4. Recurrences [Cormen 2001]

- As we saw previously when an algorithm contains a recursive call to itself its running time can often be described by a <u>recurrence</u>

- <u>Recurrence</u> - equation or inequality that describes a function in terms of its value on smaller inputs.

- Remember Merge Sort? $T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n), & n > 1 \end{cases}$

- We will three methods to solve recurrences:

<u>Substitution method</u> - guess a bound and use mathematical induction to prove it.

<u>Recursion-tree method</u> - convert the ~~tree~~ recurrence into a tree whose nodes represent the costs incurred at various levels of the recursion

<u>Master method</u> - provides bounds for recurrences of the form $T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$ where $a \geq 1$, $b > 1$ and $f(n)$ is a given function. Requires memorization of three cases.

Technicalities

- We won't assume that the domain is $\mathbb{N}$
- Boundary conditions will be ignored, i.e. we will assume that the base case for stopping the recursion will always cost constant time;
- Don't consider floors, ceilings and boundary conditions. Usually these don't matter.

4.1 The substitution method

- Two step procedure:
  1. Guess the form of the solution (your intuition)
  2. Use mathematical induction ## to prove that the guess works
- Can only be applied when it is easy to guess the answer

Example:

- $T(n) = \begin{cases} \Theta(1), & n=1 \\ 2T(\lfloor n/2 \rfloor) + n, & n>1 \end{cases}$ (merge sort)

  Our guess is going to be $O(n \log n)$

- Our method:

  Prove that $T(n) \leq cn \log n$ for a $c > 0$

- Assume that our guess holds for $\frac{n}{2}$, i.e.

  $T(\lfloor \frac{n}{2} \rfloor) \leq c \lfloor \frac{n}{2} \rfloor \log \lfloor \frac{n}{2} \rfloor$

Substituting into the recurrence yields:

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

$$\leq 2\left(c\left\lfloor \frac{n}{2} \right\rfloor \log\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\right) + n \quad \text{(we can drop the floor functions now)}$$

$$\leq cn \log\left(\frac{n}{2}\right) + n = cn[\underbrace{\log n}_{\text{Assume these are base 2}} - \underbrace{\log 2}_{\text{Assume these are base 2}}] + n$$

$$= cn(\log n - 1) + n = cn \log n \underbrace{- cn + n}_{\text{always negative when } c \geq 1}$$

$$\leq cn \log n \quad , c \geq 1$$

Now we need to check the <u>boundary conditions ($n = 1$)</u>

- $T(n) \leq cn \log n = c \cdot 1 \cdot \log 1 = 0 \neq 1$ (From ☺(1))
- I.e. our guess does not hold. What to do? Answer: "Remove the difficult base case"
- Recall that from <u>asymptotic notation</u>: $T(n) \leq cn \log n \;,\; n \geq n_0$

  Ideal if $n=1$ does not work maybe we can find other n_0.
- Observe that:

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

for $\underline{n = 2} \Rightarrow T(2) = 2T\left(\left\lfloor \frac{2}{2} \right\rfloor\right) + n = 2\,\underline{T(1)} + 2$, still depends on $\underline{T(1)}$

for $\underline{n = 3} \Rightarrow T(3) = 2T\left(\left\lfloor \frac{3}{2} \right\rfloor\right) + n = 2\,\underline{T(1)} + 3$

for $\underline{n = 4} \Rightarrow T(4) = 2T\left(\left\lfloor \frac{4}{2} \right\rfloor\right) + n = 2T(2) + n$ does not depend on $T(1)$!!

