CS3000: Algorithms & Data
Jonathan Ullman

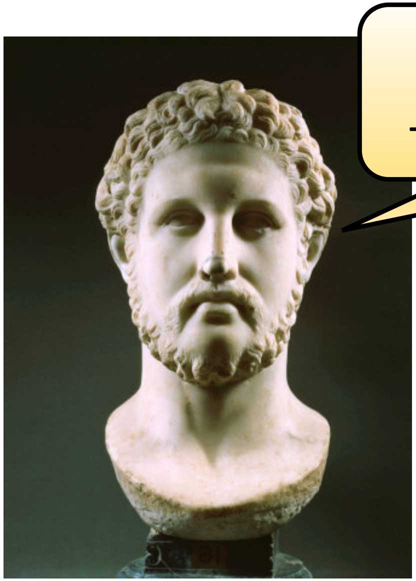
Lecture 4:

- Divide and Conquer: Karatsuba
- Solving Recurrences

Sep 18, 2018

Mergesort Wrapup

Divide and Conquer Algorithms



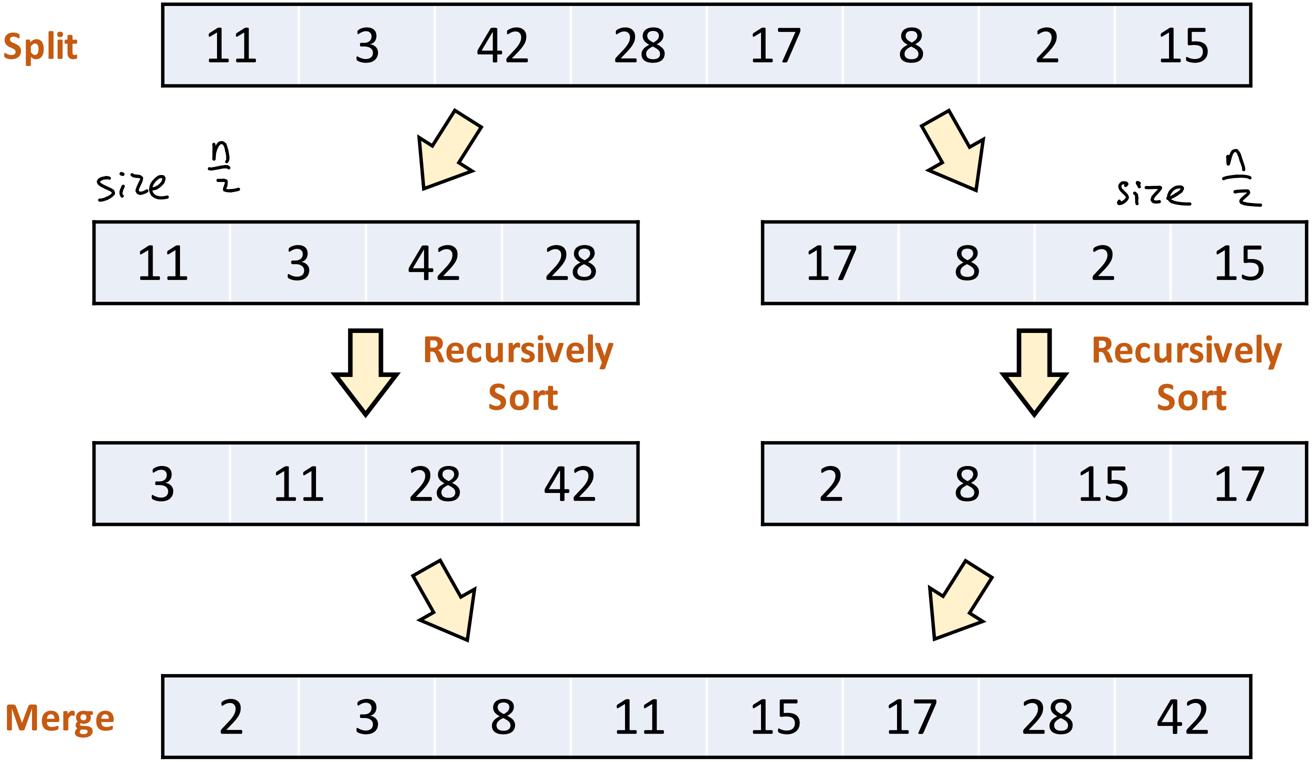
Divide et impera!
-Philip II of Macedon

- Split your problem into smaller subproblems
- Recursively solve each subproblem
- Combine the solutions to the subprobelms

