Solve the Recurrence

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 2T(n/2) + cn & \text{if } n > 1 \end{cases}$$

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + n & \text{if } n > 1 \end{cases}$$

T(n) = n + 2T(n/2)

T(n/2) = n/2 + 2T(n/2/2)

T(n/2) = n/2 + 2T(n/4)

T(n/4) = n/4 + 2T(n/4/2)

T(n/4) = n/4 + 2T(n/8)

For k = logn => n = 2^k

T(n) = n + 2T(n/2)

T(n) = n + 2(n/2+2T(n/4))

T(n) = n + n + 4T(n/4)

T(n) = 2n + 4(n/4 + 2T(n/8))

T(n) = 2n + n + 8T(n/8)

T(n) = 3n + 8(n/8 + 2T(n/16))

T(n) = 4n + 2^4 T(n/2^4)

.....

T(n) = kn + 2^k.T(n/2^k)

T(n) = n.logn + T(1)

T(n) = O(nlogn)

Solve the Recurrence (Practice example)

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/2) + n & \text{if } n > 1 \end{cases}$$

T(n) = n + T(n/2)

T(n) = n + (n/2+T(n/4))

T(n) = n + n/2 + T(n/4)

T(n) = n + n/2 + n/4 + T(n/8))

T(n) = n + n/2 + n/4 + T(n/8).

T(n) =n (1 + 1/2 + 1/4) + T(n/2^3)

…..

T(n) = n(1 + ½ + ¼ …….. + 1/k) + T(n/2^k)

For k = log n

T(n) = n(1+ ½ + ¼ + ….. + 1/n) + T(1)

T(n) = n.(1+1) + 1

T(n) = Theta(n)

Geometric Series (Aside)

$$1 + x + x^2 + \ \dots + \ x^n = \frac{1 - \ x^{n+1}}{1 - x} \ for \ x \ \neq 1$$

OR

$$1 + x + x^2 + \ \dots + \ x^n = \frac{x^{n+1} \ - 1}{x \ - 1} \ for \ x \ \neq 1$$

$$1 + x + x^2 + \ \dots + \ x^n = \frac{1}{1 - x} \ for \ x < 1$$

Solve the Recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 4T(n/2) + n & \text{if } n > 1 \end{cases}$$

$T(n) = n + 4T(n/2)$

$T(n/2) = n/2 + 4T(n/2/2)$

$T(n/2) = n/2 + 4T(n/4)$

$T(n/4) = n/4 + 4T(n/4/2)$

$T(n/4) = n/4 + 4T(n/8)$

=====================

$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \ldots + 2^k$

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

=====================

For $k = \log n \Rightarrow n = 2^k$

$T(n) = n + 4T(n/2)$

$T(n) = n + 4(n/2 + 4T(n/4))$

$T(n) = n + 4n/2 + 16T(n/4)$

$T(n) = n + 2n + 4^2(n/4 + 4T(n/8))$

$T(n) = n + 2n + 4n + 4^3\, T(n/8)$

$T(n) = n + 2n + 4 + 4^3(n/8 + 4T(n/16))$

$T(n) = n + 2n + 4n + 8n + 4^4\, T(n/2^4)$

$T(n) = n\,(1 + 2 + 4 + 8) + 4^4 T(n/ 2^4)$

...

$T(n) = n.\ \frac{2^{k+1} - 1}{2 - 1} + 4^k.T(n/2^k)$

$T(n) = n.\ (2^{logn+1} - 1\,) + 4^{\log n}\ .\ T(1)$

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

Solve the Recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/2) + 1 & \text{if } n > 1 \end{cases}$$

T(n) = 1 + T(n/2)

T(n) = 1 + (1+T(n/4))

T(n) = 2 + T(n/4)

T(n) = 3 + T(n/8)

T(n) = 3 + T(n/2^3)

…..

T(n) = k + T(n/2^k)

