CS 381 – FALL 2019

Week 3.1, Wednesday, Sept 4

Announcement: Homework 2 released soon (tonight)
Due: September 16th, 2019 @ 11:59PM on Gradescope

Example: $T(n) = 4T(n/2) + n$ (n is a power of 2)

Claim: $T(n) = O(n^3)$
Induction Hypothesis:
Assume $T(k) \le ck^3$ for all $k<n$, for some constant c

$$\mathbf{T(n) = 4T(n/2) + n}$$
$$\le 4\,c\,(n/2)^3 + n$$
$$= cn^3/2 + n$$
$$= cn^3 - (cn^3/2 - n)$$
$$\mathbf{\le cn^3}$$

Need to show $cn^3/2 - n \ge 0$. True for $c \ge 2$

But $O(n^3)$ is not a tight bound! $\theta(n^3)$ does not hold.

Show that $T(n) = O(n^2)$

Claim: $T(n) \le cn^2$

$$\mathbf{T(n) = 4T(n/2) + n \le cn^2}$$
$$\le 4\,c\,(n/2)^2 + n$$
$$= cn^2 + n$$
$$\le cn^2$$

NO!

Wrong argument: *we just made the constant c larger*

To show that it is $O(n^2)$, we need to subtract lower order terms!

Claim: $\mathbf{T(n) \le c_1 n^2 - c_2 n}$, for some constants c_1 and c_2

Inductive hypothesis: $T(k) \le c_1 k^2 - c_2 k$ for all $k<n$

$$
\begin{aligned}
\mathbf{T(n) = 4T(n/2) + n} \\
&\le 4\,(c_1(n/2)^2 - c_2(n/2)) + n \\
&= c_1 n^2 - 2c_2 n + n \\
&= \mathbf{c_1 n^2 - c_2 n} - c_2 n + n \\
&\le \mathbf{c_1 n^2 - c_2 n}
\end{aligned}
$$

Need $-c_2 n + n \le 0$:
true if $c_2 \ge 1$

Recursion Tree Method

Use the recursion tree to find the solution to a recurrence

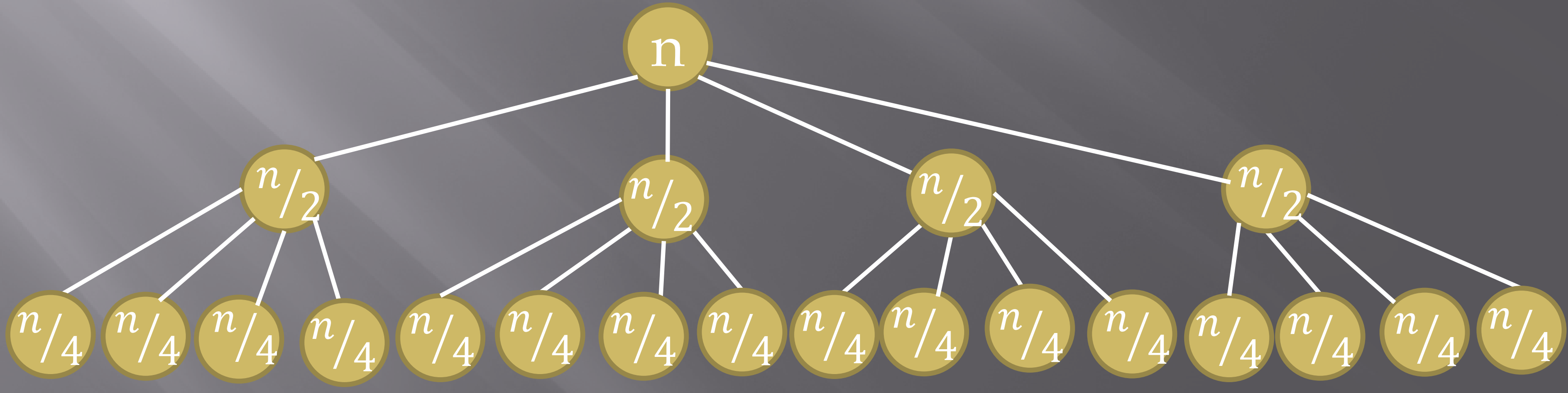
- Tree represents a model of the cost of the recursive algorithm
- Getting a closed form can be messy
- Insight obtained from tree can give a good initial guess to be used in an induction

$T(n) = 4T(n/2) + n$, $n=2^k$

Work done at each level

- Level 0: n
- Level 1: 2n (4 instances of size n/2 each)
- Level 2: 4n (4^2 =16 instances of size n/4 each)
- **Clicker Question:** How much total work is done at Level 3?