Often combining is easier than solving

Mergesort

List A of n numbers



Split

11	3	42	28	17	8	2	15

size $\frac{n}{2}$

11	3	42	28

size $\frac{n}{2}$

17	8	2	15

Recursively Sort

3	11	28	42

Recursively Sort

2	8	15	17

Merge

2	3	8	11	15	17	28	42

Can merge in time $O(n)$

Running Time of Mergesort

$T(n)$ = running time of mergesort on a list of size n

1 op

O(n) ops

$2 \times T(\frac{n}{2})$

O(n) ops

```
MergeSort(A):
    If (n = 1): Return A

    Let   $m \leftarrow \lceil n/2 \rceil$
          L ← A[1:m]
          R ← A[m+1:n]

    Let L ← MergeSort(L)
    Let R ← MergeSort(R)
    Let A ← Merge(L,R)

    Return A
```

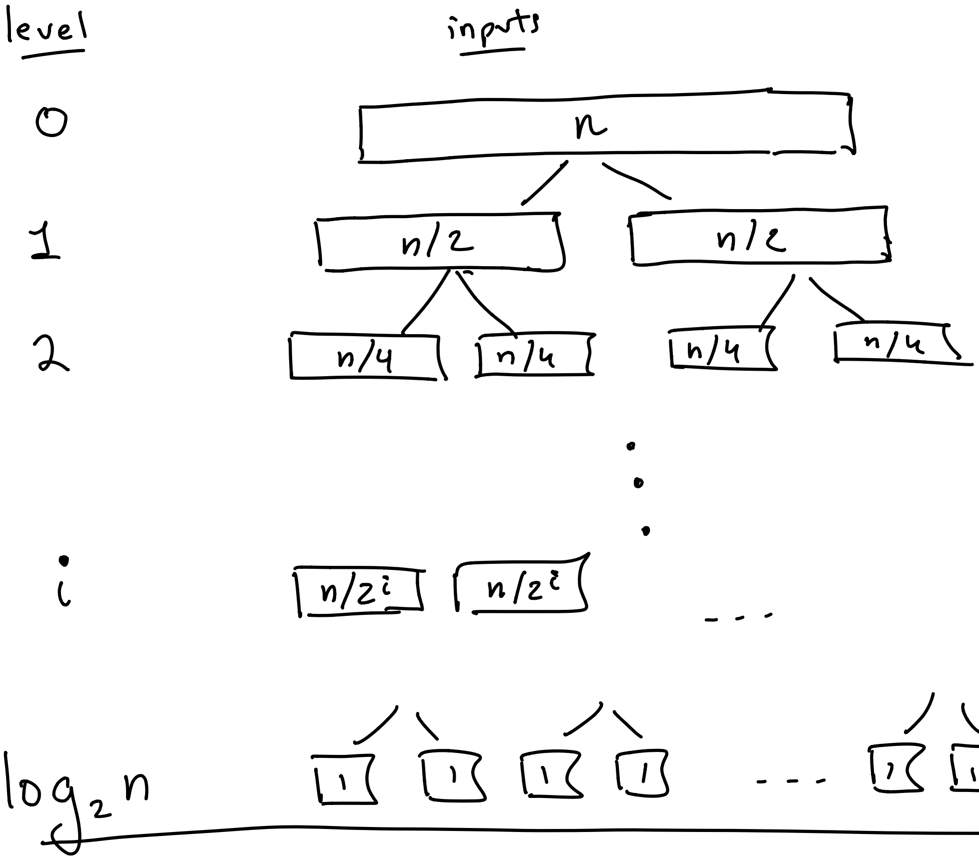
$$T(n) = 2 \times T\left(\frac{n}{2}\right) + Cn$$