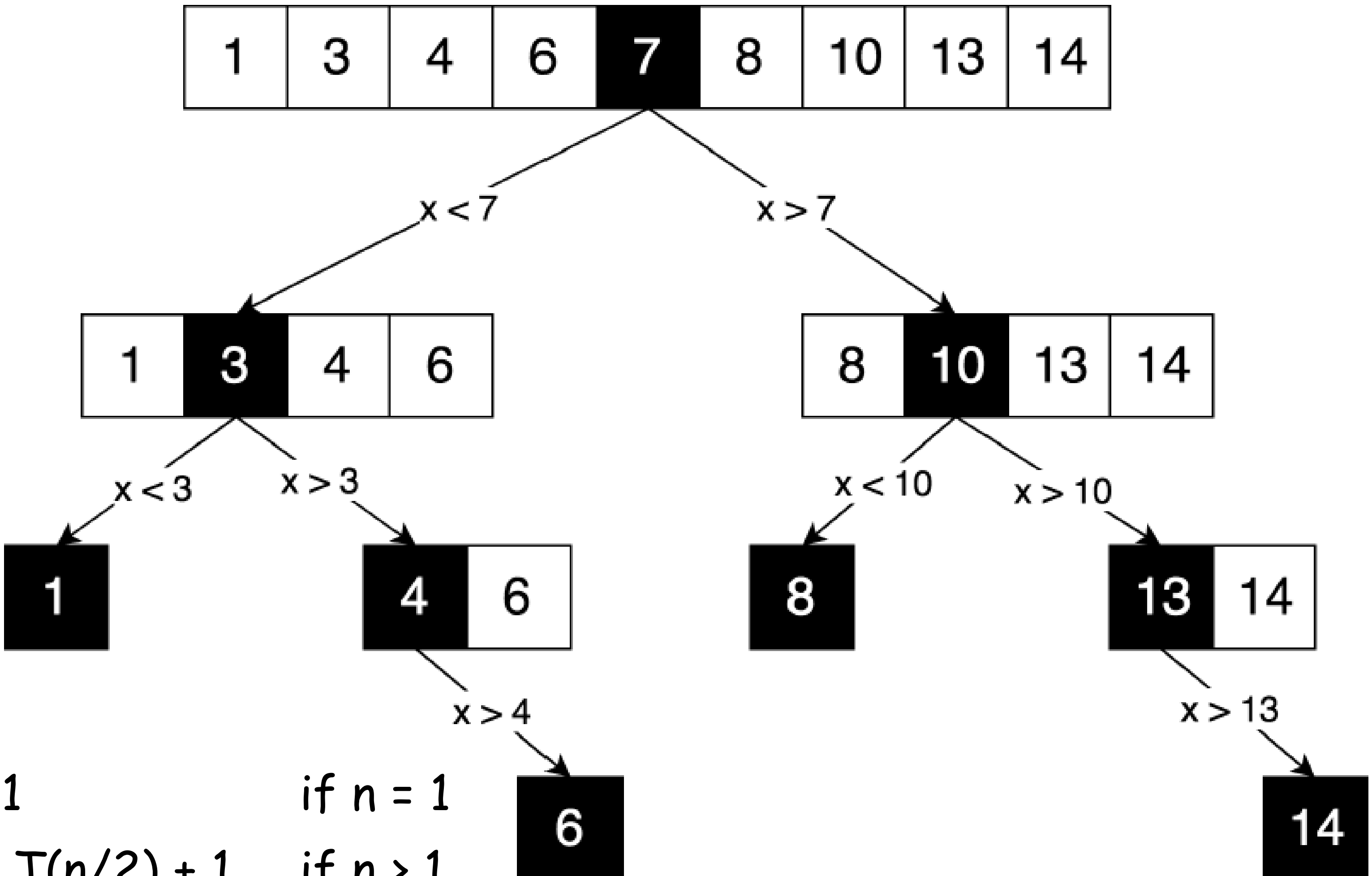
For k = logn => n = 2^k

T(n) = logn + T(1)

T(n) = 1 + log(n)

i.e. T(n) = Theta (logn)

Binary search example



$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/2) + 1 & \text{if } n > 1 \end{cases}$$

The master theorem

- Suppose that $a \geq 1, b > 1,$ and d are constants (independent of n).
- Suppose $T(n) = a \cdot T\left(\frac{n}{b}\right) + O\left(n^d\right)$. Then

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

Three parameters:
a : number of subproblems
b : factor by which input size shrinks
d : need to do n^d work to create all the subproblems and combine their solutions.

We can also take n/b to mean either $\left\lfloor \frac{n}{b} \right\rfloor$ or $\left\lceil \frac{n}{b} \right\rceil$ and the theorem is still true.

The master theorem (Limitations)

You *cannot* use the Master Theorem if

- $T(n)$ is not monotone, ex: $T(n) = \sin n$
- $f(n)$ is not a polynomial, ex: $T(n) = 2T(\frac{n}{2}) + 2^n$
- b cannot be expressed as a constant, ex: b=2n

Examples

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d).$$

$$T(n) = \begin{cases} O(n^d \log(n)) & \text{if } a = b^d \\ O(n^d) & \text{if } a < b^d \\ O(n^{\log_b(a)}) & \text{if } a > b^d \end{cases}$$

- An example
  - $T(n) = 4\,T(n/2) + O(n)$
  - $T(n) = O(n^2)$

$a = 4$, $b = 2$, $d = 1$ — $a > b^d$ ✓

- Binary Search
  - $T(n) = T(n/2) + c$
  - $T(n) = O(\log(n))$

$a = 1$, $b = 2$, $d = 0$ — $a = b^d$ ✓

- MergeSort
  - $T(n) = 2T(n/2) + O(n)$
  - $T(n) = O(n\log(n))$

$a = 2$, $b = 2$, $d = 1$ — $a = b^d$ ✓

- That other one
  - $T(n) = T(n/2) + O(n)$
  - $T(n) = O(n)$

$a = 1$, $b = 2$, $d = 1$ — $a < b^d$ ✓

Solve the Recurrence

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

$T(n) = 1 + (1+T(n-2))$

$T(n) = 2 + T(n-2)$

$T(n) = 3 + T(n-3)$

$T(n) = 4 + T(n-4)$

…..

$T(n) = k + T(n-k)$

For $k = (n-1)$

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

$T(n) = n$

i.e. $T(n) = \text{Theta}(n)$

For $n - k = 1$

So, $k = n-1$

Solve the Recurrence

For n – k = 1

So, k = n-1

T(n) = n + T(n-1)

T(n) = n + (n-1+T(n-2))

T(n) = 2n – 1 + T(n-2)

T(n) = 2n -1 + ((n-2) + T(n-3))

T(n) = 3n -3 + T(n-3)

T(n) = 3n -3 + n-3 + T(n-4))

T(n) = 4n – 6 + T(n-4)

…..

T(n) = kn – c + T(n-k)

T(n) = (n – 1).n - n – c + T(1)

T(n) = n^2 - n – c + T(1)

i.e. T(n) = Theta(n^2)

Solve the Recurrence

$T(n) = 2T(n-1)$

$T(n) = 2(2T(n-2))$

$T(n) = 4T(n-2)$

$T(n) = 4(2T(n-3))$

$T(n) = 8T(n-3)$

$T(n) = 8(2T(n-4))$

$T(n) = 16T(n-4)$

$T(n) = 2^4 T(n-4)$

……..

$T(n) = 2^k.T(n-k)$

$T(n) = 2^{(n-1)}.T(1)$

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

For $n – k = 1$

So, $k = n-1$

Master Theorem Example

Let $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n} + 42$. What are the parameters?

$$\begin{aligned} a &= \\ b &= \\ d &= \end{aligned}$$

Therefore which condition?

Master Theorem Example

- Solution

Let $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n} + 42$. What are the parameters?

$$\begin{aligned} a &= 2 \\ b &= 4 \\ d &= \frac{1}{2} \end{aligned}$$

Therefore which condition?

Since $2 = 4^{\frac{1}{2}}$, case 2 applies.

Thus we conclude that

$$T(n) \in \Theta(n^d \log n) = \Theta(\sqrt{n} \log n)$$

Master Theorem 4th Case

- Fourth Condition:

Recall that we cannot use the Master Theorem if $f(n)$ (the non-recursive cost) is not polynomial.

There is a limited 4-th condition of the Master Theorem that allows us to consider polylogarithmic functions.

Corollary

If $f(n) \in \Theta(n^{\log_b a} \log^k n)$ *for some* $k \geq 0$ *then*

$$T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$$

This final condition is fairly limited and we present it merely for completeness.

