Week 7: Recursive Algorithm Analysis

CSC 236:Introduction to the Theory of Computation

Summer 2024

Instructor: Lily

Announcement

- Email note: please include your UTORID
- A3 peer reviews extended to Thursday (maybe the last one)
- A4 due date extended to July 11th
- Midterm 1 results end of this week or start of next
  - Still finishing up oral exams; solutions out at the end of the week
- Midterm 2: 6:00~8:00pm EX 100 (July 17)
  - Covers proof-of-correctness material (weeks 4~7, assignments 3 and 4)
  - Same structure as midterm 1 (5 questions), one sided aid sheet
- Tutorials cover examples. This week: run-time analysis

Quick Sort (Worst Case)

assume: partition always returns last index of interval

```
def partition(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: index p so that A[k] < A[p] for
  # k = i, …, p-1 and A[l] > A[p] for
  # l = p+1, …, j
  p = i
  pivot = A[j-1]
  for k in range(i, j-1):
    if pivot > A[k]:
      swap(A, p, k)
      p += 1
  swap(A, p, j-1)
  return p
```

```
def quicksort(A, i, j):
  # Pre: list A. i, j indices
  # Post: A is sorted
  if j-i <= 1:
    return
  p = partition(A, i, j)      Θ(n)
  quicksort(A, i, p)          1.
  quicksort(A, p+1, j)        2.
```

partition quicksort 1.

$$T(n) = \Theta(n) + T(n-1) + T(1)$$

quicksort 2.

Substitution Method

Goal: find closed form expression for $T(n)$

eg. $T(n) = \Theta(f(n))$

$$T(n) = \Theta(n) + T(n-1) + T(1)$$

$g(n)$

$T(1)$: $O(1)$

$$= \Theta(n) + T(n-1)$$

Recall: $g(n) \in \Theta(n)$ means $\exists c_1, c_2$

$$c_1 n \leq g(n) \leq c_2 n$$

Want to show (i) $T(n) \leq O(f(n))$ use $c_2 n$

(ii) $T(n) \geq \Omega(f(n))$ use $c_1 n$

Substitution Method

WTS $T(n) = \Omega(f(n))$

guess: $f(n) = n^2$

$$T(n) = \Theta(n) + T(n-1) + T(1)$$

$\geq c_1 n$

$$\geq c_1 n + T(n-1)$$

$$\geq c_1 n + c_1(n-1) + T(n-2)$$

$$\geq c_1 n + c_1(n-1) + c_1(n-2) + T(n-2)$$

$$\vdots$$

$$\geq c_1(n + n-1 + n-2 + \cdots + 1) = c_1 \frac{(n+1)n}{2} \in \Omega(n^2)$$

Pitfalls

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

Guess: $T(n) = O(n)$

i.e. find c: $T(n) \leq cn$

$$T(n) \leq 2\left(c\left\lfloor n/2 \right\rfloor\right) + n$$

$$\leq cn + n \not\leq O(n)$$

$$\leq (c+1)n$$

$$\leq cn \quad (\text{WTS})$$

cannot change constant in the middle.

Pitfalls

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

CORRECT:

guess $T(n) = O(n \log_2 n)$

$P(n) := \exists c : T(n) \leq cn \log_2 n$

Base case: $T(2)$ is const.

Inductive step: Suppose $k \geq 3$ and $P(2) \wedge \cdots \wedge P(k-1)$ is true.
show $P(k)$ is true.

$$T(k) \leq 2c\lfloor k/2 \rfloor \log_2 \lfloor k/2 \rfloor + k \quad \text{(IH)}$$

$$\leq ck \log_2 k/2 + k$$

$$= ck \log_2 k - ck + k$$

$$\leq ck \log_2 k$$

(if $c \geq 1$ this term ≤ 0)