  A. 6n B. 8n C. 9n D. 16n E. 32n

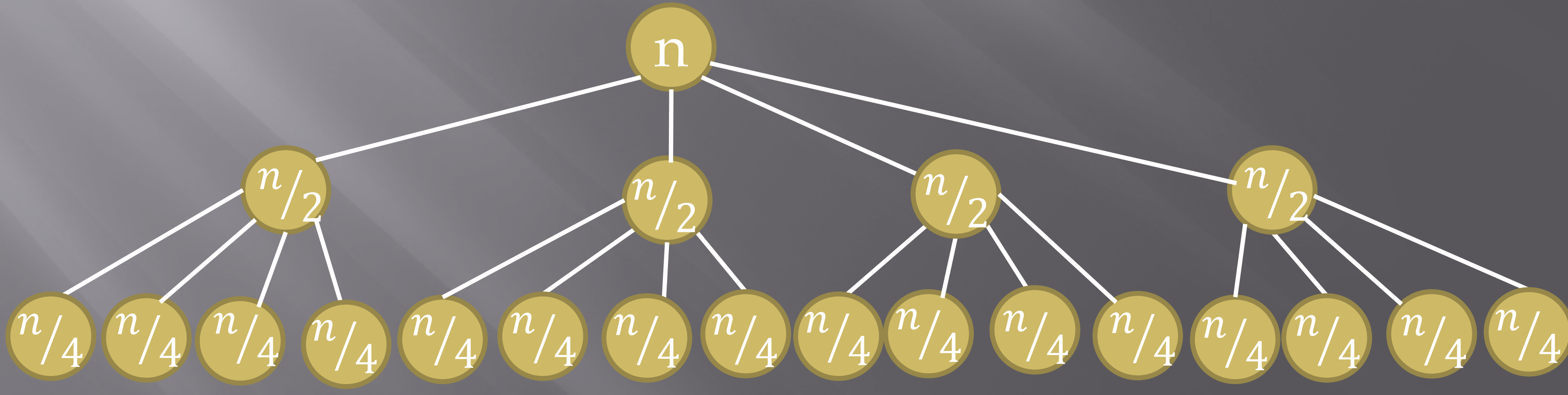


$T(n) = 4T(n/2) + n$, $n=2^k$

Work done at each level

- Level 0: n
- Level 1: 2n (4 instances of size n/2 each)
- Level 2: 4n ($4^2 = 16$ instances of size n/4 each)
- **Clicker Question:** How much total work is done at Level 3?

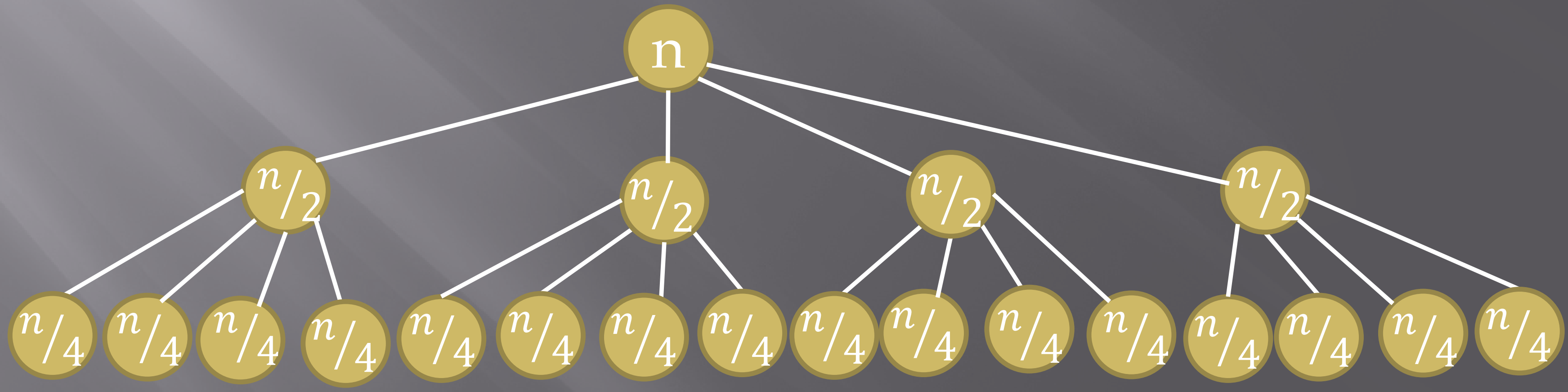
  A. 6n **B. 8n** C. 9n D. 16n E. 32n



T(n) = 4T(n/2) + n, $n=2^k$

Work done at each level

- Level 0: n
- Level 1: 2n (4 instances of size n/2 each)
- Level 2: 4n (4^2 =16 instances of size n/4 each)
- Level 3: 8n (4^3 instances of size $n/2^3$ each)
- at level i, there are 4^i instances of size $n/2^i$ results in 2^in total work for level i



