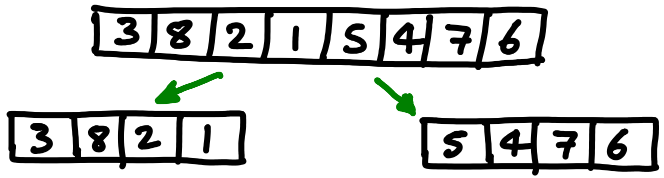
Back to Sorting: Merge Sort

a Divide-and-Conquer algorithm.

Back to Sorting: Merge Sort

a Divide-and-Conquer algorithm.

1) Divide the problem into 2 smaller instances.

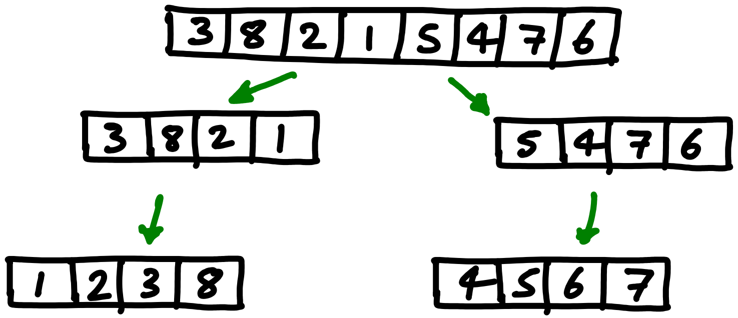


Back to Sorting: Merge Sort

a Divide-and-Conquer algorithm.

1) Divide the problem into 2 smaller instances.

2) "Conquer" = solve the smaller problems



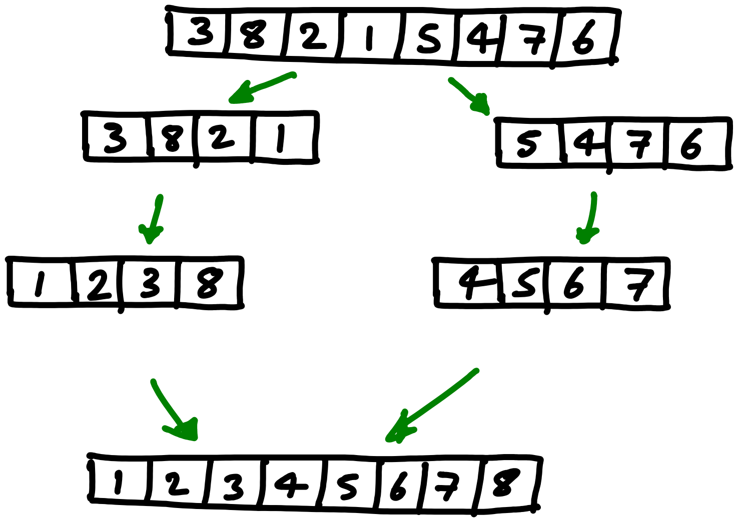
Back to Sorting : Merge Sort

a Divide-and-Conquer algorithm.

1) Divide the problem into 2 smaller instances.

2) "Conquer" = solve the smaller problems

3) Combine = merge the solutions



Merge two sorted arrays

A:

1	2	3	5	8	9	14	16	17	18

B:

4	6	7	10	11	12	13	15	19	20

Merge two sorted arrays

Smallest # is at beginning of A or B : $\underset{A}{1}$ v. 4

A:

1	2	3	5	8	9	14	16	17	18

B:

4	6	7	10	11	12	13	15	19	20

Merge two sorted arrays

A:	1	2	3	5	8	9	14	16	17	18
B:	4	6	7	10	11	12	13	15	19	20

Smallest # is at beginning of A or B : $\underset{A}{1}$ v. 4

Increment index in A

$\underset{A}{2}$ v 4

Merge two sorted arrays

A:

1	2	3	5	8	9	14	16	17	18

B:

4	6	7	10	11	12	13	15	19	20

Smallest # is at beginning of A or B : $\underset{A}{1}$ v. 4

Increment index in A $\underset{A}{2}$ v 4

Increment A $\underset{A}{3}$ v 4

Merge two sorted arrays

A:

1	2	3	5	8	9	14	16	17	18

B:

4	6	7	10	11	12	13	15	19	20

Smallest # is at beginning of A or B : $\underset{A}{1}$ v. 4

Increment index in A $\underset{A}{2}$ v 4

Increment A $\underset{A}{3}$ v 4

Increment A 5 v $\underset{B}{4}$

Merge two sorted arrays

A:

1	2	3	5	8	9	14	16	17	18

B:

4	6	7	10	11	12	13	15	19	20

Smallest # is at beginning of A or B : $\underset{A}{1}$ v. 4

Increment index in A $\underset{A}{2}$ v 4

Increment A $\underset{A}{3}$ v 4

Increment A 5 v $\underset{B}{4}$

Increment B $\underset{A}{5}$ v 6

Merge two sorted arrays

A:	1	2	3	5	8	9	14	16	17	18
B:	4	6	7	10	11	12	13	15	19	20

Smallest # is at beginning of A or B : $\underset{A}{1}$ v. 4

Increment index in A $\underset{A}{2}$ v 4

Increment A $\underset{A}{3}$ v 4

Increment A   5 v $\underset{B}{4}$

Increment B $\underset{A}{5}$ v 6

etc

each time store smallest in a new array

$\Theta(n)$ time

MergeSort time complexity $= T(n)$

MergeSort time complexity $= T(n)$

1) Divide : ?

MergeSort time complexity $= T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

Merge Sort time complexity $= T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

2) Conquer : ?

MergeSort time complexity $= T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

2) Conquer : $\Theta(1) + 2T\left(\frac{n}{2}\right)$ to make 2 recursive calls

MergeSort time complexity $= T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

2) Conquer : $\Theta(1) + 2T\left(\frac{n}{2}\right)$ to make 2 recursive calls

3) Merge : ?

Merge Sort time complexity = $T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

2) Conquer : $\Theta(1) + 2T\left(\frac{n}{2}\right)$ to make 2 recursive calls

3) Merge : $\Theta(n)$

time = ?

MergeSort time complexity = $T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

2) Conquer : $\Theta(1) + 2T\left(\frac{n}{2}\right)$ to make 2 recursive calls

3) Merge : $\Theta(n)$

time = ?

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

Merge Sort time complexity $= T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

2) Conquer : $\Theta(1) + 2T\left(\frac{n}{2}\right)$ to make 2 recursive calls

3) Merge : $\Theta(n)$

time = ?

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

$$T(1) = \Theta(1)$$

Merge Sort time complexity $= T(n)$

1) Divide : $\Theta(1)$ time (identify index of split)

2) Conquer : $\Theta(1) + 2T\left(\frac{n}{2}\right)$ to make 2 recursive calls

3) Merge : $\Theta(n)$

time = ?

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

} Actually $T(n) = T\left(\lceil \frac{n}{2} \rceil\right) + T\left(n - \lceil \frac{n}{2} \rceil\right) + \Theta(n)$

this is just a detail

$$T(1) = \Theta(1)$$

How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$ $[T(1) = \Theta(1)]$

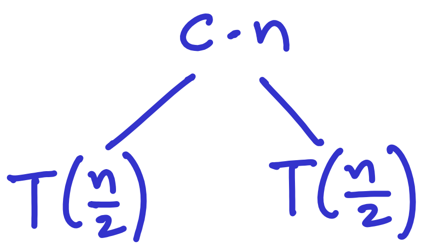
first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T(\frac{n}{2}) + c \cdot n$

How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree: first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$

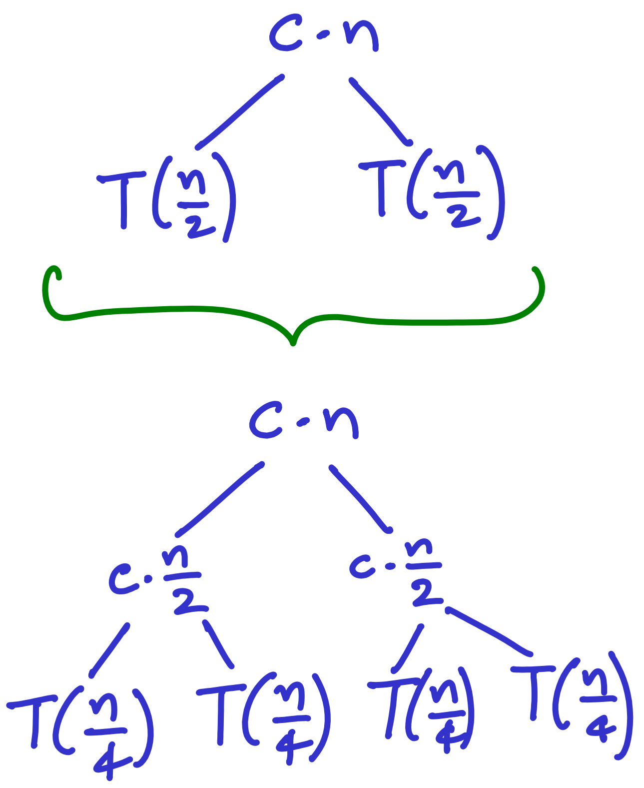


How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \rightarrow c \cdot n$

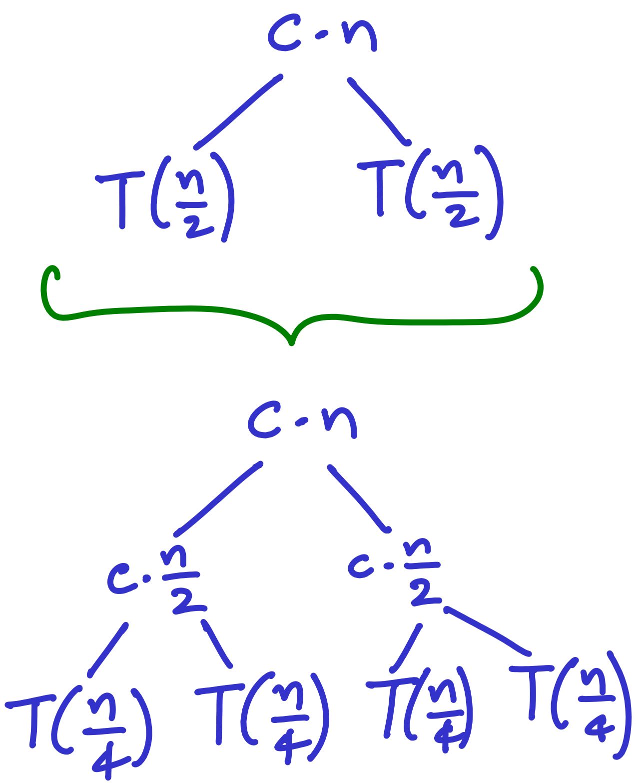
i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$

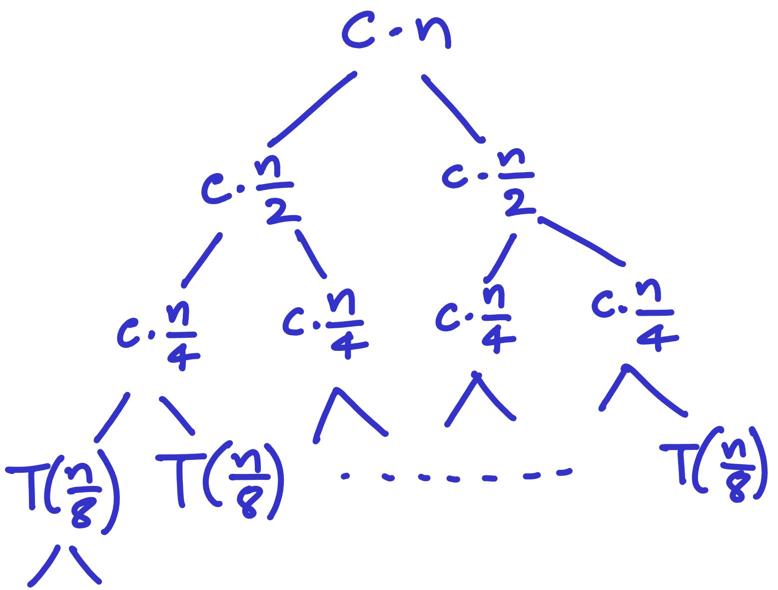


How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree: first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$



