Recursion Tree-

Like **Master's Theorem**, Recursion Tree is another method for solving the recurrence relations.

A recursion tree is a tree where each node represents the cost of a certain recursive sub-problem.

- We sum up the values in each node to get the cost of the entire algorithm.

-

Steps to solve Recurrence relations using Recursion tree method-

-

Step-01:

-

Draw a recursion tree based on the given recurrence relation.

-

-

Step-02:

-

Determine-

- Cost of each level
- Total number of levels in the recursion tree
- Number of nodes in the last level
- Cost of the last level

-

Step-03:

-

Add cost of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation.

-

Following problems clearly illustrates how to apply these steps.

-

PRACTICE PROBLEMS BASED ON RECURSION TREE-

-

Problem-01:

Solve the following recurrence relation using recursion tree method-

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

-

Solution-

-

Step-01:

-

Draw a recursion tree based on the given recurrence relation.

-

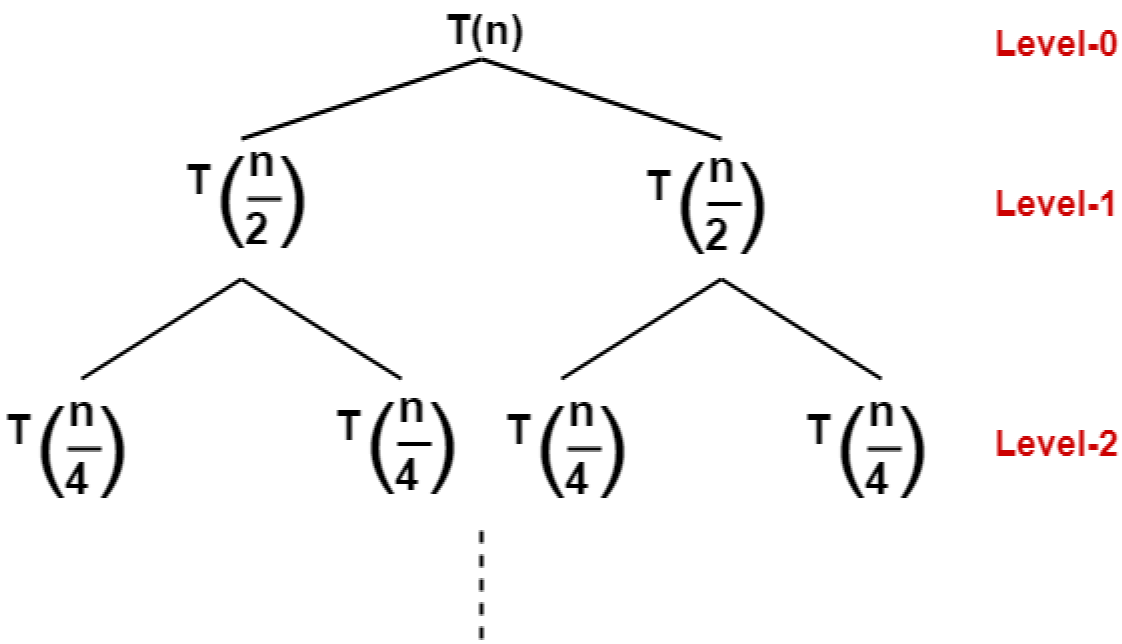
The given recurrence relation shows-

- A problem of size n will get divided into 2 sub-problems of size n/2.
- Then, each sub-problem of size n/2 will get divided into 2 sub-problems of size n/4 and so on.
- At the bottom most layer, the size of sub-problems will reduce to 1.