Total work done at all levels of the recursion tree

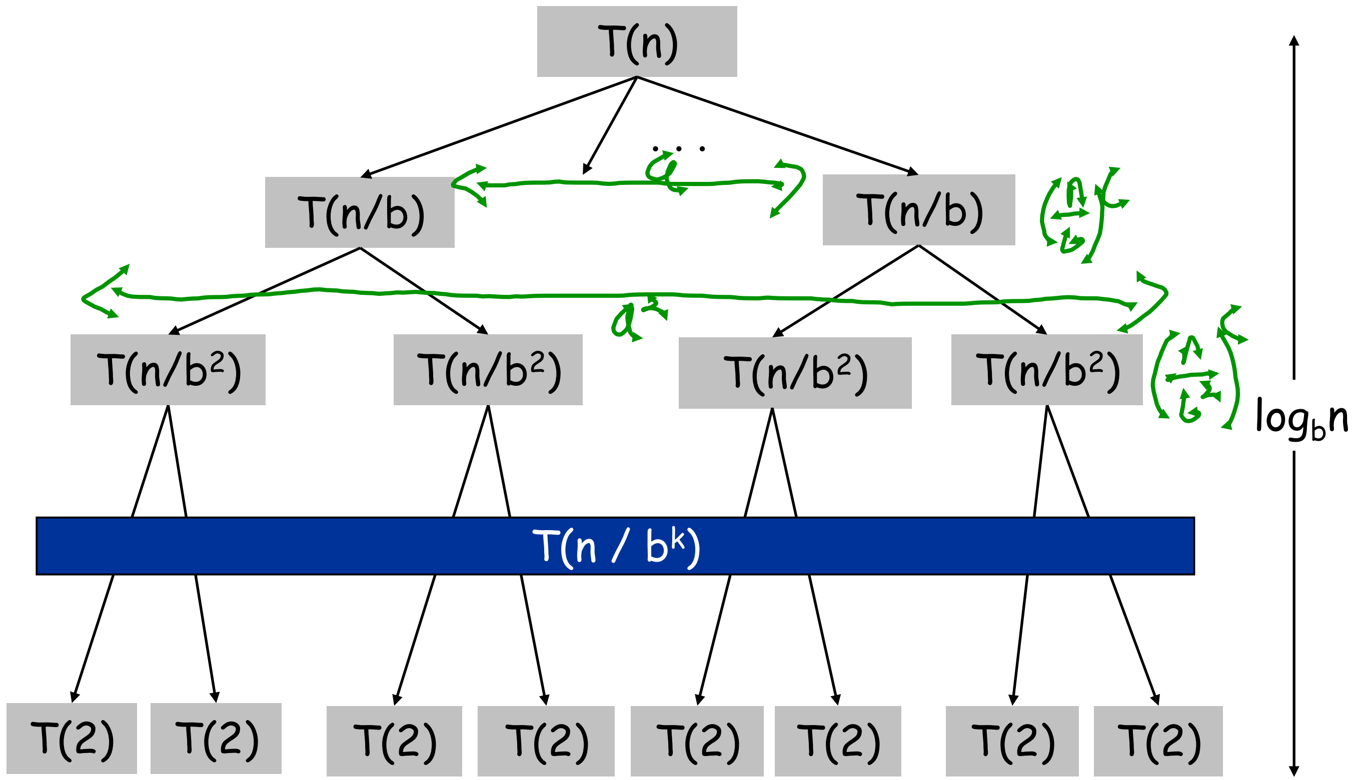
at level i of the tree

- there are 4^i nodes, each doing work of size $n/2^i$
- results in $n2^i$ total work for level i

$$\sum_{i=0}^{k} n\,2^i = n\left(2^{k+1}-1\right) = n(2n-1) = 2n^2 - n$$

This gives $T(n) = \theta(n^2)$
Prove this correct by induction as an exercise.

More General Analysis



Merge Step Cost

n^c

$a(n/b)^c = n^c(a/b^c)$

$a^2(n/b^2)^c =$ **$n^c(a/b^c)^2$**

. . .

$n^c(a/b^c)^k$

. . .

$n^c(a/b^c)^{\log_b n}$

$$T(n) \le \begin{cases} 1 & if\ n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & otherwise \end{cases}$$

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

A Helpful Identity

Fact: If $\gamma \neq 1$ then

$$1 + \gamma^1 + \gamma^2 \ldots + \gamma^k = \frac{1 - \gamma^{k+1}}{1 - \gamma}$$

Observation 1: If $\gamma = 1$ then $1 + \gamma^1 + \gamma^2 \ldots + \gamma^k = k + 1 \in \Theta(k)$