Recurrence Relation

$$T(n) = \Theta(n \log n)$$

Recursion Tree

$$T(n) = 2 \cdot T(n/2) + Cn$$
$$T(1) = C$$



work excl. recursive calls

$Cn = Cn$

$2 \times \left(\frac{Cn}{2}\right) = Cn$

$4 \times \left(\frac{Cn}{4}\right) = Cn$

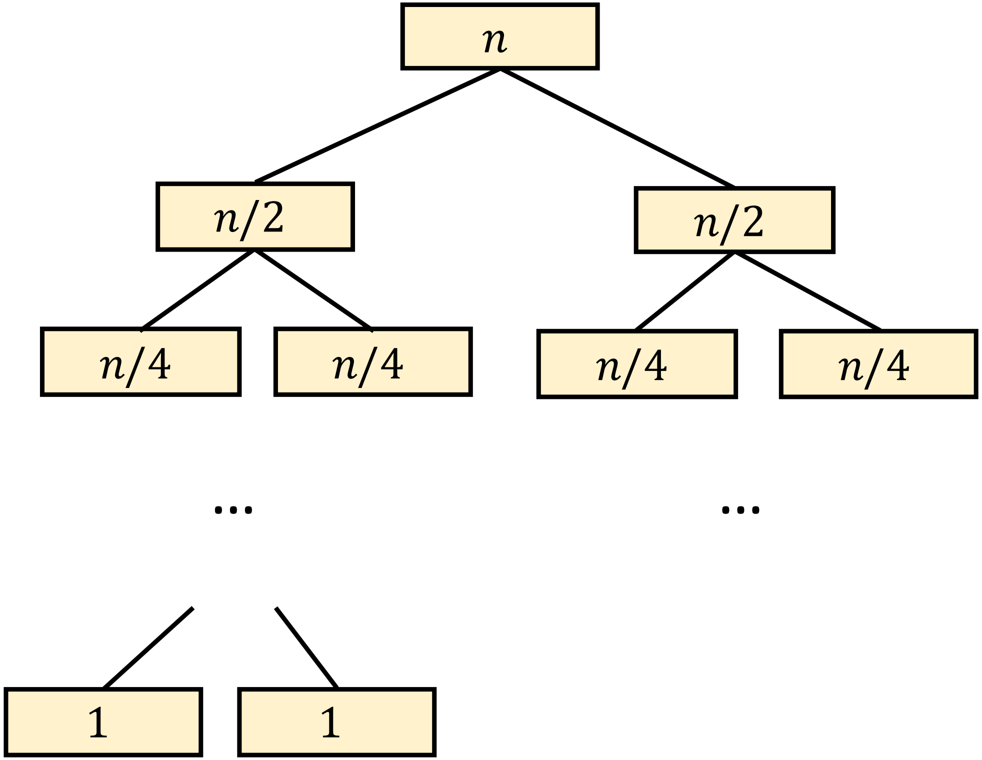
$2^i \times \left(\frac{Cn}{2^i}\right) = Cn$

$n \times C = Cn$

Total Work is $Cn(\log_2 n + 1)$

Recursion Tree

$$T(n) = 2 \cdot T(n/2) + Cn$$
$$T(1) = C$$



Level	Input Size at this Level	Work at this Level
0	n	Cn
1	$n/2$, $n/2$	$2 \cdot \left(\frac{Cn}{2}\right) = Cn$
2	$n/4$, $n/4$, $n/4$, $n/4$	$4 \cdot \left(\frac{Cn}{4}\right) = Cn$
i	… …	$2^i \cdot \left(\frac{Cn}{2^i}\right) = Cn$
$\log_2 n$	1, 1	$2^{\log_2 n} \cdot C = Cn$

Proof by Induction

$$T(n) = 2 \cdot T(n/2) + Cn$$
$$T(1) = C$$

- **Claim:** $T(n) = Cn \log_2 2n$

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

$$= 2 \cdot \left(C \cdot \frac{n}{2} \cdot \log_2 n\right) + Cn$$

$$= Cn \left(\log_2 n + 1\right)$$

$$= Cn \times \log_2 2n$$

Mergesort Summary

- Sort a list of n numbers in $O(n \log n)$ time
  - Can actually sort anything that allows comparisons
  - Any comparison based algorithm is $\Omega(n \log n)$ time
- Divide-and-conquer approach
  - Break the list into two halves, sort each one and merge
  - Key Fact: merging is easier than sorting
- Proof of correctness
  - Proof by induction
- Analysis of running time
  - Solve a recurrence

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

Integer Multiplication: Karatsuba's Algorithm

Addition

- Given n-digit numbers x, y output $x + y$

$$\begin{array}{ccccc} & 1 & 2 & 3 & 4 \\ + & 1 & 1 & 2 & 2 \\ \hline = & 2 & 3 & 5 & 6 \end{array}$$

