CS4800: Algorithms & Data
Jonathan Ullman

Lecture 3:

- Divide and Conquer: Mergesort
- Solving Recurrences, Master Theorem

Jan ~~[illegible]~~, 2018

16

Ask the Audience!

$\Theta(\log n)$ good to remember

$O(\log_2 n)$ and $\Omega(\log_2 n)$

- $\boxed{\text{True}}$ or False?

  $\forall a, b, c > 1,\ a \cdot \log_b(c \cdot n) = \Theta(\log_2 n)$

$$\boxed{f(n) = O(g(n)) \text{ if } \exists\, n_0,\ k > 0 \text{ s.t. } \forall n \geq n_0 \quad f(n) \leq k \cdot g(n)}$$

$$a \cdot \log_b(c \cdot n) = \frac{a \cdot \log_2(c \cdot n)}{\log_2(b)} = \frac{a \log_2(n) + a \log_2(c)}{\log_2(b)}$$

$$\underbrace{\frac{a}{\log_2(b)} \cdot \log_2(n) + \frac{a \log_2(c)}{\log_2(b)}}_{f(n)}$$

$$\underbrace{\frac{2a}{\log_2(b)} \cdot \log_2(n)}_{k \cdot g(n)}$$

choose n_0 s.t.

$$\frac{2a}{\log_2(b)} \cdot \log_2(n) \geq \frac{a}{\log_2(b)} \cdot \log_2(n) + \frac{a \log_2(c)}{\log_2(b)}$$

Divide and Conquer Algorithms

James Gillray. *Plumb Pudding in Danger.* 1805

- Divide your problem into simpler *subproblems*
- Recursively solve each subproblem
- Combine the solutions to the subprobelms

subproblems

Divide and Conquer Algorithms

- Examples
  - Karatsuba's Algorithm (Multiplication)
  - Mergesort / Binary Search
  - Median
  - …
- Key Tools
  - Proof by Induction (Correctness)
  - Recurrences (Running Time) / Master Theorem

Sorting

n numbers / elements / items

11	3	42	28	17	8	2	15	A

2	3	8	11	15	17	28	42

Simple Algorithm: Insertion Sort

Time $\Omega(n^2)$

Scan to find the max

11	3	$\boxed{42}$	28	17	8	2	15	A

Place the max

11	3	15	28	17	8	2	$\boxed{42}$

finding the max in step i

takes $n-i+1$ time

$$\sum_{i=1}^{n-1} n-i+1 = \sum_{i=1}^{n-1} i = \frac{(n-1)(n)}{2} = \Omega(n^2)$$

Repeat $n-1$ times.

2	3	8	11	15	17	28	42

Simple Algorithm: Insertion Sort

L R

Scan to find the max

11	3	42	28	17	8	2	15

A

Place the max

11	3	15	28	17	8	2	42

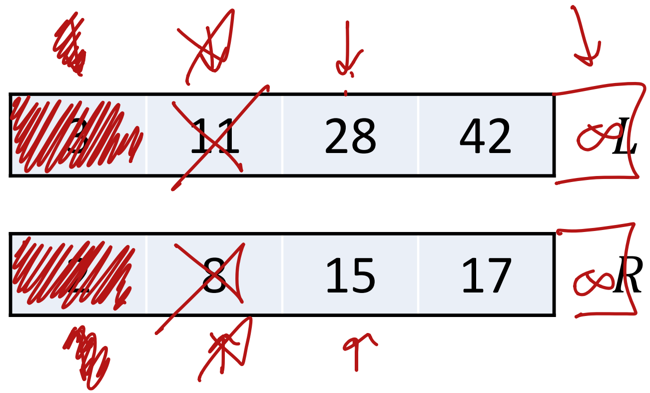
Suppose, by magic (aka recursion)
L and R were sorted.

Repeat
$n - 1$ times.

2	3	8	11	15	17	28	42

Divide and Conquer: Mergesort

- Key Idea: If L, R are sorted lists of length n, then we can merge them into a single sorted list A in $O(n)$ time.



faster than the time to sort $L \cup R$ from scratch

2	3	8	11	15	17	28	42	A

To place each elt of A, only need $O(1)$ work.

