ADVANCED ANALYSIS OF ALGORITHMS
CPS 5440

OMAR DIB

SUBSTITUTION METHOD

SUBSTITUTION METHOD: CONCEPT

- Substitution method for solving recurrences consists of two steps:
- Guess the form of the solution, e.g., $T(n) = O\big(g(n)\big)$,then
- Use mathematical induction to find constants $(c \; and \; n_0)$ in the form and show that the solution works
  - **Step 1 (Base step)** – Prove that the guess is true for the initial value
  - **Step 2 (Inductive step)** – Prove that if the guess is true for $T(k) \le \mathrm{c}\, g(k)$, $\forall\, k < n$, then this implies that $T(n) \le \mathrm{c}\, g(n)$, for some $c > 0$ and $n \ge n_0$
- The inductive hypothesis is applied to smaller values, similar like recursive calls bring us closer to the base case
- The substitution method is a powerful way to establish lower or upper bounds on a recurrence
- It applies in cases when it is easy to guess the form of the solution

SUBSTITUTION METHOD: MAKING A GOOD GUESS

- There is no general way to guess the correct solution to recurrences.
- Guessing a solution takes experience and, occasionally, creativity.
- There are some heuristics that can help us make a good guess ($e.g.$, Recursion Tree)
- If a recurrence is similar to a one, we have seen before, then guessing a similar solution is reasonable
- For example, $T(n) = 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 17) + n$, we make the guess that $T(n) = O(n \lg n)$ like Merge Sort
- Another way to make a good guess is to prove the loose upper and lower bounds on the recurrence and then reduce the range of uncertainty. For example:
  - Start with and prove the initial lower bound of $T(n) = \Omega(n)$ for the recurrence
  - Start with and prove the initial upper bound of $T(n) = O(n^2)$ for the recurrence
  - Then gradually lower the upper bound and raise the lower bound until convergence to correct, asymptotically tight solution of $T(n) = \Theta(n \lg n)$
- Sometimes the correct guess at an asymptotic bound on the solution of a recurrence does not work. This can be solved by revising the guess and subtracting a lower-order term in the guess.

SUBSTITUTION METHOD: MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$
- Guess: $T(n) = O(n \lg n)$, or $T(n) \leq \mathrm{c}\,.\,n \lg n$, for some constant c and $n_0 \leq n$
- Hypothesis: $T(k) \leq \mathrm{c}\,.\,k \lg k\,,\ \forall\ k < n,$ we will use $k = \frac{n}{2}$
- Inductive Step:

$$
\begin{aligned}
T(n) &= 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\
&\leq 2\,.\,c\,.\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\
&\leq c\,.\,n \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\
&= c\,.\,n \lg n - c\,.\,n \lg 2 + n \\
&= c\,.\,n \lg n - c\,.\,n + n \\
&\leq \mathrm{c}\,.\,n \lg n \qquad if: -c.n + n \leq 0 \Rightarrow c \geq 1
\end{aligned}
$$

SUBSTITUTION METHOD: MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

- Guess: $T(n) = O(n \lg n)$, or $T(n) \leq \mathrm{c}\,.\,n \lg n$, for some constant c and $n_0 \leq n$
- Hypothesis: $T(k) \leq \mathrm{c}\,.\,k \lg k\,,\ \forall\ k < n$
- From inductive step: $T(n) \leq \mathrm{c}\,.\,n \lg n$ when $c \geq 1$
- Base step: $T(1) \leq \mathrm{c}\,.\,1 \lg 1\,?$
  - Impossible as $T(1) = 1 \nleq \mathrm{c}\,.\,1 \lg 1 = 0$. (Problem!)
  - But we only want to show that $T(n) \leq \mathrm{c}\,.\,n \lg n$ for sufficiently large values of n; $i.e.,\forall\ n \geq n_0$.
  - Solution: Try $n_0 > 1$
- Base steps (check boundaries)
  - We must check both $T(2)$ and $T(3)$ simultaneously because of the nature of the recursive equation
  - Check $T(2)$ and $T(3)$

SUBSTITUTION METHOD : MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

- Guess: $T(n) = O(n \lg n)$, or $T(n) \leq \mathrm{c}\,.\,n \lg n$, for some constant c and $n_0 \leq n$
- Hypothesis: $T(k) \leq \mathrm{c}\,.\,k \lg k\,,\ \forall\ k < n$
- From inductive step: $T(n) \leq \mathrm{c}\,.\,n \lg n$ when $c \geq 1$
- **Base step:** $T(1) \leq \mathrm{c}\,.\,n \lg n\ \forall\ n \geq 1\ (n_0 > 1)$
- Base step boundaries:
  - $T(1) = 1 \Longrightarrow \begin{cases} T(2) & = 4 \\ T(3) & = 5 \end{cases}$
  - We want to satisfy simultaneously
  - $\begin{cases} 4 = T(2) \leq \mathrm{c}\,.2 \lg 2 \\ 5 = T(3) \leq \mathrm{c}\,.3 \lg 3 \end{cases} \Longrightarrow \begin{cases} c \geq 2 \\ c \geq 1.052 \end{cases} \Longrightarrow c \geq 2$
- We prove that $T(n) \leq \mathrm{c}\,.\,n \lg n$, *with* $c = 2$, *and* $n_0 = 2$, *So* $T(n) = O(n \lg n)$

SUBSTITUTION METHOD : MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

What happens if we make a wrong guess?

- Guess: what if we choose a wrong guess, $e.g.,\ T(n) = O(n)$, or $T(n) \le c\,.\,n$, for some constant c and $n_0 \le n$
- Hypothesis: $T(k) \le c\,.\,k,\ \forall\ k < n,$ we will use $k = \frac{n}{2}$
- Inductive Step:

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

$$\le 2\,.\,c\,.\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

$$\le c\,.\,n + n$$

$$= (c + 1).\,n$$

$\not\le c\,.\,n$ *The above inequality does not hold because $c + 1$ cannot be less than c (Contradiction).*

Hence $T(n) \ne O(n)$

SUBSTITUTION METHOD : MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

Substitution method can also be used to guess a lower bound!

- Let's guess a lower bound
- We want to show $T(n) \geq \mathrm{c}\,.\,n \lg n$, for some constant c and $n_0 \leq n$. (Assume that n is a power of 2)
- Hypothesis: $T(k) \geq \mathrm{c}\,.\,k \lg k\,,\ \forall\ k < n,$

we will use $k = \frac{n}{2}$

- Inductive Step: $T(n) = 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$

$$\geq 2\,.\ c\,.\ \left(\left\lfloor \frac{n}{2} \right\rfloor\right) \lg \left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

$$\geq c\,.\,n \lg \left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

$$= c\,.\,n \lg n\ - c\,.\,n \lg 2 + n$$

$$= c\,.\,n \lg n\ - c\,.\,n + n$$

$$= c\,.\,n \lg n\ - n(c-1)$$

$$\geq \mathrm{c}\,.\,n \lg n$$

True *as long as* $(c\ - 1) \leq 0\ \Rightarrow c \leq 1$

SUBSTITUTION METHOD: MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

Substitution method can also be used to guess a lower bound!

- Let's guess lower bound
- We want to show $T(n) \geq c.n \lg n$, for some constant c and $n_0 \leq n$. (Assume that n is a power of 2)
- Hypothesis: $T(k) \geq \mathrm{c}\,.\,k \lg k\,, \forall\ k < n,$ we will use $k = \frac{n}{2}$
- From inductive step: $T(n) \geq c.n \lg n$ when $c \leq 1$
- Base step: $T(1) \geq c.1 \lg 1\ ?$
  - True as $T(1) = 1 \geq c.1 \lg 1 = 0.$
- Check boundaries
  - We also want to satisfy the boundary condition $(T(2) = 4)$
  - $T(2) \geq c.2 \lg 2\ ? \Longrightarrow 4 \geq 2.c?$ (True as long as $c < 2$. By the requirement of the inductive step $c \leq 1 \Longrightarrow c = 1$)

SUBSTITUTION METHOD: MERGE SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & n > 1 \end{cases}$

To prove a bound, we must also prove that that $T(n)$ is strictly increasing!