Basic Operation: adding digits w/ carry

Running Time: $O(n)$

Multiplication

- Given n-digit numbers x, y output $x \cdot y$

$$
\begin{array}{cccccccc}
 & & & & 1 & 2 & 3 & 4 \\
 & & & \times & 1 & 1 & \textcircled{2} & \textcircled{2} \\
\cline{4-8}
 & & & & 2 & 4 & 6 & 8 \\
+ & & & 2 & 4 & 6 & 8 & 0 \\
+ & & 1 & 2 & 3 & 4 & 0 & 0 \\
+ & 1 & 2 & 3 & 4 & 0 & 0 & 0 \\
\hline
 & 1 & 3 & 8 & 4 & 5 & 4 & 8
\end{array}
$$

Running Time: $\Theta(n^2)$

Divide and Conquer Multiplication

	1	2	3	4
x	1	1	2	2

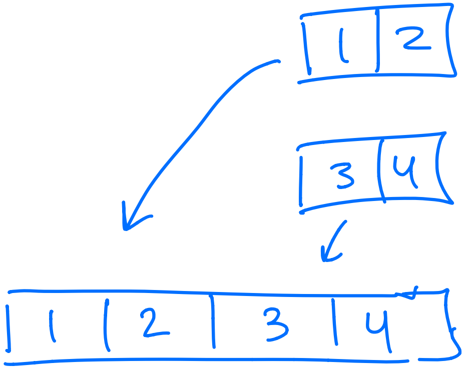
$$\boldsymbol{x} = 10^2 \cdot 12 + 34$$

$$\boldsymbol{y} = 10^2 \cdot 11 + 22$$

	$\boldsymbol{a}$	$\boldsymbol{b}$
x	$\boldsymbol{c}$	$\boldsymbol{d}$

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$

$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$



Divide and Conquer Multiplication

	1	2	3	4
x	1	1	2	2

$$x = 10^2 \cdot 12 + 34$$

$$y = 10^2 \cdot 11 + 22$$

	a	b
x	c	d

$$x = 10^{n/2}a + b$$

$$y = 10^{n/2}c + d$$

$$x \cdot y = (10^2 \cdot 12 + 34)(10^2 \cdot 11 + 22)$$

$$= 10^4 \cdot (\underline{12 \times 11}) + 10^2 \cdot (\underline{12 \times 22} + \underline{11 \times 34}) + (\underline{34 \times 22})$$

4 mults of 2-digits + 3 adds + shifts

Divide and Conquer Multiplication

	$\boldsymbol{a}$	$\boldsymbol{b}$
x	$\boldsymbol{c}$	$\boldsymbol{d}$

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$

$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$

$$\boldsymbol{x} \cdot \boldsymbol{y} = (10^{n/2}\boldsymbol{a} + \boldsymbol{b})(10^{n/2}\boldsymbol{c} + \boldsymbol{d})$$

$$= 10^{n}\underline{\boldsymbol{ac}} + \; 10^{n/2}(\underline{\boldsymbol{ad}} + \underline{\boldsymbol{bc}}) + \underline{\boldsymbol{bd}}$$

- Four $n/2$-digit mults, three n-digit adds
  - Multiplying by 10^n is “free” because it’s a shift
- Recurrence: $T(n) = 4T\left(\frac{n}{2}\right) + 3n$

Divide and Conquer Multiplication

$$T(n) = 4 \cdot T(n/2) + 3n$$
$$T(1) = 1$$

- **Claim:** $T(n) \geq n^2$

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

Too many recursive calls.

Karatsuba's Algorithm

Dont need each on its own

	a	b
x	c	d

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$

$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$

$$x \cdot y = 10^{n}\boldsymbol{ac} + \ 10^{n/2}(\boldsymbol{ad} + \boldsymbol{bc}) + \boldsymbol{bd}$$

- Key Identity
  - $(\boldsymbol{b} - \boldsymbol{a})(\boldsymbol{c} - \boldsymbol{d}) = \boldsymbol{ad} + \boldsymbol{bc} - \boldsymbol{ac} - \boldsymbol{bd}$

- Only three $n/2$-digit mults (plus some adds)!
  - ac
  - bd
  - (b-a)(c-d)

