We again will be dealing with divide-and-conquer algorithms.

When we left off last time, we were discussing two divide-and-conquer algorithms for multiplying $n \times n$ matrices:

- Straightforward approach with 8 recursive multiplications:
  - $T(n) = 8T(n/2) + \Theta(n^2)$

- Strassen's approach with 7 recursive multiplications:
  - $T(n) = 7T(n/2) + \Theta(n^2)$

The recursion trees of these algorithms are more complicated to analyze than in other examples we looked at last time.

Another approach which can help us to analyze the running time of a divide-and-conquer algorithm is known as the **master method**:

Suppose the running time of an algorithm is of the form $T(n) = aT(n/b) + f(n)$ where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

Intuitively, we will compare the asymptotic growths of the functions $f(n)$ and $n^{\log_b a}$, and in many cases we will be able to directly determine the asymptotic growth of $T(n)$.

$f(n)$: time to solve one subproblem of size n.

$n^{\log_b a}$: number of subproblems.

Master Theorem

$T(n) = aT(n/b) + f(n)$ has the following asymptotic bounds.

1) If $f(n) \in O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then $T(n) \in \Theta(n^{\log_b a})$

2) If $f(n) \in \Theta(n^{\log_b a})$, then $T(n) \in \Theta(n^{\log_b a} \cdot \log n)$

3) If $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$ for a $c < 1$, then $T(n) \in \Theta(f(n))$

regularity condition

Ex: $T(n) = 4T(\frac{n}{2}) + n$

$a = 4,\ b = 2,\ f(n) = n^1,\ n^{\log_b a} = n^{\log_2 4} = n^2$

$n^1 \in O(n^{2-\varepsilon})$ for $\varepsilon = 1$ any $\varepsilon \in (0, 1]$

Case 1 holds $\Rightarrow T(n) \in \Theta(n^2)$.

Merge Sort: $T(n) = 2T(n/2) + O(n)$

- We showed via recursion tree and proof by induction that the running time of merge sort is $\Theta(n \log n)$.
- By master method:

$a=2,\ b=2,\ n^{\log_b a} = n^1,\ f(n) = n^1$

Case 2 holds $\Rightarrow T(n) \in \Theta(n \log n)$

Computing nth power of a: $T(n) = T(n/2) + O(1)$

- We showed via recursion tree that the running time of merge sort is $\Theta(\log n)$.
- By master method:

$a=1,\ b=2,\ n^{\log_b a} = n^0$

$f(n) = n^0$

Case 2 $\Rightarrow T(n) \in \Theta(\log n)$

Multiplying Matrices:

- Straightforward approach with 8 recursive multiplications:
  - $T(n) = 8T(n/2) + \Theta(n^2)$

$a=8$, $b=2$, $n^{\log_2 8} = n^3$

$f(n) \in n^2$

$n^2 \in O(n^{3-\varepsilon})$ for $\varepsilon = 1$

Case 1 holds $\Rightarrow T(n) \in \Theta(n^3)$

- Strassen's approach with 7 recursive multiplications:
  - $T(n) = 7T(n/2) + \Theta(n^2)$

$a=7$, $b=2$, $n^{\log_2 7}$

$f(n) = n^2$,

$n^2 \in O(n^{\log_2 7 - \varepsilon})$ for $\varepsilon = 0.001$

Case 1 holds: $\Theta(n^{\log_2 7})$