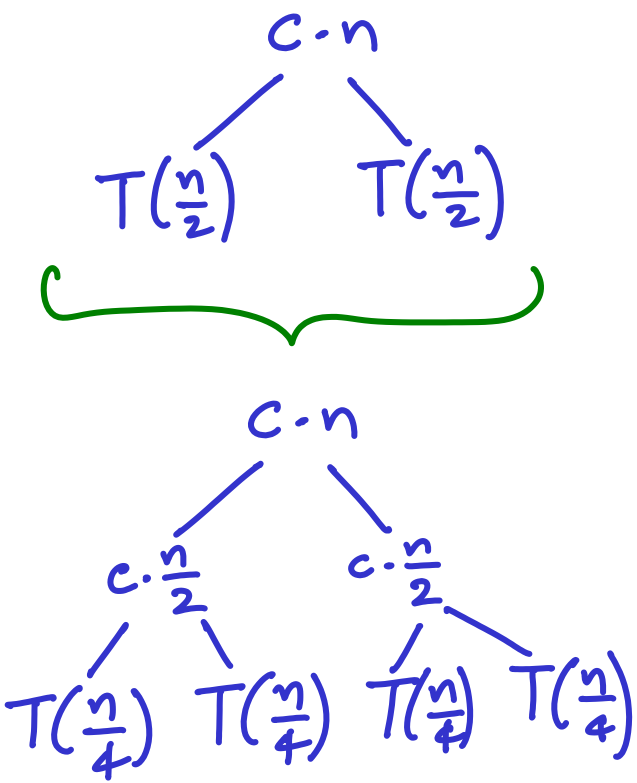


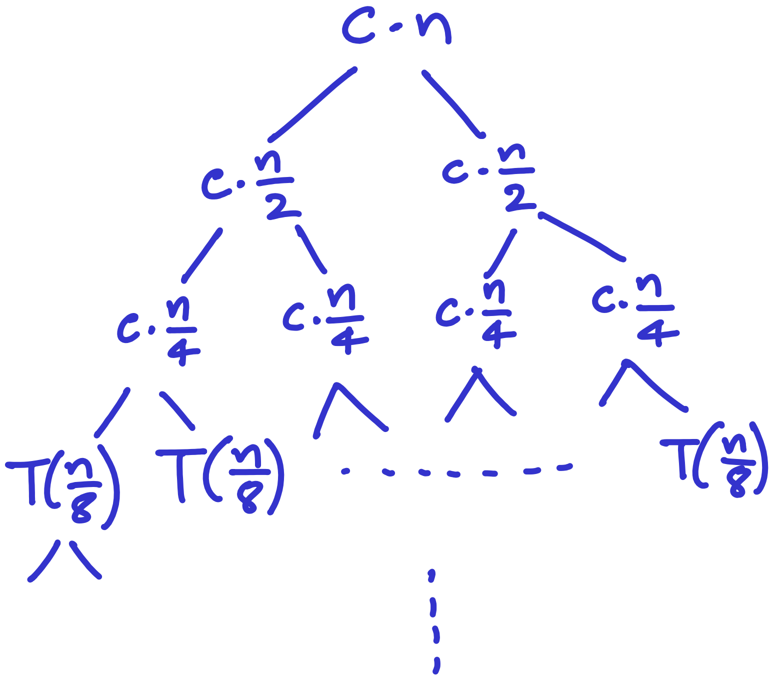
How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T(\frac{n}{2}) + c \cdot n$





$\Theta(1)$ for leaves

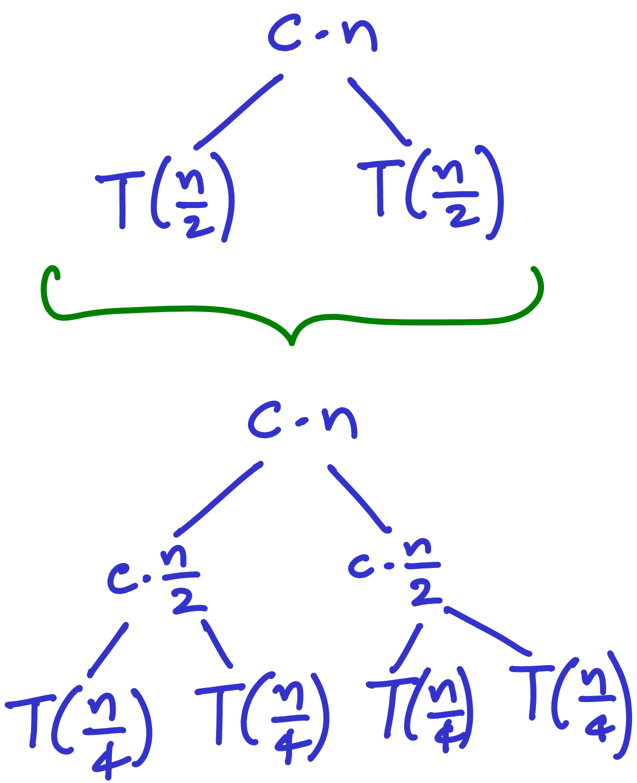
#leaves?

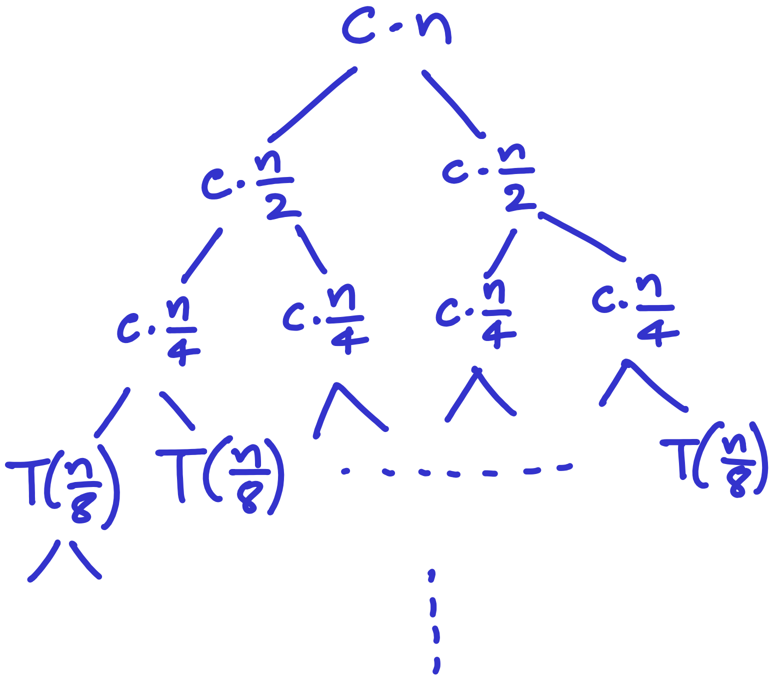
How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$





$\Theta(1)$ for leaves

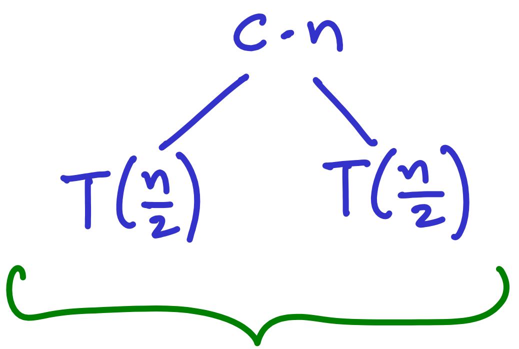
#leaves? ↳ n

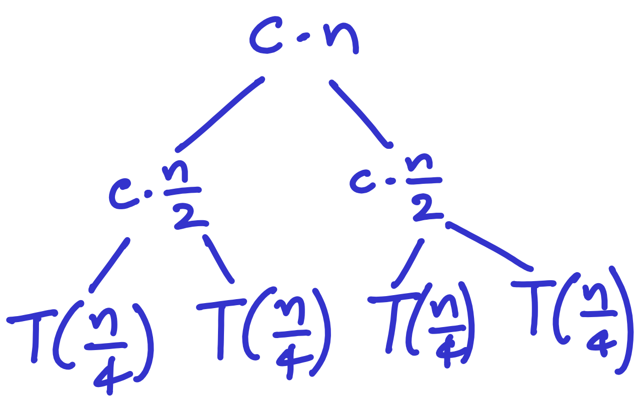
disjoint subproblems

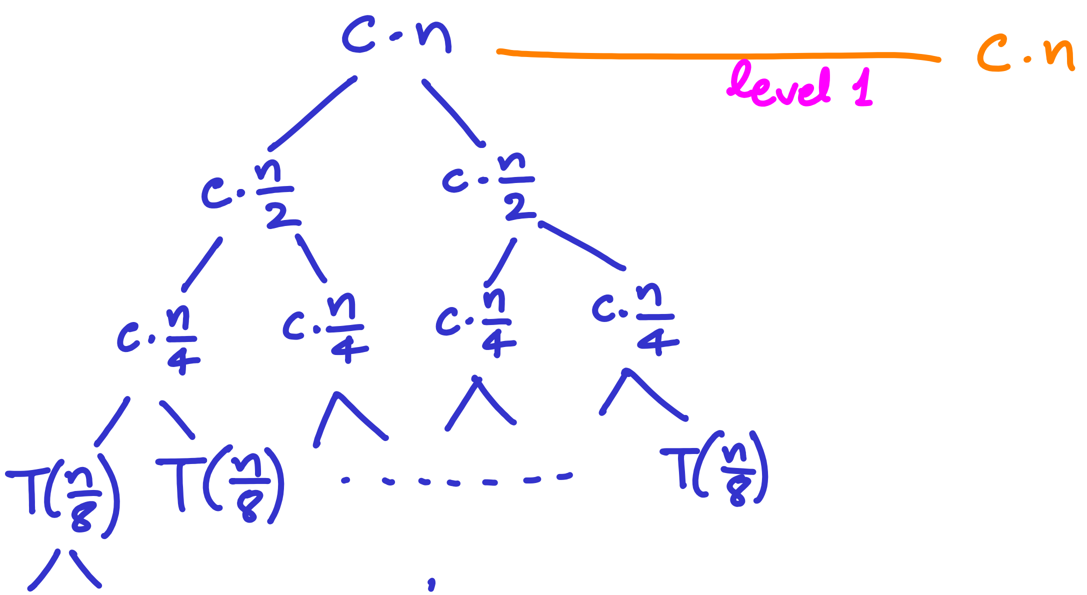
How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree: first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$







$\Theta(1)$ for leaves

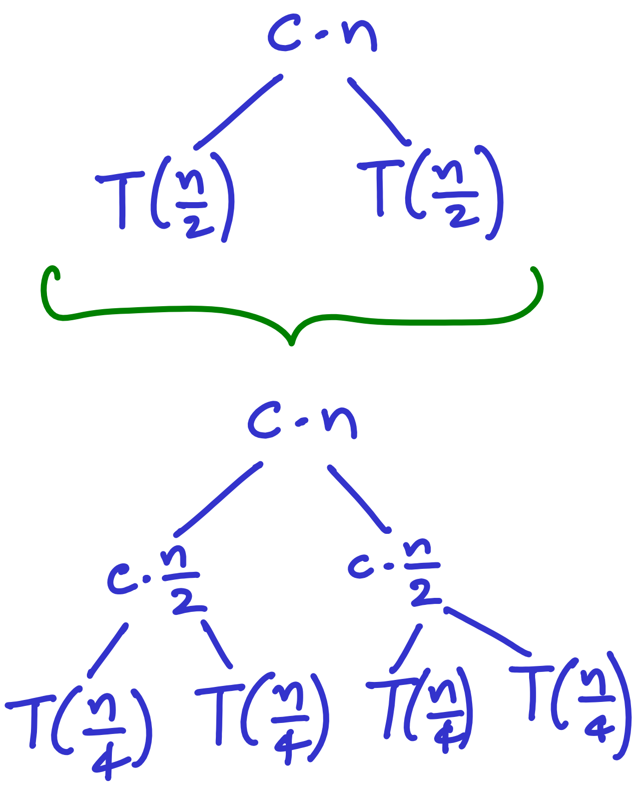
#leaves?
↳ n

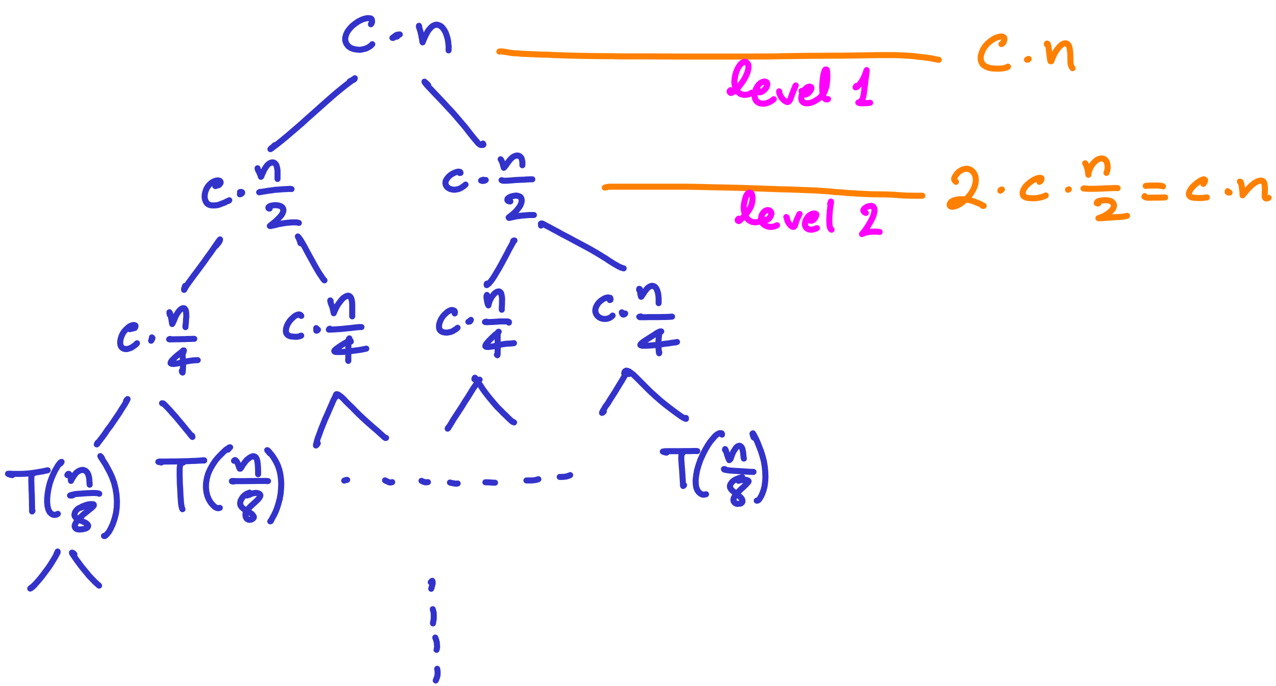
How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$ $[T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T(\frac{n}{2}) + c \cdot n$