- I.e. we now know that for $n > 3$ our guess will hold. We can thus replace $T(1)$ by $T(2)$ and $T(3)$ as the base case in the inductive proof.

$$T(2) = 2T(1) + 2 = 2 \cdot 1 + 2 = 4$$

$$T(3) = 2T(1) + 3 = 2 \cdot 1 + 3 = 5$$

- "New recurrence" Tm:

$$\begin{cases} 4 & , n = 2 \\ 5 & , n = 3 \\ 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & , n > 3 \end{cases}$$

- Now we can try our induction hypothesis for the base cases again:

Hypothesis: $T(n) \leq c\, n \log n$

For $n = 2 \Rightarrow T(2) \leq c \cdot 2 \underbrace{\log 2}_{1} = 2c$

But we know that $T(2) = 4$

This implies that $c \geq 2$ ($4 \leq 2c \Leftrightarrow c \geq 2$)

For $n = 3$: $T(3) \leq c \cdot 3 \log 3$

But we know that $T(3) = 5$

This implies that $c \geq 5/(3\log 3)$ ($5 \leq c \cdot 3 \log 3 \Leftrightarrow c \geq \frac{5}{3\log 3}$)

Our induction was thus proven for all the cases of the recurrence (end of the example)

Question: How can we make a good guess? → No easy answer...

- Experience
- Creativity
- ...

Example:

- $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor + 17\right) + n$
- What would be a good guess?
  - For $n \to \infty$, $2T\left(\left\lfloor \frac{n}{2} \right\rfloor + 17\right) + n \simeq 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$
  - This is the recurrence from the previous example
  - Then our guess would be $T(n) = O(n \log n)$

- We can also use recurrence trees to make a good guess
- We can also use a zooming method.
  1) Assume "loose" bounds and prove them
  2) Proceed to assume "tighter" bounds and prove them
  3) Eventually the proof will fail and we will obtain the adequate bound.

Example:

- $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1$
  - Guess (Initial Guess): $T(n) = O(n) \Rightarrow T(n) = cn$
  - Proof: $T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1 \leq c\left\lfloor \frac{n}{2} \right\rfloor + c\left\lceil \frac{n}{2} \right\rceil + 1 \leq c\frac{n}{2} + c\frac{n}{2} + 1$

$= cn + 1$ ← This is different by a factor of 1 from our guess

Induction proof fails!! →

Question: What can we do then?

- Maybe our guess needs to also contemplate a constant factor
- New guess $T(n) \leq \boxed{cn - b}$ (This is still $T(n) = O(n)$, however our induction guess now incorporates a constant factor)

$$
\begin{aligned}
- \; T(n) &\leq \left(c\left\lfloor \tfrac{n}{2} \right\rfloor - b\right) + \left(c\left\lceil n/2 \right\rceil - b\right) + 1 \\
&\leq c\,\frac{n}{2} - b + c\,\frac{n}{2} - b + 1 \\
&= cn - 2b + 1 \\
&\leq \boxed{cn - b} \quad \forall b \geq 1
\end{aligned}
$$

Our guess was now proven by induction !! (end of example)

Example:

- Algebraic manipulation:
  - $T(n) = 2T\left(\left\lfloor \sqrt{n} \right\rfloor\right) + \log n$
- Renaming $m = \log n$ yields
  - $m = \log_2 n \Rightarrow n = 2^m$
  - $T(n) = 2T\left(\sqrt{n}\right) + \log n$

$$
\begin{aligned}
\Leftrightarrow T(2^m) &= 2T\left(\sqrt{2^m}\right) + m \\
\Leftrightarrow T(2^m) &= 2T\left(2^{m/2}\right) + m
\end{aligned}
$$

- We can now rename $S(m) = T(2^m)$ to produce the new recurrence:

$S(m) = T(2^m)$ ↪ $T(2^m) = 2T(2^{m/2}) + m$

↪ $S(m) = 2S(\frac{m}{2}) + m$

- This is the recurrence we saw in a previous example ($T(n) = 2T(\frac{n}{2}) + n$)
- Therefore our guess would be $T(n) = O(n \log n)$
- We only need to convert to the new names:

$$S(m) = O(m . \log m)$$

- But we know that $S(m) = T(2^m)$ and $m = \log_2 n$

Therefore:

$$S(m) = O(m . \log m)$$

$$\Leftrightarrow T(2^m) = O(\log n . \log(\log n))$$

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

4.2 The recursion-tree method

- Substitution-method makes it difficult to come up with a good guess
- We can also draw a recurrence tree:
  - each node represents the cost of a subproblem
  - We sum the costs within each level of the tree to obtain a set of per-level costs
  - We sum all the per-level costs to obtain total time.
- Particularly useful for divide-and-conquer algorithms