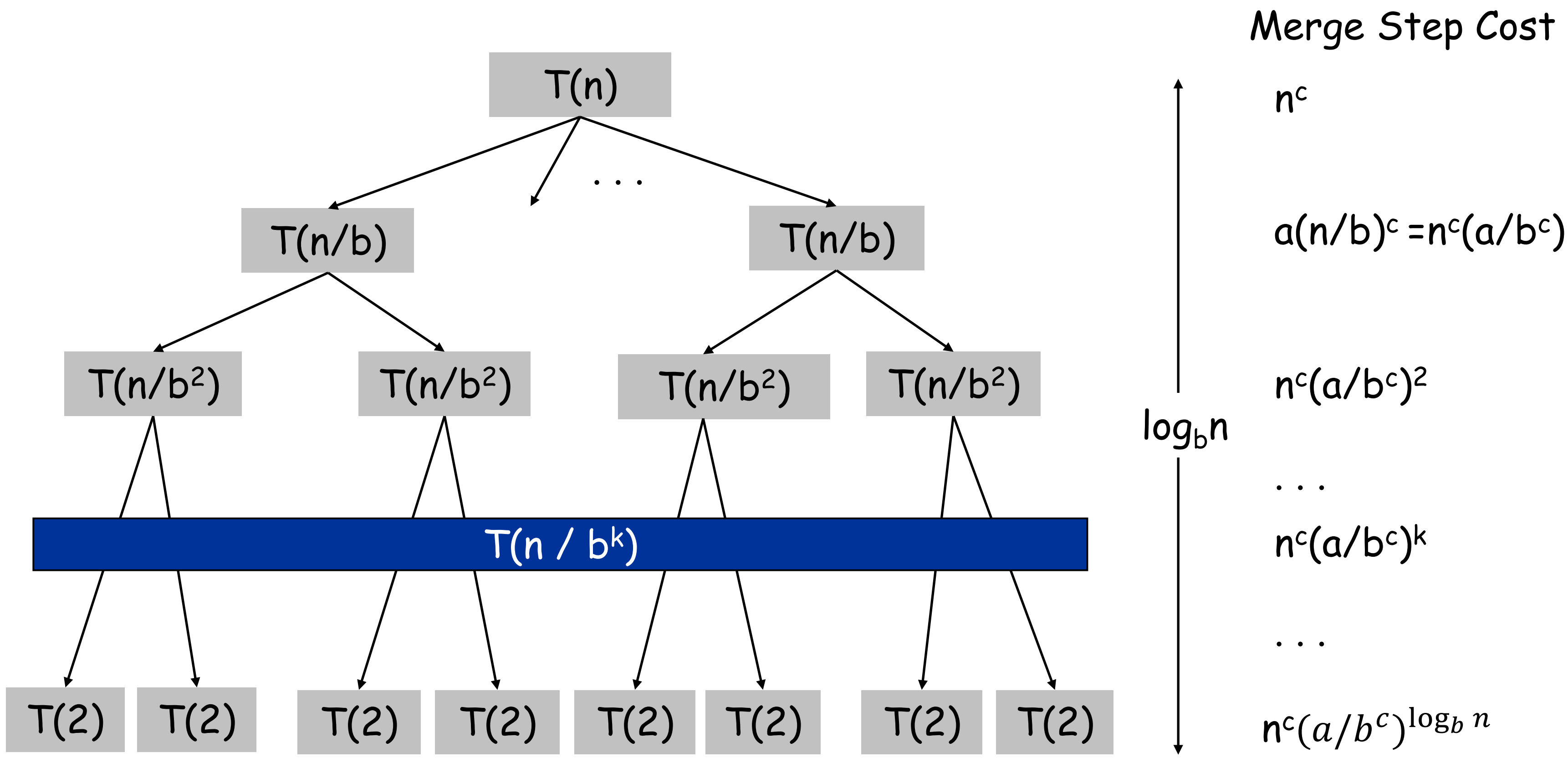
Observation 2: If $0 < \gamma < 1$ then $\mathbf{1} + \gamma^1 + \gamma^2 \ldots + \gamma^k \approx \frac{1}{1-\gamma} \in \Theta(1)$

Observation 3: If $1 < \gamma$ then $1 + \gamma^1 + \gamma^2 \ldots + \boldsymbol{\gamma^k} \approx \frac{\gamma^{k+1}}{\gamma-1} \in \Theta(\gamma^k)$

Observation 4: In our case $k = \log_b n$ and $\gamma = \left(\frac{a}{b^c}\right)$

$$T(n) \le n^c \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i = n^c \left(\frac{1 - \gamma^{k+1}}{1 - \gamma}\right)$$