Karatsuba’s Algorithm

```
Karatsuba(x,y,n):
  If (n = 1): Return $x \cdot y$                    // Base Case

  Let $m \leftarrow \lceil n/2 \rceil$                    // Split
  Write $x = 10^m a + b$,  $y = 10^m c + d$

  Let e ← Karatsuba(a,c,m)                         // Recurse
      f ← Karatsuba(b,d,m)
      g ← Karatsuba(b-a,c-d,m)

  Return $10^{2m} e + 10^m (e + f + g) + f$         // Merge
```

Correctness of Karatsuba

- **Claim:** The algorithm **`Karatsuba`** is correct
- $\forall n$ $\forall$ x, y with n-digits Karatsuba$(x, y, n) = x \cdot y$

Proof:

Inductive Hypotheses: $H(n) = \forall x, y$ with n digits, $K(x, y, n) = x \cdot y$

Base Case: $H(1)$ is true, obviously

Correctness of Karatsuba

- **Claim:** The algorithm **`Karatsuba`** is correct

Proof Cont'd:

Inductive Step: Assume $H(1) \wedge \dots \wedge H(n)$, meaning that Karatsuba is correct for all x, y with $\le n$ digits. Want to show $H(n+1)$. Consider any x, y with $n+1$ digits.

- By definition, a, b, c, d, $b-a$, $c-d$ all have $\le n$ digits.
- Therefore, by $H(1) \wedge \dots \wedge H(n)$, $e = a \cdot b$, $f = c \cdot d$, $g = (b-a)(c-d)$.
- Therefore, $K(x, y, n+1) = 10^{2m} e + 10^{m}(e + f + g) + f$

$$= 10^{2m} a \cdot b + 10^{m}(ad + bc) + c \cdot d = x \cdot y \quad \square$$

Running Time of Karatsuba

$O(1)$ →

$O(n)$

$3 \times T(\frac{n}{2})$

```
Karatsuba(x,y,n):
    If (n = 1): Return $x \cdot y$

    Let $m \leftarrow \lceil n/2 \rceil$
    Write $x = 10^m a + b$,  $y = 10^m c + d$

    Let e ← Karatsuba(a,c,m)
        f ← Karatsuba(b,d,m)
        g ← Karatsuba(b-a,c-d,m)

    Return $10^{2m} e + 10^m (e + f + g) + f$
```