$\Rightarrow$ $O(n)$ time algorithm.

Divide and Conquer: Mergesort

front of L front of R next slot in A

Merge($L[1 \dots \ell], R[1, \dots, r]$):

Let $\boxed{i}, \boxed{j}, \boxed{k} \leftarrow 1, A[1, \dots, n]$ // A will be the output

For $k = 1, \dots, \ell + r$ // Loop over elts

If $i > \ell$: // L is empty

$A[k] = R[j], j \leftarrow j + 1$

Elif $j > r$: // R is empty

$A[k] = L[i], i \leftarrow i + 1$

Elif $L[i] < R[j]$: // L is smaller

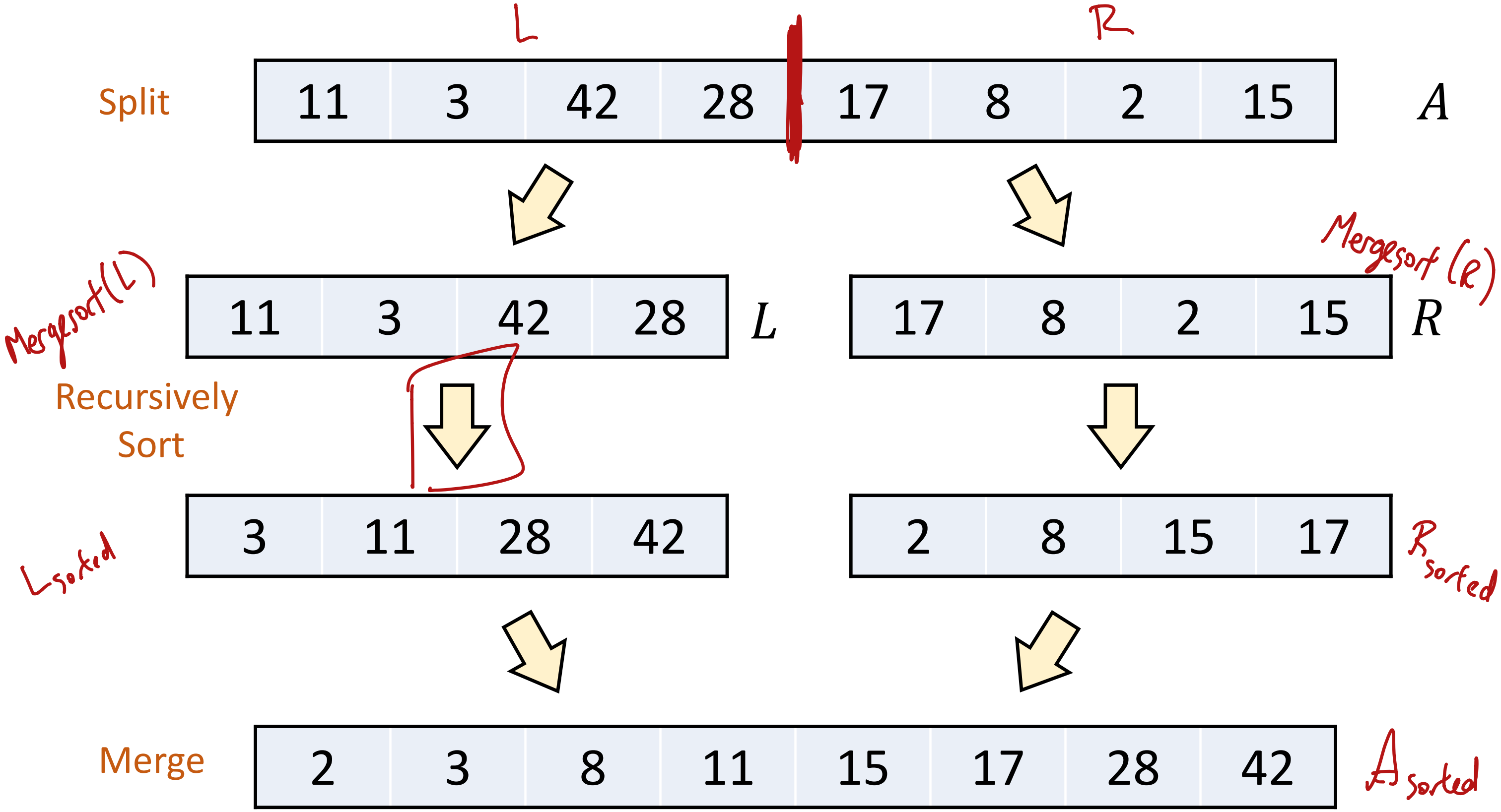
$A[k] = L[i], i \leftarrow i + 1$

Else: // R is smaller

$A[k] = R[j], j \leftarrow j + 1$

Return A

Divide and Conquer: Mergesort



Divide and Conquer: Mergesort

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

If $n = 1$, return A // Base case (Doesn't have to be $n=1$)

$\ell \leftarrow \left\lceil \frac{n}{2} \right\rceil$ // Split into two lists

$L \leftarrow A[1, \dots, \ell], R \leftarrow A[\ell + 1, n]$

$L \leftarrow Mergesort(L)$ // Sort recursively

$R \leftarrow Mergesort(R)$

$A \leftarrow Merge(L, R)$ // Merge

return A

Correctness of Mergesort

Thm: $\forall n \in \mathbb{N}$, and all A of size n,
Mergesort(A) returns A in sorted order.

```
Mergesort(A[1, ..., n]):
    If n = 1, return A
    ℓ ← ⌈n/2⌉  L ← A[1, ..., ℓ], R ← A[ℓ + 1, n]

    L ← Mergesort(L)
    R ← Mergesort(R)
    A ← Merge(L, R)
    return A
```

$|A| = k+1$

$|L|, |R| \le k$

$\rightarrow$ returns L, R in sorted order

Proof by induction on n:

$H(n)$: $\forall A$ of size n, mergesort works

$H(1) \Rightarrow H(2) \Rightarrow H(3) \Rightarrow \ldots$