MASTER THEOREM

- $T(n) = 2 \cdot T(n/2) + n \log n$

$a = ?$

$b = ?$

$d = ?$

MASTER THEOREM

- **$T(n) = 2 \cdot T(n/2) + n \log n$**

a = 2
b = 2
d = n/a
f(n) is not polynomial

i.e. $f(n) = \Theta(n \log n)$

So k = 1 therefore, fourth condition of master theorem

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

Recursion Tree

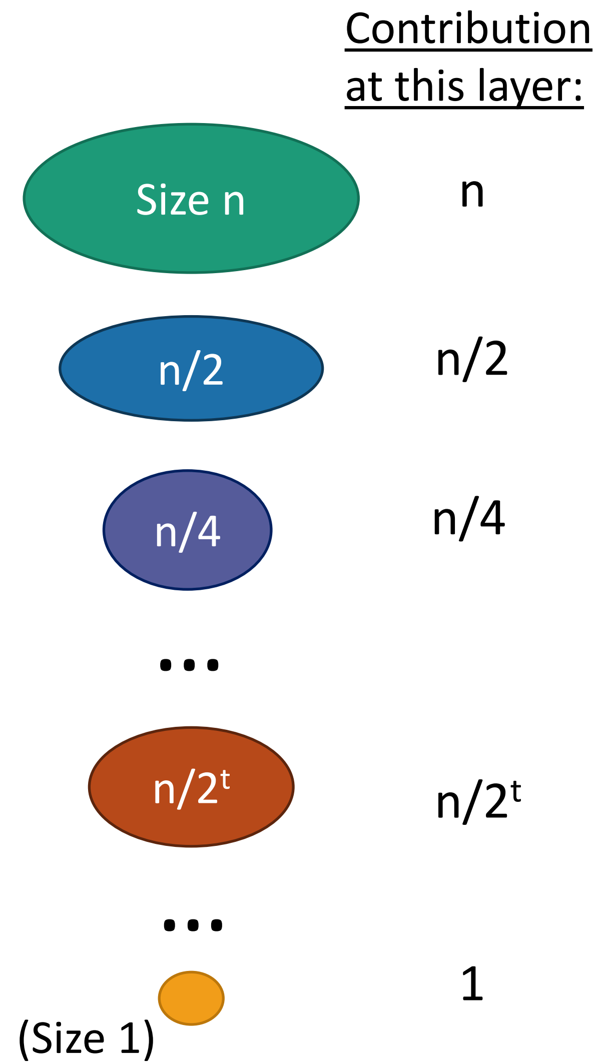
Recursion Tree method

- $T_1(n) = T_1\left(\frac{n}{2}\right) + n, \quad T_1(1) = 1.$
- Adding up over all layers:

$$\sum_{i=0}^{\log(n)} \frac{n}{2^i}$$

$$= n. \sum_{i=0}^{\log(n)} \frac{1}{2^i} = n.2$$

- So $T_1(n) = O(n).$



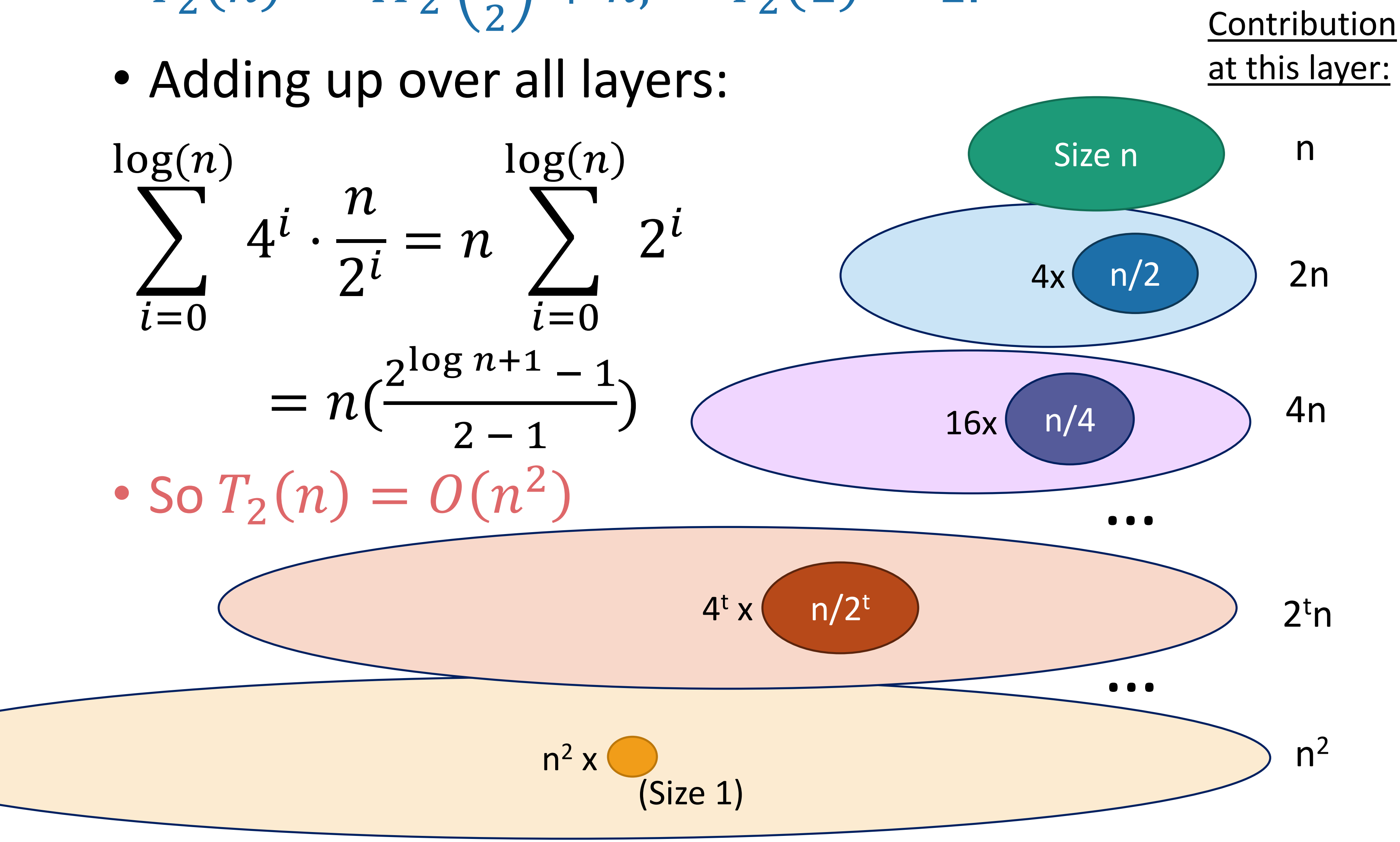
Recursion Tree method

- $T_2(n) = 4T_2\left(\frac{n}{2}\right) + n, \quad T_2(1) = 1.$
- Adding up over all layers:

$$\sum_{i=0}^{\log(n)} 4^i \cdot \frac{n}{2^i} = n \sum_{i=0}^{\log(n)} 2^i$$

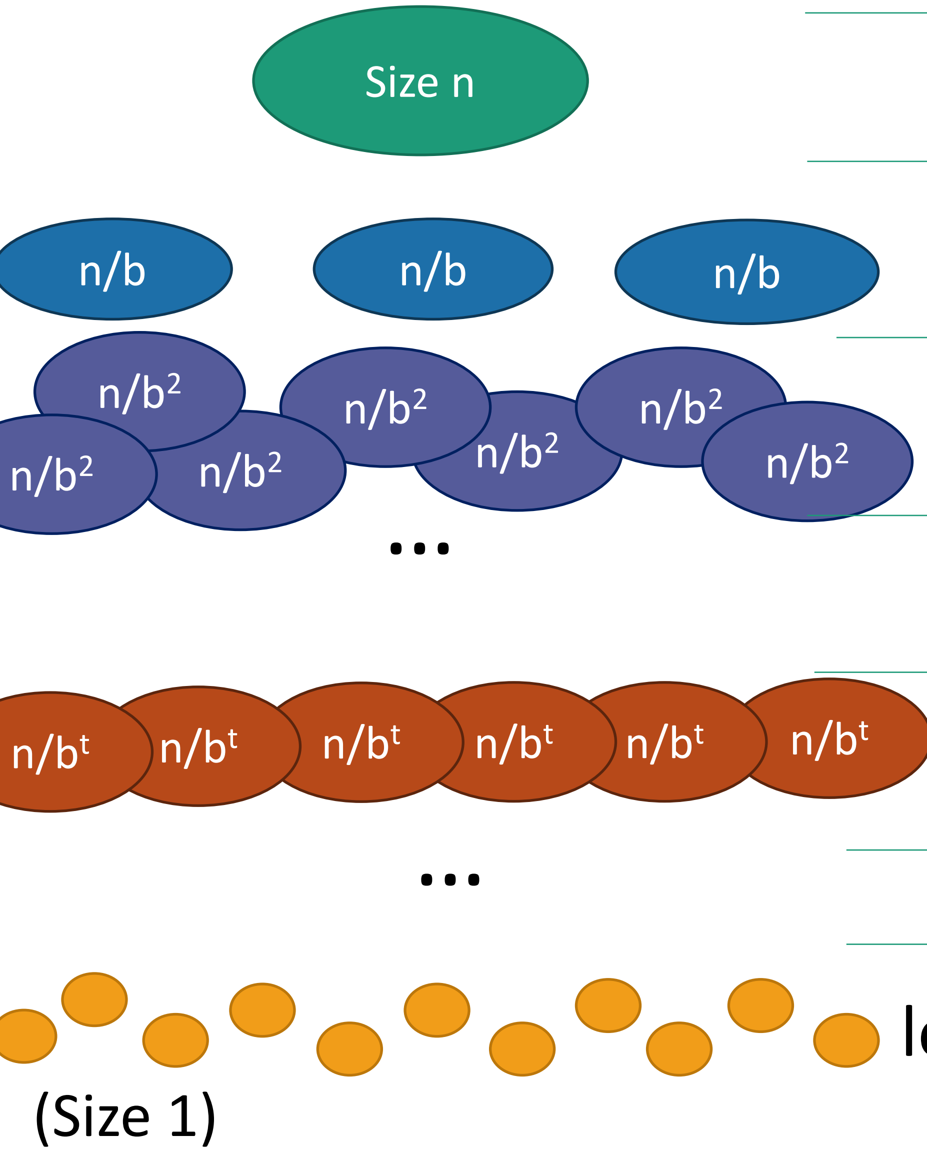
$$= n\left(\frac{2^{\log n + 1} - 1}{2 - 1}\right)$$