$\Theta(1)$ for leaves

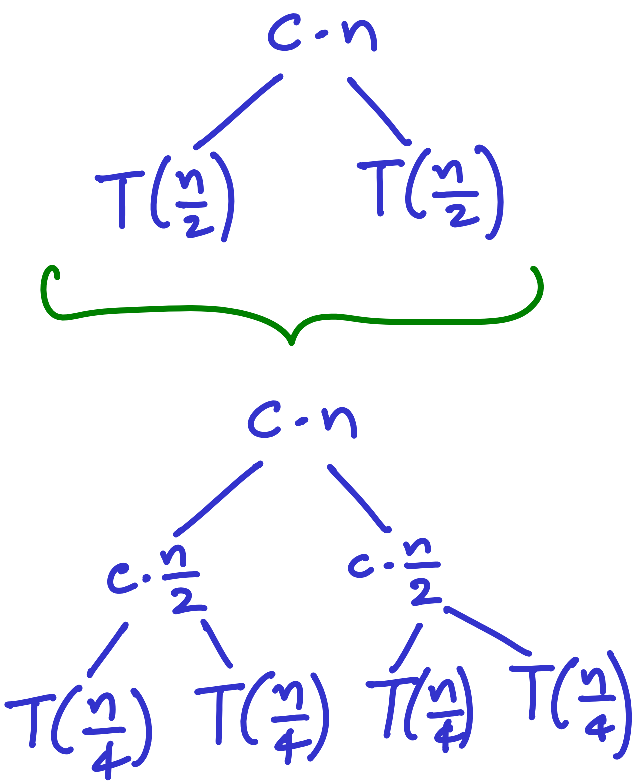
#leaves? $\hookrightarrow n$

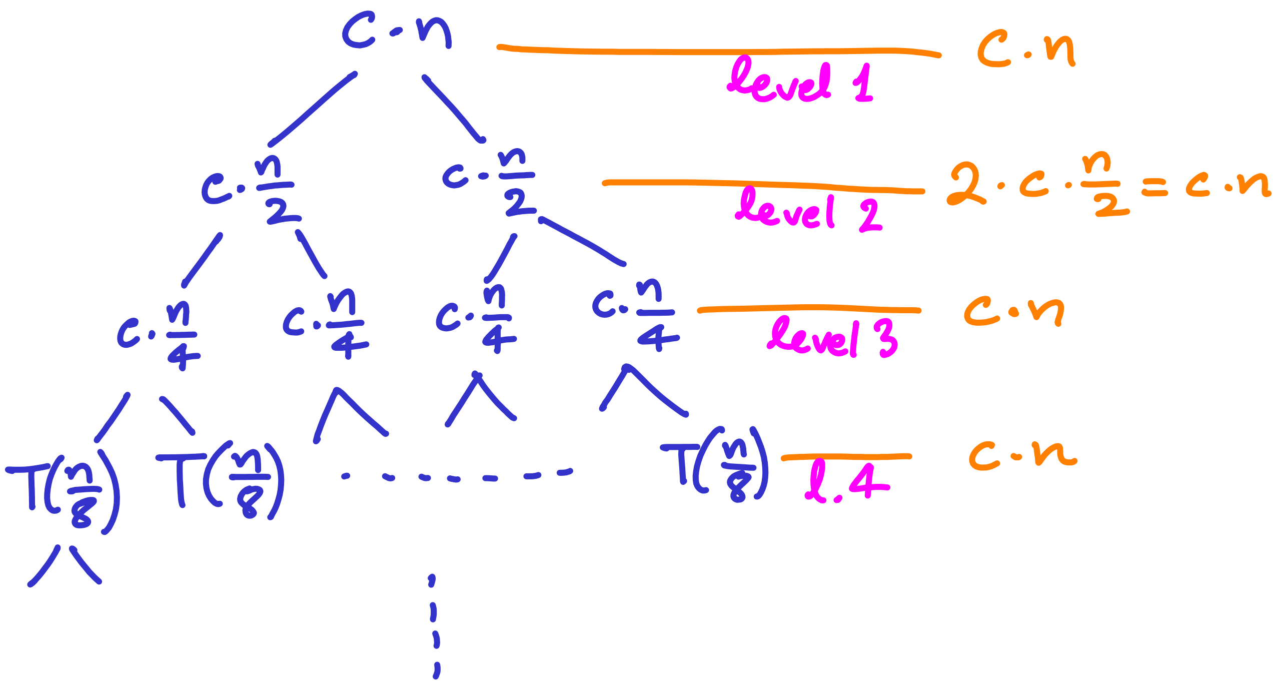
How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$





$\Theta(1)$ for leaves

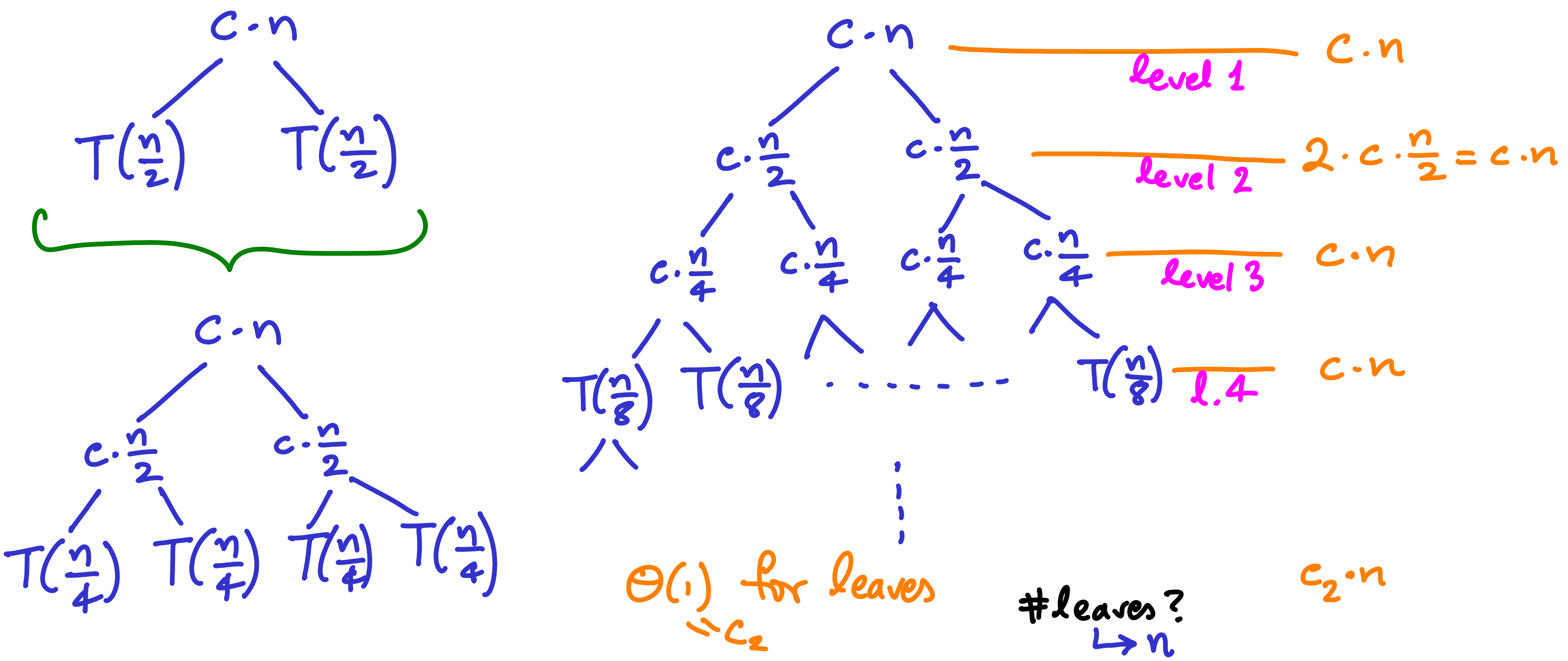
#leaves? ↳ n

How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \to c \cdot n$

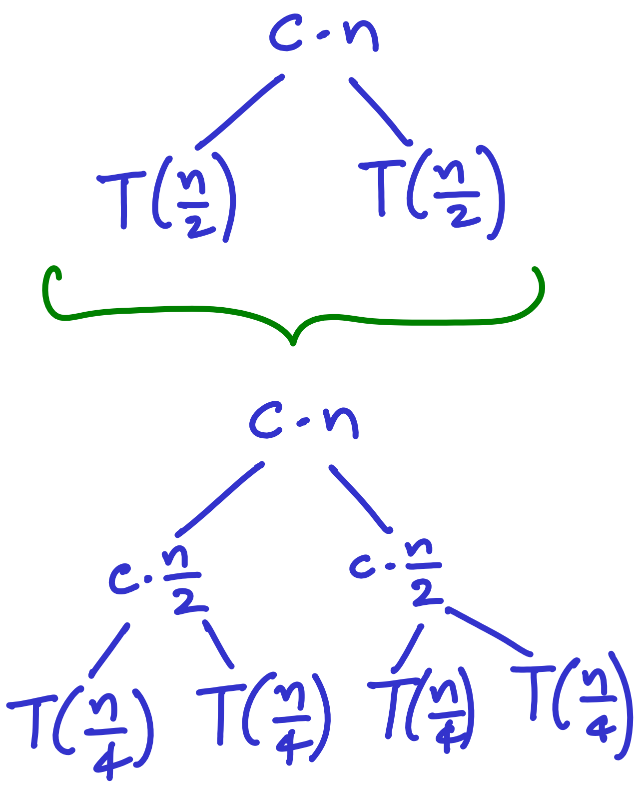
i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$

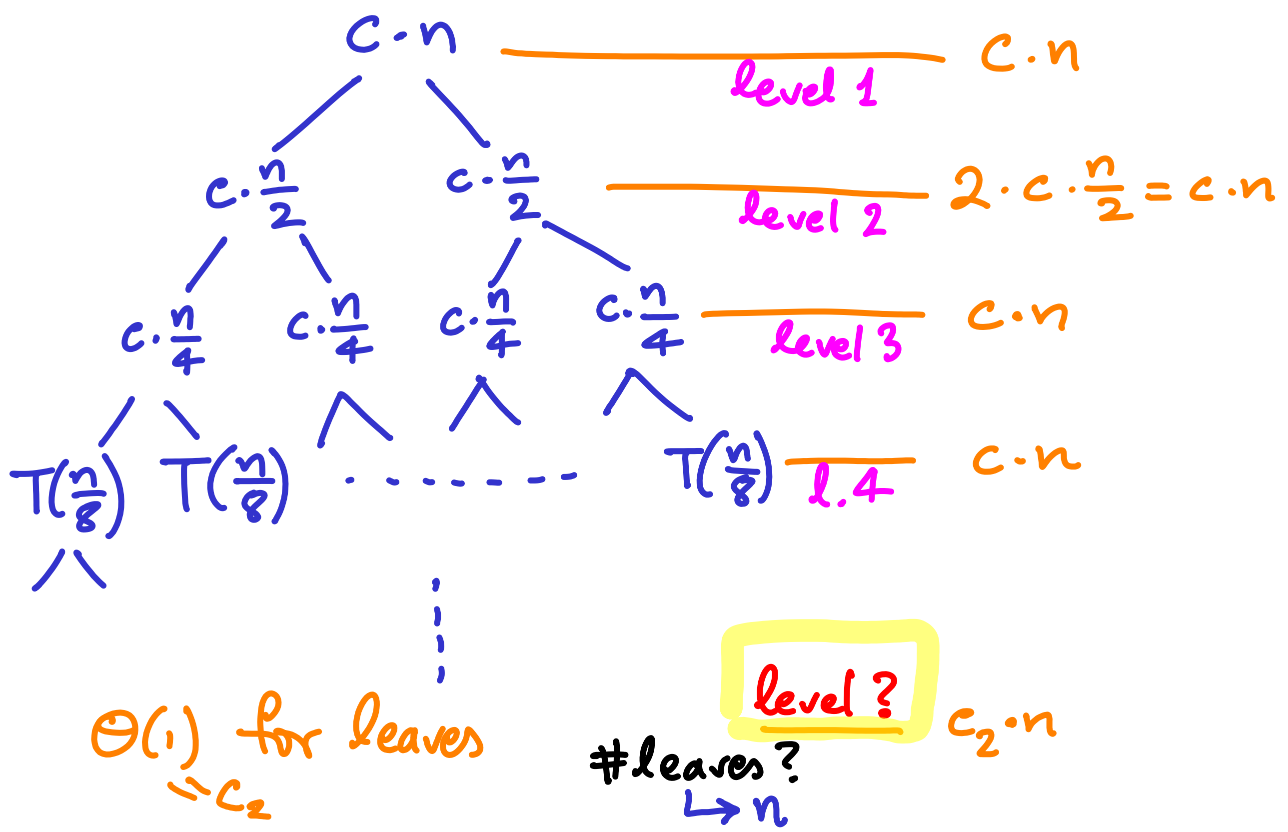


How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive **recursion tree**: first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$



