# CS4800: Algorithms & Data
# Jonathan Ullman

HW1 Due @ 4:59pm!

HW2 Out shortly after class

Lecture 4:
- Recurrences: Master Theorem
- Divide and Conquer: Median

Jan 19, 2018

## Recap

Example of a divide-and-conquer algorithm: Mergesort

Analyzed its running time:

1. guess and check by induction
2. recursion tree

## Today:

- Review tree method
- "Master Theorem" (recipe for solving most recurrences that come up in D-and-C algorithms)
- Another Example: Finding the median of a list of numbers

**Tuesday:** Wrap up D-and-C w/ more examples

# Recurrences

- Mergesort:
  - $T(n) = 2T(n/2) + Cn$
  - $T(n) = \Theta(n \log n)$

split input into two pieces of size $\frac{n}{2}$ and use $O(n)$ work to combine

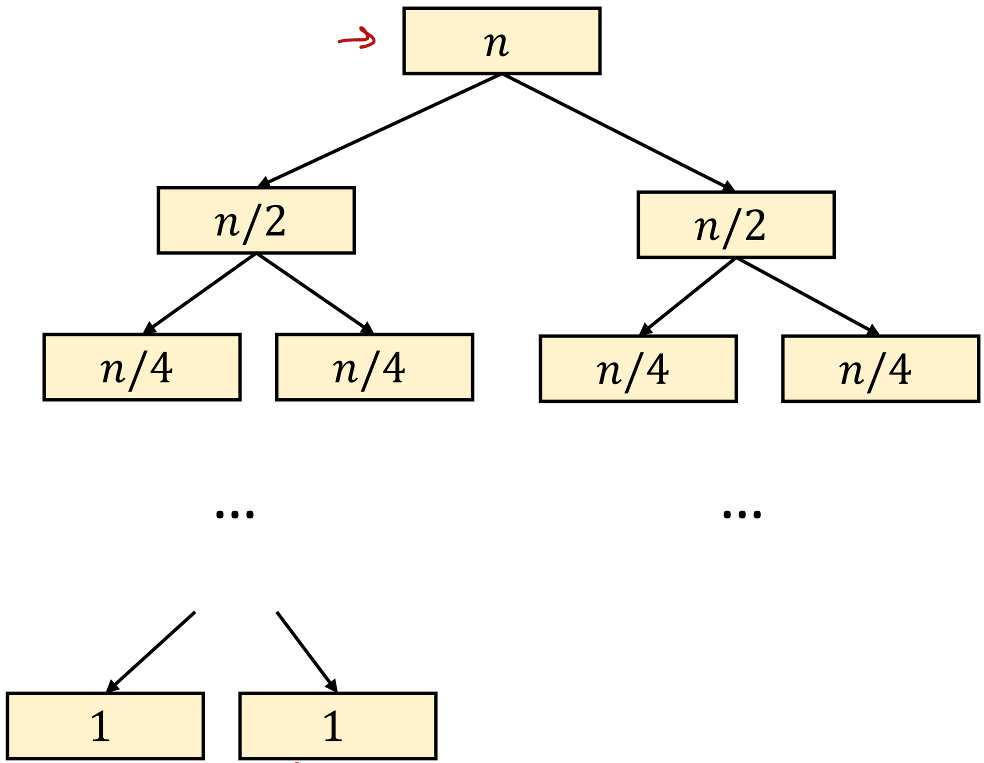
- Karatsuba's Algorithm:
  - $T(n) = 3T(n/2) + Cn$
  - $T(n) = \Theta(n^{\log_2 3})$

split input into three pieces of size $\frac{n}{2}$ and use $O(n)$ work to combine.

- How would we arrive at these answers?

