CSCE 411: Design and Analysis of Algorithms

Lecture 2: Divide and Conquer, Part II

Date: *Lecturer: Nate Veldt*

Course Logistics

- Read section 4.6 if you are interested in the proof of the Master Theorem.
- Continue skimming chapters 1-3
- Begin reading Chapter 15 for our next unit on Dynamic Programming
- Syllabus quiz is due Sat, Aug 24. HW 1 and intro video due Fri, Aug 30

1 Continued Analysis of Merge Sort

Merge Sort. Given n numbers to sort

- Divide the sequence of length n into arrays of length $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$
- Recursively sort the two halves
- (Merge Procedure) Combine the two halves by sorting them.

Question 1. What is the runtime of the *merge procedure* in Merge Sort?

A $\Theta(1)$

B $\Theta(n \lg n)$

C $\Theta(n)$

D $\Theta(n^2)$

Proof: At each step we compare two numbers and write one of them to a master array. This takes $\Theta(1)$ time. There are n steps since the master array has length n, so $\Theta(n)$ time total.

2 Recurrence Analysis for Divide and Conquer

Runtimes for divide and conquer algorithms can be described in terms of a recurrence relation, which lists the runtime for a problem in terms of the runtime for a smaller instance of the problem.

Let $T(n)$ denote the runtime for a problem of ~~instance~~ size n.

Example: merge-sort Assume for this analysis that $n = 2^p$ where $p \in \mathbb{N}$. $\lg n = p$

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

$$\begin{aligned}
T(n) &= 2\underbrace{T(n/2)}+\Theta(n) & & 2^1 T(n/2^1) + 1 \cdot \Theta(n) \\
&= 2\left[2T(n/4) + \Theta\left(\tfrac{n}{2}\right)\right] + \Theta(n) \\
&= 4T(n/4) + 2\Theta\left(\tfrac{n}{2}\right) + \Theta(n) \\
&= 4T(n/4) + 2\Theta(n) & \longrightarrow \; & 2^2 T(n/2^2) + 2\Theta(n) \\
&= 4\left[2T(n/8) + \Theta\left(\tfrac{n}{4}\right)\right] + 2\Theta(n) \\
&= 8T(n/8) + 3\Theta(n) & \longrightarrow \; & 2^3 T(n/2^3) + 3\Theta(n) \\
&\;\;\vdots \\
&= 2^p T(n/2^p) + p\,\Theta(n) \\
&= n\,T(1) + \lg n\,\Theta(n) \\
&= \Theta(n \log n)
\end{aligned}$$

3 Three methods for solving recurrences

Given a recurrence relation, there are three approaches to finding the overall runtime.

- **Recursion tree**: as above
- **Substitution method**: guess and prove (typically by induction)
- **Master theorem**: someone else did it for you
  (provided a template you just need to apply)

n^2 "<<" $n^{2.0000001}$

n^2 not polynomially smaller than $n^2 \log n$

4 The Master Theorem for Recurrence Relations

Theorem 4.1. *Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the relation:*

$g(n) = n^{\log_b a}$

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

$f(n)$ "<<" $g(n)$

1. *If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then* $T(n) = \Theta(n^{\log_b a}) = \Theta(g(n))$

$f(n)$ "=" $g(n)$

2. *If $f(n) = \Theta(n^{\log_b a})$, then* $T(n) = \Theta(n^{\log_b a} \log n)$

$f(n)$ ">>" $g(n)$

3. *If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then* $T(n) = \Theta(f(n))$

4.1 Example: Merge-Sort

Recall that Merge-Sort satisfies the following recurrence:

$$T(n) = \begin{cases} \theta(1) & \text{if } n = 1 \\ 2T(n/2) + \theta(n) & \text{if } n > 1 \end{cases} \tag{1}$$

We can apply the master theorem with: $a = 2$ $b = 2$ $g(n) = n^{\log_2 2} = n$

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

$T(n) = \Theta(n^{\log_2 2} \log n) = \Theta(n \log n)$.

4.2 What to know about the master method?

Proof idea: Follows by applying the recursion tree method on a general recursion relationship.

A full proof can be found in Section 4.6 of the textbook.

What's more important: Knowing how to apply it.

$$\boxed{T(n) = \underline{a}\,T(n/\underline{b}) + f(n)}$$

4.3 Examples

Apply the master theorem to the following recurrences:

a	b	$g(n)$	
9	3	$n^{\log_b a} = n^{\log_3 9} = \underline{n^2}$	
3	4	$n^{\log_4 3}$ "<<" $n \log n$	
7	2	$n^{\log_2 7}$ ">>" $\Theta(n^2)$	

$$T(n) = \underline{9}T(n/\underline{3}) + \underline{n} \quad \text{(a)} \qquad f(n) = n \tag{2}$$

$$T(n) = 3T(n/4) + n \log n \quad \text{(b)} \tag{3}$$

$$T(n) = 7T(n/2) + \underline{\underline{\Theta(n^2)}} \quad \text{(c)} \tag{4}$$