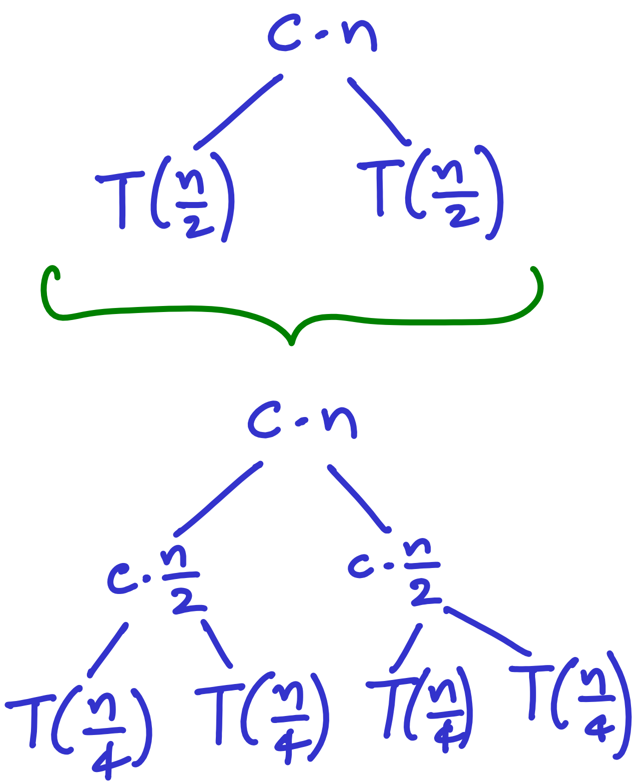


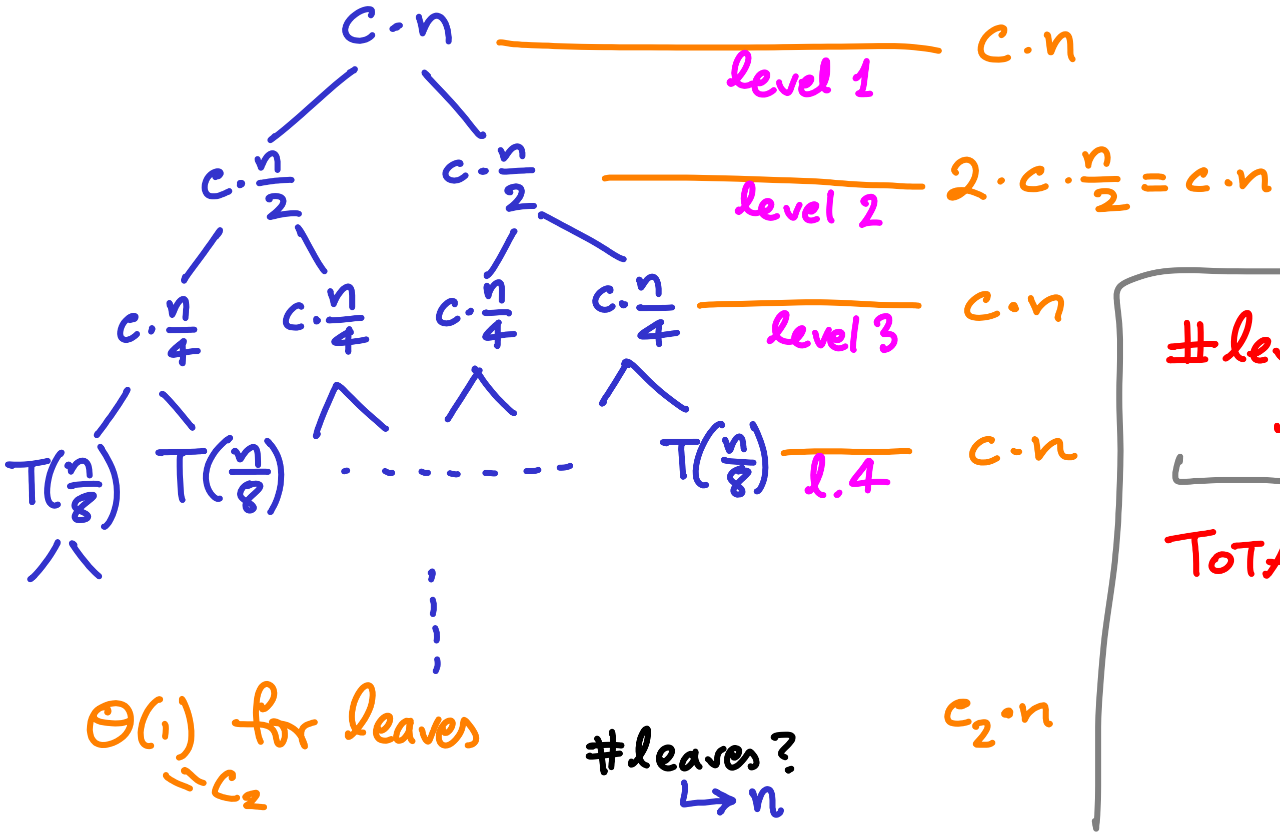
How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \to c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$





$\#$levels $= \log_2 n$

TOTAL WORK

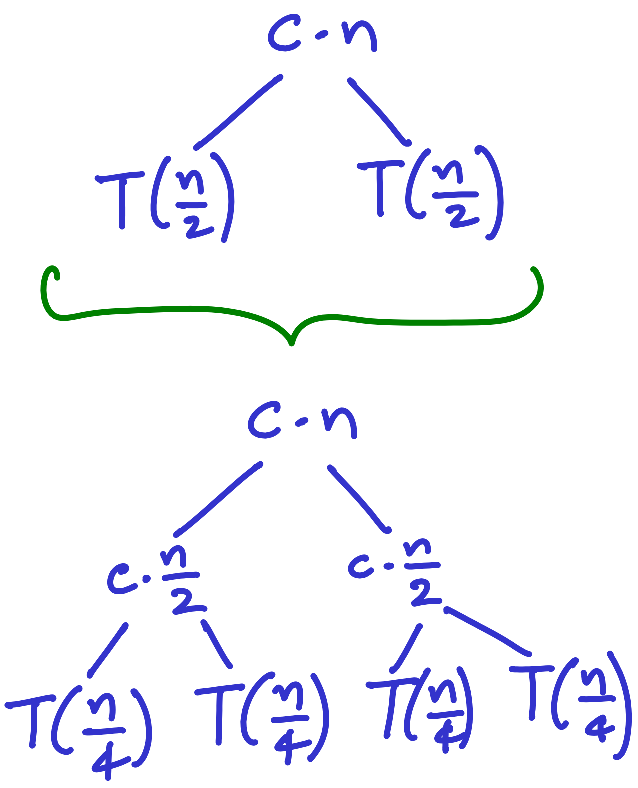
?

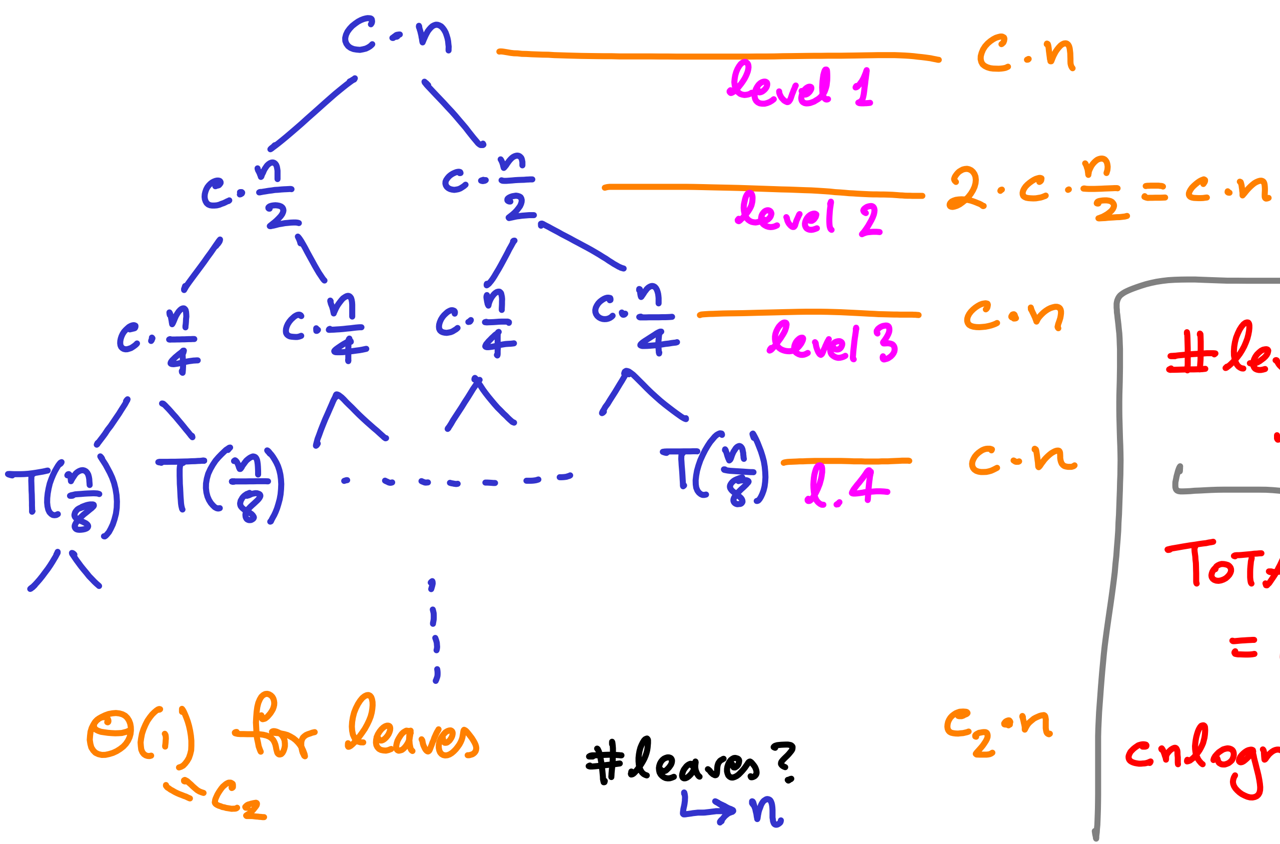
How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ $\quad [T(1) = \Theta(1)]$

The intuitive recursion tree:

first specify $\Theta(n) \rightarrow c \cdot n$

i.e. $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$





#levels $= \log_2 n$

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

$cn \log n + c_2 n$

How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

The more formal <u>substitution method</u>

How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Just focus on upper bound

The more formal <u>substitution method</u>

Start by guessing the answer. Maybe $O(n \log n)$?

How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$

The more formal substitution method

Start by guessing the answer. Maybe $O(n\log n)$?

Use induction : assume that for $k<n$ $T(k) \leq c \cdot k\log k$.

How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$

The more formal substitution method

Start by guessing the answer. Maybe $O(n\log n)$?

Use induction : assume that for $k<n$ $T(k) \leq c \cdot k\log k$.

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

How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$

The more formal substitution method

Start by guessing the answer. Maybe $O(n\log n)$?

Use induction: assume that for $k<n$ $T(k) \leq c \cdot k\log k$.

Substitute: $T(n) \leq 2 \cdot c \cdot \frac{n}{2} \log \frac{n}{2} + \Theta(n) \leq c \cdot n \cdot \log \frac{n}{2} + d \cdot n$

How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

The more formal <u>substitution method</u>

Start by guessing the answer. Maybe $O(n\log n)$?

Use induction : assume that for $k<n$ $T(k) \leq c \cdot k\log k$.

Substitute : $T(n) \leq 2\cdot c \cdot \frac{n}{2}\log\frac{n}{2} + \Theta(n) \leq c\cdot n \cdot \log\frac{n}{2} + d\cdot n$

$$= \underbrace{c\cdot n\log n}_{\text{desired form}} - c\cdot n\log 2 + d\cdot n$$

How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$

The more formal substitution method

Start by guessing the answer. Maybe $O(n\log n)$?

Use induction: assume that for $k<n$ $T(k) \le c\cdot k\log k$.

Substitute: $T(n) \le 2\cdot c\cdot \frac{n}{2}\log\frac{n}{2} + \Theta(n)$

$$\le c\cdot n\cdot \log\frac{n}{2} + d\cdot n$$

$$= c\cdot n\log n - c\cdot n\log 2 + d\cdot n$$

$$= c\cdot n\log n - (c\cdot n - dn)$$

How to solve $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

The more formal substitution method

Start by guessing the answer. Maybe $O(n \log n)$?