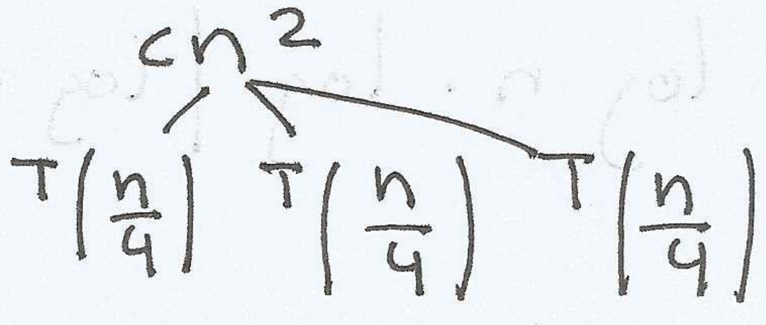
Example:

$$T(n) = 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2) \overset{\text{We can ignore approximations and convert the } \Theta \text{ notation}}{=} 3T\left(\frac{n}{4}\right) + cn^2$$

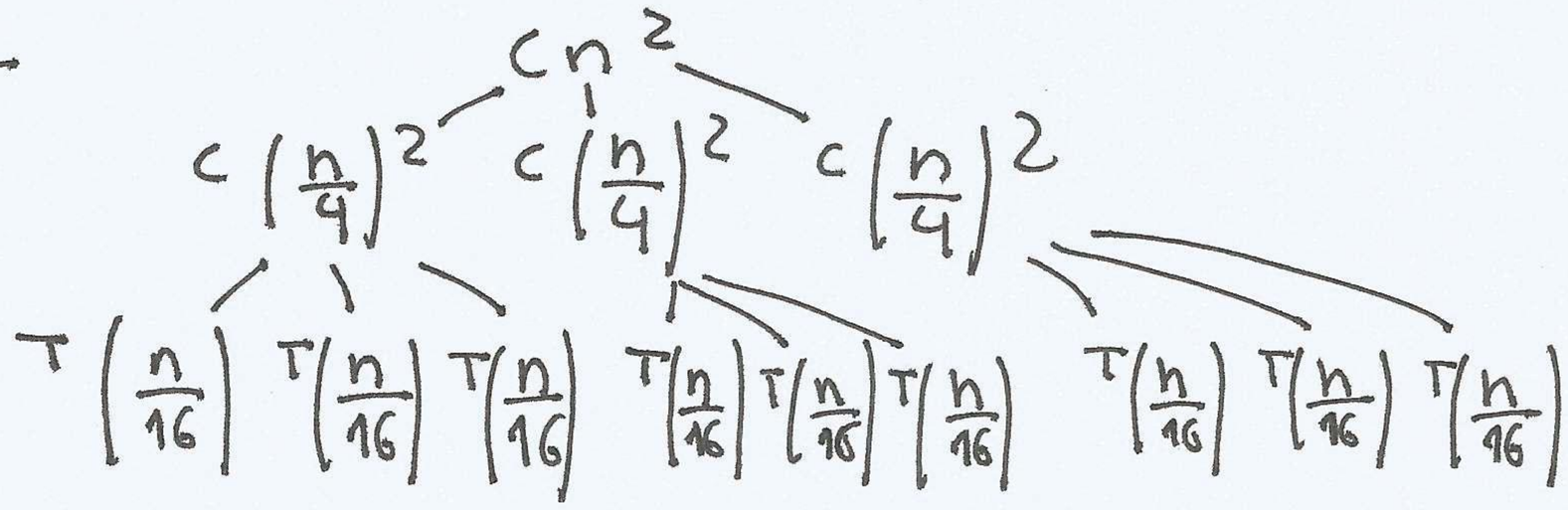
Let's draw each iteration:

1st $T(n)$

2nd

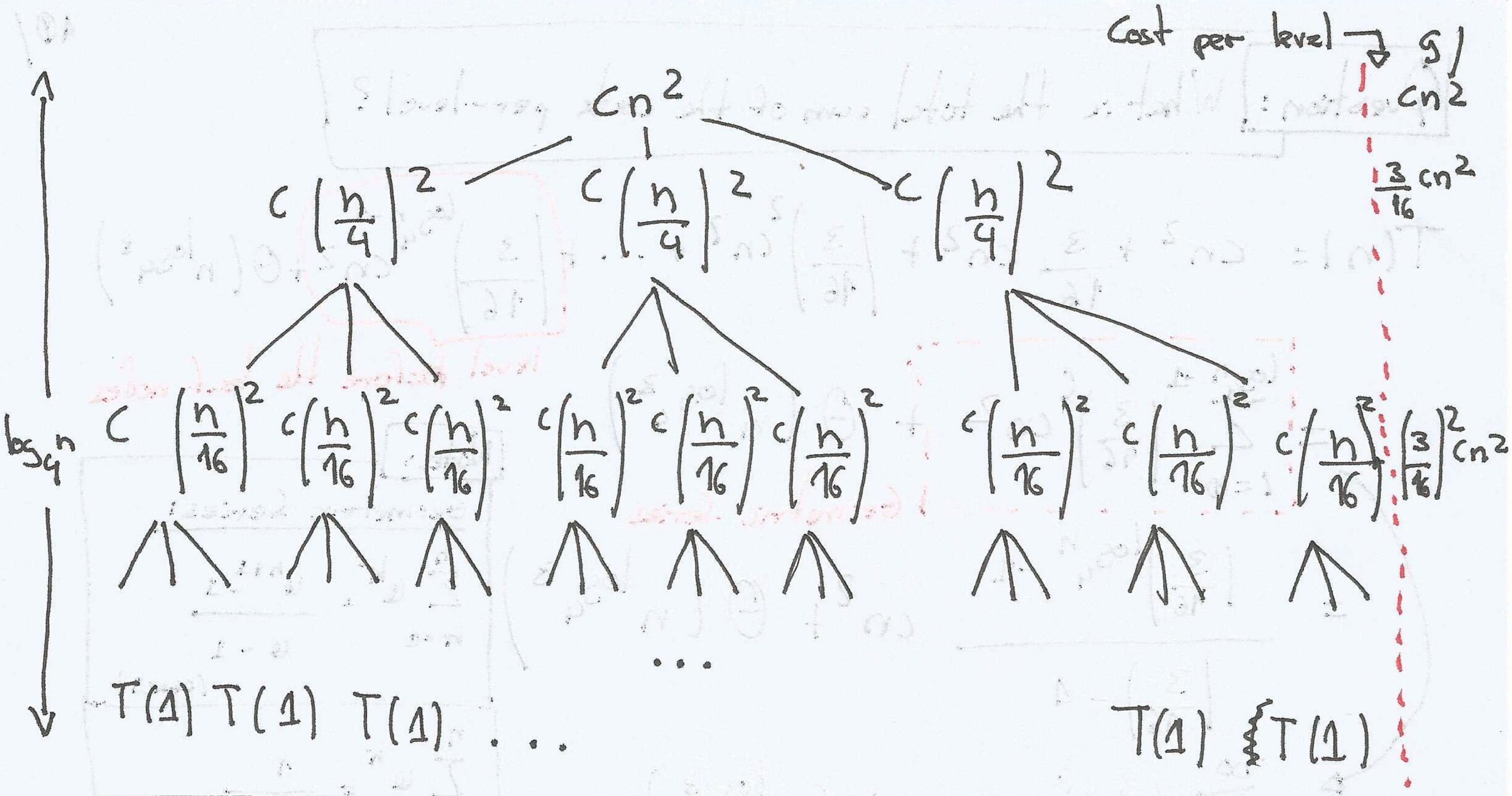


3rd



⋮

↳



Question: How many levels?

~~height 0: 1 node = 3^0~~
~~height 1: 3 nodes = 3^1~~
~~height 2: 9 nodes = 3^2~~
~~height 3: 27 nodes = 3^3~~
~~"~~
height d: 3^d nodes

height 0: interval length: $\frac{n}{4^0} = \frac{n}{1}$
" 1: " " : $\frac{n}{4} = \frac{n}{4^1}$
" 2: " " : $\frac{n}{16} = \frac{n}{4^2}$
" 3: " " : $\frac{n}{64} = \frac{n}{4^3}$
⋮
height d: " " : $\frac{n}{4^d}$

This implies that $d = \log_4 n$ levels

Question: What is the per-level cost?