$O(n)$

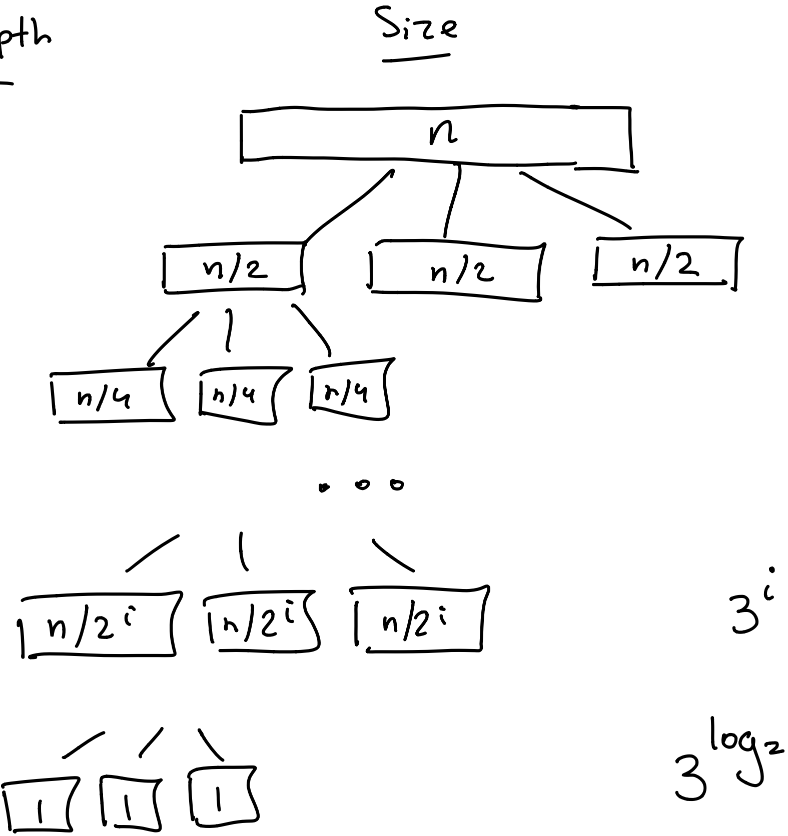
$O(n)$

Shifts (mult by 10^m) is free

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

Recursion Tree

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



Recursion Depth	Work
0	Cn
1	$3 \times \left(\frac{Cn}{2}\right) = \frac{3Cn}{2}$
2	$9 \times \left(\frac{Cn}{4}\right) = \left(\frac{3}{2}\right)^2 \cdot Cn$
⋮	
i	$3^i \times \left(\frac{Cn}{2^i}\right) = \left(\frac{3}{2}\right)^i \cdot Cn$
$\log_2 n$	$3^{\log_2 n} \times C = n^{\log_2 3} \times C$

Geometric Series

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

$r \neq 1$

Karatsuba's: $\sum_{i=0}^{\log_2 n} \left(\frac{3}{2}\right)^i \times C_n$

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

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

$$C_n \left(\frac{\left(\frac{3}{2}\right)^{(\log_2 n)+1} - 1}{\frac{1}{2}} \right)$$

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

$$= O\left(n \cdot \frac{n^{\log_2 3}}{n}\right)$$

$$= O\left(n^{\log_2 3}\right)$$

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

$r < 1$

$S = O(1)$

$r > 1$

$S = O(r^\ell)$

Karatsuba Wrapup

- Multiply n digit numbers in $O(n^{1.59})$ time
  - Improves over naïve $O(n^2)$ time algorithm
  - **Fast Fourier Transform:** multiply in $\approx O(n \log n)$ time
- Divide-and-conquer approach
  - Uses a clever algebraic trick to split
  - **Key Fact:** adding is faster than multiplying
- Prove correctness via induction
- Analyze running time via recursion tree
  - $T(n) = 3T(n/2) + Cn$

Solving Recurrences:
“The Master Theorem”

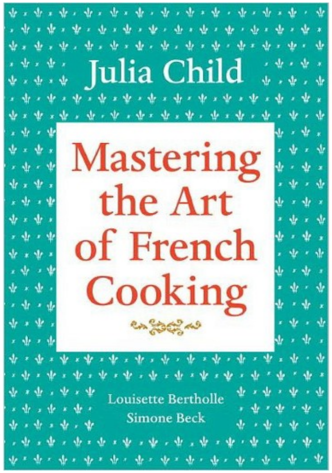
The "Master Theorem"

Mergesort: $a=2$, $b=2$, $d=1$

Karatsuba: $a=3$, $b=2$, $d=1$

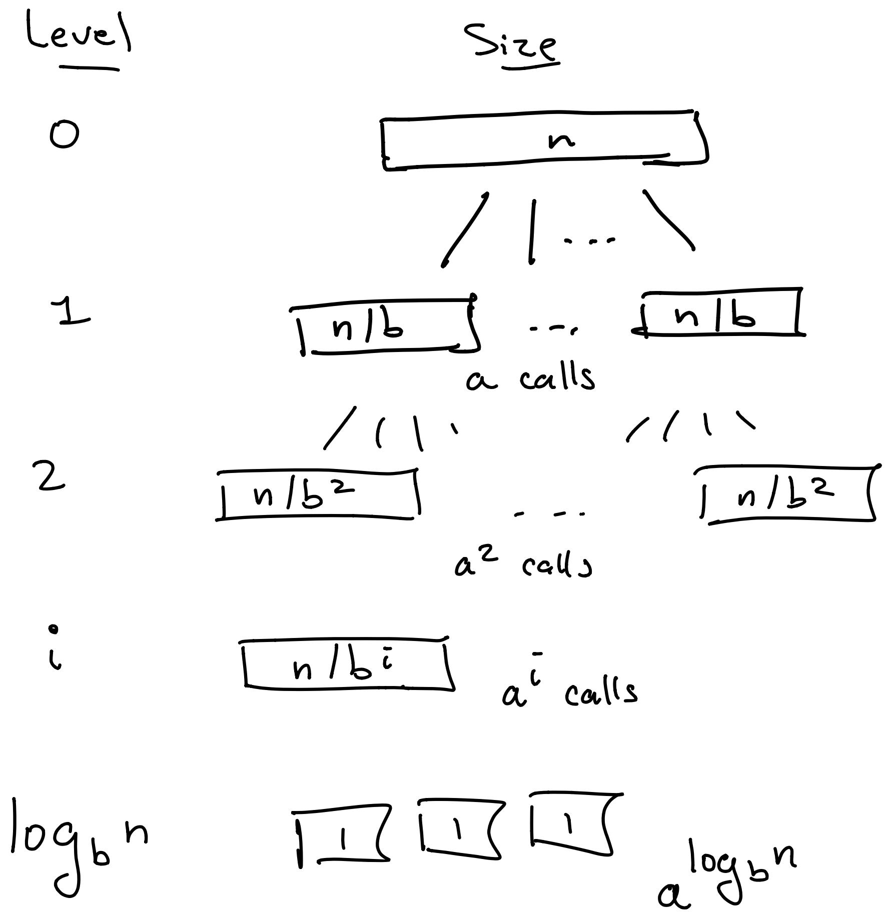
- Generic divide-and-conquer algorithm:
  - Split into $\boldsymbol{a}$ pieces of size $\frac{\boldsymbol{n}}{\boldsymbol{b}}$ and merge in time $O\left(n^{\boldsymbol{d}}\right)$
- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$