Use induction: assume that for $k<n$ $T(k) \leq c \cdot k \log k$.

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

$$\leq c \cdot n \cdot \log \frac{n}{2} + d \cdot n$$

$$= c \cdot n \log n - c \cdot n \log 2 + d \cdot n$$

$$= c \cdot n \log n - (c \cdot n - dn)$$

$$= c \cdot n \log n - \underbrace{(c-d)}_{?} n$$

How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$

The more formal substitution method

Start by guessing the answer. Maybe $O(n\log n)$?

Use induction : assume that for $k<n$ $T(k) \le c \cdot k\log k$.

Substitute : $T(n) \le 2\cdot c \cdot \frac{n}{2}\log\frac{n}{2} + \Theta(n)$

$$\begin{aligned}
&\le c\cdot n\cdot\log\frac{n}{2} + d\cdot n \\
&= c\cdot n\log n - c\cdot n\log 2 + d\cdot n \\
&= c\cdot n\log n - (c\cdot n - dn) \\
&= c\cdot n\log n - (c-d)n
\end{aligned}$$

For $c > d$ we get $T(n) \le cn\log n$

done

How to solve $T(n) = 2T(\frac{n}{2}) + \Theta(n)$

The more formal <u>substitution method</u>

Start by guessing the answer. Maybe $O(n\log n)$?

Just focus on upper bound

Use induction: assume that for $k<n$ $T(k) \le c \cdot k\log k$.

Substitute: $T(n) \le 2 \cdot c \cdot \frac{n}{2} \log \frac{n}{2} + \Theta(n)$

$$\begin{aligned}
&\le c \cdot n \cdot \log \frac{n}{2} + d \cdot n \\
&= c \cdot n\log n - c \cdot n\log 2 + d \cdot n \\
&= c \cdot n\log n - (c \cdot n - dn) \\
&= c \cdot n\log n - (c-d)n
\end{aligned}$$

For $c>d$ we get $T(n) \le cn\log n$

done

In this case, you can get a lower bound in a similar way.

That is often not the case

RECURRENCES - SUBSTITUTION METHOD (guessing)

more examples

RECURRENCES - SUBSTITUTION METHOD (guessing)

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

$$T(1) = \Theta(1)$$

RECURRENCES - SUBSTITUTION METHOD (guessing)

$$\left.\begin{array}{l} T(n) = 4T\left(\frac{n}{2}\right) + n \\ T(1) = \Theta(1) \end{array}\right\}$$

twice the input → four times the work (sort of)

↳ suggestions?

Recurrences - Substitution Method (guessing)

$T(n) = 4T(\frac{n}{2}) + n$ twice the input → four times the work (sort of)

$T(1) = \Theta(1)$ suggestions? → guess $O(n^2)$

1	1
2	4*1+2 = 6
4	4*6+4 = 28
8	4*28+8 = 120
16	4*120+16 = 496
32	4*496+32 = 2016 starting to look like 2*n^2

RECURRENCES - SUBSTITUTION METHOD (guessing)

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

twice the input → four times the work (sort of)

suggestions? → guess $O(n^2)$

Try $O(n^3)$ first: assume $T(k) \le c \cdot k^3$ for $k < n$

Recurrences - Substitution Method (guessing)

$T(n) = 4T\left(\frac{n}{2}\right) + n$ — twice the input → four times the work (sort of)

$T(1) = \Theta(1)$ — suggestions? → guess $O(n^2)$

Try $O(n^3)$ first: assume $T(k) \le c \cdot k^3$ for $k < n$

$$T(n) \le 4 \cdot \underbrace{c \cdot \left(\frac{n}{2}\right)^3} + n$$

RECURRENCES - SUBSTITUTION METHOD (guessing)

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

twice the input → four times the work (sort of)

suggestions? → guess $O(n^2)$

Try $O(n^3)$ first: assume $T(k) \le c \cdot k^3$ for $k < n$

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

RECURRENCES - SUBSTITUTION METHOD (guessing)

$T(n) = 4T(\frac{n}{2}) + n$
$T(1) = \Theta(1)$

twice the input → four times the work (sort of)

↳ suggestions? → guess $O(n^2)$

Try $O(n^3)$ first: assume $T(k) \le c \cdot k^3$ for $k < n$

$$T(n) \le 4 \cdot c \cdot \left(\frac{n}{2}\right)^3 + n = \frac{1}{2} \cdot c \cdot n^3 + n = \underbrace{c \cdot n^3}_{\text{want this}} \underbrace{- \frac{1}{2} c n^3 + n}_{\text{minus stuff}}$$

RECURRENCES - SUBSTITUTION METHOD (guessing)

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad T(1) = \Theta(1)$$

twice the input → four times the work (sort of)

suggestions? → guess $O(n^2)$

Try $O(n^3)$ first: assume $T(k) \le c \cdot k^3$ for $k < n$

$$T(n) \le 4 \cdot c \cdot \left(\frac{n}{2}\right)^3 + n = \frac{1}{2} \cdot c \cdot n^3 + n = c \cdot n^3 - \frac{1}{2} c n^3 + n = cn^3 - \underbrace{\left(\frac{1}{2} cn^3 - n\right)}_{?}$$

RECURRENCES - SUBSTITUTION METHOD (guessing)

$$\left.\begin{array}{l} T(n) = 4T\left(\frac{n}{2}\right) + n \\ T(1) = \Theta(1) \end{array}\right\}$$

twice the input → four times the work (sort of)

suggestions? → guess $O(n^2)$

Try $O(n^3)$ first: assume $T(k) \leq c \cdot k^3$ for $k < n$

$$T(n) \leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^3 + n = \frac{1}{2} \cdot c \cdot n^3 + n = c \cdot n^3 - \frac{1}{2}cn^3 + n = \underbrace{cn^3 - \underbrace{\left(\frac{1}{2}cn^3 - n\right)}_{\geq 0 \text{ if } c \geq 2}}_{\text{DONE}}$$

RECURRENCES - SUBSTITUTION METHOD (guessing)

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

twice the input → four times the work (sort of)

↳ suggestions? → guess $O(n^2)$

Try $O(n^3)$ first: assume $T(k) \leq c \cdot k^3$ for $k < n$

$$T(n) \leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^3 + n = \frac{1}{2} \cdot c \cdot n^3 + n = c \cdot n^3 - \frac{1}{2}cn^3 + n = cn^3 - \underbrace{\left(\frac{1}{2}cn^3 - n\right)}_{\geq 0 \text{ if } c \geq 2}$$

Notice c depends on base case

... assume $T(1) \leq c \cdot 1^3$

Pause for a minute : why assume $T(k) \leq c \cdot k^3$ instead of $T(k) = O(k^3)$

↳ $T(n) \leq 4T\left(\frac{n}{2}\right) + n \leq 4 \cdot \underbrace{O\left(\left(\frac{n}{2}\right)^3\right)} + n$

Pause for a minute: why assume $T(k) \leq c \cdot k^3$ instead of $T(k) = O(k^3)$

$\hookrightarrow T(n) \leq 4T\left(\frac{n}{2}\right) + n \leq 4 \cdot O\left(\left(\frac{n}{2}\right)^3\right) + n = \underbrace{O(n^3)} + n = O(n^3)$

done ?!

Pause for a minute : why assume $T(k) \leq c \cdot k^3$ instead of $T(k) = O(k^3)$

$\hookrightarrow$ $T(n) \leq 4T\left(\frac{n}{2}\right) + n \leq 4 \cdot O\left(\left(\frac{n}{2}\right)^3\right) + n = O(n^3) + n = O(n^3)$!?

What about $T(n) = n$? Prove it's $O(1)$: Base case : $T(1) = \text{const} = O(1)$ ✓

Pause for a minute : why assume $T(k) \leq c \cdot k^3$ instead of $T(k) = O(k^3)$

$\hookrightarrow T(n) \leq 4T\left(\frac{n}{2}\right) + n \leq 4 \cdot O\left(\left(\frac{n}{2}\right)^3\right) + n = O(n^3) + n = O(n^3)$!?

What about $T(n) = n$? Prove it's $O(1)$: Base case : $T(1) = \text{const} = O(1)$ ✓

Assume $T(k) = O(1)$ for $k < n$

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

Pause for a minute : why assume $T(k) \leq c \cdot k^3$ instead of $T(k) = O(k^3)$

$T(n) \leq 4T(\frac{n}{2}) + n \leq 4 \cdot O((\frac{n}{2})^3) + n = O(n^3) + n = O(n^3)$!?

What about $T(n) = n$? Prove it's $O(1)$: Base case : $T(1) = \text{const} = O(1)$ ✓

Assume $T(k) = O(1)$

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

so $n = (n-1) + 1 = O(1) + 1 = O(1)$

THIS IS INCORRECT

Don't use Big-O within induction proof

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$

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

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^2 + n$

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^2 + n = c \cdot n^2 + \underbrace{n}_{\text{never negative!}}$

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^2 + n = c \cdot n^2 + \underbrace{n}_{\text{never negative!}}$

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot (\frac{n}{2})^2 + n = c \cdot n^2 + \underbrace{n}_{\text{never negative!}}$

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

we can show $T(n) = c \cdot n^2 + n < c \cdot n^2 + n^2 = (c+1) \cdot n^2$

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^2 + n = c \cdot n^2 + n$

never negative!

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

we can show $T(n) = c \cdot n^2 + n < c \cdot n^2 + n^2 = (c+1) \cdot n^2$

OR $T(n) = cn^2 + n = cn^2 + \frac{1}{n} \cdot n^2 = \left(c + \frac{1}{n}\right) \cdot n^2$

Not good enough

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot (\frac{n}{2})^2 + n = c \cdot n^2 \underbrace{+n}_{\text{never negative!}}$

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

$T(n) = c \cdot n^2 + n < c \cdot n^2 + n^2 = (c+1) \cdot n^2$

Not good enough

Assume $T(k) \leq c_1 \cdot k^2 - c_2 \cdot k$

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad : \text{try for } O(n^2)$$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^2 + n = c \cdot n^2 + \underbrace{n}_{\text{never negative!}}$

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

$$T(n) = c \cdot n^2 + n < c \cdot n^2 + n^2 = (c+1) \cdot n^2$$

Not good enough

Assume $T(k) \leq c_1 \cdot k^2 - c_2 \cdot k$

$$T(n) \leq 4 \cdot \underbrace{\left(c_1 \cdot \left(\frac{n}{2}\right)^2 - c_2 \cdot \frac{n}{2}\right)}_{} + n$$

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot (\frac{n}{2})^2 + n = c \cdot n^2 + \underbrace{n}_{\text{never negative!}}$

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

$T(n) = c \cdot n^2 + n < c \cdot n^2 + n^2 = (c+1) \cdot n^2$

Not good enough

Assume $T(k) \leq c_1 \cdot k^2 - c_2 \cdot k$

$T(n) \leq 4 \cdot \left(c_1 \cdot (\frac{n}{2})^2 - c_2 \cdot \frac{n}{2}\right) + n$

$= c_1 \cdot n^2 - 2c_2 \cdot n + n$

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \leq c \cdot k^2$, so $T(n) \leq 4 \cdot c \cdot (\frac{n}{2})^2 + n = c \cdot n^2 \underbrace{+n}_{\text{never negative!}}$

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

$T(n) = c \cdot n^2 + n < c \cdot n^2 + n^2 = (c+1) \cdot n^2$

Not good enough

Assume $T(k) \leq c_1 \cdot k^2 - c_2 \cdot k$

$$T(n) \leq 4 \cdot \left(c_1 \cdot \left(\frac{n}{2}\right)^2 - c_2 \cdot \frac{n}{2}\right) + n$$

$$= c_1 \cdot n^2 - 2c_2 \cdot n + n$$

$$= \underbrace{c_1 \cdot n^2 - c_2 \cdot n}_{\text{same as } T(k)} \underbrace{- (c_2 - 1)}_{<0 \text{ if } c_2 > 1} \cdot n$$

DONE

$T(n) = 4T(\frac{n}{2}) + n$: try for $O(n^2)$

Assume $T(k) \le c \cdot k^2$, so $T(n) \le 4 \cdot c \cdot \left(\frac{n}{2}\right)^2 + n = c \cdot n^2 + \underbrace{n}_{\text{never negative!}}$

$cn^2 + n$ is $O(n^2)$ but we have commited to a constant, c.