More General Analysis

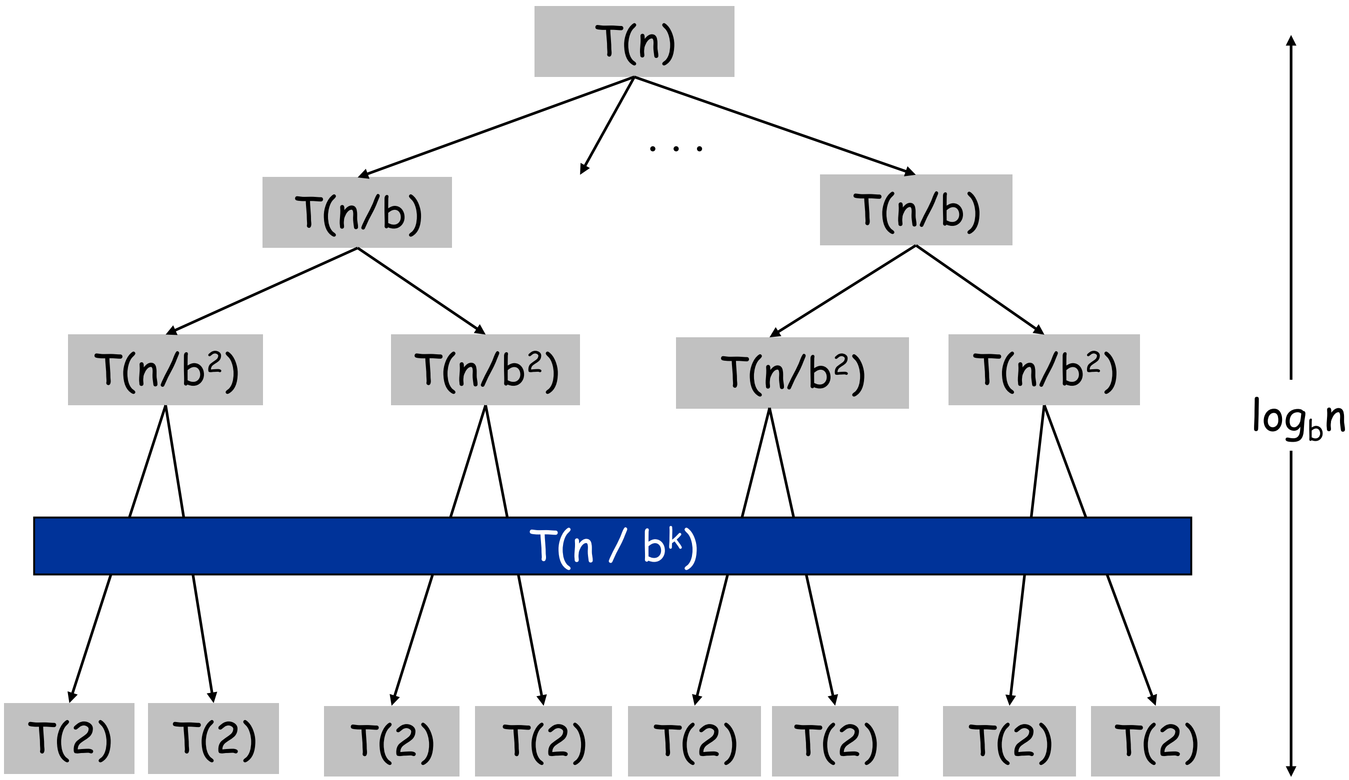


$$T(n) \le \begin{cases} 1 & if\ n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & otherwise \end{cases}$$

Case 1: $\gamma = \left(\frac{a}{b^c}\right) = 1$

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

More General Analysis



Merge Step Cost

n^c

$a(n/b)^c = n^c(a/b^c)$

$n^c(a/b^c)^2$

. . .

$n^c(a/b^c)^k$

. . .