# Recursion Tree

- $T(n) = 2T(n/2) + \boxed{Cn}$
- $T(1) = C$



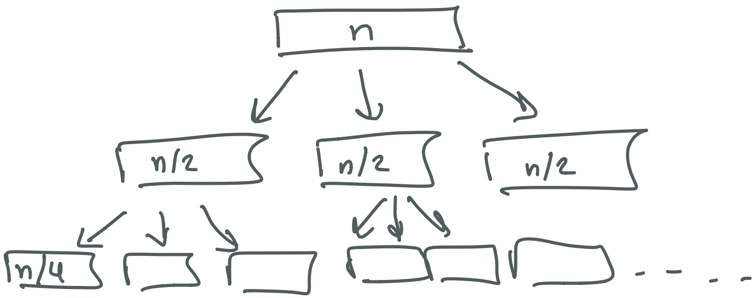

| Level | Work at level i |
|---|---|
| 0 | $Cn$ |
| 1 | $\boxed{2} \cdot \left(\frac{Cn}{\boxed{2}}\right) = Cn$ |
| 2 | $4 \cdot \left(\frac{Cn}{4}\right) = Cn$ |
| $i$ | $\boxed{2^i} \cdot \left(\frac{Cn}{\boxed{2^i}}\right) = Cn$ |
| $\ell =$ $\log_2 n$ | $2^{\log_2 n} \cdot C = Cn$ |

Total Work: $\sum_{i=0}^{\ell} Cn = (\ell+1) \cdot Cn = \Theta(n \log n)$

# Recursion Tree

- $T(n) = 3T(n/2) + Cn$
- $T(1) = C$

| level | | work |
|---|---|---|
| 0 | | $C \cdot n = O(n)$ |
| 1 | | $3 \times \left(\frac{Cn}{2}\right) = \left(\frac{3}{2}\right) \cdot Cn$ |
| 2 | | $9 \times \left(\frac{Cn}{4}\right) = \left(\frac{3}{2}\right)^2 Cn$ |
| $i$ | $3^i$ recursive calls of size $\frac{n}{2^i}$ | $3^i \times \left(\frac{Cn}{2^i}\right) = \left(\frac{3}{2}\right)^i \cdot Cn$ |
| $\vdots$ | | |
| $l = \log_2(n)$ | $3^{\log_2 n}$ recursive calls of size 1 | $3^{\log_2 n} \times C = \Omega\left(n^{\log_2 3}\right)$ |




$$T(n) = \sum_{i=0}^{l} \left(\frac{3}{2}\right)^i \times Cn = Cn \times \sum_{i=0}^{l} \left(\frac{3}{2}\right)^i = Cn \times O\left(\left(\frac{3}{2}\right)^l\right)$$

$$= Cn \times O\left(\left(\frac{3}{2}\right)^{\log_2(n)}\right)$$

# Geometric Series

- Series $S = \sum_{i=0}^{\ell} r^i$

$$S \quad = 1 + r + r^2 + \cdots + r^\ell$$

$$rS = \qquad r + r^2 + \cdots + r^\ell + r^{\ell+1}$$

- Solution $\boxed{S = \frac{1-r^{\ell+1}}{1-r}} = \frac{r^{\ell+1}-1}{r-1}$

Case ①: $r < 1$

$S \leq \frac{1}{1-r} = O(1)$

Case ②: $r = 1$

$S = (\ell+1) \cdot r = O(\ell)$

Case ③: $r > 1$

$S \geq r^\ell$

$S \leq \frac{r^{\ell+1}}{1-r}$ $\quad S = \Theta(r^\ell)$

# Ask the Audience!

- Solve using the recursion tree method:
  - $T(n) = \overset{a}{8} \cdot T\left(\frac{n}{\underset{b}{4}}\right) + Cn$
- $T(n) = \Theta(\ldots)$?

$= \Theta(n^{3/2})$

$= \Theta(n^{\log_4(8)}) = \Theta(n^{\log_b(a)})$

Answers
- $2^n \times C$
- $\boxed{n^{3/2}}$
- $2^{\log_8(n)} = n^{1/3}$
- $n^2$

# Recursion Tree

- $T(n) = 8T(n/4) + Cn$
- $T(1) = C$

level | | work
--- | --- | ---
$i$ | $8^i$ | $8^i \times \left(\frac{Cn}{4^i}\right) = \left(\frac{8}{4}\right)^i \times Cn$

bottom level $= \log_4(n)$

$$T(n) = \sum_{i=0}^{\log_4(n)} \left(\frac{8}{4}\right)^i \cdot Cn = Cn \times \Theta\left(\left(\frac{8}{4}\right)^{\log_4(n)}\right)$$

$$= \cancel{Cn} \times \Theta\left(\frac{8^{\log_4(n)}}{\cancel{4^{\log_4(n)}}}\right)$$

$$= \Theta\left(8^{\log_4(n)}\right) = \Theta\left(n^{\log_4(8)}\right)$$

$$= \Theta\left(n^{1.5}\right)$$

# Recurrences

- Mergesort:
  - $T(n) = 2T(n/2) + Cn$
  - $T(n) = \Theta(n \log n)$

- Karatsuba's Algorithm:
  - $T(n) = 3T(n/2) + Cn$
  - $T(n) = \Theta(n^{\log_2 3})$