$T(n) = c \cdot n^2 + n < c \cdot n^2 + n^2 = (c+1) \cdot n^2$

Not good enough

Assume $T(k) \le c_1 \cdot k^2 - c_2 \cdot k$

$$T(n) \le 4 \cdot \left(c_1 \cdot \left(\frac{n}{2}\right)^2 - c_2 \cdot \frac{n}{2}\right) + n$$

$$= c_1 \cdot n^2 - 2c_2 \cdot n + n$$

$$= \underbrace{c_1 \cdot n^2 - c_2 \cdot n}_{\text{same as } T(k)} \underbrace{- (c_2 - 1) \cdot n}_{<0 \text{ if } c_2 > 1}$$

DONE

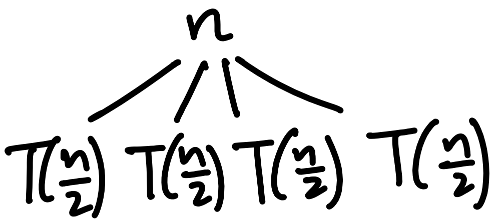
base case

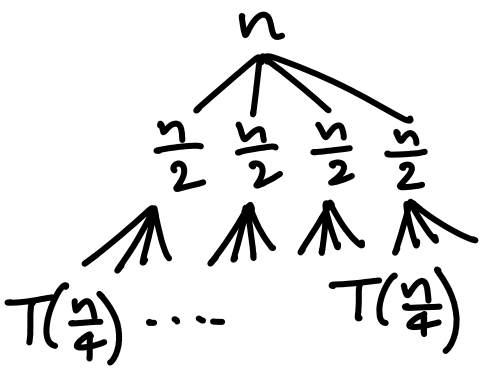
$\underbrace{T(1) \le c_1 - c_2}_{\text{forces a bound on } c_1}$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$ by recursion tree

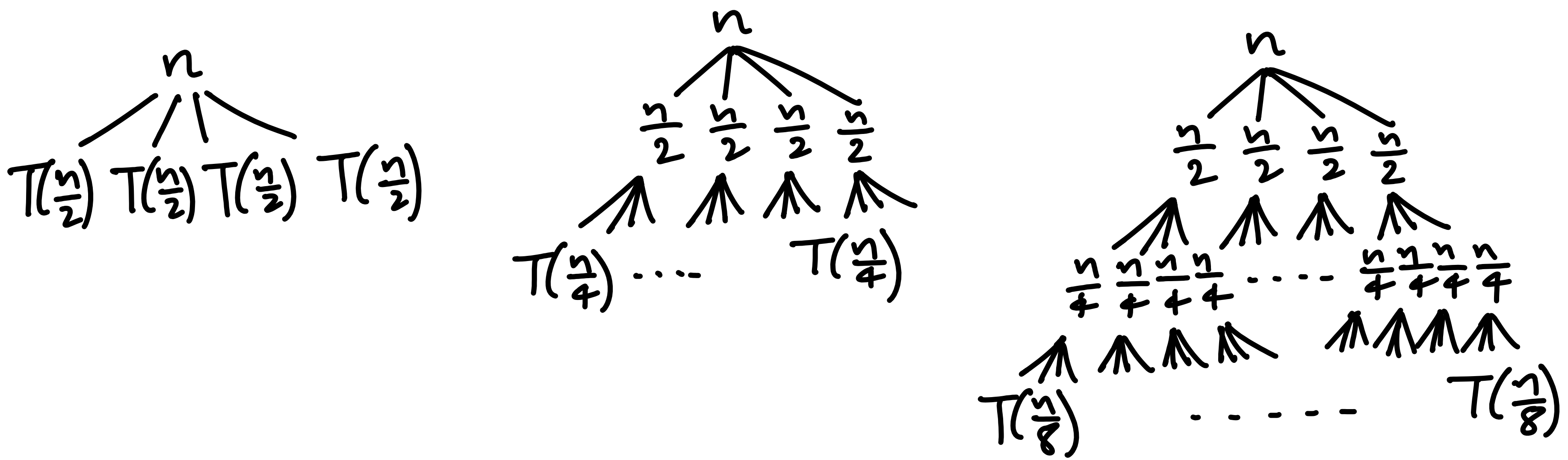
$$\begin{tikzcd}[column sep=small]
 & & n \arrow[dll, no head] \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\
T\left(\frac{n}{2}\right) & T\left(\frac{n}{2}\right) & T\left(\frac{n}{2}\right) & T\left(\frac{n}{2}\right)
\end{tikzcd}$$

$T(n) = 4T\left(\frac{n}{2}\right) + n$ by recursion tree

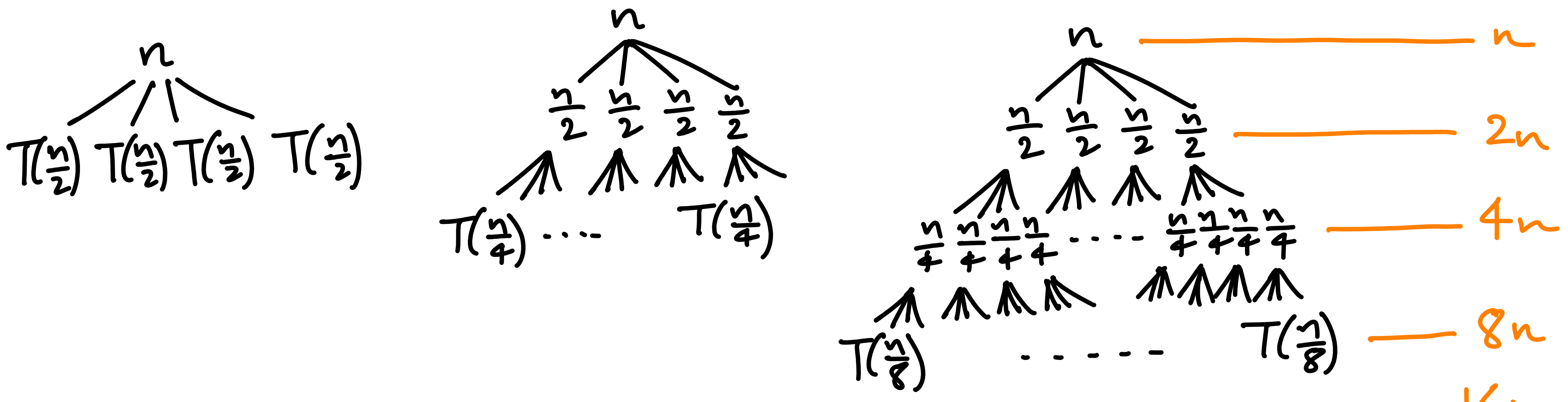




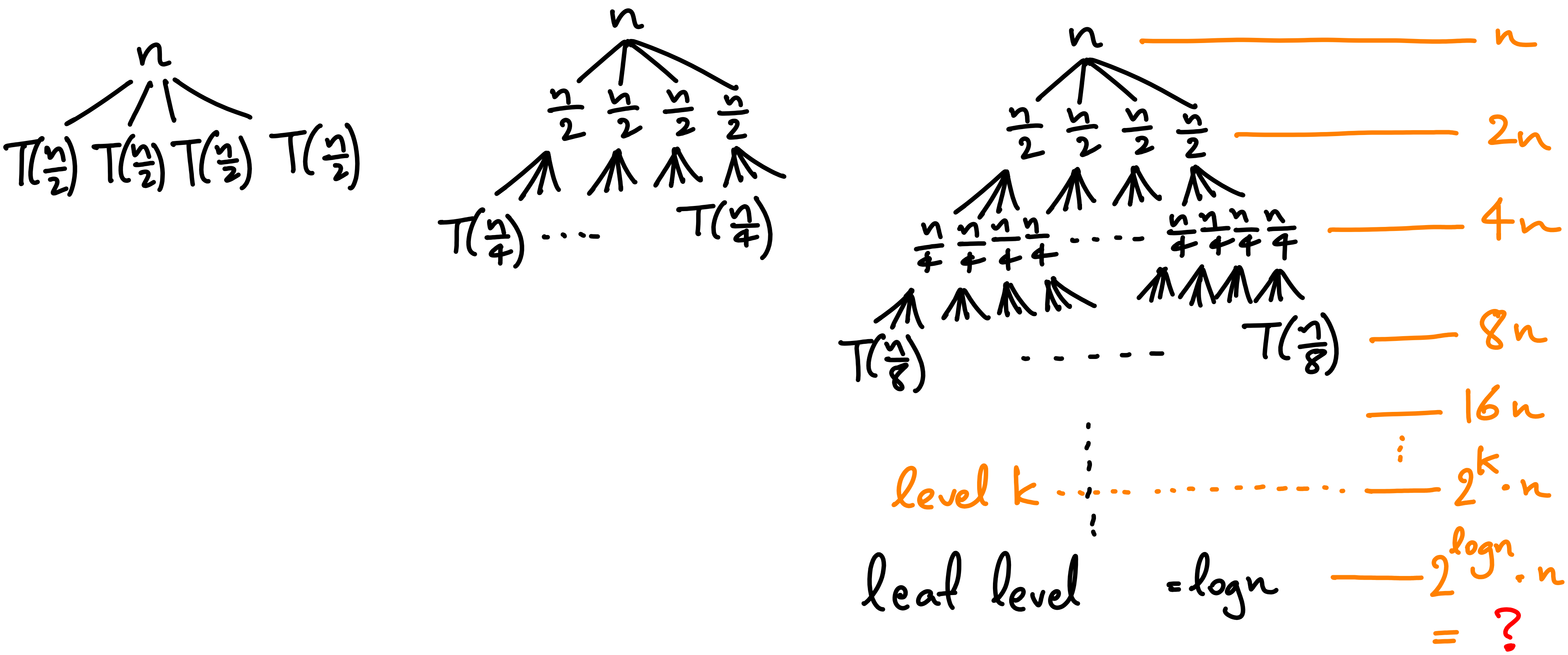
$T(n) = 4T(\frac{n}{2}) + n$ by recursion tree



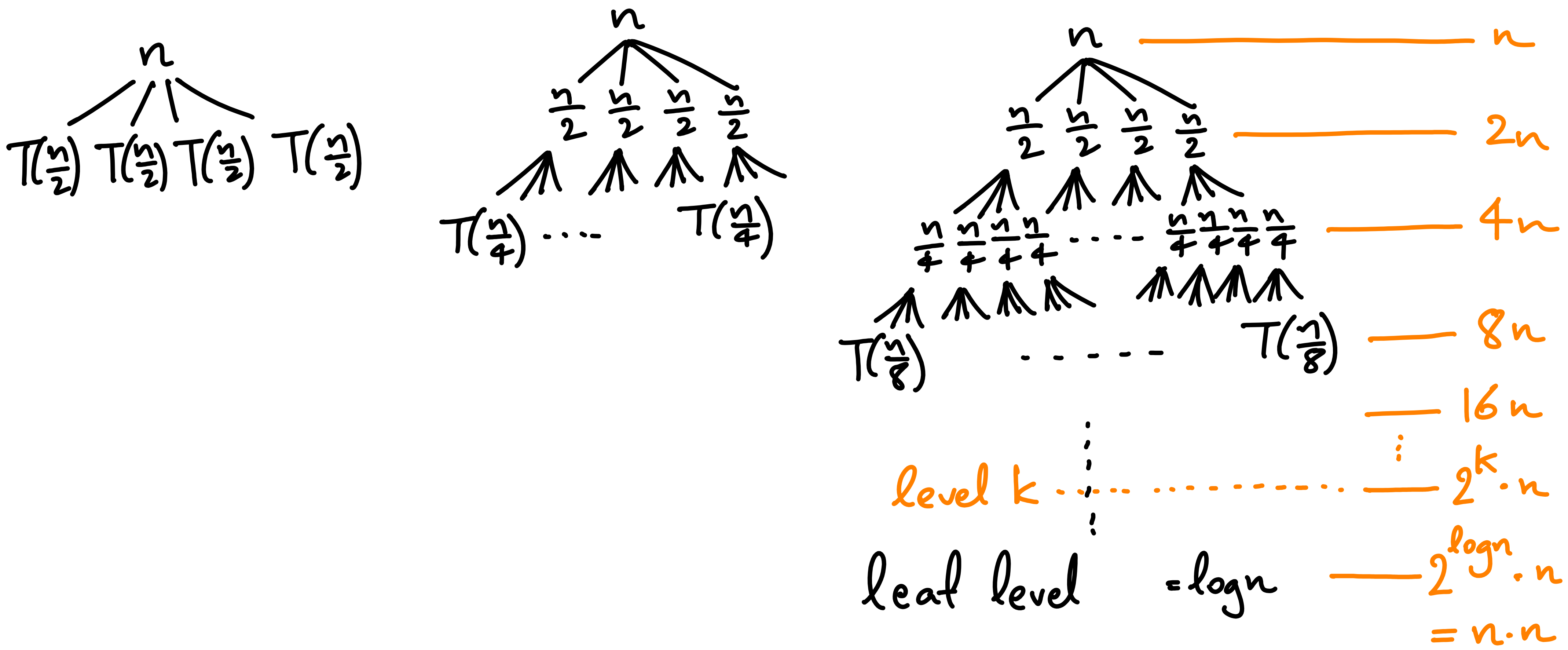
$T(n) = 4T(\frac{n}{2}) + n$ by recursion tree