- So $T_2(n) = O(n^2)$



Recursion tree

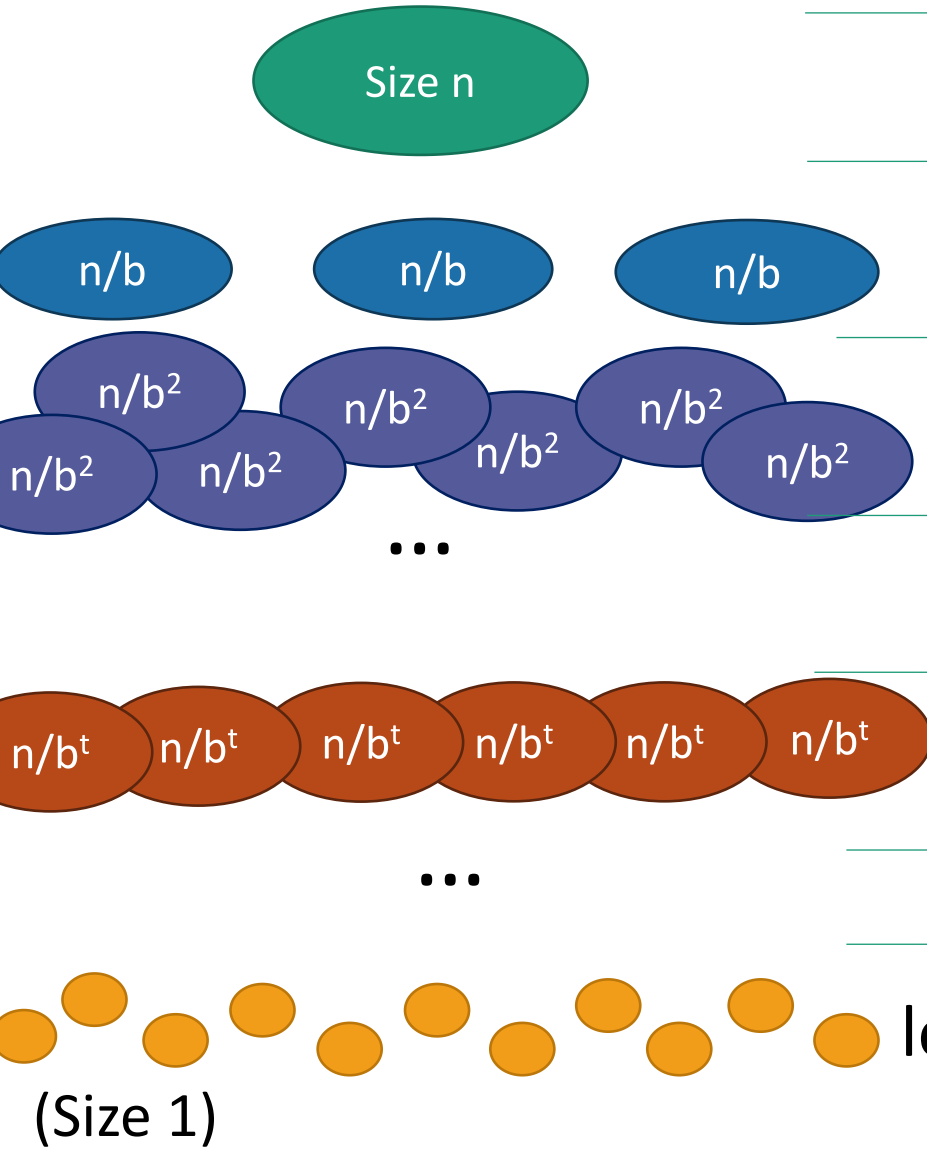
$$T(n) = a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$$



Level	# problems	Size of each problem	Amount of work at this level
0	1	n	
1	a	n/b	
2	a^2	n/b^2	
t	a^t	n/b^t	
…			
$\log_b(n)$	$a^{\log_b(n)}$	1	

Recursion tree

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

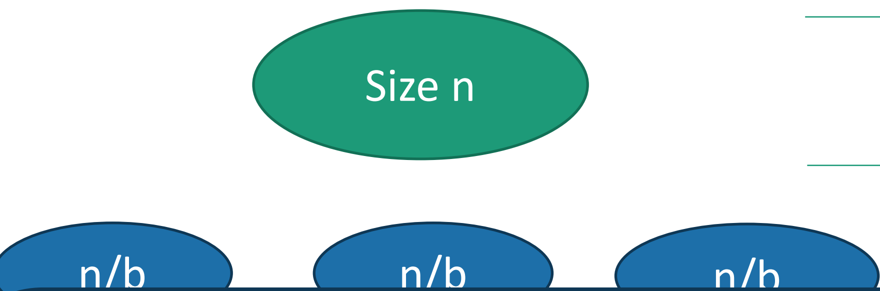


Level	# problems	Size of each problem	Amount of work at this level
0	1	n	$c \cdot n^d$
1	a	n/b	$ac\left(\frac{n}{b}\right)^d$
2	a^2	n/b^2	$a^2 c\left(\frac{n}{b^2}\right)^d$
t	a^t	n/b^t	$a^t c\left(\frac{n}{b^t}\right)^d$
…			
$\log_b(n)$	$a^{\log_b(n)}$	1	$a^{\log_b(n)} c$

(Let's pretend that the base case is T(1) = c for convenience).

Recursion tree

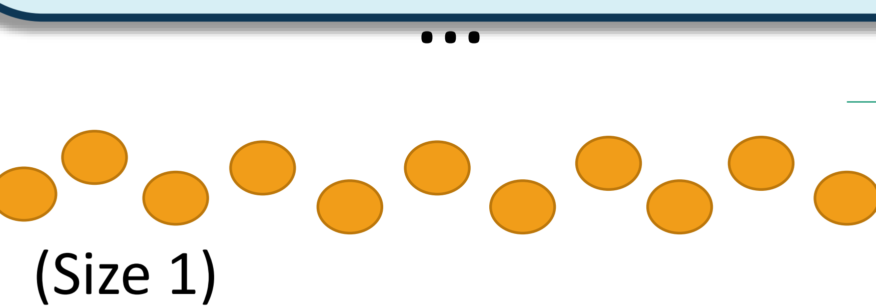
$T(n) = a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$



Level	# problems	Size of each problem	Amount of work at this level
0	1	n	$c \cdot n^d$
1	a	n/b	$ac\left(\frac{n}{b}\right)^d$
			$a^2 c\left(\frac{n}{b^2}\right)^d$
			$a^t c\left(\frac{n}{b^t}\right)^d$
…			
$\log_b(n)$	$a^{\log_b(n)}$	1	$a^{\log_b(n)} c$ (Let's pretend that the base case isT(1) = c for convenience).

Total work (derivation on board) is at most:

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



Now let's check all the cases

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

Case 1: $a = b^d$

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

- $T(n) = c \cdot n^d \cdot \sum_{t=0}^{log_b(n)} \left(\frac{a}{b^d}\right)^t$ Equal to 1!

$= c \cdot n^d \cdot \sum_{t=0}^{log_b(n)} 1$

$= c \cdot n^d \cdot (\log_b(n) + 1)$

$= c \cdot n^d \cdot \left(\frac{\log(n)}{\log(b)} + 1\right)$

$= \Theta\left(n^d \log(n)\right)$

Case 2: $a < b^d$

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

- $T(n) = \ c \cdot n^d \cdot \sum_{t=0}^{log_b(n)} \left(\frac{a}{b^d}\right)^t$ Less than 1!

