Find The Asymptotic Upper Bound Of The Following Recurrence Using The Master Method: A. T(n)=3T(n/4)+nlog(n)

Find The Asymptotic Upper Bound Of The Following Recurrence Using The Master Method: A. T(n)=3T(n/4)+nlog(n)

Introduction to Recurrence Relations and Their Significance

Recurrence relations are fundamental tools in analyzing the time complexity of recursive algorithms. They define the running time of a problem in terms of the running time of smaller subproblems, providing a mathematical framework for understanding algorithm efficiency. In computer science, mastering the methods to solve these relations—particularly the Master Method—is essential for evaluating and optimizing recursive algorithms.

This article focuses on applying the Master Method to determine the asymptotic upper bound of the recurrence:

$$T(n) = 3T(n/4) + n \log n$$

Our goal is to analyze this recurrence relation to find its tight asymptotic bound, specifically its Big O notation, which describes the upper limit of algorithm performance as input size grows large.

Understanding the Master Method

The Master Method provides a straightforward way to solve divide-and-conquer recurrences of the form:

$$T(n) = aT(n/b) + f(n)$$

where:

- $a \geq 1$ is the number of subproblems at each recursion level.
- $b > 1$ is the factor by which the problem size decreases at each recursive call.
- $f(n)$ is the cost of dividing the problem and combining solutions at each level.

The Master Method compares $f(n)$ with $n^{\log_b a}$ (the critical exponent) to determine the asymptotic bound. It involves three cases:

1. Case 1: If $f(n) = O(n^{\log_b a - e})$ for some $e > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. Case 2: If $f(n) = \Theta(n^{\log_b a} \log^k n)$ for some $k \geq 0$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
3. Case 3: If $f(n) = \Omega(n^{\log_b a + e})$ for some $e > 0$, and if regularity condition holds, then $T(n) = \Theta(f(n))$.

Applying these cases requires calculating a, b, and $f(n)$, then comparing $f(n)$ to $n^{\log_b a}$.

Analyzing the Given Recurrence

Let's analyze the recurrence:

$$T(n) = 3T(n/4) + n \log n$$

Identify the parameters:

- $a = 3$ (number of recursive calls)
- $b = 4$ (problem size reduction factor)
- $f(n) = n \log n$ (cost at each level)

Our first step is to compute $n^{\log_b a}$.

Calculating log_b a

Since $a = 3$ and $b = 4$, we have:

$$\log_b a = \log_4 3$$

Using change of base formula:

$$\log_4 3 = \frac{\log 3}{\log 4}$$

Assuming logarithm base 2 for simplicity:

- $\log_2 3 \approx 1.58496$
- $\log_2 4 = 2$

Thus:

$$\log_4 3 = \frac{1.58496}{2} \approx 0.7925$$

Therefore:

$$n^{\log_b a} = n^{0.7925}$$

This critical exponent will help us compare $f(n)$ to determine the case.

Comparing f(n) with n^{log_b a}

Recall:

- $f(n) = n \log n$
- $n^{\log_b a} \approx n^{0.7925}$

Now, analyze the growth of $f(n)$ relative to $n^{0.7925}$.

Growth Rate Analysis

- n^{0.7925} grows faster than any polylogarithmic function multiplied by a polynomial with an exponent less than 1.
- f(n) = n log n = Θ(n log n)

Compare n log n to n^{0.7925}:

- Since n^{0.7925} ≈ n^{0.79}, which is polynomially smaller than n, the n log n function grows faster than n^{0.7925} but slower than n^1.

Thus, f(n) is asymptotically larger than n^{log_b a}:

    f(n) = Ω(n^{log_b a + e})

for some e > 0, since:

    n log n = Ω(n^{0.7925 + e})

for e ≈ 0.2, because:

- n^{0.7925 + 0.2} = n^{0.9925}
- n log n grows slightly faster than n^{0.7925} but less than n^{1}.

Therefore, f(n) dominates n^{log_b a}.

Verifying the Regularity Condition

Before applying Case 3, we need to verify the regularity condition:

    a f(n/b) ≤ c f(n), for some constant c < 1 and sufficiently large n

Calculate a f(n/b):

    a f(n/b) = 3 (n/4) log(n/4)

Simplify:

    = 3 (n/4) (log n - log 4) = (3n/4) (log n - 2)

Compare with f(n) = n log n:

- The ratio is:

    \frac{a f(n/b)}{f(n)} = \frac{(3n/4) (log n - 2)}{n log n} = \frac{(3/4) (log n - 2)}{log n} = \frac{3}{4} \left(1 - \frac{2}{log n} \right)

As n grows large, log n □ □ , so:

    \frac{a f(n/b)}{f(n)} \to \frac{3}{4} 1 = 0.75 < 1

This satisfies the regularity condition (since the ratio is less than 1 for sufficiently large n). Therefore, the conditions for Case 3 are satisfied.

Applying the Master Method

Given our analysis:

- $f(n)$ dominates $n^{\log_b a}$ (since $n^{0.7925}$ grows slower than $n \log n$).
- The regularity condition holds, and $f(n)$ is polynomially larger than $n^{\log_b a}$.

