Math 416

Worksheet 6. The D&C Master Theorem

The Divide & Conquer method takes a problem of size n, breaks it up into a subproblems each of size n/b, and combines the answers to those in $f(n)$ time.[1]

Problem 1. What are a and b and $f(n)$ in the case of Strassen's algorithm for matrix multiplication?

Problem 2. Explain why, in the general case, the worst-case running time $T(n)$ of the D&C algorithm described above obeys the recurrence

$$T(n) = aT(n/b) + f(n). \tag{R}$$

There are essentially three cases to deal with:

(a) The time to solve subproblems dominates the time to assemble their solutions.
(b) The time to solve subproblems is approximately the same as the time to assemble their solutions.
(c) The assembly time dominates the subproblem-solving time.

Problem 3. Consider case (a). We might think that the assembly time is effectively 0 compared to solving time. Assuming that $f(n)$ is actually 0 (this is an oversimplification!), show that $T(n) \in \Theta(n^{\log_b a})$ solves the main recurrence (R) in this case.

Use this to make a guess about the running time of Strassen's Algorithm.

Theorem (The Divide-and-Conquer Master Theorem)**.** Suppose that $a \geq 1$ and $b > 1$ are constants and that $f\colon \mathbb{R} \to \mathbb{R}$ is an increasing, nonnegative function. Suppose further that T is a function satisfying a recurrence

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

(where $T(n/b)$ can be taken to mean either $T(\lceil n/b \rceil)$ or $T(\lfloor n/b \rfloor)$).

(a) If $f(n)$ is $O(n^\gamma)$ for some $\gamma < \log_b a$, then $T(n)$ is $\Theta(n^{\log_b a})$.
(Recursion dominates)
(b) If $f(n)$ is $\Theta(n^{\log_b a})$, then $T(n)$ is $\Theta(n^{\log_b a} \log n)$.
(c) If $f(n)$ is $\Omega(n^\gamma)$ for some $\gamma > \log_b a$ AND there is a $c < 1$ for which $af(n/b) \leq cf(n)$ for all sufficiently large n, then $T(n)$ is $\Theta(f(n))$.
(Assembly dominates)

Problem 4. What does the Master Theorem say about the running time of Mergesort? Which case does it fall under?

Problem 5. Case (c) deals with the case where assembling solutions to subproblems (and subsubproblems...) is much more costly than solving the subproblems. Explain why (in Case (c)) we would expect the running time to behave asymptotically like the function

$$\sum_{i=0}^{\log_b n} a^i f(n/b^i) = f(n) + af(n/b) + a^2 f(n/b^2) + \cdots + n^{\log_b a} f(1).$$

(*Hint:* Add up all the assembly costs.)

Do you see what might go wrong without this extra condition about the constant c in Case (c)?

[1] Typically $f(n)$ is $O(n^d)$ for some d.

Problem 6. Prove that the recurrence $T(n) = aT(n/b) + f(n)$ has explicit solution

$$T(n) = n^{\log_b a} T(1) + \sum_{i=0}^{\log_b n} a^i f(n/b^i).$$

Problem 7. Prove that in Case (c) $T(n)$ is $O(f(n))$. Your proof will need to use the additional condition about the $c < 1$.

(*Hint:* Start by explaining why $a^i f(n/b^i) < c^i f(n)$. Use Problem 6 and the geometric series formula!)

Problem 8. Use recursion trees to solve each of the following recurrences

(a) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$
(b) $T(n) = 2T(n/2) + n\log(n)$
(c) $T(n) = 2T(n/2) + n/\log(n)$.