

CSC 301: Analysis of Algorithms

Lecture 7: Analyzing Divide and Conquer Algorithms

William Rebelsky
[rebelskyw]

Grinnell College

Monday 5th February, 2024

Reminders

- Homework 1 is due Wednesday, you should have all the material to solve the problems
- We have a course website: `rebelskyw.cs.grinnell.edu`
- Tokens: Let me know if you have any events you would like me to add
 - CS Table Tuesday at 12
- Quiz 1 is due Tonight
- Office Hours: 1604 Wednesday and Friday 1-2:50

Comments on Homework 0

- For the proof: This was mostly for me to get an idea of your proof skills. If you attempted the proof, you got an S, but I would ask that you pay attention to the comments. Some of the proofs would not be sufficient for credit later in this course
- We hope to have the grades out by the end of the week.

Table of Contents

1 Divide and Conquer Basics

2 Matrix Multiplication

- Loop
- Divide and Conquer

3 Substitution Method

4 Recursion Tree Method

Divide and Conquer

Recursion!

- Base case: small problem, solve directly
- Recursive case:
 - **Divide** the problem into smaller subproblems
 - **Conquer** the subproblems recursively
 - **Combine** the subproblems to form a solution to the original problem

Algorithmic Recurrences

A recurrence $T(n)$ is algorithmic if for every sufficiently large threshold $n_0 > 0$ the following hold:

- For all $n < n_0$ we have $T(n) = \Theta(1)$
- For all $n \geq n_0$ every path of recursion terminates in a defined base case in a finite number of recursive invocations

(abuse of notation, constrained to values of n for which $T(n)$ is defined)

Table of Contents

1 Divide and Conquer Basics

2 Matrix Multiplication

- Loop
- Divide and Conquer

3 Substitution Method

4 Recursion Tree Method

Matrix Multiplication

Algorithm 1: Matrix-Multiply(A,B,C,n)

Input: nxn matrices A, B, C, and integer n**Output:** $A \cdot B + C$

```
1 for  $i=1$  to  $n$  do
2   |   for  $j=1$  to  $n$  do
3     |   |   for  $k=1$  to  $n$  do
4       |   |   |    $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$ 
5 return (C)
```

Matrix Multiplication Loop Analysis

- 3 loops, size n each
- inner loop 3 constant actions
- $\Theta(n^3)$

Matrix Multiplication

Algorithm 2: Matrix-Multiply-Recursive(A, B, C, n)

Input: $n \times n$ matrices A , B , C , and integer n **Output:** $A \cdot B + C$

```
1 if  $n=1$  // Base case then
2   |    $c_{11} = c_{11} + a_{11} \cdot b_{11}$ , return ()
3 //Implicit Else: Divide
4 partition  $A$ ,  $B$ , and  $C$  into  $\frac{n}{2} \times \frac{n}{2}$  submatrices
5 // Conquer
6 Matrix-Multiply-Recursive( $A_{11}, B_{11}, C_{11}, \frac{n}{2}$ )
7 Matrix-Multiply-Recursive( $A_{11}, B_{12}, C_{12}, \frac{n}{2}$ )
8 Matrix-Multiply-Recursive( $A_{21}, B_{11}, C_{21}, \frac{n}{2}$ )
9 Matrix-Multiply-Recursive( $A_{21}, B_{12}, C_{22}, \frac{n}{2}$ )
10 Matrix-Multiply-Recursive( $A_{12}, B_{21}, C_{11}, \frac{n}{2}$ )
11 Matrix-Multiply-Recursive( $A_{12}, B_{22}, C_{12}, \frac{n}{2}$ )
12 Matrix-Multiply-Recursive( $A_{22}, B_{21}, C_{21}, \frac{n}{2}$ )
13 Matrix-Multiply-Recursive( $A_{22}, B_{22}, C_{22}, \frac{n}{2}$ )
```

Matrix Multiplication Recursion Analysis

- Base case: $n=1$, one scalar multiplication, one scalar addition:

$$T(1) = \Theta(1)$$

- Recursive case: 8 calls each with Matrices of size $\frac{n}{2}$, Partitioning time of $\Theta(1)$. Total time:
$$T(n) = \Theta(1) + 8T(\frac{n}{2})$$
- Combination step: Terms are added in place, no combine step
- Total runtime: $T(n) = n^3$. Why?