-

This is illustrated through following recursion tree-

-



-

The given recurrence relation shows-

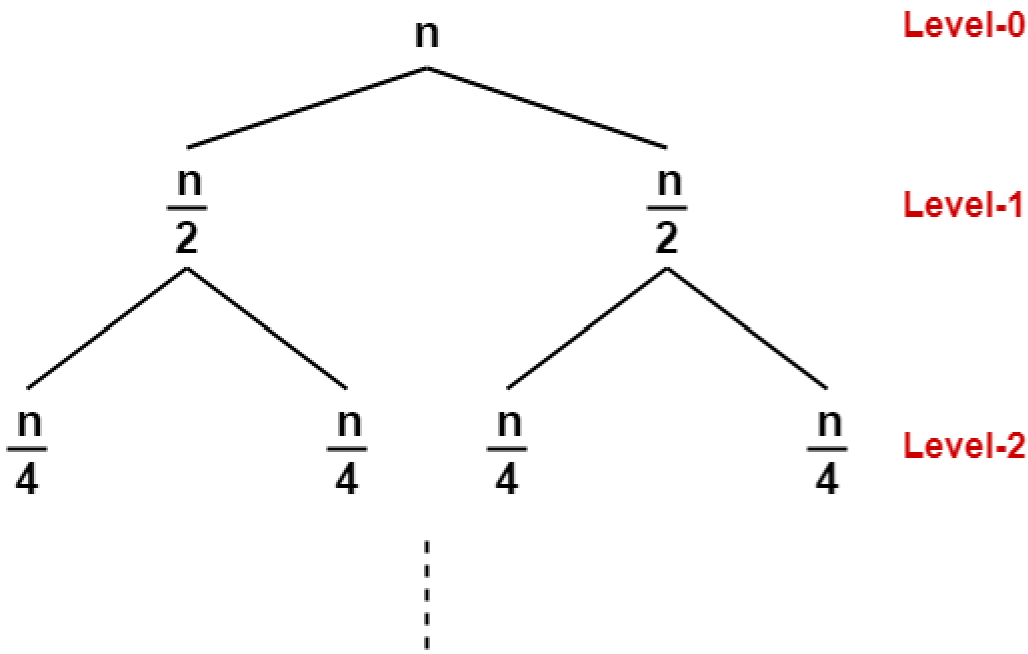
- The cost of dividing a problem of size n into its 2 sub-problems and then combining its solution is n.

- The cost of dividing a problem of size n/2 into its 2 sub-problems and then combining its solution is n/2 and so on.

-

This is illustrated through following recursion tree where each node represents the cost of the corresponding sub-problem-

-



-

Step-02:

-

Determine cost of each level-

- Cost of level-0 = n
- Cost of level-1 = n/2 + n/2 = n
- Cost of level-2 = n/4 + n/4 + n/4 + n/4 = n and so on.

-

Step-03:

-

Determine total number of levels in the recursion tree-

- Size of sub-problem at level-0 = $n/2^0$
- Size of sub-problem at level-1 = $n/2^1$
- Size of sub-problem at level-2 = $n/2^2$

-

Continuing in similar manner, we have-

Size of sub-problem at level-i = $n/2^i$

Suppose at level-x (last level), size of sub-problem becomes 1. Then-

$n / 2^x = 1$

$2^x = n$

Taking log on both sides, we get-

$x\log 2 = \log n$

$x = \log_2 n$

-

∴ Total number of levels in the recursion tree = $\log_2 n + 1$

-

Step-04:

-

Determine number of nodes in the last level-

- Level-0 has 2^0 nodes i.e. 1 node
- Level-1 has 2^1 nodes i.e. 2 nodes
- Level-2 has 2^2 nodes i.e. 4 nodes

-

Continuing in similar manner, we have-

Level-$\log_2 n$ has $2^{\log_2 n}$ nodes i.e. n nodes

-

Step-05:

-

Determine cost of last level-

Cost of last level = n x T(1) = θ(n)

-

Step-06:

-

Add costs of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation-

-

$$T(n) = \underbrace{\{\, n + n + n + \ldots\ldots \,\}}_{\text{For } \log_2 n \text{ levels}} + \theta\,(n)$$

-

$= n \times \log_2 n + \theta\ (n)$

$= n\log_2 n + \theta\ (n)$

$= \theta\ (n\log_2 n)$

-

Problem-02:

Solve the following recurrence relation using recursion tree method-

$T(n) = T(n/5) + T(4n/5) + n$

-

Solution-

-

Step-01:

-

Draw a recursion tree based on the given recurrence relation.