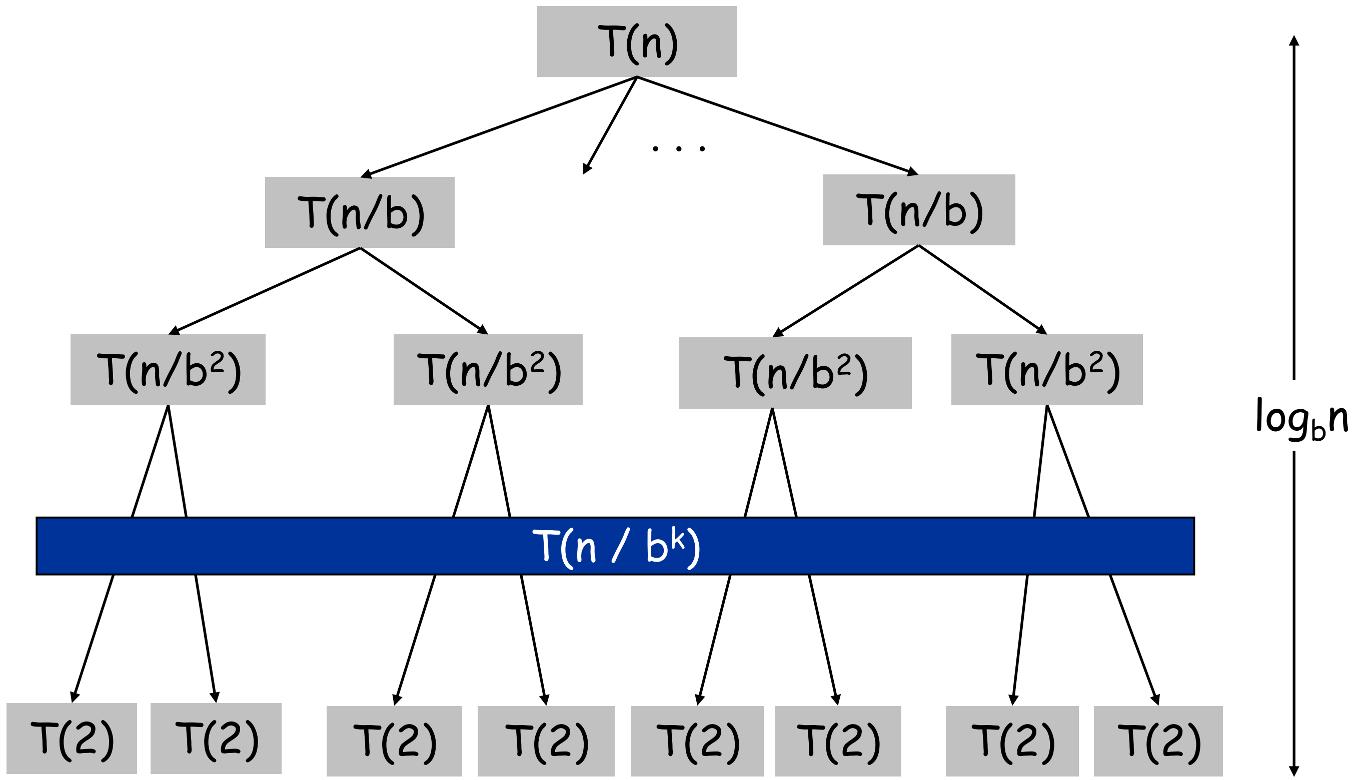
$n^c(a/b^c)^{\log_b n}$

$$T(n) \leq \begin{cases} 1 & if\ n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & otherwise \end{cases}$$

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

Case 2: $\gamma = \left(\frac{a}{b^c}\right) < 1$

More General Analysis



Merge Step Cost

n^c

$a(n/b)^c = n^c(a/b^c)$

$n^c(a/b^c)^2$

. . .

$n^c(a/b^c)^k$

. . .

$n^c(a/b^c)^{\log_b n}$

$$T(n) \le \begin{cases} 1 & if\ n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & otherwise \end{cases}$$

Case 3: $\gamma = \left(\frac{a}{b^c}\right) > 1$

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

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

Implications for Divide and Conquer Analysis

- Merge Cost: $O(n^c)$ (want c to be small)
- Branching Factor: a (smaller branching factor → faster)
- Reduction in Input Size: b (bigger is better)
  - Key Ratio: a/b^c

$$T(n) \le \begin{cases} 1 & if\ n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & otherwise \end{cases}$$

Case 1: $\left(\frac{a}{b^c}\right) < 1$ $\quad T(n) = \Theta(n^c)$

Case 2: $\gamma = \left(\frac{a}{b^c}\right) = 1$ $\quad T(n) = \Theta(n^c \log n)$

Case 3: $\left(\frac{a}{b^c}\right) > 1$ $\quad T(n) = \Theta\left(n^{\log_b a}\right)$

Clicker Question

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

a. $\Theta(n^2)$ b. $\Theta(n^3)$ c. $\Theta(n^2 \log n)$ d. $\Theta\left(n^{\ln 4}\right)$

e. none of the above

Master Theorem

$$T(n) \le \begin{cases} 1 & if\ n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & otherwise \end{cases}$$

Case 1: $\left(\frac{a}{b^c}\right) < 1$ $\quad T(n) = \Theta(n^c)$

Case 2: $\gamma = \left(\frac{a}{b^c}\right) = 1$ $\quad T(n) = \Theta(n^c \log n)$

Case 3: $\left(\frac{a}{b^c}\right) > 1$ $\quad T(n) = \Theta\left(n^{\log_b a}\right)$

Clicker Question

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

a. $\Theta(n^2)$ b. $\Theta(n^3)$ c. $\boldsymbol{\Theta(n^2 \log n)}$ d. $\Theta(n^{\ln 4})$

e. none of the above

Master Theorem

$$T(n) \leq \begin{cases} 1 & if\ n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & otherwise \end{cases}$$

Case 1: $\left(\frac{a}{b^c}\right) < 1$ $\quad T(n) = \Theta(n^c)$

Case 2: $\gamma = \left(\frac{a}{b^c}\right) = 1$ $\quad T(n) = \Theta(n^c \log n)$

Case 3: $\left(\frac{a}{b^c}\right) > 1$ $\quad T(n) = \Theta\left(n^{\log_b a}\right)$

Implications for Divide and Conquer Analysis

- Merge Cost: $O(n^c)$ (want c to be small)
- Branching Factor: a (smaller branching factor → faster)
- Reduction in Input Size: b (bigger is better)
  - Key Ratio: a/b^c

$$T(n) \le \begin{cases} \mathbf{1000000} & \boldsymbol{if\ n \le 100} \\ a \times T\left(\frac{n}{b} + \mathbf{50}\right) + n^c & otherwise \end{cases}$$