This aligns with Case 3 of the Master Method, which states:

> If $f(n) = \Omega(n^{\log_b a + e})$ for some $e > 0$, and the regularity condition holds, then:

$$T(n) = \Theta(f(n))$$

Therefore, the asymptotic upper bound of the recurrence is:

$$T(n) = \Theta(n \log n)$$

Final Result and Interpretation

The recurrence relation:

$$T(n) = 3T(n/4) + n \log n$$

has an asymptotic upper bound of $\Theta(n \log n)$. This means that, for large input sizes, the running time of the algorithm modeled by this recurrence grows proportionally to $n \log n$, up to constant factors.

Implications:

- The algorithm exhibits a near-linearithmic time complexity.
- The recursive division and combination costs dominate the overall runtime.
- Understanding this bound aids in optimizing the algorithm or comparing it with other approaches.

Summary and Key Takeaways

- The recurrence relation was identified and parameters a and b were extracted.
- Calculated $\log_b a$ as approximately 0.7925.
- Compared $f(n) = n \log n$ to $n^{0.7925}$ to determine dominance.
- Verified the regularity condition for the Master Method's Case 3.
- Concluded that the asymptotic upper bound is $\Theta(n \log n)$.

Mastering the application of the Master Method simplifies analyzing recursive algorithms, especially when dealing with complex costs like $n \log n$

Frequently Asked Questions

What is the recurrence relation given in the problem?

The recurrence relation is $T(n) = 3T(n/4) + n \log n$.

How do we identify the parameters a, b, and f(n) in the recurrence $T(n) = aT(n/b) + f(n)$?

In this recurrence, $a = 3$, $b = 4$, and $f(n) = n \log n$.

Which case of the Master Theorem applies to this recurrence?

We compare $f(n) = n \log n$ with $n^{\log_b a} = n^{\log_4 3}$ to determine the applicable case.

What is the value of log_b a in this recurrence?

$\log_4 3$ is approximately 0.792, since $\log_4 3 = \ln 3 / \ln 4$.

How does $f(n) = n \log n$ compare to $n^{\log_b a}$ in this case?

Since $n \log n$ grows faster than $n^{0.792}$ but slower than n^{1}, $f(n)$ is polynomially larger than $n^{\log_b a}$ but not as large as $n^{1 + e}$.

What is the asymptotic upper bound for T(n) using the Master Method?

Since $f(n) = \Theta(n \log n)$ and it dominates $n^{\log_b a} = \Theta(n^{0.792})$, and $f(n)$ is polynomially larger than $n^{\log_b a}$, the recurrence falls under case 3, giving $T(n) = \Theta(f(n)) = \Theta(n \log n)$.

Additional Resources

Master Method Application for Recurrence Relations: Analyzing $T(n) = 3T(n/4) + n \log n$

Introduction

Recurrence relations are fundamental to understanding the runtime complexities of divide-and-conquer algorithms. They describe how the problem size reduces at each recursive step and how the work accumulates. The Master Method, introduced by Michael T. Clancy and others, provides a straightforward way to determine asymptotic bounds for recurrence relations of the form:

$$
T(n) = aT\left(\frac{n}{b}\right) + f(n)
$$

where:

- $a \geq 1$ is the number of subproblems,
- $b > 1$ indicates the factor by which the problem size reduces,
- $f(n)$ is the non-recursive work done at each level.

In this detailed review, we analyze the recurrence:

$$
T(n) = 3T\left(\frac{n}{4}\right) + n \log n
$$

using the Master Method to find its asymptotic upper bound.

Understanding the Components of the Recurrence

Before applying the Master Method, it's essential to understand the components:

- Number of subproblems (a): 3
- Problem size reduction factor (b): 4
- Non-recursive work $f(n)$: $n \log n$

The goal is to compare $f(n)$ with $n^{\log_b a}$ to determine the dominating term.

Calculating $\log_b a$

The core of the Master Method revolves around the critical exponent:

$$
d = \log_b a
$$

Given $a=3$ and $b=4$:

$$
d = \log_4 3
$$

To compute this logarithm, we use the change of base formula:

$$
\log_4 3 = \frac{\log 3}{\log 4}
$$

Using natural logs or logs base 2 (both valid):

- $\log_2 3 \approx 1.58496$
- $\log_2 4 = 2$

Thus,

$$
\log_4 3 = \frac{\log_2 3}{\log_2 4} = \frac{1.58496}{2} \approx 0.79248
$$

So,

$$
n^{\log_b a} = n^{0.79248}
$$

This term serves as a benchmark for comparison against $f(n) = n \log n$.