- Just Now:
  - $T(n) = 8T(n/4) + Cn$
  - $T(n) = \Theta(n^{1.5})$

# The "Master Theorem"

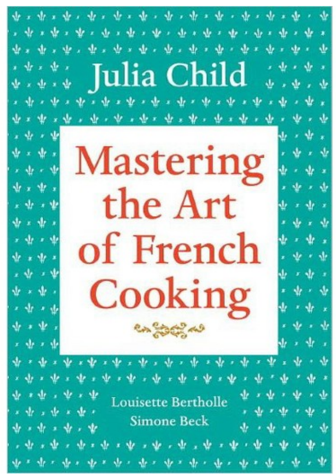
- Let's solve *all* recurrences of the form
  - $T(n) = a \cdot T(n/b) + n^c$

Important:

a, b, c are constants

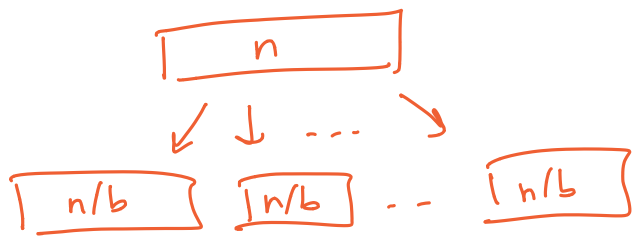
e.g. $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + O(n)$

Split into a pieces of size $\frac{n}{b}$ and use $O(n^c)$ work to combine.



# The "Master Theorem"

- Let's solve *all* recurrences of the form
  - $T(n) = a \cdot T(n/b) + n^c$




$n^c$

$a \times \left(\frac{n}{b}\right)^c$

$= \frac{a}{b^c} \times n^c$

- if $\frac{a}{b^c} > 1$ the most work is at the bottom
- if $\frac{a}{b^c} < 1$ the most work is done at the top
- if $\frac{a}{b^c} = 1$ then every level does the same work

# Recursion Tree

- $T(n) = aT(n/b) + n^c$
- $\left(\frac{a}{b^c}\right) > 1$

| level | | work |
|---|---|---|
| 0 | $n$ | $n^c$ |
| 1 | $n/b$ - - - $n/b$ | $a \times \left(\frac{n}{b}\right)^c = \left(\frac{a}{b^c}\right) * n^c$ |
| $i$ | $a^i$ copies of size $\frac{n}{b^i}$ | $\left(\frac{a}{b^c}\right)^i \times n^c$ |

bottom level is $\ell = \log_b(n)$

$$T(n) = \sum_{i=0}^{\log_b(n)} \left(\frac{a}{b^c}\right)^i \times n^c = n^c \times \Theta\left(\left(\frac{a}{b^c}\right)^{\log_b n}\right)$$

$$= \cancel{n^c} \times \Theta\left(\frac{a^{\log_b n}}{\cancel{\left(b^{\log_b n}\right)^c}}\right) = \Theta\left(a^{\log_b n}\right)$$

$$= \Theta\left(n^{\log_b a}\right)$$

# Recursion Tree

- $T(n) = aT(n/b) + n^c$
- $\left(\frac{a}{b^c}\right) = 1$

| level | | work |
|---|---|---|
| 0 | $\boxed{n}$ | $n^c$ |
| 1 | $\boxed{n/b}$ - - - $\boxed{n/b}$ | $a \times \left(\frac{n}{b}\right)^c = \frac{a}{b^c} \times n^c$ |
| $i$ | $a^i$ copies of size $\frac{n}{b^i}$ | $a^i \times \left(\frac{n}{b^i}\right)^c = \left(\frac{a}{b^c}\right)^i \times n^c = n^c$ |

bottom level is $\ell = \log_b n$

$$T(n) = n^c \times \sum_{i=0}^{\log_b n} 1^i = n^c \times O(\log_b n)$$

$$= \Theta(n^c \log n)$$

# Recursion Tree

- $T(n) = aT(n/b) + n^c$
- $\left(\frac{a}{b^c}\right) < 1$

0 $\quad \boxed{n} \quad n^c$

1 $\quad \boxed{n/b} \; \ldots \; \boxed{n/b} \quad a \times \left(\frac{n}{b}\right)^c = \left(\frac{a}{b^c}\right) n^c$