Quick Sort (Expected Case)

assume partition always returns "middle" index.

```
def partition(A, i, j):
  # Pre: i, j indices of A (i <= j)
  # Post: index p so that A[k] < A[p] for
  # k = i, …, p-1 and A[l] > A[p] for
  # l = p+1, …, j
  p = i
  pivot = A[j-1]
  for k in range(i, j-1):
    if pivot > A[k]:
      swap(A, p, k)
      p += 1
  swap(A, p, j-1)
  return p
```

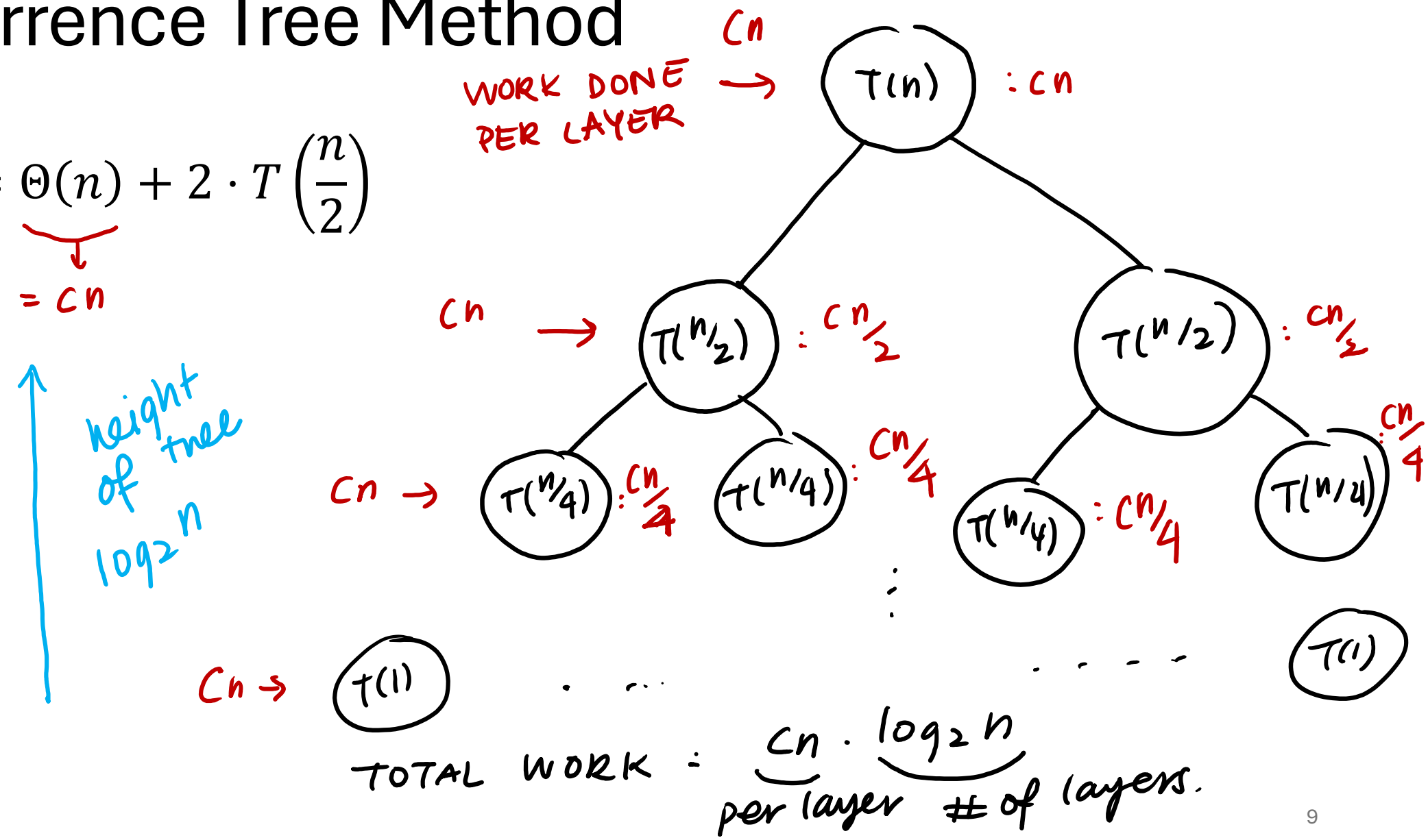
```
def quicksort(A, i, j):
  # Pre: list A. i, j indices
  # Post: A is sorted
  if j-i <= 1:
    return
  p = partition(A, i, j)
  quicksort(A, i, p)
  quicksort(A, p+1, j)
```

partition — 2x call to quicksort.