-

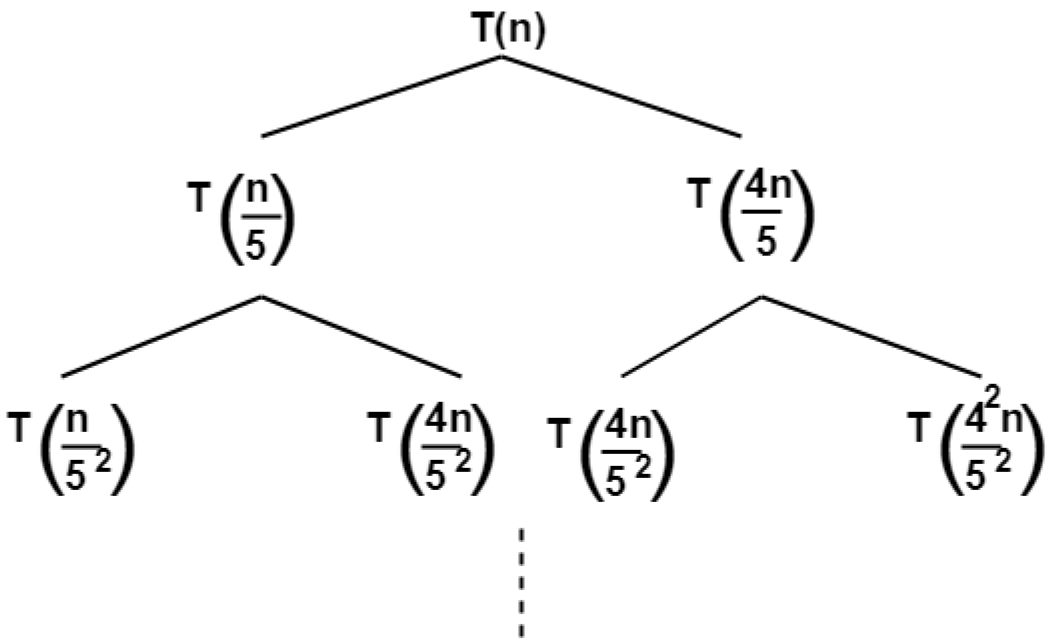
The given recurrence relation shows-

- A problem of size n will get divided into 2 sub-problems- one of size n/5 and another of size 4n/5.
- Then, sub-problem of size n/5 will get divided into 2 sub-problems- one of size $n/5^2$ and another of size $4n/5^2$.
- On the other side, sub-problem of size 4n/5 will get divided into 2 sub-problems- one of size $4n/5^2$ and another of size $4^2n/5^2$ and so on.
- At the bottom most layer, the size of sub-problems will reduce to 1.

-

This is illustrated through following recursion tree-

-



-

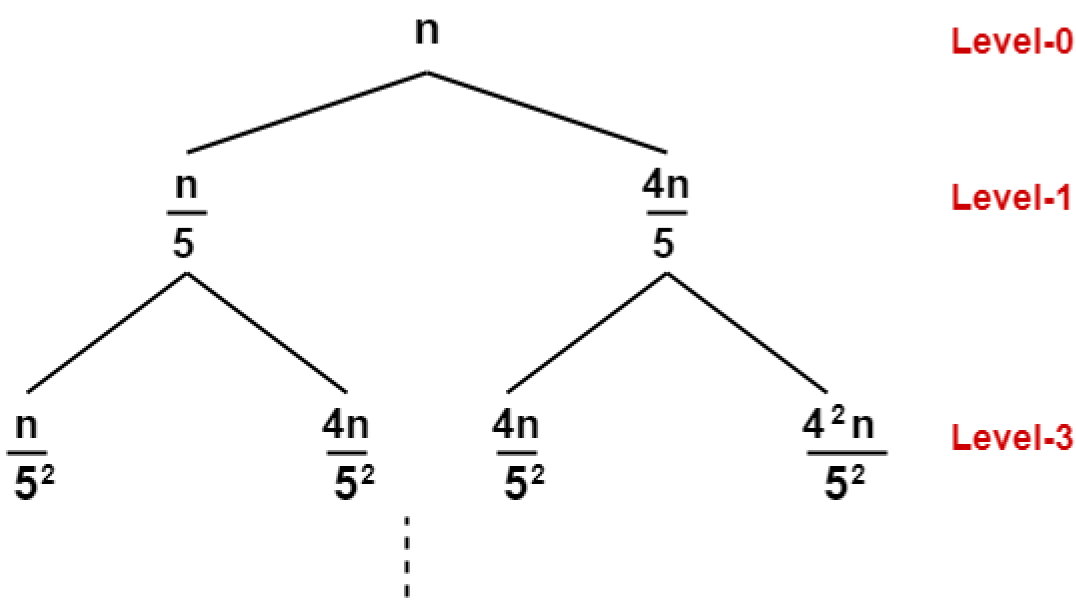
The given recurrence relation shows-

- The cost of dividing a problem of size n into its 2 sub-problems and then combining its solution is n.
- The cost of dividing a problem of size n/5 into its 2 sub-problems and then combining its solution is n/5.
- The cost of dividing a problem of size 4n/5 into its 2 sub-problems and then combining its solution is 4n/5 and so on.