Comparison of $f(n)$ with $n^{\log_b a}$

The next step involves comparing $f(n)$ with $n^{\log_b a}$:

- $f(n) = n \log n$
- $n^{\log_b a} \approx n^{0.79248}$

Since $n \log n$ grows faster than $n^{0.79248}$ for large n, we see:

$$
f(n) = n \log n = \Omega(n^{0.79248 + \varepsilon}) \quad \text{for some} \quad \varepsilon > 0
$$

In particular, because $\log n$ grows slowly but unboundedly, for sufficiently large n:

$$
f(n) \gg n^{0.79248}
$$

This indicates that $f(n)$ dominates $n^{\log_b a}$.

Applying the Master Method: Three Cases

The Master Method provides three cases to determine the asymptotic bound:

Case 1: $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$

- $f(n)$ is polynomially smaller than $n^{\log_b a}$.
- Then, $T(n) = \Theta(n^{\log_b a})$.

Case 2: $f(n) = \Theta(n^{\log_b a} (\log n)^k)$

- $f(n)$ matches $n^{\log_b a}$ up to polylogarithmic factors.
- Then, $T(n) = \Theta(n^{\log_b a} (\log n)^{k+1})$.

Case 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and regularity condition holds

- $f(n)$ dominates $n^{\log_b a}$.
- $T(n) = \Theta(f(n))$.

Determining the Correct Case

In our scenario:

$$
f(n) = n \log n
$$

$$
n^{\log_b a} \approx n^{0.79248}
$$

Since $n \log n$ grows faster than $n^{0.79248}$, for large n, there exists $\varepsilon > 0$ such that:

$$
f(n) = \Omega(n^{0.79248 + \varepsilon})
$$

In particular, choosing $\varepsilon = 0.2$:

$$
n^{0.79248 + 0.2} = n^{0.99248}
$$

And since $n \log n \gg n^{0.99248}$ for large n, the regularity condition needed for Case 3 (which ensures polynomial dominance) holds.

Verifying the Regularity Condition

The regularity condition for Case 3 requires that:

$$
a \, f\left(\frac{n}{b}\right) \leq c \, f(n)
$$

for some constant $c < 1$, and sufficiently large n.

Calculate $a \, f(n/b)$:

$$
a \, f(n/b) = 3 \times \left(\frac{n}{4}\right) \log \left(\frac{n}{4}\right)
$$

Expressed explicitly:

$$
3 \times \frac{n}{4} \left(\log n - \log 4\right) = \frac{3n}{4} (\log n - 2)
$$

Compare this with $f(n) = n \log n$:

$$
\frac{a \, f(n/b)}{f(n)} = \frac{\frac{3n}{4} (\log n - 2)}{n \log n} = \frac{3}{4} \times \frac{\log n - 2}{\log n}

As $n \to \infty$, $\log n \to \infty$, so:

$$
\frac{\log n - 2}{\log n} \to 1
$$

Therefore,

$$
\frac{a \, f(n/b)}{f(n)} \to \frac{3}{4} \times 1 = 0.75 < 1
$$

This satisfies the regularity condition for some $c \approx 0.75$, confirming that Case 3 applies.

Conclusion: Asymptotic Upper Bound

Based on the above analysis:

- $f(n) = n \log n$ grows faster than $n^{\log_b a} \approx n^{0.79248}$,
- The regularity condition holds,
- Hence, Case 3 of the Master Theorem applies.

Therefore, the asymptotic upper bound for the recurrence:

$$
T(n) = 3T(n/4) + n \log n
$$

is:

$$
\boxed{
T(n) = \Theta(n \log n)
}
$$

This indicates that the recurrence's solution grows on the order of $n \log n$, matching the non-recursive work at each level, scaled appropriately.

Implications and Practical Significance

Understanding this recurrence informs us about the efficiency of divide-and-conquer algorithms with similar structures. For example:

- Algorithms that split data into 3 parts, each of size $n/4$,
- and perform work proportional to $n \log n$ at each step,

will have an overall time complexity that scales roughly linearly with $n \log n$.

This insight aids algorithm designers in estimating performance and optimizing strategies, especially when choosing the number of subproblems or the work per level.

Additional Considerations and Alternatives

While the Master Method provides a quick route to the asymptotic bound, it hinges on certain assumptions:

- The recurrence must be of the form \(aT(n/b) + f(n)\).
- The regularity condition must hold for the third case.

In cases where the recurrence doesn't fit this mold or the regularity condition fails, alternative methods such as the recursion tree method, substitution, or Akra-Bazzi theorem may be necessary.

For our specific recurrence, however, the Master Method sufficed and yielded a clear result.

Summary

- The recurrence \(T(n) = 3T(n/4) + n \

Find The Asymptotic Upper Bound Of The Following Recurrence Using The Master Method A T N 3t N 4 Nlog N

Find The Asymptotic Upper Bound Of The Following Recurrence Using The Master Method A T N 3t N 4 Nlog N

Related Articles

- An Entrepreneur Has Two Projects To Choose Between. Both Require An Investment Of $1 Which Must Be Borrowed.
- What Does It Mean That "American Subsidized Bythe United States Government?" Why Do American Farm Subsidies
- Read The Conversation.CHAIRMAN REED. Id Like To Call This Meeting To Order. Mr. Hollembeak, Your Willingness

Back to Home