11.3.1
Solving the recurrences for fast multiplication

Analyzing the Recurrences

1. Basic divide and conquer: $T(n) = 4T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^2)$.
2. Saving a multiplication: $T(n) = 3T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^{1+\log 1.5})$

Use recursion tree method:

1. In both cases, depth of recursion $L = \log n$.
2. Work at depth i is $4^i n/2^i$ and $3^i n/2^i$ respectively: number of children at depth i times the work at each child
3. Total work is therefore $n \sum_{i=0}^{L} 2^i$ and $n \sum_{i=0}^{L} (3/2)^i$ respectively.

Analyzing the Recurrences

1. Basic divide and conquer: $T(n) = 4T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^2)$.
2. Saving a multiplication: $T(n) = 3T(n/2) + O(n)$, $T(1) = 1$. **Claim:** $T(n) = \Theta(n^{1+\log 1.5})$

Use recursion tree method:

1. In both cases, depth of recursion $L = \log n$.
2. Work at depth i is $4^i n/2^i$ and $3^i n/2^i$ respectively: number of children at depth i times the work at each child
3. Total work is therefore $n\sum_{i=0}^{L} 2^i$ and $n\sum_{i=0}^{L}(3/2)^i$ respectively.

Analyzing the recurrence with four recursive calls

$T(n) = 4T(n/2) + O(n)$, $T(1) = 1$

Analyzing the recurrence with three recursive calls

$T(n) = 3T(n/2) + O(n)$, $T(1) = 1$

Analyzing the recurrence with two recursive calls

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

THE END

...

(for now)