  $= c \cdot n^d \cdot [\text{some constant}]$

  $= \Theta\left(n^d\right)$

Geometric Series (Aside)

$$1 + x + x^2 + \ ... + \ x^n = \frac{1 - \ x^{n+1}}{1 - x} \ for \ x \ \neq 1$$

OR

$$1 + x + x^2 + \ ... + \ x^n = \frac{x^{n+1} \ - 1}{x \ - 1} \ for \ x \ \neq 1$$

$$1 + x + x^2 + \ ... + \ x^n = \frac{1}{1 - x} \ for \ x < 1$$

Case 3: $a > b^d$

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

- $T(n) = \ c \cdot n^d \cdot \sum_{t=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^t$

Larger than 1!

$$= c \cdot n^d \left(\frac{\left(\frac{a}{b^d}\right)^{\log_b(n)+1} - 1}{\frac{a}{b^d} - 1} \right)$$

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

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

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

Understanding the Master Theorem

- Let $a \geq 1, b > 1,$ and d be constants.
- Suppose $T(n) = a \cdot T\left(\frac{n}{b}\right) + O\left(n^d\right)$. Then

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

- What do these three cases mean?

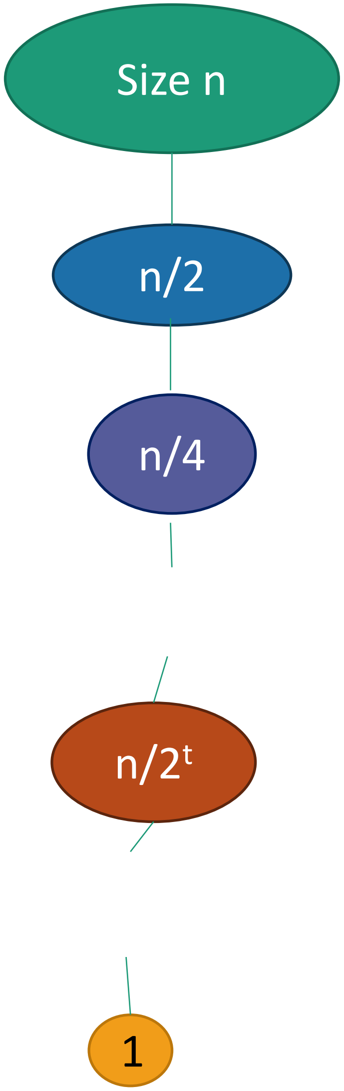
Consider our three warm-ups

1. $T(n) = T\left(\frac{n}{2}\right) + n$
2. $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$
3. $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n$

First example

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

1. $T(n) = T\left(\frac{n}{2}\right) + n, \qquad \left(a < b^d\right)$



- The amount of work done at the top (the biggest problem) is higher than the amount of work done anywhere else.

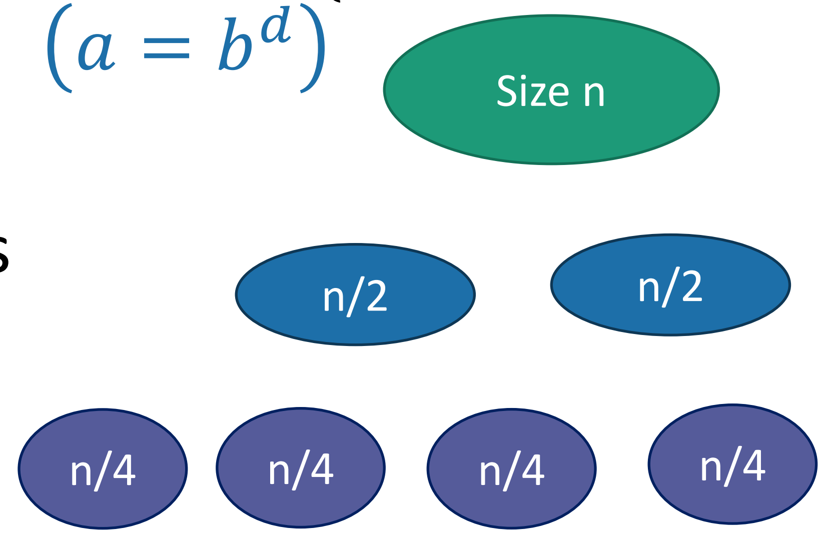
- T(n) = O(work at top) = O(n)

Most work at the top of the tree!

Second example

$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

2. $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n, \qquad \left(a = b^d\right)$



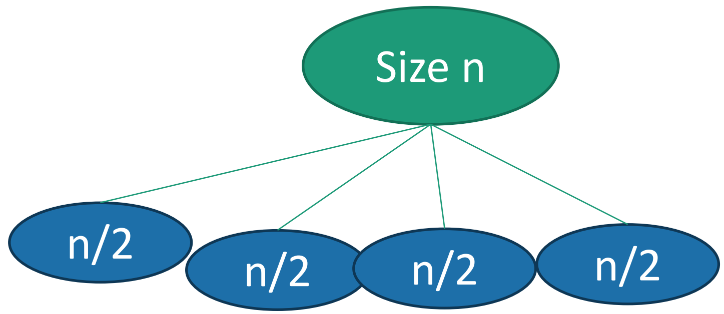
- The branching **just** balances out the amount of work.
- The same amount of work is done at every level.

- T(n) = (number of levels) * (work per level)
- = log(n) * O(n) = O(nlog(n))

Third example

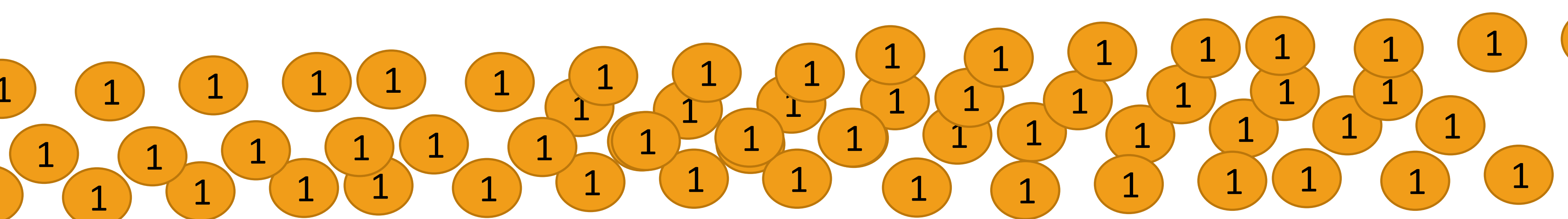
$$T(n) = \begin{cases} O\left(n^d \log(n)\right) & \text{if } a = b^d \\ O\left(n^d\right) & \text{if } a < b^d \\ O\left(n^{\log_b(a)}\right) & \text{if } a > b^d \end{cases}$$

3. $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n, \qquad \left(a > b^d\right)$



Most work at the bottom of the tree!