Case 1: $\left(\frac{a}{b^c}\right) < 1$ $\quad T(n) = \Theta(n^c)$

Case 2: $\gamma = \left(\frac{a}{b^c}\right) = 1$ $\quad T(n) = \Theta(n^c \log n)$

Case 3: $\left(\frac{a}{b^c}\right) > 1$ $\quad T(n) = \Theta\left(n^{\log_b a}\right)$

Implications for Divide and Conquer Analysis

- Merge Cost: $O(n^c)$ (want c to be small)
- Branching Factor: a (smaller branching factor → faster)
- Reduction in Input Size: b (bigger is better)
  - Key Ratio: a/b^c $\log_b a \geq c \leftrightarrow \frac{a}{b^c} \geq 1$

$$T(n) \leq \begin{cases} \mathbf{1000000} & \boldsymbol{if\ n \leq 100} \\ a \times T\left(\frac{n}{b} + \mathbf{50}\right) + f(n) & otherwise \end{cases}$$

Assume $f(n) \geq 0$

Case 1: $f(n) = \Omega\left(n^{\varepsilon + \log_b a}\right)$ $T(n) = \Theta(f(n))$

Case 2: $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ $T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$
(assumes $k \geq 0$)

Case 3: $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ $T(n) = \Theta\left(n^{\log_b a}\right)$

Other Recurrences

- $T(n) = T(n-1) + 1$ (Unroll: $T(n) = O(n)$)
  - $T(n) = T(n-1) + 1 = T(n-2) + 1 + 1$
    $= T(n-3) + 1 + 1 + 1 = \cdots = T(n-k) + k$
    $= T(1) + n - 1$

- $T(n) = 2 \times T(n-10)$ (Exponential)

Two branches

Only constant reduction in input size

Other Recurrences

- $T(n) = 2 \times T(n - 10)$ (Exponential)

Two branches

Only constant reduction in input size

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

- How to find c? [Trick]

$$2 = \frac{T(n)}{T(n-10)} = \frac{c^n}{c^{n-10}}$$

$$\rightarrow c^{10} = 2$$

$$\rightarrow c = \sqrt[10]{2} \approx 1.07177$$

(Root of Characteristic Equation)