- We will prove that $T(n)$ is strictly increasing
- Assuming for all $k \leq n$ it holds $T(k) > T(k-1)$, we want to show that $T(n+1) > T(n)$
- For the base case, note that $T(1) = 1 < 4 = T(2)$
- $T(n+1) = 2\,T\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) + n + 1$

$= 2\,T\left(\frac{n}{2}\right) + n + 1$ //Note: $T\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) = T\left(\frac{n}{2}\right)$ (assuming n is a power of 2)

$= \left[2\,T\left(\frac{n}{2}\right) + n\right] + 1$

$= T(n) + 1$

$> T(n)$

- We prove that $T(n) \geq c\,.\,n \lg n$, *with* $c = 1$, *and* $n_0 = 2$, *So* $T(n) = \Omega(n \lg n)$

SUBSTITUTION METHOD: QUICK SORT

- $T(n) = \begin{cases} 1 & n = 1 \\ T(n-1) + n & n > 1 \end{cases}$

- Guess: $T(n) = O(n^2)$, or $T(n) \leq \mathrm{c}\,.\,n^2$, for some constant c and $n_0 \leq n$
- Hypothesis: $T(k) \leq \mathrm{c}\,.\,k^2, \forall\, k < n,$ we will use $k = n - 1$
- Inductive Step: $T(n) = T(n-1) + n$

$$\leq c\,.\,(n-1)^2 + n$$

$$= c\,.\,(n^2 - 2.n + 1) + n$$

$$= c\,.\,n^2 - 2.c.n + c + n$$

$$\leq \mathrm{c}\,.\,n^2 \qquad \text{True } if: -2.c.n + c + n \leq 0 \;\Rightarrow c \geq 1$$

- Base Step: $T(1) = 1 \leq c.(1)^2 \Longrightarrow T(1) \leq c$
- We prove that $T(n) \leq \mathrm{c}\,.\,n^2$, *with* $c = 1$, *and* $n_0 = 1$, *So* $T(n) = O(n^2)$

SUBSTITUTION METHOD: LOOSE BOUND

- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\frac{n}{2}\right) + 1 & n > 1 \end{cases}$
- Guess: $T(n) = O(n)$, or $T(n) \leq c\,.\,n$, for some constant c and $n_0 \leq n$
- Hypothesis: $T(k) \leq c\,.\,k,\ \forall\ k < n$, we will use $k = \frac{n}{2}$
- Inductive Step: $T(n) = 2\,T\left(\frac{n}{2}\right) + 1$

$$\leq 2.\,c\,.\left(\frac{n}{2}\right) + 1$$

$$= c.\,n + 1$$

$$\not\leq c\,.\,n$$

- Which does not imply that $T(n) \leq c\,.\,n$, for any c. We need to show the exact form.
- To overcome this hurdle:
  - Revise our guess: say $T(n) = O(n^2)$. However, our original guess was correct!
  - Sometimes it is easier to prove something stronger!

SUBSTITUTION METHOD: LOOSE BOUND

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

- Solution: Try tighter bound
- Guess: $T(n) = O(n)$. Let's try $T(n) \leq \mathrm{c}\,.\,n - b$, where b is another constant
- Hypothesis: $T(k) \leq \mathrm{c}\,.\,k - b,\ \forall\ k < n,$ we will use $k = \frac{n}{2}$
- Inductive Step: $T(n) = 2\,T\left(\frac{n}{2}\right) + 1$

$$\leq 2.\left[c\,.\left(\frac{n}{2}\right) - b\right] + 1$$

$$= c.n - 2.b + 1$$

$$\leq \mathrm{c}\,.\,n - b \qquad \text{True } if: -b + 1 \leq 0\ \Rightarrow b\ \geq 1$$

- Base Step: $T(1) = 1\ \leq c.1 - b \Longrightarrow 1 \leq c - b \Longrightarrow b \leq c$

We prove that $T(n) \leq \mathrm{c}\,.\,n - b$, *with* $c = 2, b = 1$ *and* $n_0 = 1$, *So* $T(n) = O(n)$

SUBSTITUTION METHOD: CHANGING VARIABLES

- Consider the recurrence $T(n) = 2\,T(\lfloor\sqrt{n}\rfloor) + \lg n$

- We can simplify the recurrence with a change of variables
- Rename $m = \lg n$. We have:

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

- Define $S(m) = T(2^m)$. We get:

$$S(m) = 2\,S\left(\frac{m}{2}\right) + m$$

- Hence, the solution is $O(m \lg m)$, or with substitution $O(\lg n \,.\, \lg(\lg n))$

SUBSTITUTION METHOD: PRACTICE QUESTIONS

- Apply the substitution method to the given recurrences and show that the given guesses are the solution to these recurrences. Assume $T(1) = 1$ as the base case.

- $T(n) = T(n-1) + T(n-2) + 1$ (Guess: $O(2^n)$)

- $T(n) = 2.T(n-1) + 1$ (Guess: $O(2^n)$)

- $T(n) = 3.T\left(\frac{n}{4}\right) + n^2$ (Guess: $O(n^2)$)

- $T(n) = 3.T\left(\frac{n}{4}\right) + n$ (Guess: $O(n)$)

RECURSION TREE METHOD

RECURSION TREE METHOD: CONCEPT

- Making a good guess is sometimes difficult with the substitution method
- Recursion tree method can be used to devise a good guess
- Recursion trees show successive expansions of recurrences using trees
- RT model the costs (time) of a recursive execution of an algorithm that is composed of two parts:
  - Cost of non-recursive part
  - Cost of recursive call on smaller input size
- A tree node represents the cost of a sub-problem (recursive function invocation)
- To determine the total cost of the recursion tree, evaluate:
  - Cost of individual node at depth "i"
  - Sum up the cost of all nodes at depth "i"
  - Sum up all per-level costs of the recursion tree

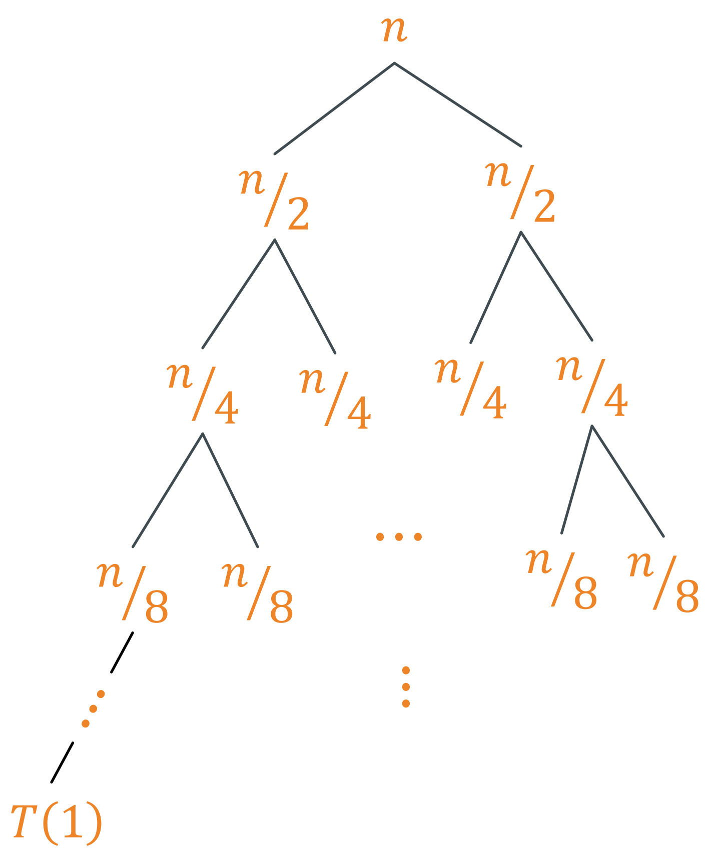
RECURSION TREE METHOD: EXAMPLE 1

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

- Solve the following recurrence using the Recurrence Tree Method
- Assumption: We assume that n is an exact power of 2
- Some Useful Properties:
  - $x^{\log_y n} \Longrightarrow n^{\log_y x}$
  - $x^0 + x^1 + x^2 + \cdots + x^n = \frac{x^{n+1}-1}{x-1} \qquad for\ x \neq 1$
  - $x^0 + x^1 + x^2 + \cdots = \frac{1}{1-x} \qquad for\ |x| < 1$

