CSCE 411: Design and Analysis of Algorithms

Lecture 2: Divide and Conquer, Part II

Date: January 16, 2025 *Nate Veldt, updated by Samson Zhou*

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, Jan 18. HW 1 and intro video due Fri, Jan 24

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 $\Theta(n^2)$

2 Recurrence Analysis for Divide and Conquer

Runtimes for divide and conquer algorithms can be described in terms of a __________________

relation, which ____

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

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

3 Three methods for solving recurrences

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

- **Recursion tree**:
- **Substitution method**:
- **Master theorem**:

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:*

____

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

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

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* ____________________________

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:

4.2 What to know about the master method?

Proof idea:

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

What's more important:

4.3 Examples

Apply the master theorem to the following recurrences:

$$T(n) = 9T(n/3) + n \tag{2}$$

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

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

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

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

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

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

A $T(n) = 4T(n/2) + O(n^2)$

B $T(n) = 8T(n/4) + O(n^2)$

C $T(n) = 8T(n/2) + O(n)$

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

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

$$C_{11} = P_5 + P_4 - P_2 + P_6$$
$$C_{12} = P_1 + P_2$$
$$C_{21} = P_3 + P_4$$
$$C_{22} = P_5 + P_1 - P_3 - P_7$$

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