$T(n) = 4T\left(\frac{n}{2}\right) + n$ by recursion tree

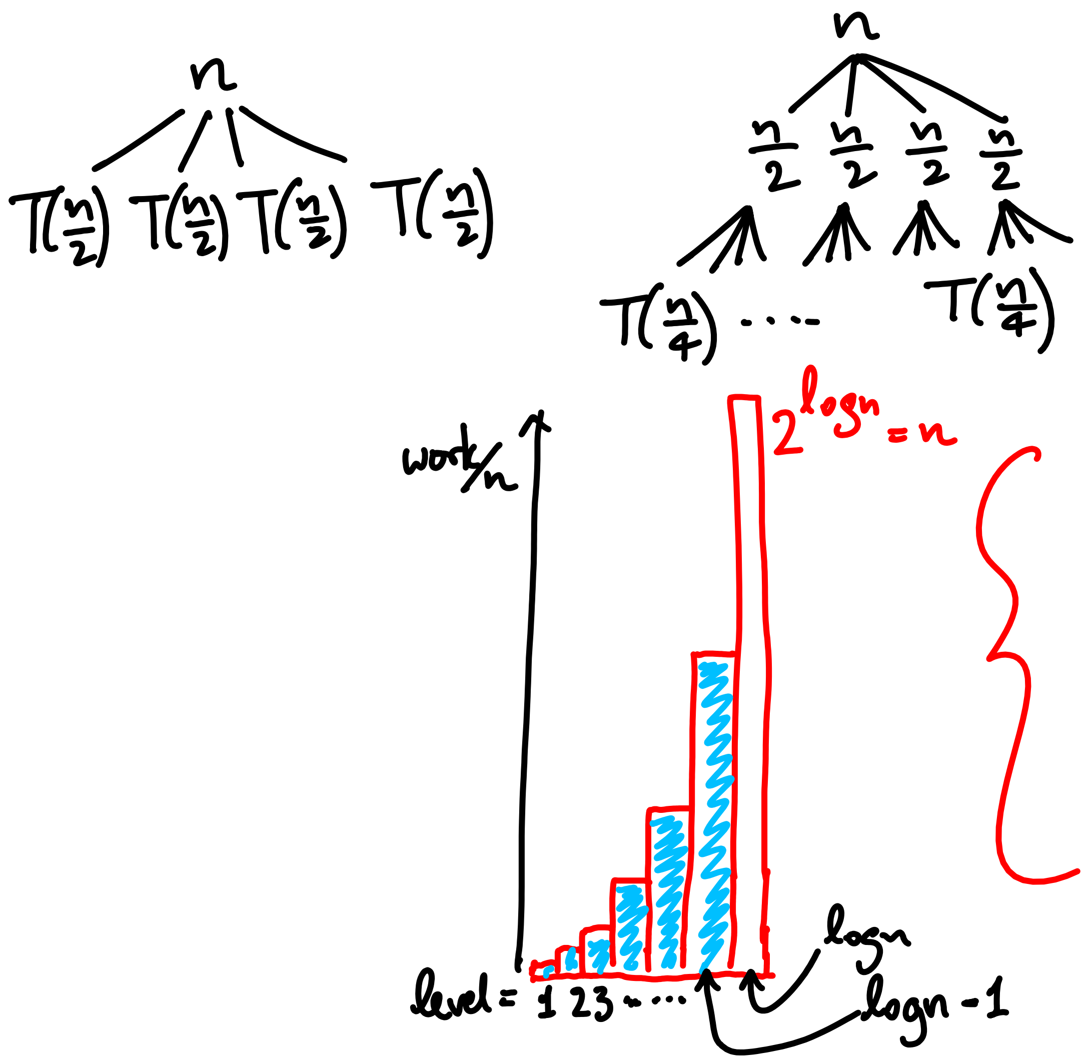


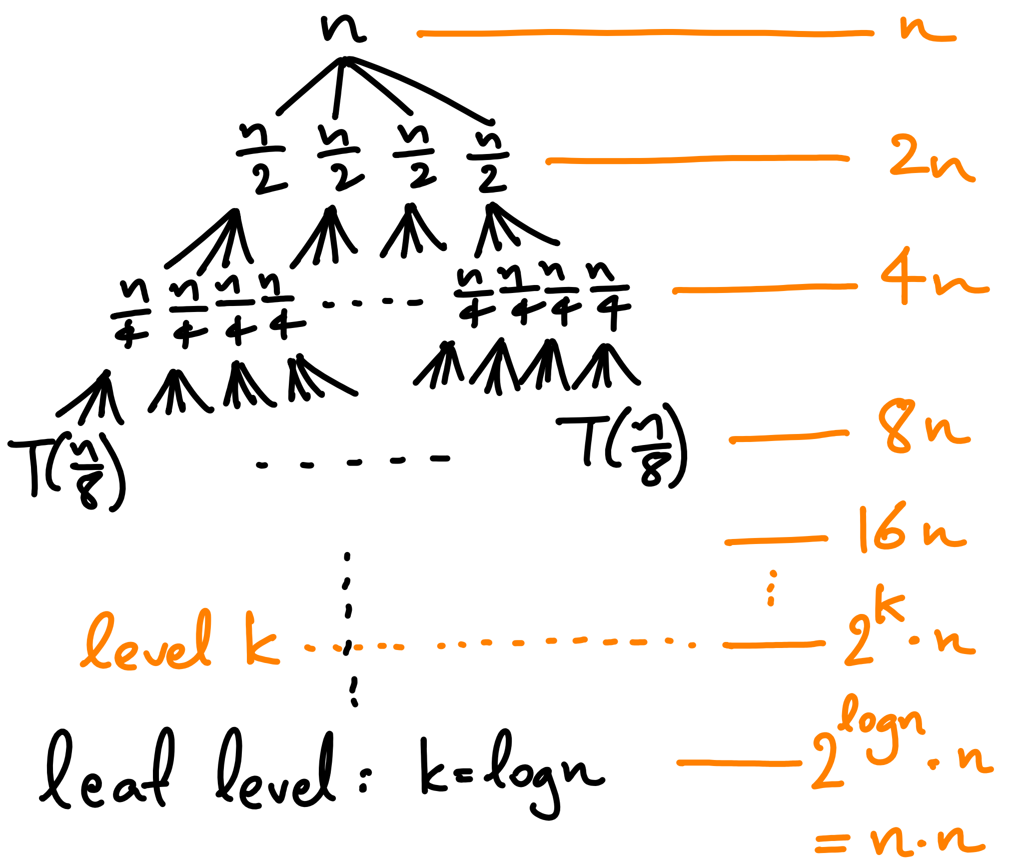
$T(n) = 4T\left(\frac{n}{2}\right) + n$ by recursion tree



$\text{Sum} = O(?)$

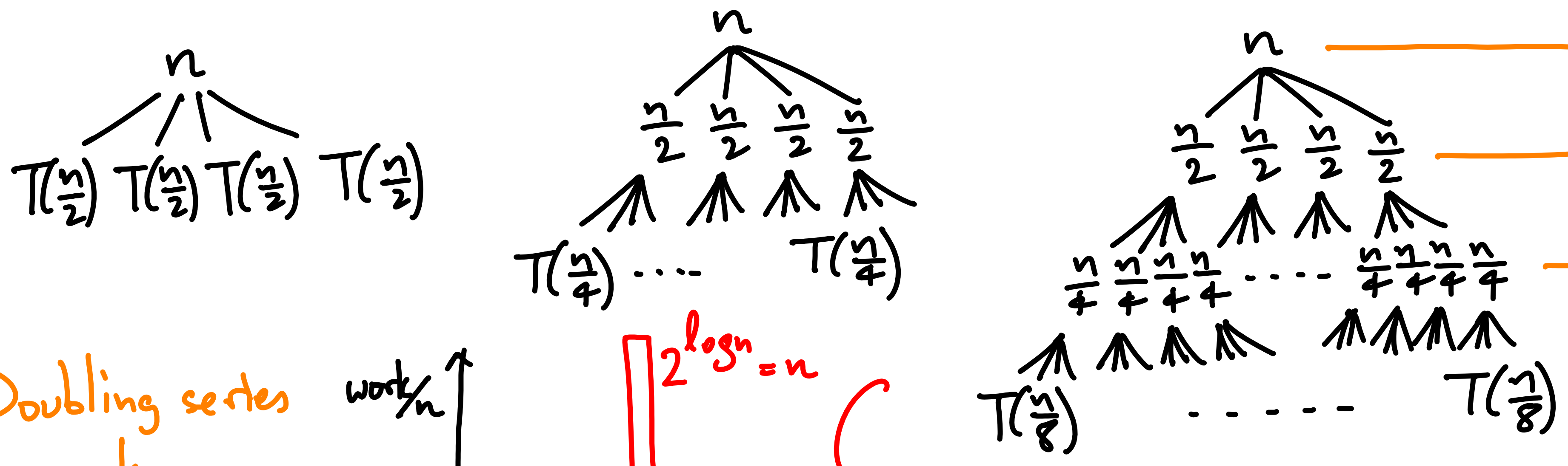
$T(n) = 4T\left(\frac{n}{2}\right) + n$ by recursion tree





$\text{Sum} = O(?)$

$T(n) = 4T(\frac{n}{2}) + n$ by recursion tree



n

2n

4n

8n

16n

⋮

level k .... $2^k \cdot n$

leaf level: $k = \log n$ $2^{\log n} \cdot n$

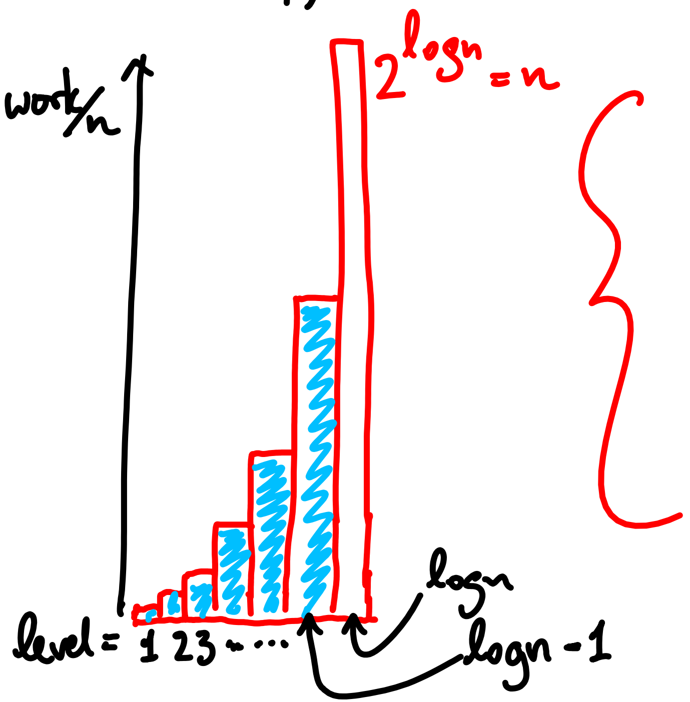
$= n \cdot n$

Sum $= O(n^2)$

Doubling series

↓

last term dominates

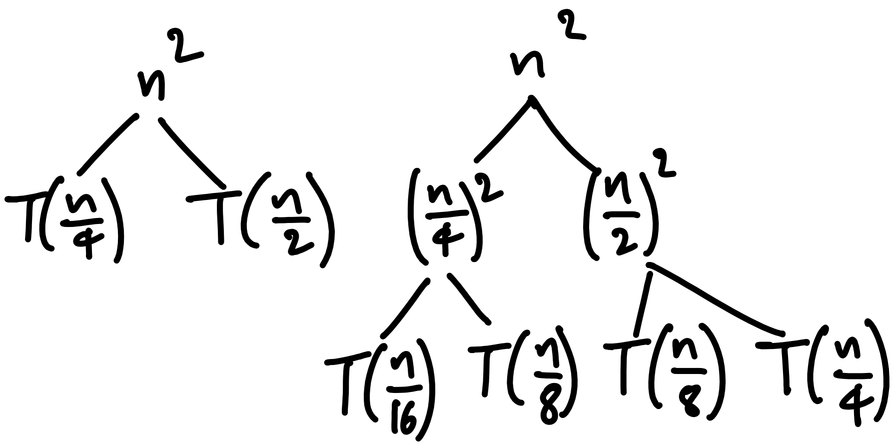


Another example: $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$

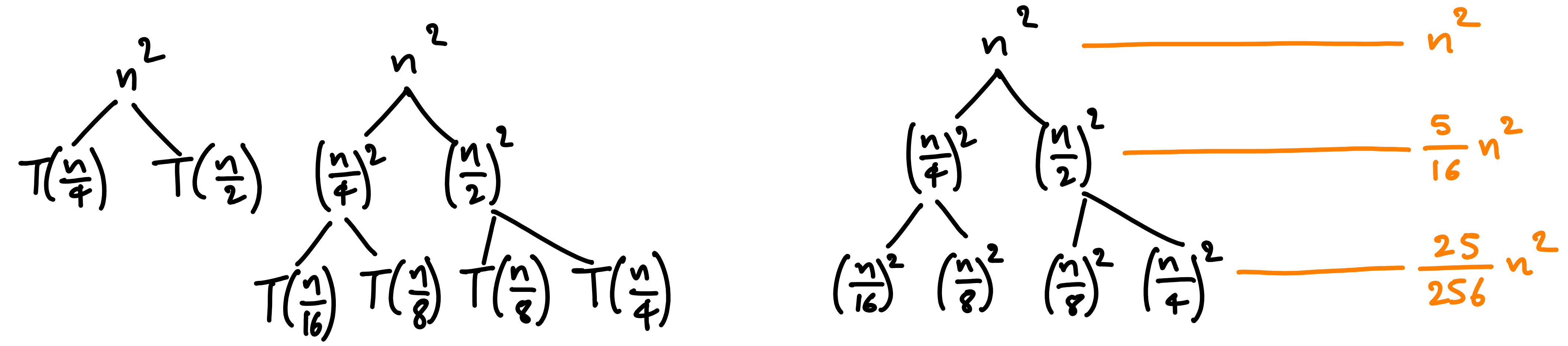
$$n^2$$

$$T\left(\frac{n}{4}\right) \qquad T\left(\frac{n}{2}\right)$$

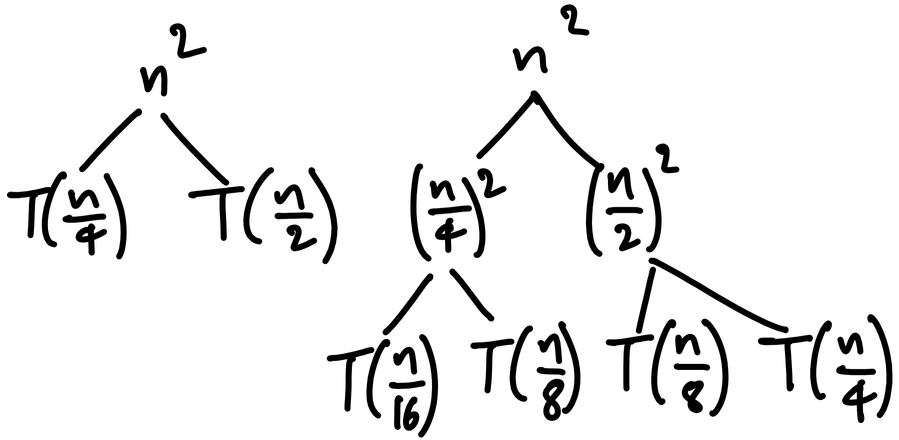
Another example: $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$