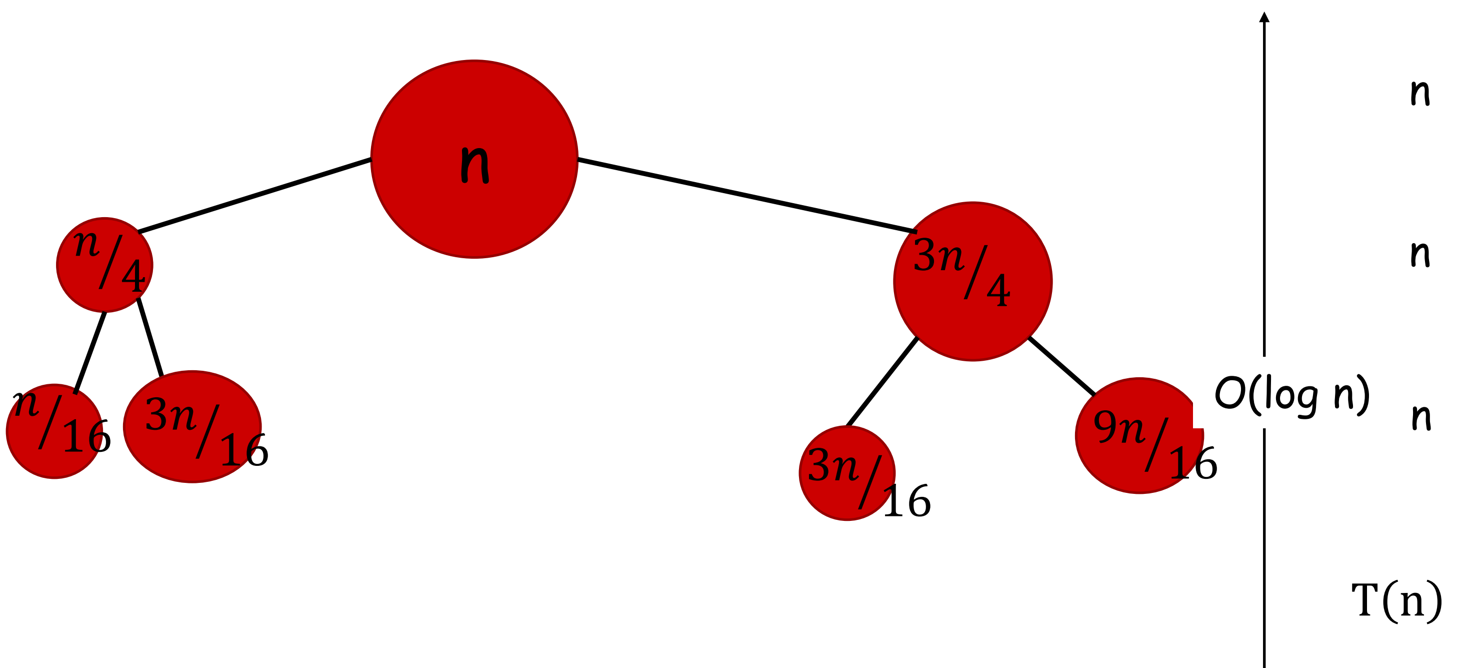
Must verify solution by induction

Other Recurrences

- MergeSort with Uneven Split: Split L into A, B of sizes n/4 and 3n/4.

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

A bit harder to Analyze with recursion tree

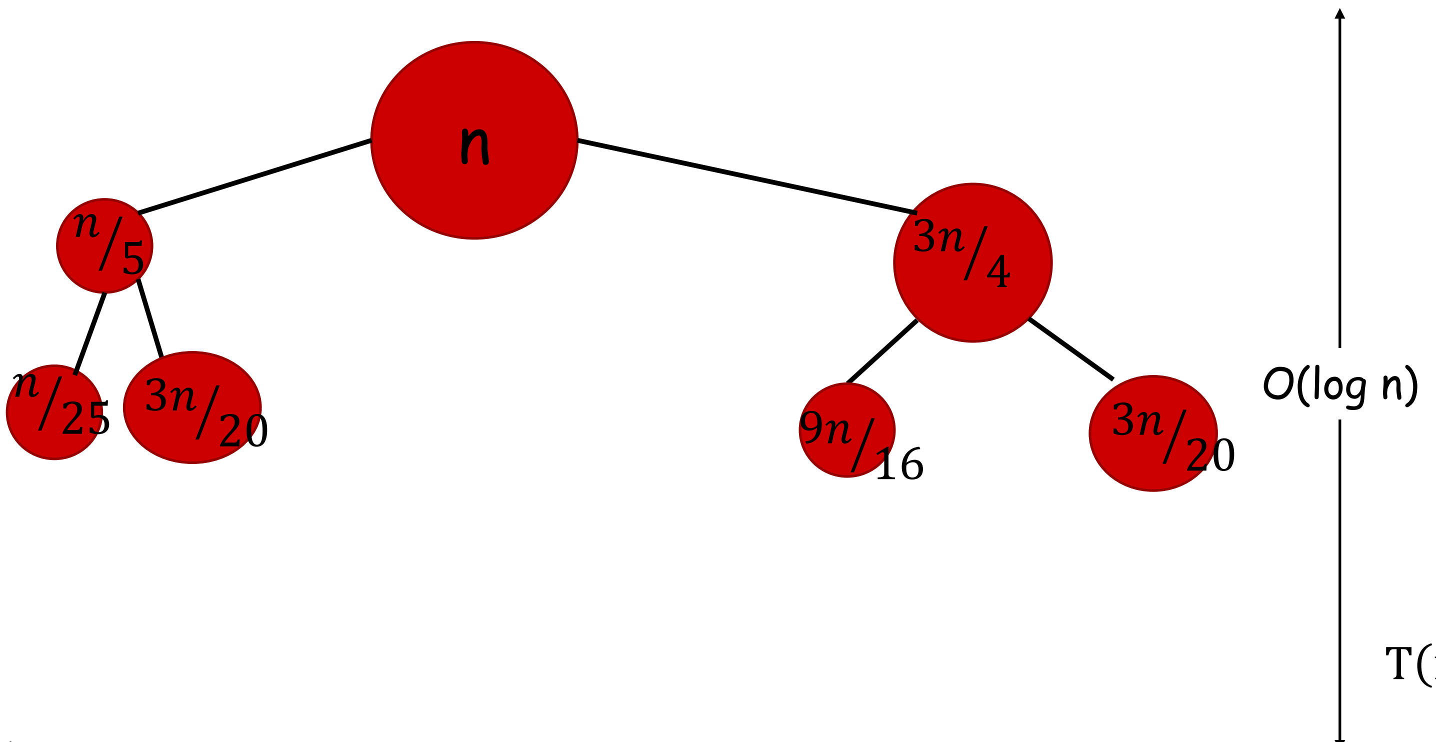


$T(n) = O(n \log n)$

Other Recurrences

- MergeSort with Uneven Split: Split L into A, B of sizes n/4 and 3n/4.

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + n$$



n

(19/20)n

$(19/20)^2$n

$$T(n) \leq n \sum_{i=0}^{\infty} \left(\frac{19}{20}\right)^i$$

Divide and conquer algorithms

- Mergesort
- Quicksort
- Binary Search
- *Linear-time selection*
- *Skyline Problem*
- *Maximum Subarray*
- *Counting inversions*