# Recurrences

Tufts University

# Warm-up question

How can we *merge* two sorted arrays?

| | | | | | |
|---|---|---|---|---|---|
| 3 | 9 | 10 | 15 | 18 | 22 |

| | | | | |
|---|---|---|---|---|
| 0 | 7 | 11 | 13 | 50 |

$$\Downarrow$$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 7 | 9 | 10 | 11 | 13 | 15 | 18 | 22 | 50 |

# Previously ...

- Proofs are an important part of 160
  - Break a big proof into Lemmas
- $\Theta$, $O$ and $\Omega$ compare growth of functions
  - Population on Earth is $\Theta(1.02^n)$, resources are $\Theta(1)$
- In 160 we look at **worst-case** runtime of algorithms
  - Runtime $R$ depends on input size ($R = R(n)$)
  - Assume $R(n) > 0$
  - $R$ defined on natural numbers

# Warm-up question

How can we *merge* two sorted arrays?

| | | | | | |
|---|---|---|---|---|---|
| 3 | 9 | 10 | 15 | 18 | 22 |

| | | | | |
|---|---|---|---|---|
| 0 | 7 | 11 | 13 | 50 |

$$\Downarrow$$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 7 | 9 | 10 | 11 | 13 | 15 | 18 | 22 | 50 |

# Warm-up question

- Create empty solution
- 1 helping index per array (current position)
- Add smallest to solution. Advance indices. Repeat

| | | | | | |
|---|---|---|---|---|---|
| 3 | 9 | 10 | 15 | 18 | 22 |

| | | | | |
|---|---|---|---|---|
| 0 | 7 | 11 | 13 | 50 |

$$\Downarrow$$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

# MergeSort

Conceptually simple. Lots of details

- If array is small (say, $n \leq 3$) solve by brute force
- Divide array into twoElse divide array into two
- Sort each subarray recursively
- Use warm-up to merge sorted arrays

# Example

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 6 | -3 | 4 | 2 | 7 | 9 | 10 | 15 | 8 | 1 |

$$\Downarrow$$

| | | | | |
|---|---|---|---|---|
| 6 | -3 | 4 | 2 | 7 |

| | | | | |
|---|---|---|---|---|
| 9 | 10 | 15 | 8 | 1 |

$$\Downarrow$$

| | | |
|---|---|---|
| 6 | -3 | 4 |

| | |
|---|---|
| 2 | 7 |

| | | |
|---|---|---|
| 9 | 10 | 15 |

| | |
|---|---|
| 8 | 1 |

# Example

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| -3 | 1 | 2 | 4 | 6 | 7 | 8 | 9 | 10 | 15 |

$$\Uparrow$$

| | | | | |
|---|---|---|---|---|
| -3 | 2 | 4 | 6 | 7 |

| | | | | |
|---|---|---|---|---|
| 1 | 8 | 9 | 10 | 15 |

$$\Uparrow$$

| | | |
|---|---|---|
| -3 | 4 | 6 |

| | |
|---|---|
| 2 | 7 |

| | | |
|---|---|---|
| 9 | 10 | 15 |

| | |
|---|---|
| 1 | 8 |

# Runtime?

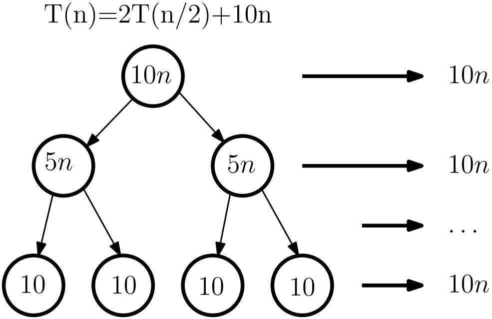
- Forget recursion
  - Runtime of merge? Let's say $10n$
- $T(n)$ = worst-case time needed to sort $n$ elements
- $T(1) = T(2) = T(3) = \Theta(1)$
- $T(n) =$

Let's draw $T(n) = 2T(\frac{n}{2}) + 10n$

# Recursion tree

Key points:

- Identify level sum (grows? shrinks? equal?)
- Identify height of tree




height $\cdot$ min(level sum) $\leq$ real cost $\leq$ height $\cdot$ max(level sum)

# Example 2: $G(n) = G(\frac{n}{2}) + \log n$

- Always assume $G(1) = \Theta(1)$
- Height of tree?
- Level sum? Grows? Shrinks?

# Proof by substitution

- More math, less drawing
- Heavily based on induction
- Confirms recursion tree (or other) hunch

$T(n) = 2T(\frac{n}{2}) + 10n$

Educated guess: $T(n) \leq cn \log n$ for some $c > 0$

**Base case**: $T(2)$

**Induction step**: $T(n)$

# Key Points

- Identify base case

  Normally T(1)

  **Beware**: may fail with $\log n$, $\sqrt{n}$, etc
- Induction step

  Use definition to make smaller

  Substitute

  Find compatible constraints on $c$

**Bonus**: can also give lower bounds!

# INCORRECT use of substitution

$T(n) = 2T(\frac{n}{2}) + 10n$

Guess: $T(n) \leq cn$ for some $c > 0$

**Base case**: $T(0)$

# INCORRECT use of substitution

$T(n) = 2T(\frac{n}{2}) + 10n$

Guess: $T(n) \leq cn$ for some $c > 0$

**Base case**: $T(1)$

**Induction step**: $T(n) = 2T(\frac{n}{2}) + 10n$

# Remember $G(n) = G(\frac{n}{2}) + \log n$?

Height: $\log n$

Level sum shrinks (min$=\Theta(1)$, max$=\log n$)

$G(n) = \Omega(\log n)$ and $G(n) = O(\log^2 n)$

Let's use substitution to prove $G(n) = \Theta(\log n)$

**Guess** $G(n) \leq c \log n$ for some $c > 0$

**Base case** $G(2)$

**Induction step** $G(n) = G(\frac{n}{2}) + \log n$

# Summary

- Recursive algorithms have complicated runtimes
- Math to the rescue!

  Recursion tree helps find intuition

  Substitution nails it down
- Practice makes you perfect

  Go to recitation!

# Additional practice questions

- What is your favorite recursive algorithm?
  1. Express its runtime as a recurrence
  2. Draw the recursion tree
  3. Prove upper/lower bounds using substitution
- Use substitution to show $T(n) = 2T(\frac{n}{2}) + 10n = \Omega(n \log n)$
- For $G(n) = 3G(\frac{n}{3}) + f(n)$, draw recursion tree for:

  $f(n) = 1$
  $f(n) = n$
  $f(n) = n^3$

  - Give upper and lower bounds for $G$ in the three cases above
  - For each of the cases above, which level sum was highest? lowest?