RECURSION TREE METHOD: EXAMPLE 1

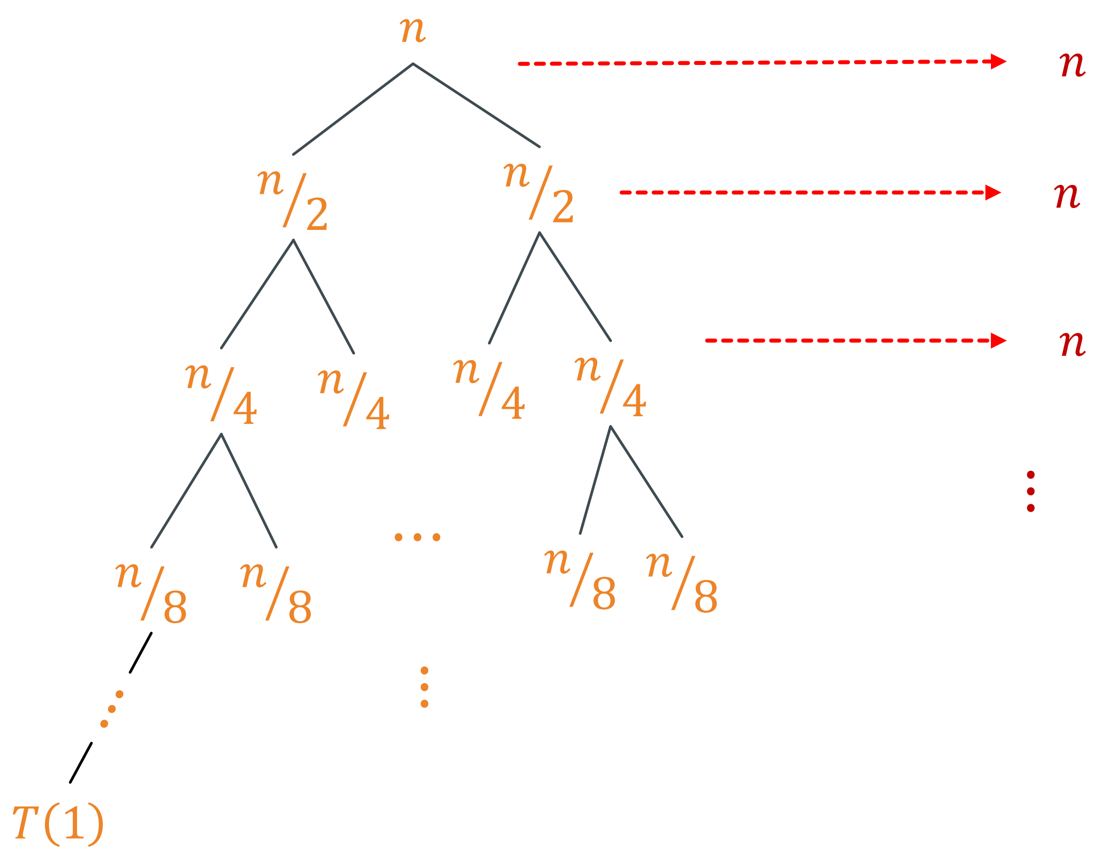
- $T(n) = 2.T\left(\frac{n}{2}\right) + n$
- $T\left(\frac{n}{2}\right) = 2.T\left(\frac{n}{2^2}\right) + \frac{n}{2}$
- $T\left(\frac{n}{2^2}\right) = 2.T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}$
- $T\left(\frac{n}{2^{k-1}}\right) = 2\,.T\left(\frac{n}{2^k}\right) + \frac{n}{2^{k-1}}$
- $T\left(\frac{n}{2^k}\right) = T(1)$



- Total cost = [Cost of Leaf Nodes] + [Cost of Internal Nodes]
- Total cost = [cost of leaf node × total leaf nodes] + [sum of costs at each level of internal nodes]
- Total cost = $L_c + I_c$

RECURSION TREE METHOD: EXAMPLE 1

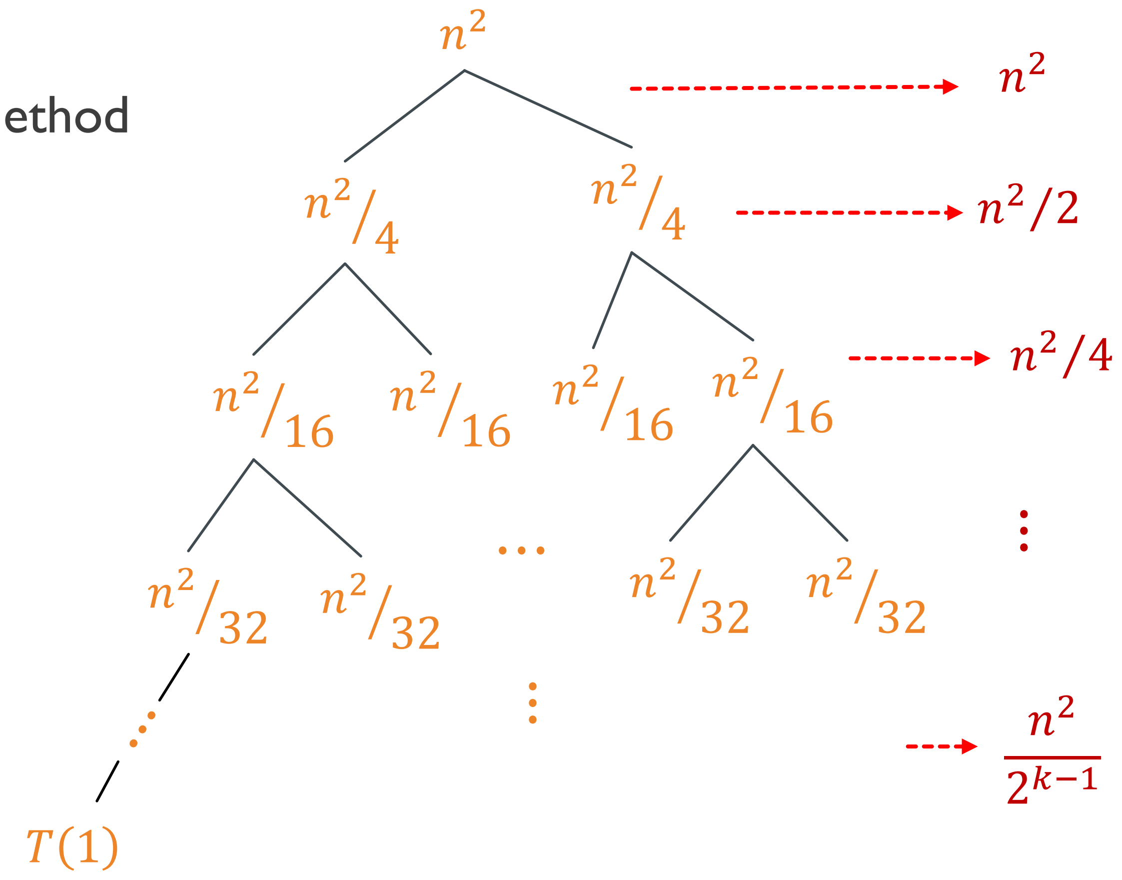
- $T\left(\frac{n}{2^k}\right) = T(1) \Longrightarrow n = 2^k \Longrightarrow k = \lg n$
- $L_c = 2^k \Longrightarrow 2^{\lg n} \Longrightarrow n^{\lg 2} \Longrightarrow n$
- $I_c = k.n = n \lg n$
- Total cost $= L_c + I_c \Longrightarrow n + n \lg n$
- Hence, $T(n) \in O(n \lg n)$