(a) $\underbrace{n^{\log_3 9} = n^2}_{g(n)}$ ">>" $\underbrace{n}_{f(n)}$ $\qquad T(n) = \Theta(g(n)) = \Theta(n^2)$

(b) Exercises

(c)

5 Strassen's Algorithm for Matrix Multiplication

Let A and B be $n \times n$ matrices, and $C = AB$. The (i, j) entry of C is defined by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\overset{A}{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}} \overset{B}{\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}} = \overset{C}{\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}}$$

$$c_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31} + a_{14} b_{41}$$

Algorithm 1 Simple Square Matrix Multiply

Input: $A, B \in \mathbb{R}^{n \times n}$
Output: $C = AB \in \mathbb{R}^{n \times n}$
Let $C = zeros(n, n)$
for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    $c_{ij} = 0$
    for $i = 1$ to n **do**
      $c_{ij} = c_{ij} + a_{ik} b_{kj}$
    end for
  end for
end for
Return C

5.1 An attempt at divide-and conquer

Assume that $n = 2^p$ for some positive integer $p > 1$.

Algorithm 2 Simple Recursive Square Matrix Multiply (SSMM)

Input: $A, B \in \mathbb{R}^{n \times n}$
Output: $C = AB \in \mathbb{R}^{n \times n}$
if $n == 1$ **then**
  $c_{11} = a_{11} b_{11}$
else
  $C_{11} = SSMM(A_{11}, B_{11}) + SSMM(A_{12}, B_{21})$
  $C_{12} = SSMM(A_{11}, B_{12}) + SSMM(A_{12}, B_{22})$
  $C_{21} = SSMM(A_{21}, B_{11}) + SSMM(A_{22}, B_{21})$
  $C_{22} = SSMM(A_{21}, B_{12}) + SSMM(A_{22}, B_{22})$
end if
Return C $= \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$

$A_{11} B_{11}$ + $A_{12} B_{21}$

Question 2. What recursion applies to the above algorithm when $n > 1$?

A $T(n) = \cancel{4}T(n/2) + O(n^2)$

B $T(n) = 8T(\cancel{n/4}) + O(n^2)$

C $T(n) = 8T(n/2) + \cancel{O(n)}$

$\boxed{\text{D}}$ $T(n) = 8T(n/2) + O(n^2)$

$$T(n) = a\, T(n/b) + f(n)$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

$$C_{11} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$C_{11} = A_{11} B_{11} + A_{12} B_{21}$$

$g(n) = n^{\log_b a} = n^{\log_2 8}$
$= n^3$

$f(n) = n^2$

By Master Thm, $T(n) = \Theta(g(n)) = \Theta(n^3)$

5.2 Strassen's Algorithm

Strassen's algorithm introduces a new way to combine matrix multiplications and additions to obtain the matrix C.

Step 1: Partition A and B as before.

Step 2: Compute S matrices

$$
\begin{aligned}
S_1 &= B_{12} - B_{22} & S_2 &= A_{11} + A_{12} \\
S_3 &= A_{21} + A_{22} & S_4 &= B_{21} - B_{11} \\
S_5 &= A_{11} + A_{22} & S_6 &= B_{11} + B_{22} \\
S_7 &= A_{12} - A_{22} & S_8 &= B_{21} + B_{22} \\
S_9 &= A_{11} - A_{21} & S_{10} &= B_{11} + B_{12}
\end{aligned}
$$

Runtime: we add (or subtract) 2 matrices of size $n/2 \times n/2$, 10 times.

Step 3: Compute P matrices

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1 = A_{11} \cdot B_{12} - A_{11} \cdot B_{22} \\
P_2 &= S_2 B_{22} \\
P_3 &= S_3 B_{11} \\
P_4 &= A_{22} S_4 \\
P_5 &= S_5 S_6 \\
P_6 &= S_7 S_8 \\
P_7 &= S_9 S_{10}
\end{aligned}
$$

Runtime: we recursively call the matrix-matrix multiplication function for 7 matrices of size $n/2 \times n/2$.

Step 4: Combine Using the P matrices, we can show that

$$
\begin{aligned}
C_{11} &= P_5 + P_4 - P_2 + P_6 \\
C_{12} &= P_1 + P_2 \\
C_{21} &= P_3 + P_4 \\
C_{22} &= P_5 + P_1 - P_3 - P_7
\end{aligned}
$$

5.3 Analysis of Strassen's Method

Question 3. Strassen's algorithm satisfies which recurrence relation for $n > 1$?

A $T(n) = 8T(n/2) + O(n^2)$

B $T(n) = 23T(n/2)$

C $T(n) = 7T(n/2) + O(n^2)$

D $T(n) = 10T(n/2) + O(n^2)$

E $T(n) = 17T(n/2) + O(1)$