- There are a HUGE number of leaves, and the total work is dominated by the time to do work at these leaves.
- T(n) = O(work at bottom) = O($4^{\text{depth of tree}}$) = O(n^2)



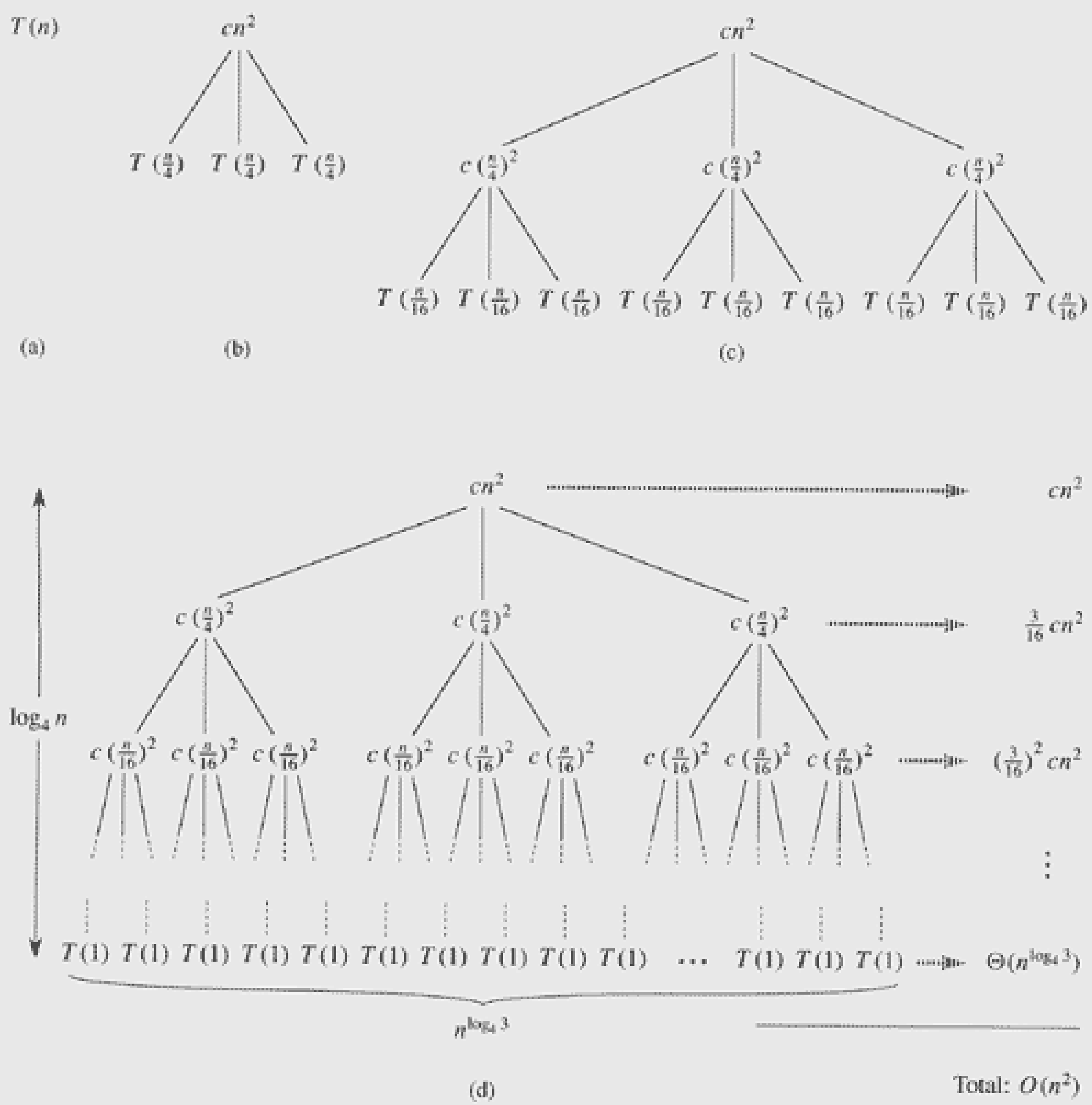


Figure 4.1 The construction of a recursion tree for the recurrence $T(n) = 3T(n/4) + cn^2$. Part **(a)** shows $T(n)$, which is progressively expanded in **(b)–(d)** to form the recursion tree. The fully expanded tree in part **(d)** has height $\log_4 n$ (it has $\log_4 n + 1$ levels).

