fit the above equality given that $\log_2 0$ is undefined, and $\log_2(1)=0$ (as noted earlier), while every other level of the recursion does. As such, this has helped cross-verify the supposition that $n_0$ should be set at n=2, which appears to be similar to the findings found while curve sketching on desmos.com.

*Substitution Method*

Having examined the recursive pattern and appropriate $n_0$, which is required to fit the definition of Big Theta, the substitution method was employed.

The first step to the substitution method is set up the recursion and estimate the time complexity. For this recursion, the following can be stated:

Recursion: $T(n+1) = T(n)(\frac{n^2+4n}{n^2+4n+3})^n \frac{n+4}{n+1}$

Guess: $C_1\log_2(n+1) \leq T(n+1) \leq C_2\log_2(n+1)$

The next step is to test the base case:

n=0, results in $C_1\log_2(n+1)$ and $C_2\log_2(n+1)$ equaling 1, with

$1 \leq 4 \leq 1$

This inequality does not hold true for the upper bounds. Therefore, the substitution will start at $n_0=1$, when n=1, T(n+1)=6.25 (see section one).

Next, the recursion is examined, with the estimated complexity substituted into the upper and lower bounds, $\log_2 n$, thus resulting in the following inequality:

$C_1\log_2 n \leq T(n)*(\frac{n^2+4n}{n^2+4n+3})^n \frac{n+4}{n+1} \leq C_2\log_2 n$ , where T(n), the recurrence takes a $\log_2 n$ complexity

Given the belief that the previous term in the recurrence, T(n), follows a $\log_2 n$ complexity, multiplied by the augmented term for every input of n, the coefficient of the upper bound must be larger than the largest value returned by the augmented term in order for the upper bound of the above inequality to hold true.

Having established earlier in the report that

$\lim_{n->\infty}(\frac{n^2+4n}{n^2+4n+3})^n \frac{n+4}{n+1}$ =1, (the augmented terms of recurrence),

, $(\frac{n^2+4n}{n^2+4n+3})^n \frac{n+4}{n+1}$ returns the largest value when n is lowest at $n_0$, and having found that the first acceptable $n_0$ is when $n_0=1$, the overall value of the recursion substituted into the upper bound inequality is the value found for T(2)=6.25, yielding

$6.25 \leq C_2$

Conversely, in order for the inequality to hold true for the lower bound, the coefficient of the lower bound, multiplied by $\log_2 n$, must be less than the smallest value returned by the augmented term, which is itself multiplied by the previous value believed to follow a $\log_2 n$ complexity.

The smallest value that the augmented term, $(\frac{n^2+4n}{n^2+4n+3})^n \frac{n+4}{n+1}$ , returns as n grows towards infinity is slightly greater than 1. As such, substituting this into the lower bounds, starting with $n_0=1$ of T(n+1) shows that:

$C_1 \log_2(2) \leq 1^* \log_2(2)$

$C_1 \leq 1$.

Finally, since a phase shift was enacted, so that time complexity could be expressed in terms of n rather than n+1, the $n_0$ shifts from 1 to 2, such that the recurrence is now expressed in terms of T(n) rather than T(n+1).

As a result of analytical analysis of the recurrence, it can be said that T(n+1) fits the time complexity of $\Theta(\log_2 n)$, with a lower bound having a positive constant, for example $C_1$=.5, and an upper bound with the constant, for example $C_2=7$, for all terms n, $n \geq n_0=2$. This output matches the expected findings from examining the graph of the recurrence, which was conducted in the numeric analysis section of this report.

*Comments on Work:*

The work done on this program was divided up evenly between all team members. Group members met throughout the week prior to submission, discussing the problem at hand and the best way to solve it.

Specifically:

Matt: Edited the final report, and calculated the limits

Girish and Muhammad crafted the recursion tree, drafted initial report.

Bhavesh helped crafted the graphs, wrote Java code

All team members assisted in reviewing calculations, making decisions regarding methodologies, and came to an agreement on the complexity findings.

The group worked well together with each team member equally pulling their weight.

*References:*

Graphs created using Desmos.

Definitions taken from: Cormen, T.H.; Leiserson; C.E.; Rives, R.L.; Stein, C (2009), Introduction to Algorithms: 3rd Edition; Cambridge, MA: MIT Press.