Base Case $(n=1)$: Goes w/o saying

Inductive Step: Assume $H(1), \ldots, H(k)$ hold, we'll prove $H(k+1)$

$\forall A$ of size $k+1$, L, R will be returned in sorted order by IH

Since L, R are in sorted order, Merge(L,R) is A in sorted order.

[Lemma: Merge is correct.]

Running Time of Mergesort

$T(n)$ = time to sort n items

$T(n) = 2 \cdot T(\frac{n}{2}) + O(n)$

$T(1) = O(1)$

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

$O(n)$

If $n = 1$, return A // Base case

$\ell \leftarrow \left\lceil \frac{n}{2} \right\rceil$ // Split into two lists

$L \leftarrow A[1, \dots, \ell], R \leftarrow A[\ell + 1, n]$

$T(\frac{n}{2})$ $L \leftarrow Mergesort(L)$ // Sort recursively

$T(\frac{n}{2})$ $R \leftarrow Mergesort(R)$

$O(n)$ $A \leftarrow Merge(L, R)$ // Merge

return A

most expensive non-recursive part

Mergesort

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

- Guess: $\forall\, n \in \mathbb{N},\ T(n) \leq Cn(\log_2 n + 1)$ $T(n) = \Theta(n \log n)$
- Proof by induction on n:

Base Case $(n=1)$: $T(1) \leq C \cdot 1 (0 + 1) = C$ ✓

Inductive Step:

$$T(k) = 2 \cdot T\left(\frac{k}{2}\right) + C \cdot k$$

$$\leq 2 \cdot \left(C \cdot \left(\frac{k}{2}\right) \cdot \left(\log_2\left(\frac{k}{2}\right) + 1\right)\right) + Ck$$

IH

$$= C \cdot k \cdot \log_2(k) + Ck$$

$$= C \cdot k \cdot (\log_2(k) + 1)$$

$\square$

Mergesort

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

- Guess: $\forall\, n \in \mathbb{N},\ T(n) \geq Cn \log_2 n$
- Proof by induction on n:

Recurrences

- Mergesort:
  - $T(n) = 2T(n/2) + Cn$ $= 2\cdot\left(2\cdot T\left(\frac{n}{4}\right) + C\cdot\left(\frac{n}{2}\right)\right) + Cn$
  - $T(n) = \Theta(n \log n)$
- Karatsuba's Algorithm:
  - $T(n) = 3T(n/2) + Cn$
  - $T(n) = \Theta(n^{\log_2 3})$
- How would we arrive at these answers?

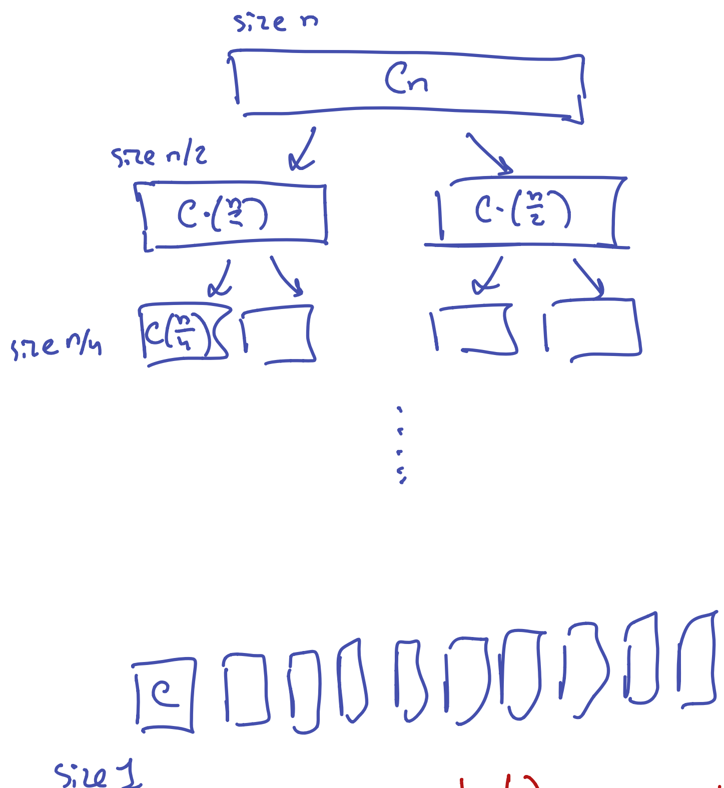
$T(n) = 11\cdot T\left(\frac{n}{4}\right) + n^3$

Visualizing recurrences

Recursion Tree

$a\cdot T(\frac{n}{b}) + c\cdot n^d$

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



level	work per call	# of calls	work
0	$C\cdot n$	1	$C\cdot n$
1	$C\cdot(\frac{n}{2})$	2	$C\cdot n$
2	$C\cdot(\frac{n}{4})$	4	$C\cdot n$
i	$C\cdot\left(\frac{n}{2^i}\right)$	2^i	$C\cdot n$
$\log_2(n)$	C	n	$C\cdot n$