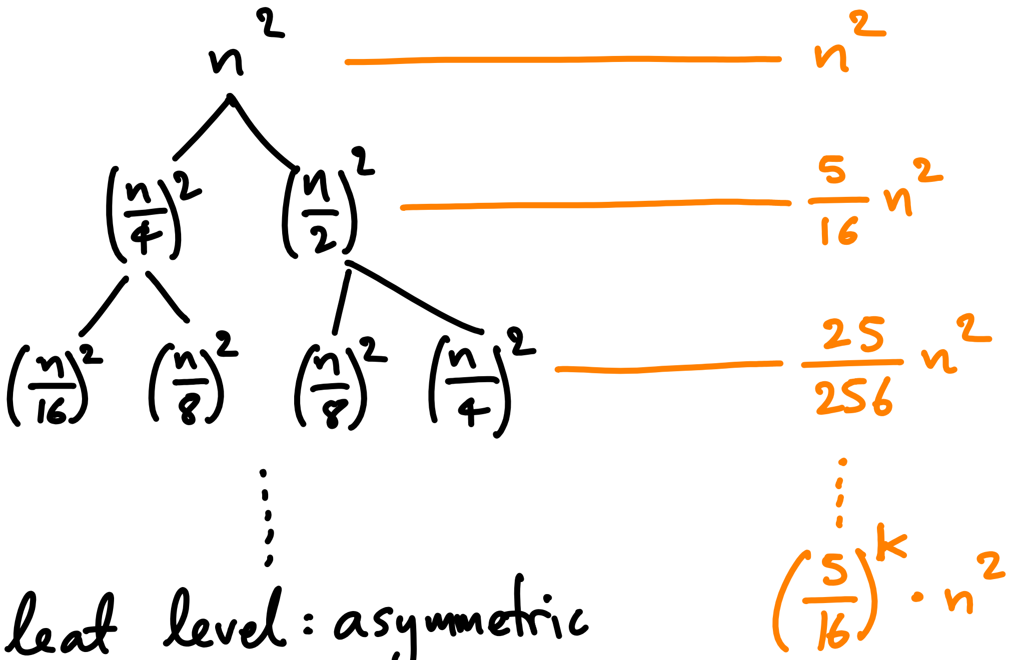


Another example: $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$



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

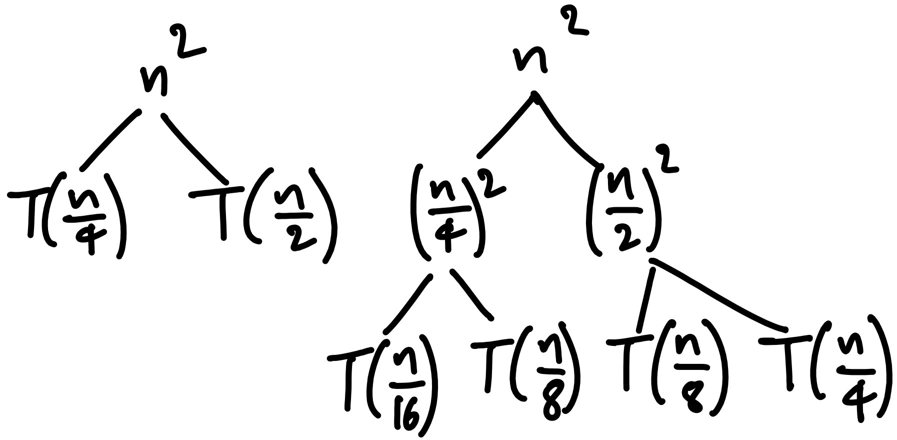


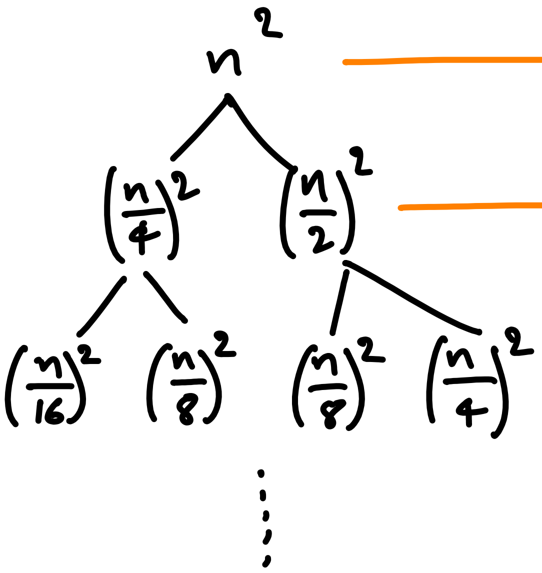


leaf level: asymmetric

deepest on right side ... $< \left(\frac{5}{16}\right)^{\log n} \cdot n^2$

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





n^2

$\frac{5}{16} n^2$

$\frac{25}{256} n^2$

$\vdots$

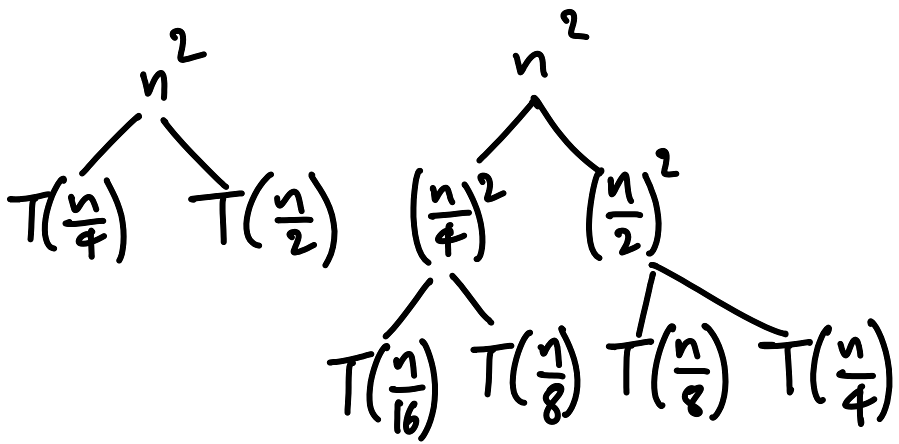
$\left(\frac{5}{16}\right)^k \cdot n^2$

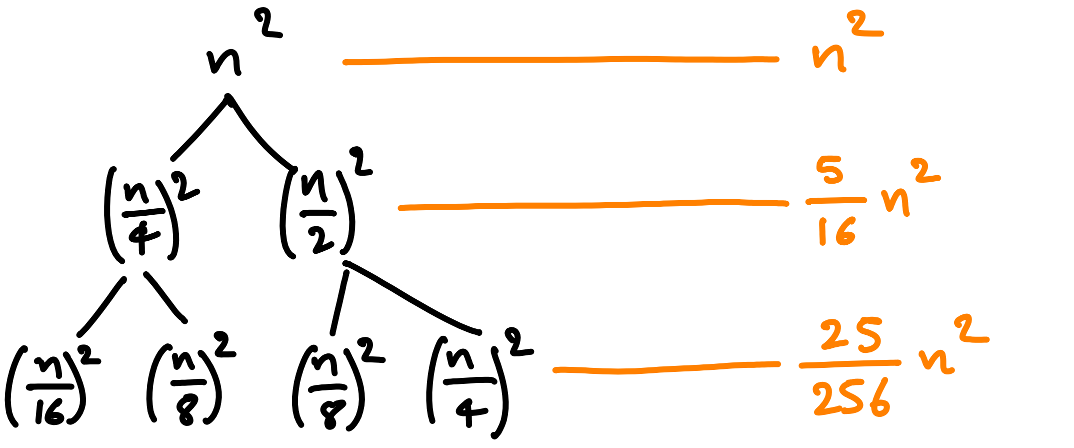
$$n^2 \cdot \left[1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \cdots \left(\frac{5}{16}\right)^k + \cdots\right]$$

leaf level: asymmetric

deepest on right side ... $< \left(\frac{5}{16}\right)^{\log n} \cdot n^2$

Another example: $T(n) = T(\frac{n}{4}) + T(\frac{n}{2}) + n^2$





$$\left(\frac{5}{16}\right)^k \cdot n^2$$

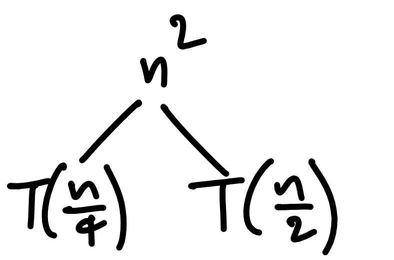
$$n^2 \cdot \left[1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \cdots \left(\frac{5}{16}\right)^k + \cdots\right]$$

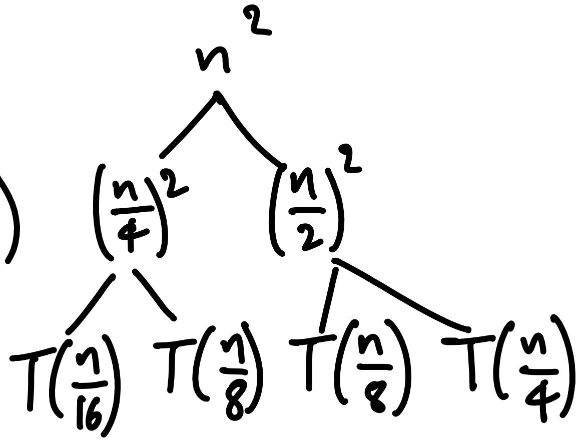
$$< n^2 \cdot \left[1 + \frac{1}{2} + \frac{1}{4} + \cdots + \left(\frac{1}{2}\right)^k + \cdots\right]$$

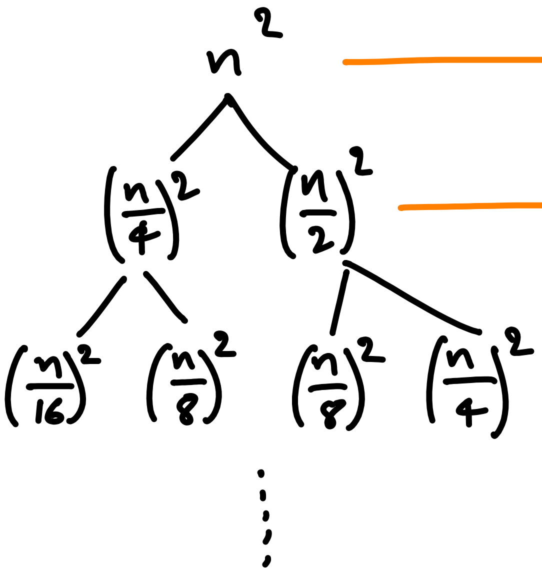
leaf level: asymmetric

deepest on right side ... $< \left(\frac{5}{16}\right)^{\log n} \cdot n^2$

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







n^2

$\frac{5}{16} n^2$

$\frac{25}{256} n^2$

$\vdots$

$\left(\frac{5}{16}\right)^k \cdot n^2$

leaf level: asymmetric

deepest on right side ... $< \left(\frac{5}{16}\right)^{\log n} \cdot n^2$

$$n^2 \cdot \left[1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \cdots \left(\frac{5}{16}\right)^k + \cdots \right]$$

$$< n^2 \cdot \left[1 + \frac{1}{2} + \frac{1}{4} + \cdots + \left(\frac{1}{2}\right)^k + \cdots \right]$$

$$= n^2 \cdot 2 = O(n^2)$$

Verify w/ substitution