RECURSION TREE METHOD: EXAMPLE 2

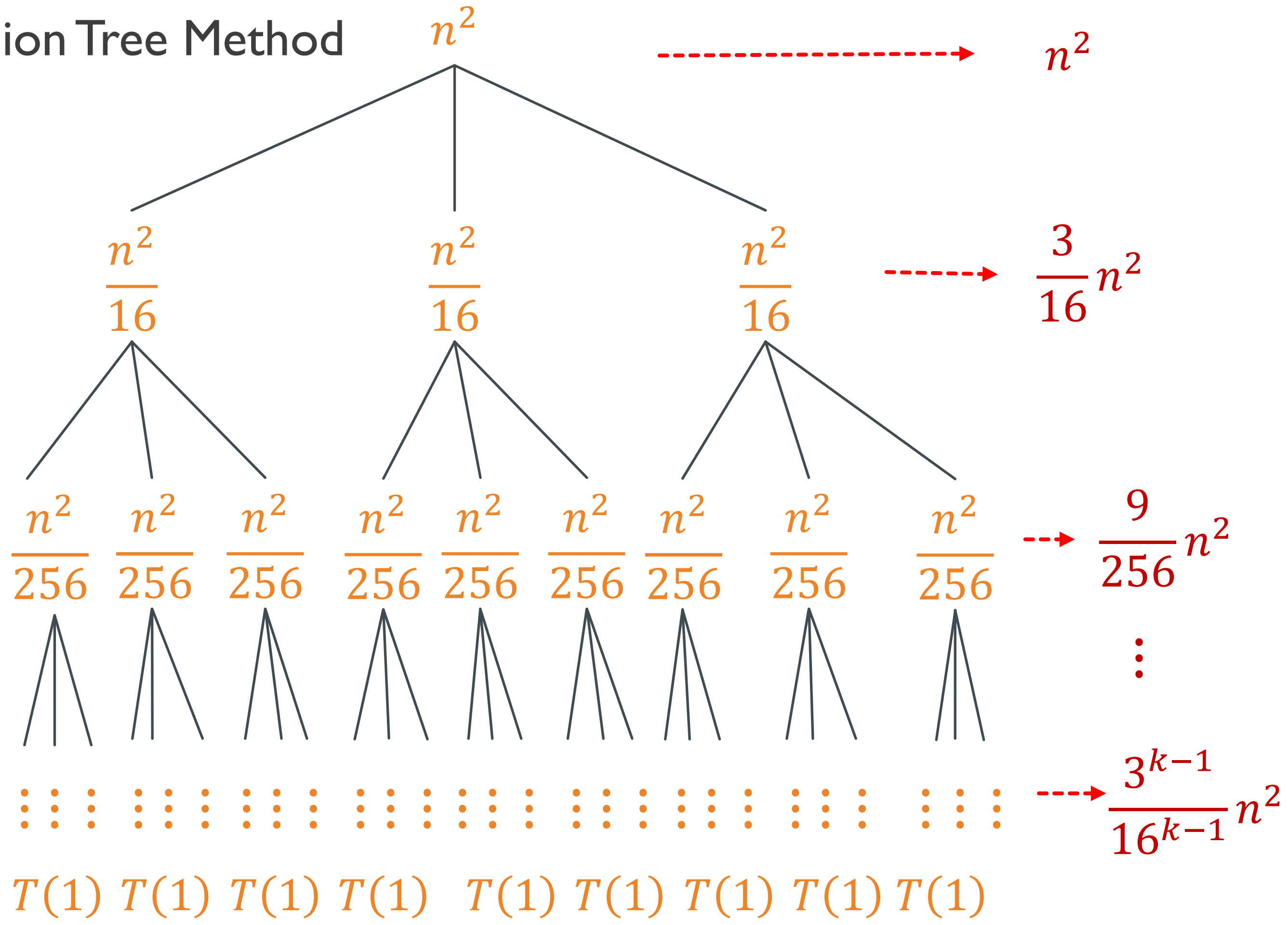
- $T(n) = \begin{cases} 1 & n = 1 \\ 2\,T\left(\frac{n}{2}\right) + n^2 & n > 1 \end{cases}$
- Solve the following recurrence using the Recursion Tree Method

- $T(n) = 2.T\left(\frac{n}{2}\right) + n^2$
- $T\left(\frac{n}{2}\right) = 2.T\left(\frac{n}{2^2}\right) + \frac{n^2}{2^2}$
- $T\left(\frac{n}{2^2}\right) = 2.T\left(\frac{n}{2^3}\right) + \frac{n^2}{4^2}$
- $T\left(\frac{n}{2^k}\right) = T(1) \Longrightarrow\ n = 2^k \Longrightarrow k = \lg n$
- $L_c = 2^k \Longrightarrow 2^{\lg n} \Longrightarrow n^{\lg 2} \Longrightarrow n$
- $I_c = n^2.\left[\left(\frac{1}{2}\right)^0 + \left(\frac{1}{2}\right)^1 + \left(\frac{1}{2}\right)^2 + \cdots + \left(\frac{1}{2}\right)^{k-1}\right] = n^2.\left[\frac{1}{1-\frac{1}{2}}\right] \Longrightarrow 2.n^2$
- Total cost = $L_c + I_c = n + 2.n^2$, Hence $T(n) \in O(n^2)$



- $T(n) = \begin{cases} 1 & n = 1 \\ 3\,T\left(\frac{n}{4}\right) + n^2 & n > 1 \end{cases}$
- Solve the following recurrence using the Recursion Tree Method
- Assumption: n is an exact power of 4

- $T(n) = 3\,.T\left(\frac{n}{4}\right) + n^2$
- $T\left(\frac{n}{4}\right) = 3.T\left(\frac{n}{4^2}\right) + \frac{n^2}{4^2}$
- $T\left(\frac{n}{4^2}\right) = 3.T\left(\frac{n}{4^3}\right) + \frac{n^2}{16^2}$
- $T\left(\frac{n}{4^k}\right) = T(1) \Rightarrow n = 4^k \Rightarrow k = \log_4 n$
- $L_c = 3^k \Rightarrow 3^{\log_4 n} \Rightarrow n^{\log_4 3} \Rightarrow n$