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

Recurrence Tree Method

$$T(n) = \underbrace{\Theta(n)}_{= cn} + 2 \cdot T\left(\frac{n}{2}\right)$$



Master Method

Recurrence relation of the form: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

branching factor

work done outside recursive call.

size of old input to new input

Case 1. (Root Heavy) If $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then

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

Case 2. (Leaf Heavy) If $f(n) = \mathrm{O}\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then

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

Case 3. (Balanced) If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

Root Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

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

$a=3$ $b=4$ $f(n) = n\log(n)$

$f(n)$ vs $n^{\log_4 3}$ < 1

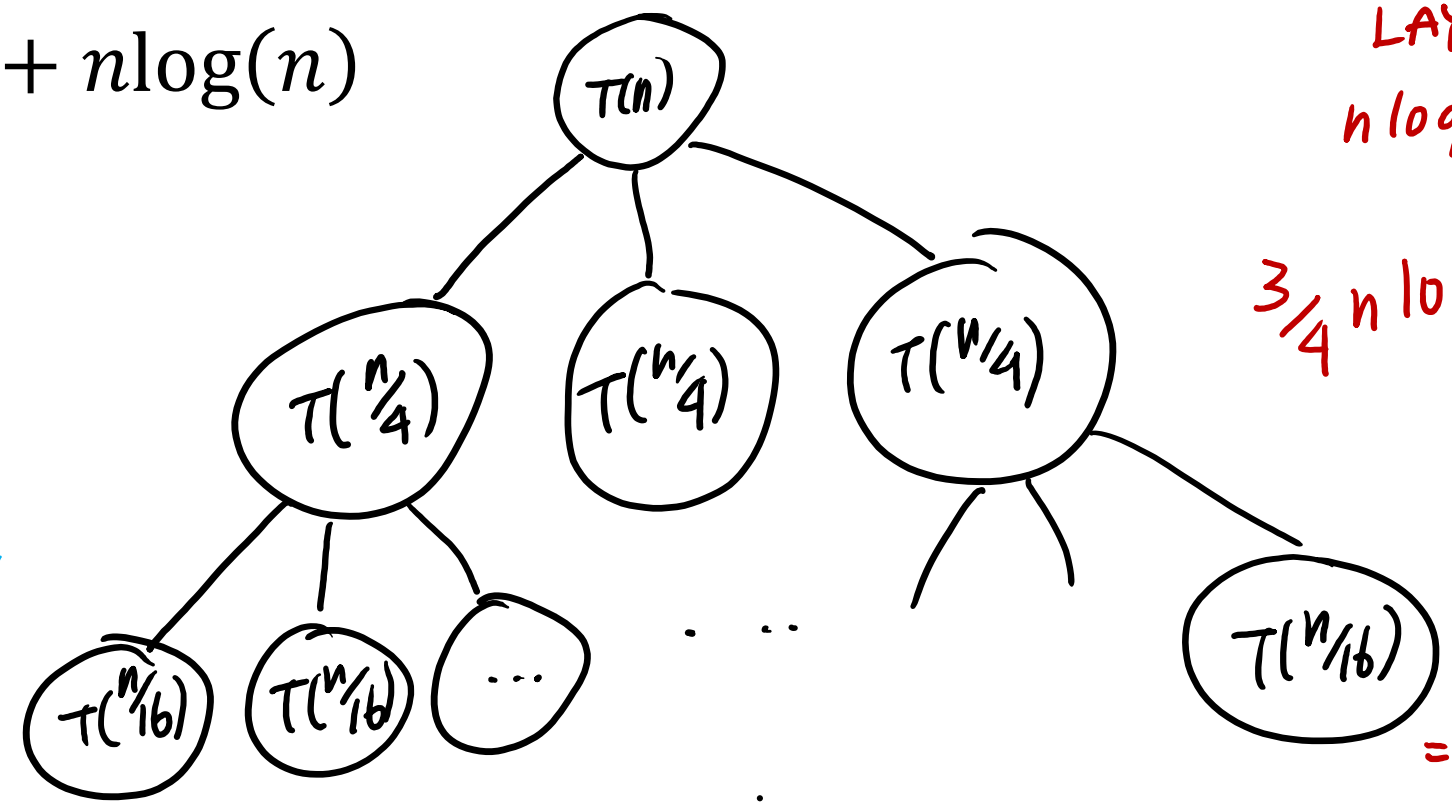
$f(n) = \Omega\left(n^{(\log_b a)+\varepsilon}\right)$