- level 0: $cn^2 = \left(\frac{3}{16}\right)^0 cn^2$
- level 1: $\frac{3}{16}cn^2 = \left(\frac{3}{16}\right)^1 cn^2$
- level 2: $\left(\frac{3}{16}\right)^2 cn^2 = \left(\frac{3}{16}\right)^2 cn^2$
⋮
- level i: $\left(\frac{3}{16}\right)^i cn^2$
(level before the leafs)
- last level of the tree: $3^{\log_4 n} = n^{\log_4 3}$ nodes each costing $T(1) \Rightarrow \Theta(n^{\log_4 3})$

Auxiliary Calculations:
$\frac{n}{4^d} = 1 \Leftrightarrow n = 4^d \Leftrightarrow \log_4 n = d$

Question: What is the total sum of the costs per-level?

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \dots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3})$$

level before the leaf nodes

$$= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

Geometric Series

$$= \frac{\left(\frac{3}{16}\right)^{\log_4 n} - 1}{\left(\frac{3}{16}\right) - 1} cn^2 + \Theta(n^{\log_4 3})$$

$$= \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{1}{1 - \frac{3}{16}} cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) = \frac{16}{13} cn^2 + n^{0,792481\dots}$$

$$= O(n^2)$$

Notes:

Geometric Series:

$$\sum_{k=0}^{n} a^k = \frac{a^{n+1} - 1}{a - 1} \quad \text{(case 1)}$$

$$\sum_{k=0}^{n} a^k = \frac{1}{1 - a} \quad \text{and } |a| < 1 \quad \text{(case 2)}$$

- Notice from our picture that the root costs $\underline{cn^2}$ and in this line we have also $\underline{cn^2}$ multiplied by $16/13$
  - That is the root contributes a constant ratio of the total cost
  - And because this is the major factor: the total cost of the tree is dominated by the cost of the root

- Now we can use the substitution method to verify our guess

  - Recurrence: $T(n) = 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2) \Rightarrow 3T\left(\frac{n}{4}\right) + cn^2$
  - Guess: $T(n) = O(n^2) \Rightarrow T(n) \leq dn^2$
  - $T(n) = 3T\left(\frac{n}{4}\right) + cn^2 \leq 3d\left(\frac{n}{4}\right)^2 + cn^2 = \frac{3}{16}dn^2 + cn^2$

    $\leq dn^2$ as long as $d \geq \frac{16}{13}c$

Auxiliary Calculations:

$\frac{3}{16}dn^2 + cn^2 \leq dn^2$

$\Leftrightarrow \left(\frac{3}{16} - \frac{1}{16}\right)dn^2 \leq -cn^2$

$\Leftrightarrow \left(\frac{13}{16}\right)d \leq -c$

$\Leftrightarrow \frac{13}{16}d \geq c$

$\Leftrightarrow d \geq \frac{16}{13}c$

Our guess thus holds $\Rightarrow T(n) = O(n^2)$

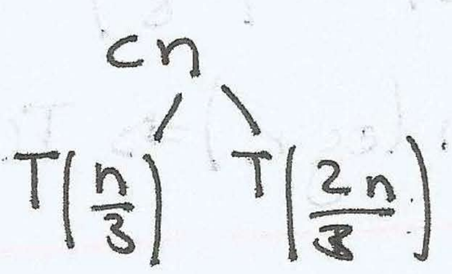
(end of example)

Example:

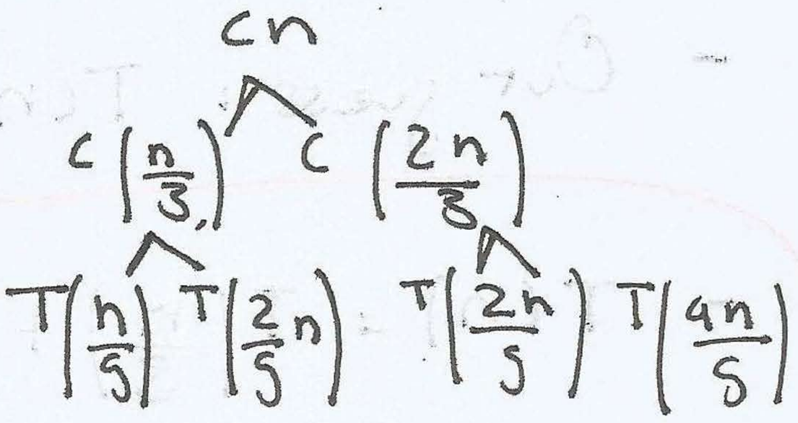
$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n)$

1st $T(n)$

2nd



3rd



Question: How many levels?

0th level: $n = \left(\frac{2}{3}\right)^0 n$

1st level: $\frac{2n}{3} = \left(\frac{2}{3}\right)^1 n$

2nd level: $\frac{4n}{9} = \left(\frac{2}{3}\right)^2 n$

3rd level: $\frac{8n}{27} = \left(\frac{2}{3}\right)^3 n$

$\vdots$

ith level: $\left(\frac{2}{3}\right)^i n$

Question: When will the interval reach size 1?

$\left(\frac{2}{3}\right)^i n = 1 \Leftrightarrow \frac{n}{\left(\frac{3}{2}\right)^i} = 1 \Leftrightarrow n = \left(\frac{3}{2}\right)^i$

$\Leftrightarrow i = \log_{3/2} n$ (maximum height of the tree)

Note:

- In this case we have $\neq$ decomposition of the input: $\left\lfloor \frac{n}{3} \right\rfloor$ and $\left\lfloor \frac{2n}{3} \right\rfloor$
- The decomposition $\frac{2n}{3}$ will result in longer intervals and therefore longer tree paths
- The longest path is thus $n \to \left(\frac{2}{3}\right)n \to \left(\frac{2}{3}\right)^2 n \to \left(\frac{2}{3}\right)^3 n \to \dots \to 1$

Question: What is the per-level cost?

0th level – cn

1st level – $c\frac{n}{3} + c\frac{2n}{3} = cn$

2nd level – $c\frac{n}{9} + c\frac{2n}{9} + c\frac{2n}{9} + c\frac{4n}{9} = c\frac{9n}{9} = cn$

⋮

Cost per-level : cn

Question: What is the total tree cost?

$$\text{TotalTreeCost} = \text{costPerLevel} \times \text{NumberOfLevels}$$

$$= cn \times \log_{3/2} n \Rightarrow O(n \log n)$$

- We can now use the substitution method for our guess
- Recurrence: $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$
- Our guess: $T(n) = O(n \log n) \Rightarrow T(n) \leq dn \log n$

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

$$\leq d\frac{n}{3}\log\frac{n}{3} + d\left(\frac{2n}{3}\right)\log\left(\frac{2n}{3}\right) + c$$

$$= d\frac{n}{3}\log n - \frac{dn}{3}\log 3 + d\left(\frac{2n}{3}\right)\log n - d\left(\frac{2n}{3}\right)\log\left(\frac{3}{2}\right) + cn$$

$$= dn\log n - d\left[\left(\frac{n}{3}\right)\log 3 + \left(\frac{2n}{3}\right)\log 3 - \left(\frac{2n}{3}\right)\log 2\right] + cn$$

$$= dn\log n - dn\left(\lg 3 - 2/3\right) + cn$$

$$\leq \boxed{dn\log n}, \text{ for } d \geq \frac{c}{\log 3 - \left(\frac{2}{3}\right)}$$

Auxiliary Calculations:

$dn\log n - dn\left(\lg 3 - \frac{2}{3}\right) + cn$

$\leq dn\log n$

$\Leftrightarrow -dn\left(\lg 3 - \frac{2}{3}\right) \leq -cn$

$\Leftrightarrow d \geq \frac{c}{\lg 3 - \frac{2}{3}}$