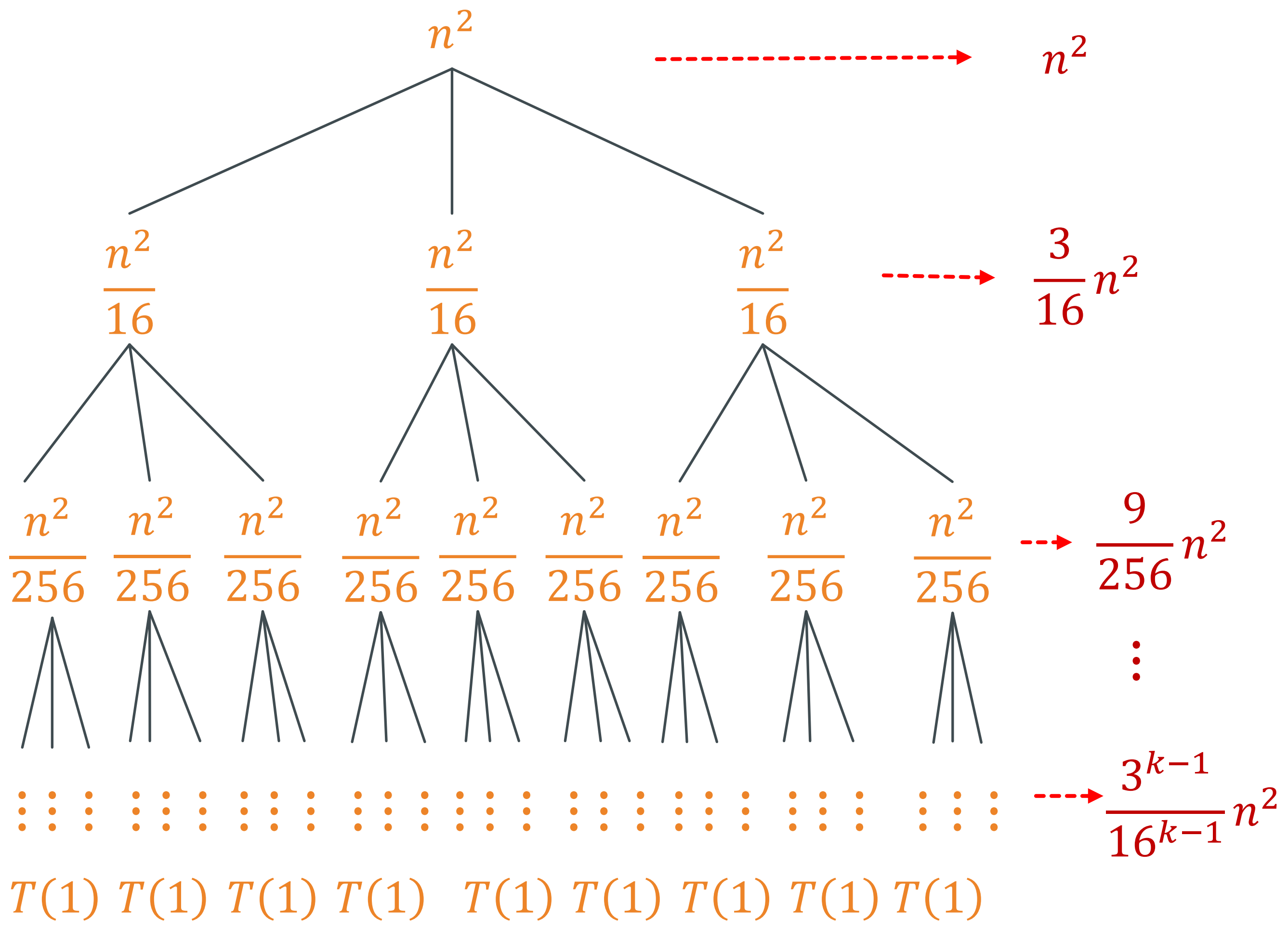


RECURSION TREE METHOD: EXAMPLE 3

- $T(n) = 3\,.T\left(\frac{n}{4}\right) + n^2$
- $T\left(\frac{n}{4}\right) = 3.T\left(\frac{n}{4^2}\right) + \frac{n^2}{4^2}$
- $T\left(\frac{n}{4^2}\right) = 3.T\left(\frac{n}{4^3}\right) + \frac{n^2}{16^2}$
- $T\left(\frac{n}{4^k}\right) = T(1) \Longrightarrow\ n = 4^k \Longrightarrow k = \log_4 n$
- $L_c = 3^k \Longrightarrow 3^{\log_4 n} \ \Longrightarrow n^{\log_4 3}$
- $I_c =$

$= n^2.\left[(\frac{3}{16})^0 + (\frac{3}{16})^1 + (\frac{3}{16})^2 + \cdots + (\frac{3}{16})^{k-1}\right]$

$= n^2.\left[\frac{1}{1-\frac{3}{16}}\right] \Longrightarrow \frac{16}{13}.n^2$



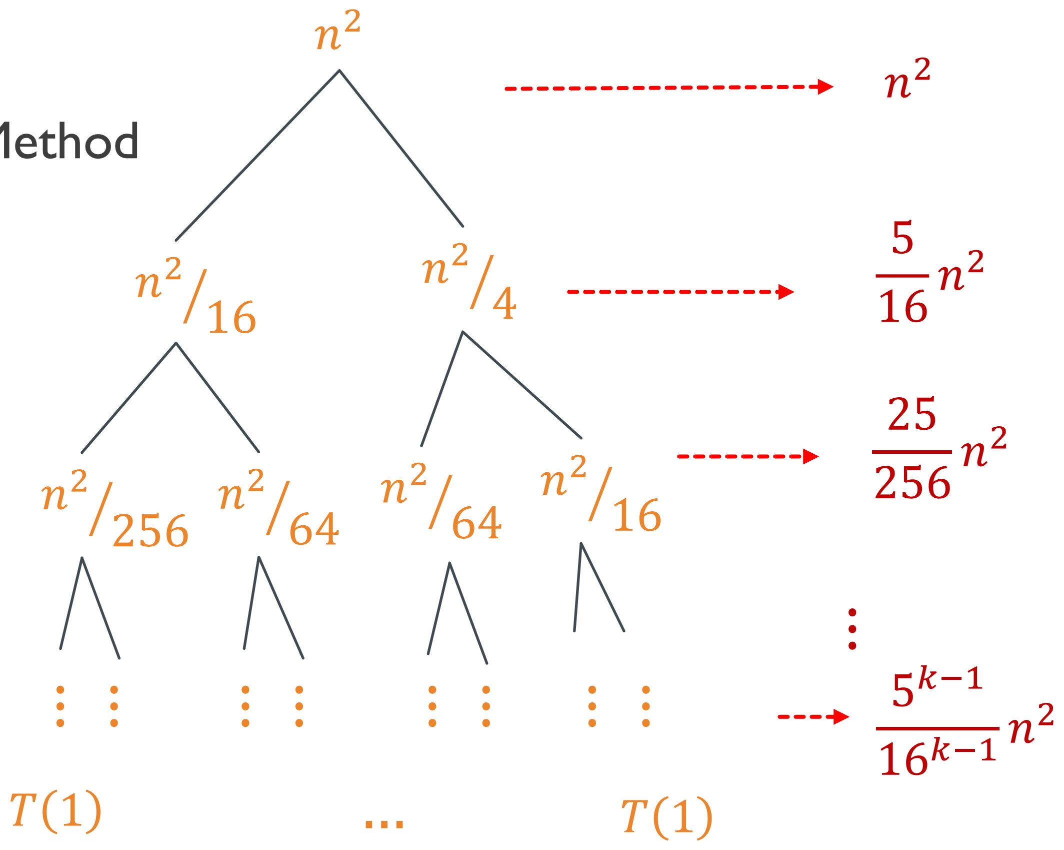
- Total cost = $L_c + I_c = n^{\log_4 3} \ + \frac{16}{13}.n^2,$

  Hence $T(n) \in \ O(n^2)$

RECURSION TREE METHOD: EXAMPLE 4

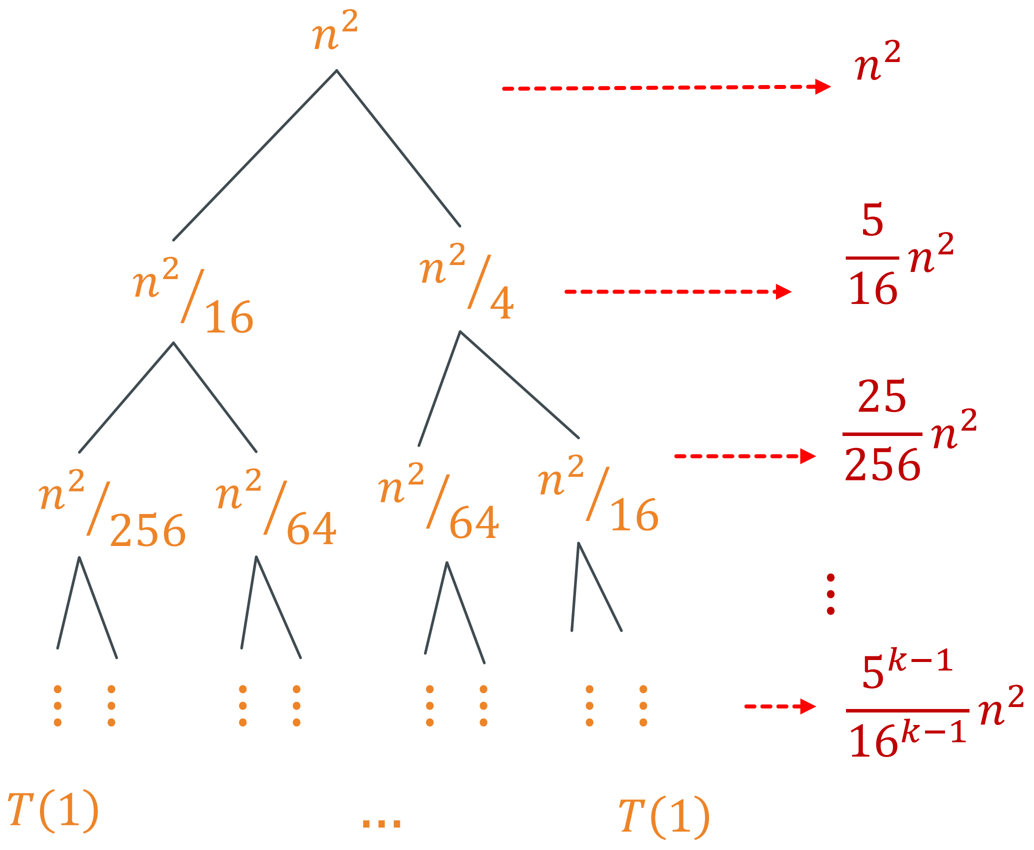
- $T(n) = \begin{cases} 1 & n = 1 \\ T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2 & n > 1 \end{cases}$