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

Root Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

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



WORK DONE PER LAYER

$n \log n$

$\frac{3}{4} n \log \frac{n}{4}$

$\frac{9}{16} n \log \frac{n}{16}$

$= \left(\frac{3}{4}\right)^i \log \frac{n}{4^i}$

$\frac{1}{1-x} = 1 + x + x^2 + \cdots$

$\sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i = \frac{1}{1-\frac{3}{4}} = 4$

TOTAL WORK: $n \log n \left(1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2 + \cdots\right) = 4n \log n$

Leaf Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

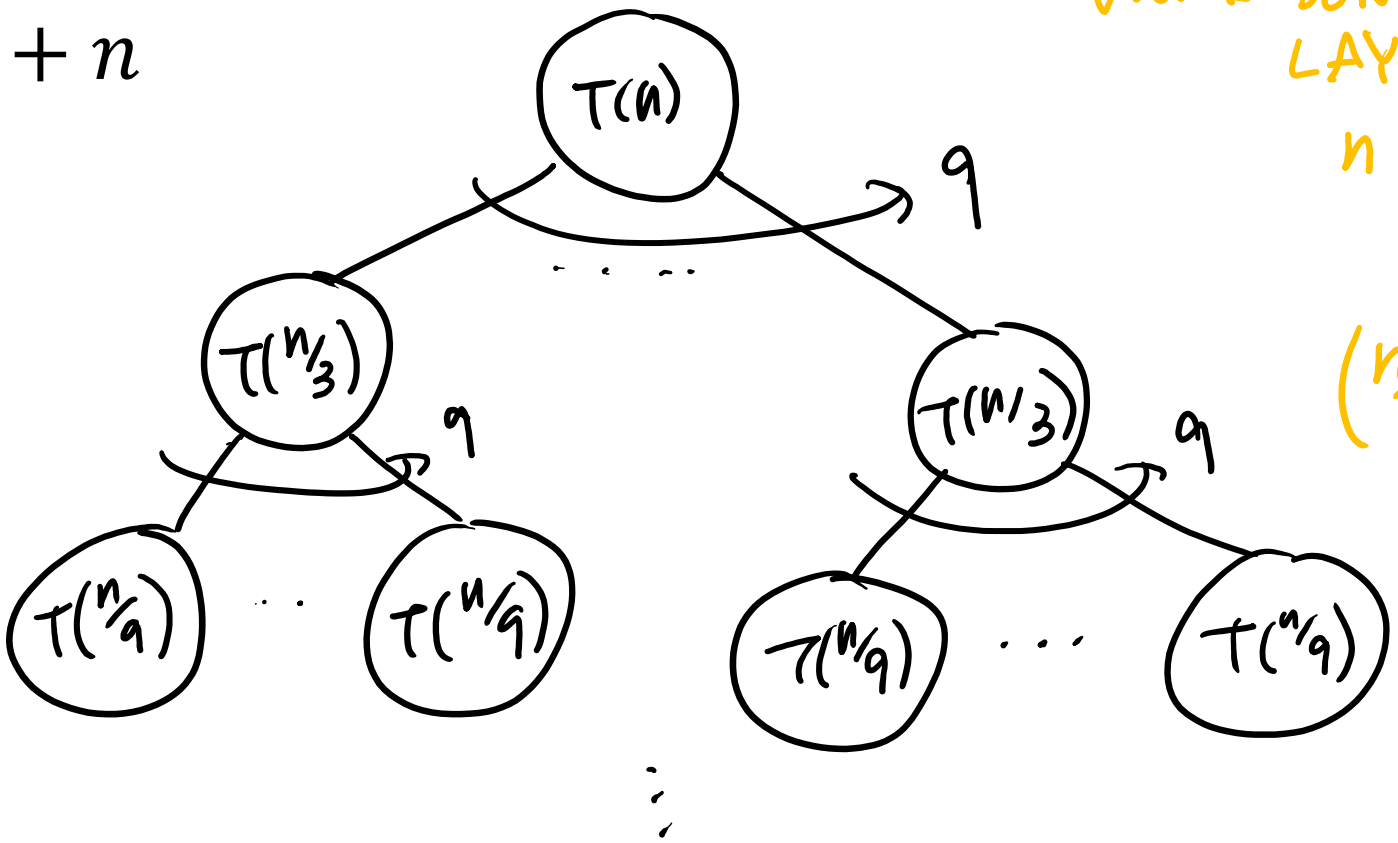
$a = 9$, $b = 3$, $f(n) = n$ vs $n^{\log_b a} = n^2$