Table of Contents

1 Divide and Conquer Basics

2 Matrix Multiplication

- Loop
- Divide and Conquer

3 Substitution Method

4 Recursion Tree Method

Substitution Method

- Most general way to solve recurrences
- Two steps:
 - ➊ Guess the form using symbolic constants
 - ➋ Use induction to prove the solution and find the constants
- Best to try to do upper and lower bounds separately

Substitution Method Example

$$T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + \Theta(n)$$

- Guess: $T(n) = O(n \lg n)$
- Inductive Hypothesis: $T(n) \leq c n \lg n$ for all $n \geq n_0$. We will choose c and n_0 later
- Assume by induction that this holds for all numbers at least as big as n_0 and less than n . In particular, therefore, if $n \geq 2n_0$ it holds for $\lfloor \frac{n}{2} \rfloor$, yielding $T(\lfloor \frac{n}{2} \rfloor) \leq c \lfloor \frac{n}{2} \rfloor \lg(\lfloor \frac{n}{2} \rfloor)$
- Substituting into the recurrence

Substitution Method Example

$$\begin{aligned}
 T(n) &= 2T(\lfloor \frac{n}{2} \rfloor) + \Theta(n) \\
 &\leq 2(c\lfloor \frac{n}{2} \rfloor \lg(\lfloor \frac{n}{2} \rfloor)) + \Theta(n) \\
 &\leq 2(c(\frac{n}{2})\lg(\frac{n}{2})) + \Theta(n) \\
 &= cn\lg(n/2) + \Theta(n) \\
 &= cn\lg n - cn\lg(2) + \Theta(n) \\
 &= cn\lg n - cn + \Theta(n) \\
 &\leq cn\lg n
 \end{aligned} \tag{1}$$

The final step holds if we constrain n_0 and c such that $n \geq 2n_0$.

To show the base case, pick $c = \max\{T(2), T(3)\}$ then

$T(2) \leq c < (2\lg 2)c$ and $T(3) \leq c < (3\lg 3)c$ since $n\lg n \geq 1$ for $n \geq 2$.

Practice

Prove the recurrence from earlier: $T(n) = \Theta(1) + 8T(\frac{n}{2})$ implies $T(n) = n^3$ using the substitution method

Tips

- Make good guesses based on functions/recurrences you've already seen
- Subtract a lower order term (e.g. $\Theta(1)$) if the inductive hypothesis fails (see page 92 in the book)
- Avoid pitfalls:
 - Don't use asymptotic notation in the inductive hypothesis, the constants are important

Table of Contents

1 Divide and Conquer Basics

2 Matrix Multiplication

- Loop
- Divide and Conquer

3 Substitution Method

4 Recursion Tree Method

Recursion Tree Method

- Best used to generate a good guess for the substitution method
- If precise when using the tree, constitutes a proof itself
- Steps:
 - 1 Start from the input
 - 2 Draw the tree going down
 - 3 Figure out the height at termination
 - 4 Sum across the rows

Recursion Tree Example

$$T(n) = 3T(n/4) + \Theta(n^2)$$

- $T(n) = 3T(n/4) + cn^2$
- Cost at row 1: cn^2
- Cost at row 2: $\frac{3}{16}cn^2$
- Cost at row 3: $(\frac{3}{16})^2cn^2$
- Height of tree: $\log_4(n)$
- Cost of last row: $3^{\log_4(n)} = n^{\log_4(3)}$ times $\Theta(1)$
- Creating the summation, can show
$$T(n) < \frac{16}{13}cn^2 + \Theta(n^{\log_4(3)})$$
 which is $O(n^2)$

Practice

Prove the recurrence from earlier: $T(n) = \Theta(1) + 8T(\frac{n}{2})$ implies $T(n) = n^3$ using the recursion tree method

Practice

Prove that $T(n) = T(n/3) + T(2n/3) + \Theta(n)$ is $\Theta(n \lg n)$

Practice

Prove that $T(n) = T(n/3) + T(2n/3) + \Theta(n)$ is $\Theta(n \lg n)$

- Drawing the tree we see that the cost of the leaves is $O(n)$ and the cost of the internal nodes is $O(n \lg n)$. Using substitution we get the result. See pages 98-100 in the textbook for more details