- Solve the following recurrence using the Recursion Tree Method
  - $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$
  - $T\left(\frac{n}{4}\right) = T\left(\frac{n}{16}\right) + T\left(\frac{n}{8}\right) + \frac{n^2}{16}$
  - $T\left(\frac{n}{2}\right) = T\left(\frac{n}{8}\right) + T\left(\frac{n}{4}\right) + \frac{n^2}{4}$
  - $T\left(\frac{n}{16}\right) = T\left(\frac{n}{64}\right) + T\left(\frac{n}{32}\right) + \frac{n^2}{256}$
  - $T\left(\frac{n}{8}\right) = T\left(\frac{n}{32}\right) + T\left(\frac{n}{16}\right) + \frac{n^2}{64}$
  - $T\left(\frac{n}{4}\right) = T\left(\frac{n}{16}\right) + T\left(\frac{n}{8}\right) + \frac{n^2}{16}$



RECURSION TREE METHOD: EXAMPLE 4

- $T(n) = \begin{cases} 1 & n = 1 \\ T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2 & n > 1 \end{cases}$



- $T\left(\frac{n}{2^k}\right) = T(1) \implies n = 2^k \implies k = \lg n$
- $L_c = 2^k \implies 2^{\lg n} \implies n^{\lg 2} \implies n$
- $I_c =$

$= n^2 . \left[\left(\frac{5}{16}\right)^0 + \left(\frac{5}{16}\right)^1 + \left(\frac{5}{16}\right)^2 + \cdots + \left(\frac{5}{16}\right)^{k-1} \right]$

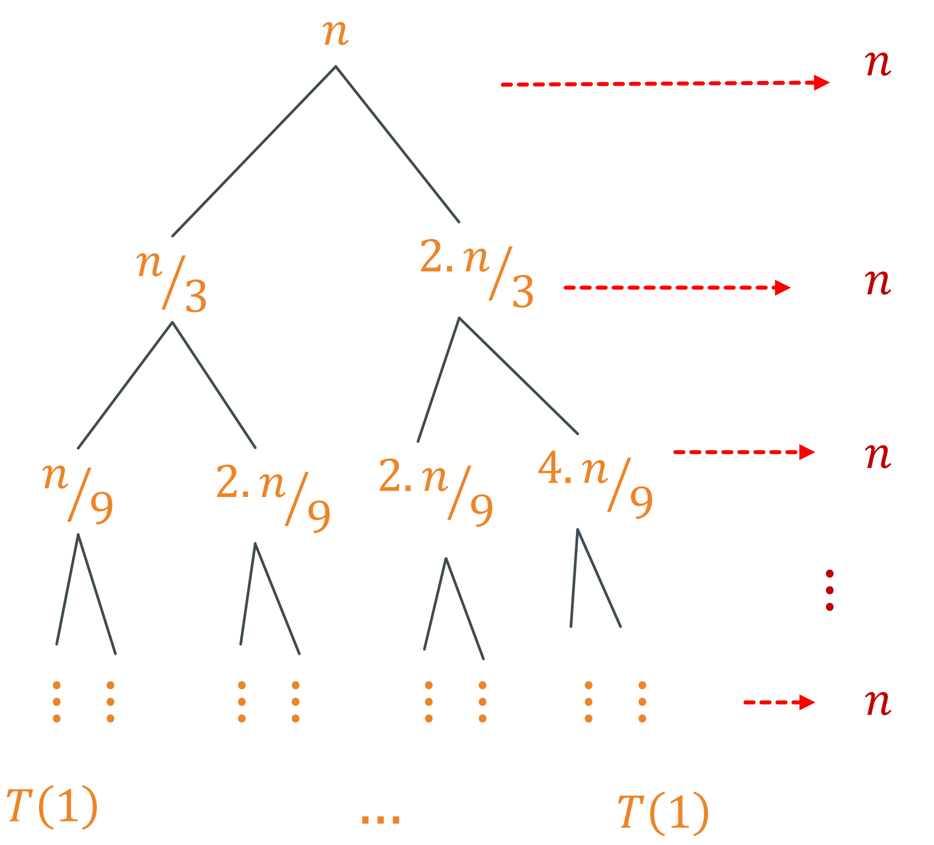
$= n^2 . \left[\frac{1}{1 - \frac{5}{16}} \right] \implies \frac{16}{11} . n^2$

- Total cost = $L_c + I_c = n + \frac{16}{11} . n^2$, Hence $T(n) \in O(n^2)$

RECURSION TREE METHOD: EXAMPLE 5

- $T(n) = \begin{cases} 1 & = 1 \\ T\left(\frac{n}{3}\right) + T\left(\frac{2.n}{3}\right) + n & n > 1 \end{cases}$

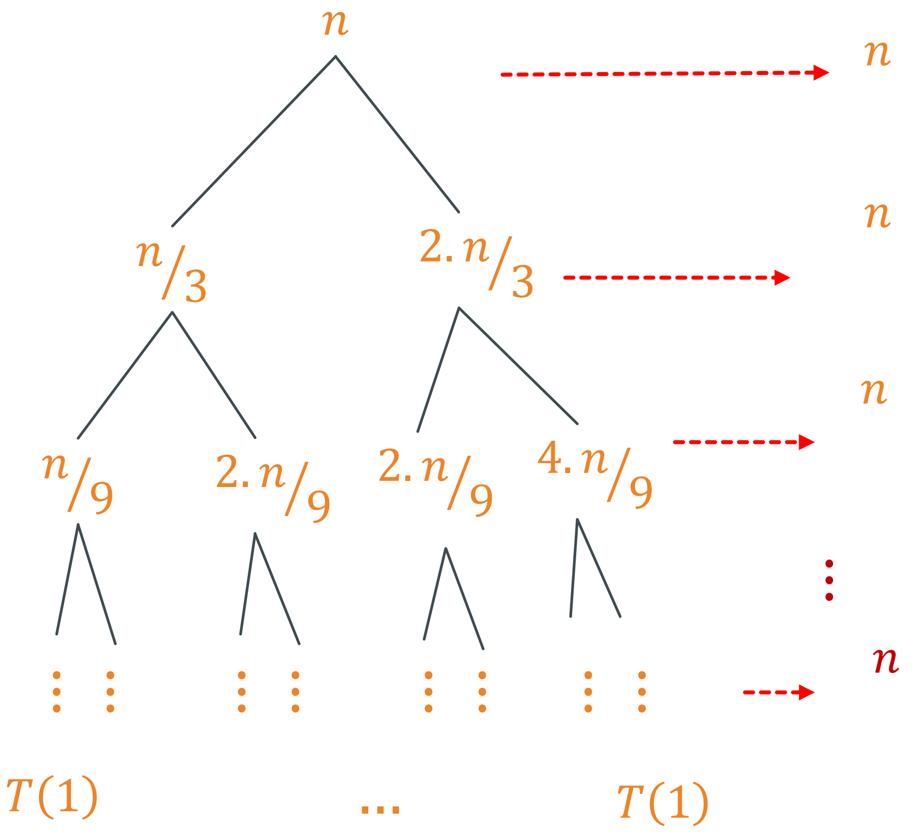
- $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2.n}{3}\right) + n$
- $T\left(\frac{n}{3}\right) = T\left(\frac{n}{9}\right) + T\left(\frac{2n}{9}\right) + \frac{n}{3}$
- $T\left(\frac{2.n}{3}\right) = T\left(\frac{2.n}{9}\right) + T\left(\frac{4.n}{9}\right) + \frac{2.n}{3}$
- $T\left(\frac{n}{9}\right) = T\left(\frac{n}{27}\right) + T\left(\frac{2.n}{27}\right) + \frac{n}{9}$
- $T\left(\frac{2.n}{9}\right) = T\left(\frac{2.n}{27}\right) + T\left(\frac{4.n}{27}\right) + \frac{2.n}{9}$
- $T\left(\frac{4.n}{9}\right) = T\left(\frac{4.n}{27}\right) + T\left(\frac{8.n}{27}\right) + \frac{4.n}{9}$