$$T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^2\right)$$

Leaf Heavy

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$



WORK DONE IN EACH LAYER

n

$\left(\frac{n}{3}\right) 9$

$\left(\frac{n}{3^2}\right) 9^2$

TOTAL WORK: $\leq n(1+3+3^2+\cdots+3^{(\log_3 n)}) \in O(n^2)$

Balanced

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

$a = 1$ $b = 3/2$

$f(n) = 1 = n^0$ F(n) vs $n^{\log_b a}$

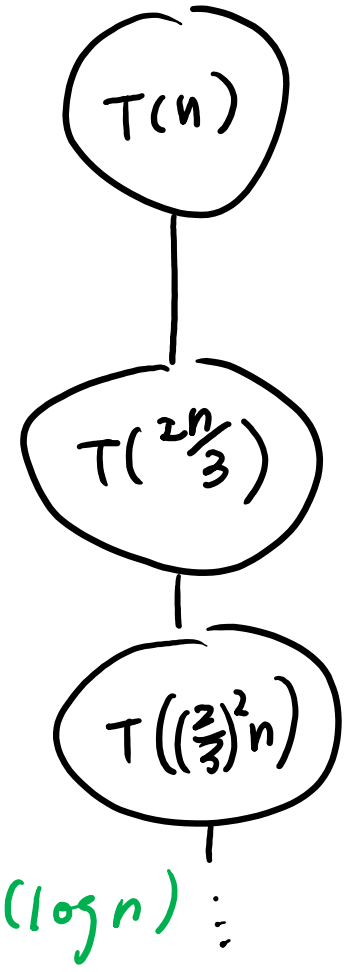
$n^{\log_{3/2} 1} = n^0$

$T(n) = \Theta(n^0 \log n) = \Theta(\log n)$

Balanced

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$



WORK DONE ON EACH LAYER

1

1

1

⋮

TOTAL WORK:

$\log_{3/2} n$ · 1

# of layers

WORK done per layer

$= \log_{3/2} n$

$\in O(\log n)$

How about...