$i$ $\quad a^i$ copies of size $\frac{n}{b^i} \quad a^i \times \left(\frac{n}{b^i}\right)^c = \left(\frac{a}{b^c}\right)^i \times n^c$

$l = \log_b n$

$$T(n) = n^c \times \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i$$

$$\leq n^c \times \sum_{i=0}^{\infty} \left(\frac{a}{b^c}\right)^i = n^c \times \frac{1}{1 - \frac{a}{b^c}}$$

$$= \Theta(n^c)$$

# The "Master Theorem"

- Let's solve *all* recurrences of the form
  - $T(n) = a \cdot T(n/b) + n^c$
- Three cases:
  - $\left(\frac{a}{b^c}\right) > 1$: $T(n) = \Theta\left(n^{\log_b a}\right)$ e.g. Karatsuba
  - $\left(\frac{a}{b^c}\right) = 1$: $T(n) = \Theta(n^c \log n)$ e.g. mergesort
  - $\left(\frac{a}{b^c}\right) < 1$: $T(n) = \Theta(n^c)$ later

# The "Master Theorem"

$T(n) = 3 \times T\left(\frac{n}{2}\right) + n \log n$

- *Even More General*: All recurrences of the form $T(n) = a \cdot T(n/b) + f(n)$

$f(n) = n \log n$

- Three cases:
  - $f(n) = O\left(n^{(\log_b a) - \varepsilon}\right)$ → for some $\varepsilon > 0$
    - $T(n) = \Theta\left(n^{\log_b a}\right)$

e.g. $3 \times T\left(\frac{n}{2}\right) + n^{1.5}$ $\quad \log_2(3) \approx 1.59$

  - $f(n) = \Theta\left(n^{\log_b a}\right)$
    - $T(n) = \Theta\left(n^{\log_b a} \log n\right)$
  - $f(n) = \Omega\left(n^{(\log_b a) + \varepsilon}\right)$ & $af\left(\frac{n}{b}\right) < Cf(n)$ for $C < 1$
    - $T(n) = \Theta\left(f(n)\right)$

# Ask the Audience!

$T(n) = a\,T\left(\frac{n}{5}\right) + n^c$

- Use the Master Theorem to Solve:
  - $T(n) = 16 \cdot T\left(\frac{n}{4}\right) + n^2$

    $a = 16$, $b = 4$, $c = 2$ $\quad \left(\frac{a}{b^c}\right) = 1 \quad T(n) = \Theta(n^2 \log n)$

  - $T(n) = 21 \cdot T\left(\frac{n}{5}\right) + n^2$

    $a = 21$, $b = 5$, $c = 2$ $\quad \left(\frac{a}{b^c}\right) = \frac{21}{25} < 1 \quad T(n) = \Theta(n^2)$

  - $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 1$

    $a = 2$, $b = 2$, $c = 0$ $\quad \left(\frac{a}{b^c}\right) = \left(\frac{2}{2^0}\right) = 2 > 1 \quad T(n) = \Theta(n)$

  - $T(n) = \underbrace{5 \cdot T\left(\frac{n}{3}\right)}_{\text{work done at the bot.}} + \underbrace{n \log_2 n}_{f(n)}$

    $\Theta(n^{\log_3 5}) \quad \log_3 5 > 1$

    $f(n) = O\left(n^{(\log_3 5) - .01}\right)$

    $T(n) = \Theta\left(n^{\log_3 5}\right)$

# Switching Gears: Selection (Median)

# Selection

- Given an array of numbers $A[1, \ldots, n]$, how quickly can I find the:
  - Smallest number? $= \Theta(n)$
  - Second smallest? $= \Theta(n)$
  - Eighth smallest? $= \Theta(n)$
  - $k$-th smallest? $= O(kn) \longrightarrow O(n \log n)$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 11 | 3 | 42 | 28 | 17 | 8 | 2 | 15 | $A$ |

# Selection

- Selecting the $k$-th smallest number is no harder than sorting. Takes $\Theta(n \log n)$ time.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 11 | 3 | 42 | 28 | 17 | 8 | 2 | 15 | $A$ |

$\Downarrow$ sort

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 8 | 11 | 15 | 17 | 28 | 42 |