RECURSION TREE METHOD: EXAMPLE 5

- $T(n) = \begin{cases} 1 & n = 1 \\ T\left(\frac{n}{3}\right) + T\left(\frac{2.n}{3}\right) + n & n > 1 \end{cases}$

- $T\left(\frac{2^k}{3^k} n\right) = T(1) \implies n = \frac{3^k}{2^k} \implies k = \log_{3/2} n$
- $L_c = 2^k \implies 2^{\log_{3/2} n} \implies n^{\log_{3/2} 2}$
- $I_c = n.k = n.\log_{3/2} n$
- Total cost = $L_c + I_c = n^{\log_{3/2} 2} + n.\log_{3/2} n,$
- $T(n) \in O(n.\log n)$? Difficult to decide which side is bigger $\implies O$ is ambiguous
- Solution: Use both terms in the total cost as guesses in the substitution method (Try it out: You will find $n.\log n$ is a correct guess …)



RECURSION TREE METHOD: CAUTION NOTE

- Recursion trees are best used to generate good guesses
  - Verify guesses using the substitution method
- A small amount of "sloppiness" can be tolerated
  - Using an infinite decreasing geometric series as an upper bound
  - Assuming "n" to be an exact power of $2, 3, or\ 4$
  - Assuming the tree is complete. In reality, the tree may have fewer internal and leaf nodes
- By carefully drawing out a recursion tree and summing the costs, the recursion tree method can be used as a direct proof of a solution to any recurrence

RECURSION TREE METHOD: PRACTICE QUESTIONS

- Solve the following recurrences using the recurrence tree method.

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

MASTER THEOREM

MASTER THEOREM

- The Master Method depends on the following theorem
- Theorem:
  Let $a \geq 1, b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the non-negative integers by the recurrence: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$
  Then $T(n)$ can be bounded asymptotically as follows:

1. $If\ f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
2. $If\ f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$
3. $If\ f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$,
   $\boldsymbol{and}$ $f(n)$ satisfies the regularity condition: $a f\left(\frac{n}{b}\right) \leq c f(n)$ for some constant $c < 1$,
   and all sufficiently large n
   then $T(n) = \Theta(f(n))$

MASTER THEOREM

- Important to note that the three cases do not cover all the possibilities
  - Gap between cases 1 and 2
    when $f(n)$ is smaller than $n^{log_b a}$ but not polynomially smaller
  - Gap between cases 2 and 3
    when $f(n)$ is larger than $n^{log_b a}$ but not polynomially larger
- If $f(n)$ falls into one of these gaps, or if the regularity condition in case 3 fails to hold, the master method cannot be used to solve the recurrence

MASTER THEOREM: EXAMPLE 1

- $T(n) = 2\,T\left(\frac{n}{2}\right) + n$ //Merge Sort
- $a \;= 2$
- $b \;= 2$
- $f(n) = n$
- $n^{log_b a} \Longrightarrow n^{log_2 2} \Longrightarrow n$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) = n^{log_b a}$ so case 2 is applied. $[f(n) \; = \Theta(n^{log_b a})]$
  - $\Longrightarrow T(n) \; = \Theta(n^{log_b a} \lg n)$

    $= \Theta(n^{log_2 2} \lg n)$

    $= \Theta(n \lg n)$
- Hence: $T(n) \; = \Theta(n \lg n)$

MASTER THEOREM: EXAMPLE 2

- $T(n) = 2\,T\left(\frac{n}{2}\right) + n^2$ //Quick Sort
- $a = 2$
- $b = 2$
- $f(n) = n^2$
- $n^{log_b a} \Longrightarrow n^{log_2 2} \Longrightarrow n$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) > n^{log_b a}$ so case 3 is applied. $[f(n) = \Omega(n^{log_b a + \varepsilon})]$
  - $\Longrightarrow T(n) = \Theta(f(n))$

    $= \Theta(n^2)$
- Hence: $T(n) = \Theta(n^2)$

Verify Regularity Condition:

- ✓ $a.\,f\left(\frac{n}{b}\right) \le c\, f(n)$
- ✓ $2.\,f\left(\frac{n}{2}\right) \le c\,.\,n^2$
- ✓ $2.\,\frac{n^2}{4} \le c\,.\,n^2$
- ✓ $\frac{1}{2} \le c$

MASTER THEOREM: EXAMPLE 3

- $T(n) = 9\,T\left(\frac{n}{3}\right) + n$
- $a \ = 9$
- $b \ = 3$
- $f(n) = n$
- $n^{log_b a} \Longrightarrow n^{log_3 9} \Longrightarrow n^2$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) < n^{log_b a}$ so case 1 is applied. $[f(n) = O(n^{log_b a\, -\, \varepsilon})]$
  - $\Longrightarrow T(n) \ = \Theta(n^{log_b a})$
    $= \Theta(n^{log_3 9})$
    $= \Theta(n^2)$

- Hence: $T(n) \ = \Theta(n^2)$

MASTER THEOREM: EXAMPLE 4

- $T(n) = T\left(\frac{n}{2}\right) + 1$ //Binary Search
- $a = 1$
- $b = 2$
- $f(n) = n^0$
- $n^{log_b a} \Longrightarrow n^{log_2 1} \Longrightarrow n^0$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) = n^{log_b a}$ so case 2 is applied. $[f(n) = \Theta(n^{log_b a})]$
  - $\Longrightarrow T(n) = \Theta(n^{log_b a} \lg n)$

    $= \Theta(n^{log_2 1} \lg n)$

    $= \Theta(n^0 \lg n)$
- Hence: $T(n) = \Theta(\lg n)$

MASTER THEOREM: EXAMPLE 5

- $T(n) = 4\,T\left(\frac{n}{2}\right) + n^3$
- $a\ = 4$
- $b\ = 2$
- $f(n) = n^3$
- $n^{log_b a} \Longrightarrow n^{log_2 4} \Longrightarrow n^2$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) > n^{log_b a}$ so case 3 is applied. $[f(n) = \Omega(n^{log_b a + \varepsilon})]$
  - $\Longrightarrow T(n)\ = \Theta(f(n))$

    $= \Theta(n^3)$
- Hence: $T(n)\ = \Theta(n^3)$

Verify Regularity Condition:

- ✓ $a.\,f\left(\frac{n}{b}\right) \le c\,f(n)$
- ✓ $4.\,f\left(\frac{n}{2}\right) \le c\,.\,n^3$
- ✓ $4.\frac{n^3}{8} \le c\,.\,n^3$
- ✓ $\frac{1}{2} \le c$