(Root Heavy) $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(f(n))$
(Leaf Heavy) $f(n) = O\left(n^{\log_b a-\epsilon}\right)$ where $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
(Balanced) $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log n\right)$

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

$a = 2$ $\quad b = 2$ $\quad f(n) = n\log n$

$f(n)$ vs $n^{\log_b a}$

$= n^{\log_2 2} = n^1$

IS IT ROOT HEAVY : $f(n) = \Omega\left(n^{1+\varepsilon}\right)$ $\varepsilon > 0$ ✗

$\log n \notin \Omega(n^{\varepsilon})$ for any ε

Now you try!

1. Recall Karatsuba's algorithm from last lecture. Use the Master Method to find its asymptotic running time.
2. Use the Master Method to compute the closed form for the following recurrence relations:

   a) $T(n) = 8T\left(\frac{n}{4}\right) + n^2 \log n$

   b) $T(n) = 3T\left(\frac{n}{9}\right) + \sqrt{n}$

   c) $T(n) = 36T\left(\frac{n}{6}\right) + n^{1.5}$

```
def karatsuba(a, b):
   # Pre: a and b are natural
number with at most n digits
   # Post: returns a*b
if |a| == 1 and |b| == 1:
  return a*b
a1, a2 = a[0..n/2], a[n/2..n]
b1, b2 = b[0..n/2], b[n/2..n]
p1 = karatsuba(a1, b1)
p2 = karatsuba(a1+a2, b1+b2)
p3 = karatsuba(a2, b2)
return p1*10^{n} + (p2-p1-
p3)*10^{n/2} + p3
```

Q1. Recall Karatsuba's algorithm from last lecture. Use the Master Method to find its asymptotic running time.

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

MASTER METHOD:

$a = 3 \quad b = 2 \quad \log_b a = \log_2 3$

$f(n) = c \cdot n = O(n^{\log_2 3})$

$T(n) = \Theta(n^{\log_2 3})$

```
def karatsuba(a, b):
    # Pre: a and b are natural number with at most n digits
    # Post: returns a*b
if |a| == 1 and |b| == 1:
  return a*b
a1, a2 = a[0..n/2], a[n/2..n]
b1, b2 = b[0..n/2], b[n/2..n]
p1 = karatsuba(a1, b1)
p2 = karatsuba(a1+a2, b1+b2)
p3 = karatsuba(a2, b2)
return p1*10^{n} + (p2-p1-p3)*10^{n/2} + p3
```

↓ better than

ELEMENTARY SCHOOL ALGOL: $\Theta(n^2)$

Q2. Use the Master Method to compute the closed form for following:

a) $T(n) = 8T\left(\frac{n}{4}\right) + n^2 \log n$

$a = 8 \quad b = 4 \quad \log_4 8 < 2$

$f(n) = n^2 \log n = \Omega(n^{\log_4 8})$

$T(n) = \Theta(n^2 \log n)$ ROOT HEAVY.

---

b) $T(n) = 3T\left(\frac{n}{9}\right) + \sqrt{n}$

$a = 3 \quad b = 9 \quad \log_9 3 = 1/2$

$f(n) = \sqrt{n} = n^{1/2} = \Theta(n^{1/2})$

$T(n) = \Theta(\sqrt{n} \log n)$ BALANCED

---

c) $T(n) = 36T\left(\frac{n}{6}\right) + n^{1.5}$

$a = 36 \quad b = 6 \quad \log_6 36 = 2$

$f(n) = n^{1.5} = O(n^2)$

$T(n) = \Theta(n^2)$ LEAF HEAVY.

How about...

$T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$?

CAN WE CHANGE THE RECURRENCE RELATION SO WE CAN APPLY MASTER METHOD?

$T(m) = 2T\left(2^{\lfloor m/2 \rfloor}\right) + m$

$S(m) = 2\,S\left(\lfloor m/2 \rfloor\right) + m$

$2^m = n$

(balanced case of master method)

$\Rightarrow S(m) = \Theta(m \log m)$

$m = \log n$

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

How about…

$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$?

Recap

- Substitution method
- Recurrence tree method (guess closed form)
- Master method + examples
- Karatsuba multiplication
- Quick Sort best and worst case

Next time… finite automaton