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

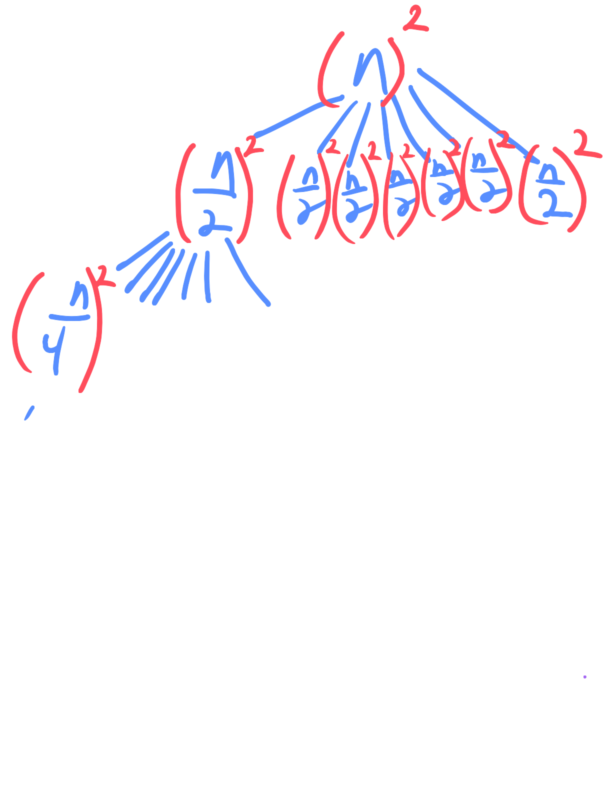
When we left off last time, we were discussing two divide-and-conquer algorithms for multiplying $n \times n$ matrices:

- 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.

$$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$$

levels	Steps
0	n^2
1	$7 \cdot \frac{n^2}{4} = \frac{7}{4} n^2$
2	$7^2 \frac{n^2}{4^2} = \left(\frac{7}{4}\right)^2 n^2$
$\vdots$	$\vdots$
	$\left(\frac{7}{4}\right)^i n^2$
	$\vdots$
$\log_2 n$	



Sum this column.

$$\sum_{i=0}^{\log_2 n} \left(\frac{7}{4}\right)^i n^2 = n^2 \underbrace{\sum_{i=0}^{\log_2 n} \left(\frac{7}{4}\right)^i}_{\text{geometric series}} = n^2 \underbrace{\left[\frac{\left(\frac{7}{4}\right)^{\log_2 n+1} - 1}{\frac{7}{4} - 1}\right]}_{o(n)} \sim o(n^3)$$

little-oh

little-oh

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 $T(n)$.

$f(n) \approx$ # of steps in one subproblem

$n^{\log_b a} \approx$ # of subproblems

Suppose $T(n) = a \cdot T(\frac{n}{b}) + f(n)$ as defined above.
Then $T(n)$ has the following asymptotic bounds.

1) If $\underbrace{f(n) \in O(n^{\log_b a - \varepsilon})}_{f(n) \text{ is smaller than } n^{\log_b a}}$ for some $\varepsilon > 0$, then $\underline{T(n) \in \Theta(n^{\log_b a})}$

2) If $\underbrace{f(n) \in \Theta(n^{\log_b a})}_{f(n) \text{ is same as } n^{\log_b a}}$, then $\underline{T(n) \in \Theta(n^{\log_b a} \cdot \log n)}$.

don't forget the log factor.

3) If $\underbrace{f(n) \in \Omega(n^{\log_b a + \varepsilon})}_{f(n) \text{ is bigger}}$ for some $\varepsilon > 0$, and if $\underbrace{a \cdot f(\frac{n}{b}) \leq c \cdot f(n)}_{\text{regularity condition}}$ for a $c < 1$, then $\underline{T(n) \in \Theta(f(n))}$.

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

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

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

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

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:

  $a=2$, $b=2$, $n^{\log_2 2} = n^1$

  $f(n) = n^1$.

  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:

  $a=1$, $b=2$, $n^{\log_2 1} = n^0$

  $f(n) = n^0$

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

Multiplying Matrices:

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

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

$f(n) = n^2$

$n^2 \in O(n^{3-\varepsilon})$ for $\varepsilon = \frac{1}{2}$. Any $\varepsilon \in (0, 1]$ works.

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 $\varepsilon = 0.8$

Case 1 holds $\Rightarrow T(n) \in \Theta(n^{\log_2 7})$

$$T(n) = 4T\left(\frac{n}{2}\right) + n^3$$

$a = 4$, $b = 2$, $n^{\log_2 4} = n^2$

$f(n) = n^3$

$n^3 \in \Omega(n^{2+\varepsilon})$ for $\varepsilon = \frac{1}{2}$.

Regularity Condition: $af\left(\frac{n}{b}\right) \leq c \cdot f(n)$ for $c < 1$.

strict.

$$4\left(\frac{n}{2}\right)^3 = 4\frac{n^3}{8} = \frac{1}{2}n^3$$

c

Case 3 holds.

$T(n) \in \Theta(n^3)$.

$T(n) = 2T(\frac{n}{2}) + n\log n.$

$a=2,\ b=2,\ n^{\log_2 2} = n^1.$

$f(n) = n\log n$

$n\log n \in \Omega(n^{1+\varepsilon})$ for $\varepsilon > 0$. Does an ε exist?

Note: $n^{1+\varepsilon} = n^1 \cdot n^\varepsilon$. $\Rightarrow \cancel{n}\log n$ vs $\cancel{n}\cdot n^\varepsilon \Rightarrow \underbrace{\log n}_{\log}$ vs $\underbrace{n^\varepsilon}_{\text{polynomial}}$

No ε exists! Master Theorem does not apply.

$T(n) = 2T(\frac{n}{2}) + n^2\log n$

$a=2,\ b=2,\ n^{\log_2 2} = n^1$

$f(n) = n^2 \log n$

$n^2\log n \in \Omega(n^{1+\varepsilon})$ for $\varepsilon > 0$. Does an ε exist?

$\cancel{n}\cdot n^1 \cdot \log n$ vs $\cancel{n}\cdot n^\varepsilon \rightarrow n^1\log n$ vs n^ε. we want $n^1\log n$ to be bigger than n^ε. What ε can I pick? $\varepsilon \in (0, 1]$

$\varepsilon = 1$ works.

$T(n) = 2T(\frac{n}{2}) + \frac{n^2}{\log n}$

$n^{\log_b a} = n^1$

$f(n) = \frac{n^2}{\log n}$

$\frac{n^2}{\log n} \in \Omega(n^{1+\varepsilon})$ Is there an $\varepsilon > 0$?

$\frac{n^{\cancel{2}}}{\log n}$ vs $\cancel{n}\cdot n^\varepsilon = \frac{n}{\log n}$ vs n^ε

We want ε so that $\frac{n}{\log n}$ is bigger than n^ε.

We can pick any $\varepsilon \in (0, 1)$.