MASTER THEOREM: EXAMPLE 6

- $T(n) = T\left(\frac{n}{2}\right) + n^2$
- $a = 1$
- $b = 2$
- $f(n) = n^2$
- $n^{log_b a} \Longrightarrow n^{log_2 1} \Longrightarrow n^0$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) > n^{log_b a}$ so case 3 is applied. $[f(n) = \Omega(n^{log_b a + \varepsilon})]$
  - $\Longrightarrow T(n) = \Theta(f(n))$
    $= \Theta(n^2)$
- Hence: $T(n) = \Theta(n^2)$

Verify Regularity Condition:

- ✓ $a.f\left(\frac{n}{b}\right) \leq c\, f(n)$
- ✓ $1.f\left(\frac{n}{2}\right) \leq c\,.\,n^2$
- ✓ $\frac{n^2}{4} \leq c\,.\,n^2$
- ✓ $\frac{1}{4} \leq c$

MASTER THEOREM: EXAMPLE 7

- $T(n) = 4.T\left(\frac{n}{2}\right) + n^2$
- $a \ = 4$
- $b \ = 2$
- $f(n) = n^2$
- $n^{log_b a} \Longrightarrow n^{log_2 4} \Longrightarrow n^2$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) = n^{log_b a}$ so case 2 is applied. $[f(n) \ = \Theta(n^{log_b a})]$
  - $\Longrightarrow T(n) \ = \Theta(n^{log_b a} \lg n)$

    $= \Theta(n^{log_2 4} \lg n)$

    $= \Theta(n^2 \lg n)$
- Hence: $T(n) \ = \Theta(n^2 \lg n)$

MASTER THEOREM: EXAMPLE 8

- $T(n) = 7.T\left(\frac{n}{3}\right) + n^2$
- $a \;= 7$
- $b \;= 3$
- $f(n) = n^2$
- $n^{log_b a} \Longrightarrow n^{log_3 7} \Longrightarrow n^{1.77}$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) > n^{log_b a}$ so case 3 is applied. $[f(n) = \Omega(n^{log_b a + \varepsilon})]$
  - $\Longrightarrow T(n) \;= \Theta(f(n))$

    $= \Theta(n^2)$

- Hence: $T(n) \;= \Theta(n^2)$

Verify Regularity Condition:

- ✓ $a.f\left(\frac{n}{b}\right) \leq c\, f(n)$
- ✓ $7.f\left(\frac{n}{3}\right) \leq c\,.n^2$
- ✓ $7.\frac{n^2}{9} \leq c\,.n^2$
- ✓ $\frac{7}{9} \leq c$

MASTER THEOREM: EXAMPLE 9

- $T(n) = 7\,T\left(\frac{n}{2}\right) + n^2$
- $a \;= 7$
- $b \;= 2$
- $f(n) = n^2$
- $n^{log_b a} \Longrightarrow n^{log_2 7} \Longrightarrow n^{2.81}$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) < n^{log_b a}$ so case 1 is applied. $[f(n) = O(n^{log_b a\,-\,\varepsilon})]$
  - $\Longrightarrow T(n) \;= \Theta(n^{log_b a})$

    $= \Theta(n^{log_2 7})$

    $= \Theta(n^{2.81})$

- Hence: $T(n) \;= \Theta(n^{2.81})$

MASTER THEOREM: EXAMPLE 10

- $T(n) = 2\,T\left(\frac{n}{2}\right) + \sqrt{n}$
- $a = 2$
- $b = 2$
- $f(n) = n^{(1/2)}$
- $n^{log_b a} \Longrightarrow n^{log_2 2} \Longrightarrow n^1$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) < n^{log_b a}$ so case 1 is applied. $[f(n) = O(n^{log_b a - \varepsilon})]$
  - $\Longrightarrow T(n) = \Theta(n^{log_b a})$

    $= \Theta(n^{log_2 2})$

    $= \Theta(n)$

- Hence: $T(n) = \Theta(n)$

MASTER THEOREM: EXAMPLE 11

- $T(n) = 3.T\left(\frac{n}{4}\right) + n\log n$
- $a \ = 3$
- $b \ = 4$
- $f(n) = n\log n$
- $n^{log_b a} \Longrightarrow n^{log_4 3} \Longrightarrow n^{0.79}$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow f(n) > n^{log_b a}$ so case 3 is applied. $[f(n) = \Omega(n^{log_b a+\varepsilon})]$
  - $\Longrightarrow T(n) \ = \Theta(f(n))$

    $= \Theta(n\log n)$

- Hence: $T(n) \ = \Theta(n\log n)$

Verify Regularity Condition:

- ✓ $a.f\left(\frac{n}{b}\right) \leq c\, f(n)$
- ✓ $3.f\left(\frac{n}{4}\right) \leq c\, .n\lg n$
- ✓ $3.\frac{n}{4}\lg\frac{n}{4} \leq c\, .n\lg n$
- ✓ $\frac{3}{4}\ [(\lg n) - 2] \leq c\, .n\lg n$
- ✓ $\frac{3}{4} \leq c$

MASTER THEOREM: EXAMPLE 12

- $T(n) = 4.T\left(\frac{n}{2}\right) + \frac{n^2}{\lg n}$
- $a \;= 4$
- $b \;= 2$
- $f(n) = \frac{n^2}{\lg n}$
- $n^{log_b a} \Longrightarrow n^{log_2 4} \Longrightarrow n^2$
- Compare $f(n)$ and $n^{log_b a}$:
  - $\Longrightarrow Non - polynomial\ difference\ between\ f(n)\ and\ n^{log_b a}$
  - Master Method does not apply
    - The difference must be polynomially larger by a factor of n^{ε} where $\varepsilon > 0$.
    - In this case the difference is only larger by a factor of $\frac{1}{\lg n}$

MASTER THEOREM: EXAMPLE 13

- $T(n) = 2.T\left(\frac{n}{2}\right) + n\log n$
- $a \; = 2$
- $b \; = 2$
- $f(n) = n\log n$
- $n^{log_b a} \Longrightarrow n^{log_2 2} \Longrightarrow n^1$
- Compare $f(n)$ and $n^{log_b a}$:
  - *Seems like case 3 should apply*
  - Master Method does not apply. Non polynomial difference between $f(n)$ and $n^{log_b a}$
    - The difference must be polynomially larger by a factor of n^{ε} where $\varepsilon > 0$.
    - In this case the difference is only larger by a factor of $\log n$

MASTER THEOREM : PRACTICE QUESTIONS

- Solve the following recurrences using the Master Method.

- $T(n) = 3.T\left(\frac{n}{2}\right) + n^2$
- $T(n) = 16.T\left(\frac{n}{4}\right) + n^2$
- $T(n) = T\left(\frac{2.n}{5}\right) + n$
- $T(n) = 3.T\left(\frac{n}{2}\right) + n$
- $T(n) = 3.T\left(\frac{n}{3}\right) + 1$
- $T(n) = 16.T\left(\frac{n}{4}\right) + n$
- $T(n) = T\left(\frac{n}{2}\right) + n$
- $T(n) = 2.T\left(\frac{n}{4}\right) + n^{0.51}$
- $T(n) = 4.T\left(\frac{n}{2}\right) + n$
- $T(n) = 3.T\left(\frac{n}{3}\right) + n$
- $T(n) = n.T\left(\frac{n}{2}\right) + n$
- $T(n) = 2.T\left(\frac{n}{2}\right) + 2^n$
- $T(n) = 4.T\left(\frac{n}{2}\right) + \lg n$
- $T(n) = 4.T\left(\frac{n}{2}\right) + \frac{n}{\lg n}$
- $T(n) = \frac{1}{2}.T\left(\frac{n}{2}\right) + \frac{1}{n}$
- $T(n) = 4.T\left(\frac{n}{2}\right) + n\lg n$
- $T(n) = 6.T\left(\frac{n}{3}\right) + n^2\lg n$
- $T(n) = 64.T\left(\frac{n}{8}\right) + n^2\lg n$

QUESTIONS/ANSWERS