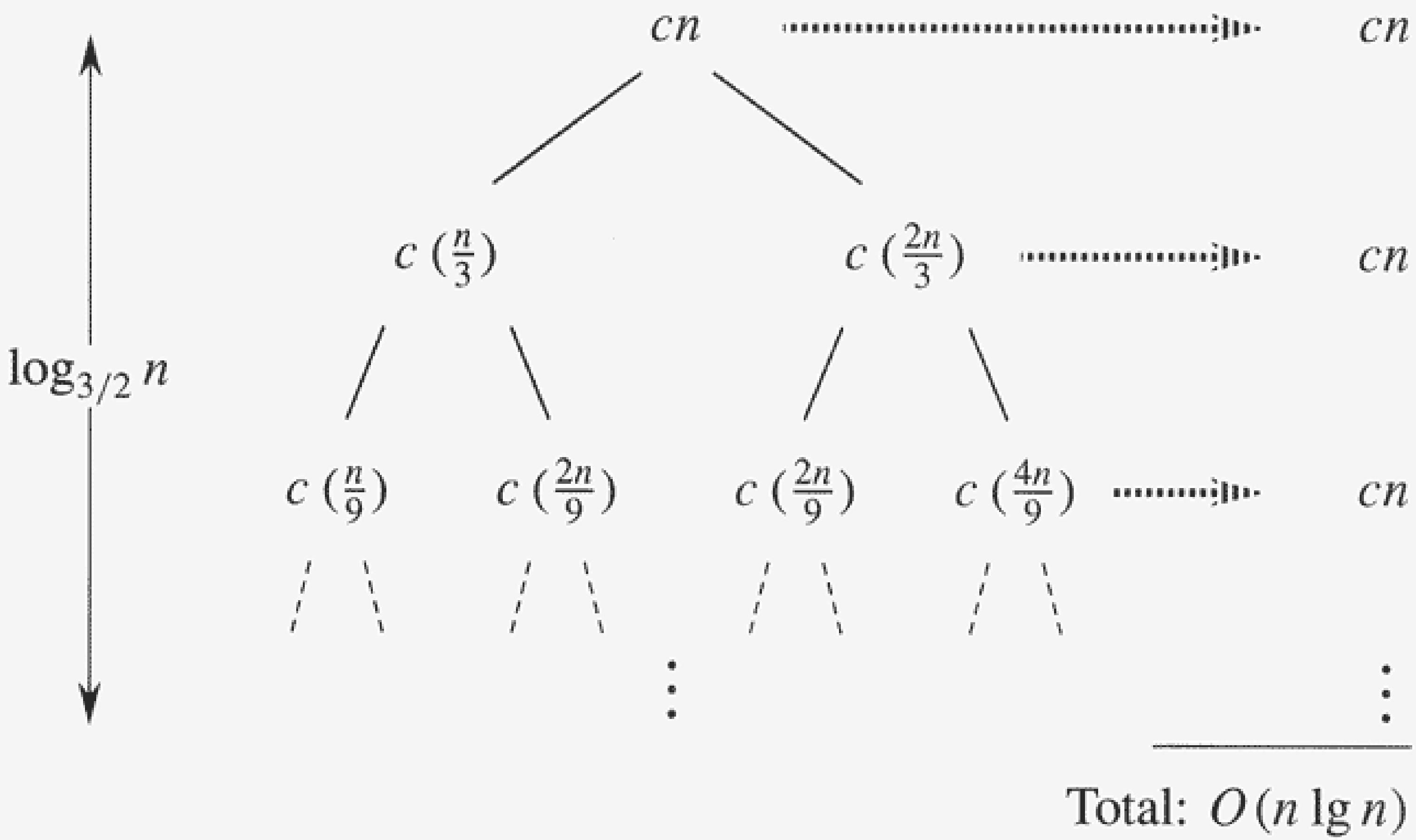


Figure 4.2 A recursion tree for the recurrence $T(n) = T(n/3) + T(2n/3) + cn$.

Substitution Method

1. Guess the form of the solution or Guess what the answer is
   (iterative substitution: iteratively apply the recurrence equation to itself to find a possible pattern)

2. Prove your guess is correct, using mathematical induction
   (Guess and Test method).

Solving Recurrences by Substitution: Guess-and-Test

$T(n) = 2T(n/2) + n$

Guess (#1)	**$T(n) = O(n)$**	
Inductive Hypothesis	**$T(n) <= cn$**	for some constant $c>0$
Inductive Step	$T(n/2) <= cn/2$	

$T(n) = 2T(n/2) + n$

$T(n) \leq 2 \cdot c(n/2) + n$

$T(n) \leq cn + n$

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

no choice of c could ever make $(c + 1)\, n \leq cn$!

Our guess was wrong!!

Solving Recurrences by Substitution: G #2

$T(n) = 2T(n/2) + n$

Guess (#2) **$T(n) = O(n^2)$**

IH **$T(n) \le cn^2$** for some constant $c>0$

Inductive Step $T(n/2) \le cn^2/4$

$$T(n) = 2T(n/2) + n$$

$$T(n) \le 2 \cdot c(n^2/4) + n$$

$$T(n) \le \frac{cn^2}{2} + n$$

Works for all n as long as c>=2 !!

$cn^2 / 2 + n \le cn^2$

Solving Recurrences by Substitution: G #3

T(n) = 2T(n/2) + n

Guess (#3) **T(n) = O(nlogn)**

IH **T(n) <= cnlogn** for some constant c>0

Inductive Step $T\left(\frac{n}{2}\right) \leq c\ \frac{n}{2}\ \log(\frac{n}{2})$

$$T(n) = 2T(n/2) + n$$

$$T(n) \leq 2 \cdot c\ \frac{n}{2}\ \log(\frac{n}{2}) + n$$

$$T(n) \leq cn\ (logn - \log 2) + n$$

$$T(n) \leq cn \log n - cn + n$$

Thus $T(n) \leq cn \log n - cn + n$ <= cnlogn

Works for all n as long as c>=1 !!

Guess and Test Method by Substitution: Ex #2, G # 1

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn\log n & \text{if } n \geq 2 \end{cases}$$

Guess (# 1) $T(n) = O(n \log n)$

(Inductive Hypothesis): $T(n) \leq c\, n \log n$ for $c > 0$

Inductive step, Assume $T\left(\frac{n}{2}\right) \leq c\, \frac{n}{2} \log(\frac{n}{2})$

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

$$T(n) \leq 2 \cdot c\, \frac{n}{2} \log(\frac{n}{2}) + bnlogn$$

$$T(n) \leq cn\,(logn - \log 2) + bnlogn$$

$$T(n) \leq cn \log n - cn + bnlogn$$

$$T(n) \leq (c + b)n \log n - cn$$

Wrong: we cannot make this last line be less than cn log n

Guess and Test Method by Substitution: Ex #2, G # 2

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn\log n & \text{if } n \geq 2 \end{cases}$$

Guess (# 1) $T(n) = O(n \log^2 n)$

(Inductive Hypothesis): $T(n) \leq c\, n \log^2 n$ for $c > 0$

Inductive step, Assume $T\left(\frac{n}{2}\right) \leq c\, \frac{n}{2} \log^2(\frac{n}{2})$

$$T(n) = 2T\left(\frac{n}{2}\right) + \text{bnlogn}$$

$$T(n) \leq 2 \cdot c\, \frac{n}{2} \log^2(\frac{n}{2}) + \text{bnlogn}$$

$$T(n) \leq cn\,(logn - \log 2)^2 + \text{bnlogn}$$

$$T(n) \leq cn\log^2 n - 2cnlogn + cn + \text{bnlogn}$$

$$T(n) \leq cn\log^2 n + (b - 2c)nlogn + cn$$

if $c > b$.

So, $T(n)$ is $O(n \log^2 n)$.

In general, to use this method, you need to have a good guess and you need to be good at induction proofs.