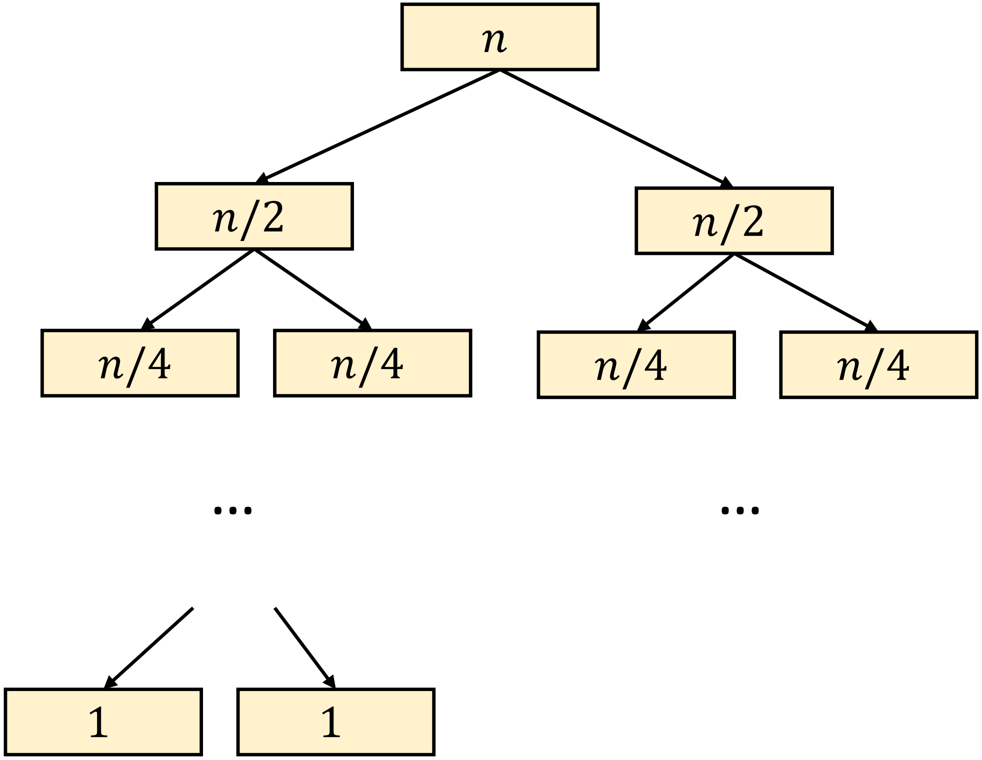
$C\cdot n\cdot \log_2(n)$

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

$$\sum_{i=0}^{\log_2(n)} C\cdot n = C\cdot n\cdot \sum_{i=0}^{\log_2(n)} 1^i$$

Recursion Tree

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



Level	Work
0	Cn
1	$2 \cdot \left(\frac{Cn}{2}\right) = Cn$
2	$4 \cdot \left(\frac{Cn}{4}\right) = Cn$
i	$2^i \cdot \left(\frac{Cn}{2^i}\right) = Cn$
$\log_2 n$	$2^{\log_2 n} \cdot C = Cn$

Recursion Tree

- $T(n) = \boxed{3}\,T(n/\boxed{2}) + Cn$
- $T(1) = C$

level	tree	work
0	$c \cdot n$	$c \cdot n$
1	$c \cdot (\frac{n}{2})$ $c \cdot (\frac{n}{2})$ $c \cdot (\frac{n}{2})$	$3 \times c \cdot (\frac{n}{2}) = c \cdot \frac{3n}{2}$
2	… $c \cdot (\frac{n}{4})$	$9 \times c \cdot (\frac{n}{4}) = c \cdot \frac{9n}{4}$
i	$\sum_{i=0}^{\log_2(n)} c \cdot n \cdot \left(\frac{3}{2}\right)^i = c \cdot n \cdot \overbrace{\sum_{i=0}^{\log_2(n)} \left(\frac{3}{2}\right)^i}^{\text{geometric series}}$	$3^i \times c \cdot \left(\frac{n}{2^i}\right) = c \cdot \left(\frac{3}{2}\right)^i \cdot n$
	$= n \cdot O\left(\left(\frac{3}{2}\right)^{\log_2(n)}\right) = O\left(3^{\log_2(n)}\right) = O\left(n^{\log_2(3)}\right)$	
$\log_2(n)$	…	$3^{\log_2(n)} \times c \cdot 1 = n^{\log_2 3} \cdot c$

Geometric Series

work in level i

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

$$S \;= 1 + r + r^2 + \cdots + r^{\ell}$$

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

$(1-r)\cdot S = 1 - r^{\ell+1}$

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

case 1: $r < 1$

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

case 2: $r > 1$

$S = \frac{r^{\ell+1}-1}{r-1} = O(r^{\ell+1})$

case 3: $r = 1$ $\quad S = (\ell+1)\cdot r$