a, b, d should be <u>independent</u> of n



Recursion Tree

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



Work

n^d

$a \times \left(\frac{n}{b}\right)^d = \left(\frac{a}{b^d}\right) \cdot n^d$

$a^2 \times \left(\frac{n}{b^2}\right)^d = \left(\frac{a}{b^d}\right)^2 \cdot n^d$

$\left(\frac{a}{b^d}\right)^i \cdot n^d$

$a^{\log_b n} = \left(\frac{a}{b^d}\right)^{\log_b n} \cdot n^d$

Recursion Tree

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

Total Work:

$$S = n^d \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i$$

case 1: $\frac{a}{b^d} > 1 \implies S = O\left(a^{\log_b n}\right) = O\left(\left(b^{\log_b a}\right)^{\log_b n}\right)$

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

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

Recursion Tree

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

$$S = n^d \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i$$

Case 3: $\frac{a}{b^d} = 1 \implies S = O(n^d \log_b n)$

Recursion Tree

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

$$S = n^d \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i$$

- Case 2: $\frac{a}{b^d} < 1 \implies S = O(n^d)$

The "Master Theorem"

- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$
- Three cases:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{d}}\right) > 1 :\ T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{d}}\right) = 1 :\ T(n) = \Theta\left(n^{\boldsymbol{d}} \log n\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{d}}\right) < 1 :\ T(n) = \Theta\left(n^{\boldsymbol{d}}\right)$

Ask the Audience!

$\frac{a}{b^d} > 1 \Rightarrow T(n) = \Theta(n^{\log_b a})$

$\frac{a}{b^d} = 1 \Rightarrow T(n) = \Theta(n^d \log n)$

$\frac{a}{b^d} < 1 \Rightarrow T(n) = \Theta(n^d)$

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

    $a = 16$, $b = 4$, $d = 2$

    $\frac{16}{4^2} = 1$

    $T(n) = \Theta(n^2 \log n)$

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

    $a = 21$, $b = 5$, $d = 2$

    $\frac{21}{5^2} < 1$

    $T(n) = \Theta(n^2)$

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

    $a = 2$, $b = 2$, $d = 0$

    $\frac{2}{2^0} > 1$

    $T(n) = \Theta(n)$

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

    $a = 1$, $b = 2$, $d = 0$

    $\frac{1}{2^0} = 1$

    $T(n) = \Theta(\log n)$

The "Master Theorem"

- **Even More General:** all recurrences of the form
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + f(n)$
- Three cases:
  - $f(n) = O\left(n^{(\log_{\boldsymbol{b}} \boldsymbol{a})-\varepsilon}\right)$:
    - $T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$
  - $f(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$:
    - $T(n) = \Theta(f(n) \cdot \log n)$
  - $f(n) = \Omega\left(n^{(\log_{\boldsymbol{b}} a)+\varepsilon}\right)$ **AND** $\boldsymbol{a}f\left(\frac{n}{\boldsymbol{b}}\right) \leq Cf(n)$ for $C < 1$
    - $T(n) = \Theta\left(f(n)\right)$