# Finding the Median

- The **median** is the $\left\lceil \frac{n}{2} \right\rceil$-th smallest number

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 11 | 3 | 42 | 28 | 17 | 8 | 2 | 15 | $A$ |

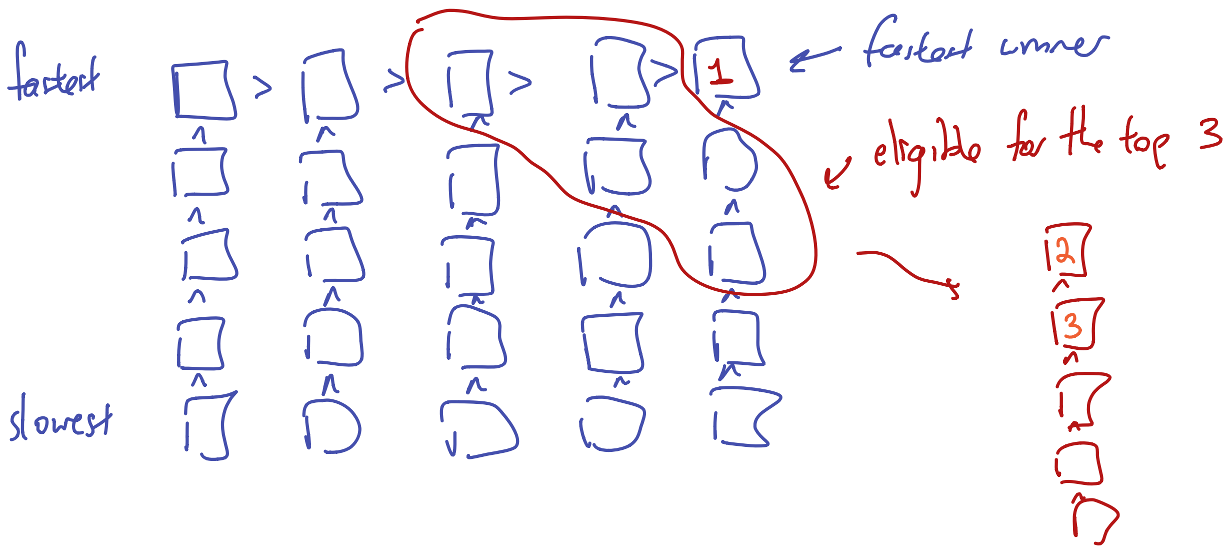
sort

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 8 | 11 | 15 | 17 | 28 | 42 |

- Today: finding the **median** in $O(n)$ time.

# Warmup

- You have 25 horses and want to find the 3 fastest.
- You do not know how fast they are, but you have a racetrack where you can race 5 horses at a time.
  - In: $\{1, 5, 6, 18, 22\}$ Out: $(6 \succ 5 \succ 18 \succ 22 \succ 1)$
- Problem: find the 3 fastest using the fewest races.



# Median Algorithm: Take I

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 17 | 3 | 42 | 11 | 28 | 8 | 2 | 15 | 13 | $A$ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 11 | 3 | ~~17~~ 15 | 13 | 2 | 8 | 17 | 28 | 42 |

Select$(A[1, \dots, n], k)$:

If $n = 1$: return $A[1]$

**Choose a pivot element** $p = A[1]$

**Partition around the pivot** $p$, let $A[r]$ be the pivot

If $k < r$: return $Select(A[1, \dots, r-1], k)$

Elif $k > r$: return $Select(A[r+1, \dots, n], k-r)$

Else: return $A[r]$

# Median Algorithm: Take I

k=4

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 17 | 3 | 42 | 11 | 28 | 8 | 2 | 15 | 13 | $A$ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 11 | 3 | 17 | 13 | 2 | 8 | 17 | 28 | 42 |

A[7]

| | | | | | |
|---|---|---|---|---|---|
| 11 | 3 | 17 | 13 | 2 | 8 |

# Median Algorithm: Take I

$k = 5$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $A$ |

$A[i]$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $k=4$ |

$A[i]$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | $k=3$ |

$\vdots$

recurse $k-1$ times

Total Time $\Omega(n^2)$

# Median Algorithm: Take II

- Need to find a pivot element $p$ that is in the middle of the sorted list in $O(n)$ time.

- Idea 1: Use the median of this list!

- Idea 2: Use the "median-of-medians" (m-o-m)!

Finish on tuesday