-

This is illustrated through following recursion tree where each node represents the cost of the corresponding sub-problem-

-



-

Step-02:

-

Determine cost of each level-

- Cost of level-0 = n
- Cost of level-1 = n/5 + 4n/5 = n
- Cost of level-2 = $n/5^2 + 4n/5^2 + 4n/5^2 + 4^2n/5^2 = n$

-

Step-03:

-

Determine total number of levels in the recursion tree. We will consider the rightmost sub tree as it goes down to the deepest level-

- Size of sub-problem at level-0 = $(4/5)^0n$
- Size of sub-problem at level-1 =$(4/5)^1n$
- Size of sub-problem at level-2 =$(4/5)^2n$

-

Continuing in similar manner, we have-

Size of sub-problem at level-i = $(4/5)^in$

Suppose at level-x (last level), size of sub-problem becomes 1. Then-

$(4/5)^xn = 1$

$(4/5)^x = 1/n$

Taking log on both sides, we get-

$x\log(4/5) = \log(1/n)$

$x = \log_{5/4}n$

-

∴ Total number of levels in the recursion tree = $\log_{5/4}n + 1$

-

Step-04:

-

Determine number of nodes in the last level-

- Level-0 has 2^0 nodes i.e. 1 node
- Level-1 has 2^1 nodes i.e. 2 nodes
- Level-2 has 2^2 nodes i.e. 4 nodes

-

Continuing in similar manner, we have-

Level-$\log_{5/4}n$ has $2^{\log_{5/4}n}$ nodes

-

Step-05:

-

Determine cost of last level-

Cost of last level = $2^{\log_{5/4}n} \times T(1) = \theta(2^{\log_{5/4}n}) = \theta(n^{\log_{5/4}2})$

-

Step-06:

-

Add costs of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation-

-

$$T(n) = \underbrace{\{ n + n + n + \ldots.. \}}_{\text{For } \log_{5/4}n \text{ levels}} + \theta(n^{\log_{5/4}2})$$

-

= $n\log_{5/4}n + \theta(n^{\log_{5/4}2})$

= $\theta(n\log_{5/4}n)$

-

Problem-03:

Solve the following recurrence relation using recursion tree method-

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

-

Solution-

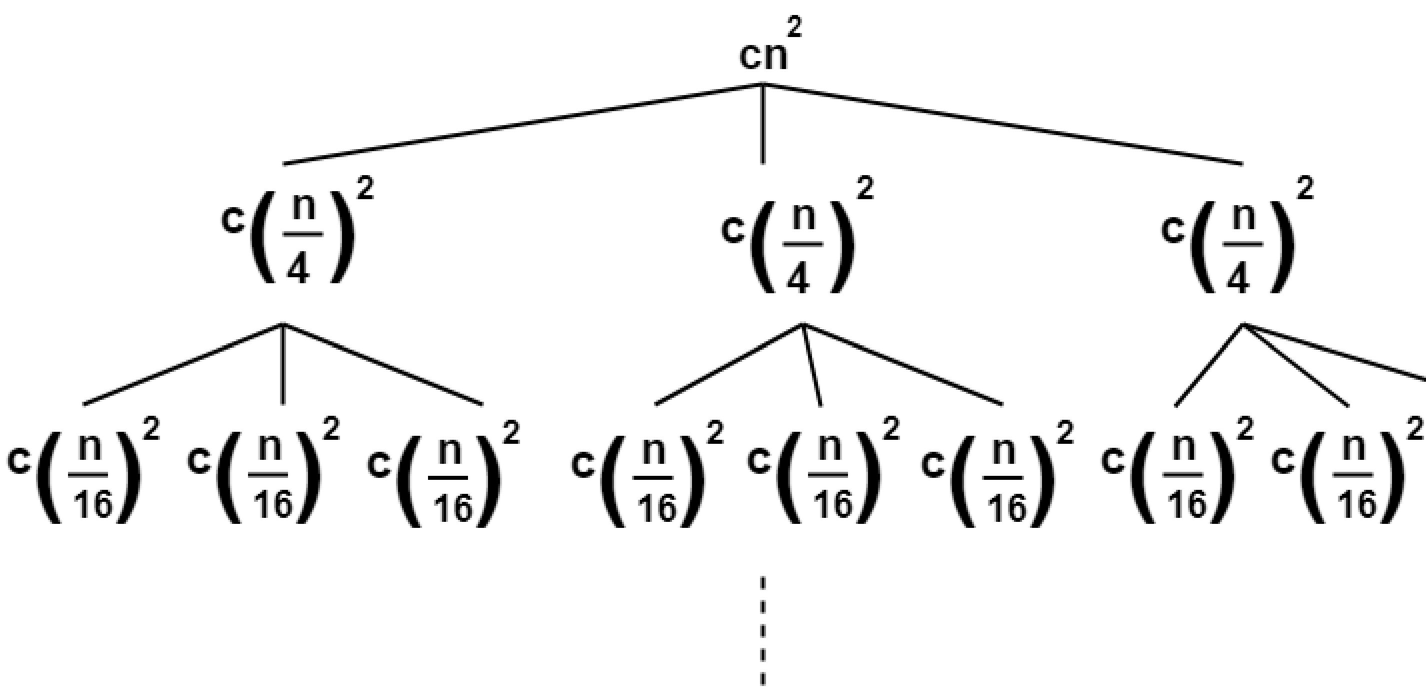
-

Step-01:

-

Draw a recursion tree based on the given recurrence relation-

-



-

(Here, we have directly drawn a recursion tree representing the cost of sub problems)

-

Step-02:

-

Determine cost of each level-

- Cost of level-0 = cn^2
- Cost of level-1 = $c(n/4)^2 + c(n/4)^2 + c(n/4)^2 = (3/16)cn^2$
- Cost of level-2 = $c(n/16)^2 \times 9 = (9/16^2)cn^2$

-

Step-03:

-

Determine total number of levels in the recursion tree-

- Size of sub-problem at level-0 = $n/4^0$
- Size of sub-problem at level-1 = $n/4^1$
- Size of sub-problem at level-2 = $n/4^2$

-

Continuing in similar manner, we have-

Size of sub-problem at level-i = $n/4^i$

Suppose at level-x (last level), size of sub-problem becomes 1. Then-

$n/4^x = 1$

$4^x = n$

Taking log on both sides, we get-

$x\log 4 = \log n$

$x = \log_4 n$

-

∴ Total number of levels in the recursion tree = $\log_4 n + 1$

-

Step-04:

-

Determine number of nodes in the last level-

- Level-0 has 3^0 nodes i.e. 1 node
- Level-1 has 3^1 nodes i.e. 3 nodes
- Level-2 has 3^2 nodes i.e. 9 nodes

-

Continuing in similar manner, we have-

Level-$\log_4 n$ has $3^{\log_4 n}$ nodes i.e. $n^{\log_4 3}$ nodes

-

Step-05:

-

Determine cost of last level-

Cost of last level = $n^{\log_4 3} \times T(1) = \theta(n^{\log_4 3})$

-

Step-06:

-

Add costs of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation-

-

$$T(n) = \underbrace{\left\{ cn^2 + \frac{3\,cn^2}{16} + \frac{9\,cn^2}{(16)^2} + \ldots\ldots\ldots\ldots \right\}}_{\text{For } \log_4 n \text{ levels}} + \theta(n^{\log_4 3})$$

-

$= cn^2 \{ 1 + (3/16) + (3/16)^2 + \ldots\ldots\ldots \} + \theta(n^{\log_4 3})$

-

Now, $\{ 1 + (3/16) + (3/16)^2 + \ldots\ldots\ldots \}$ forms an infinite Geometric progression.

-

On solving, we get-

$= (16/13)cn^2 \{ 1 – (3/16)^{\log_4 n} \} + \theta(n^{\log_4 3})$

$= (16/13)cn^2 – (16/13)cn^2 (3/16)^{\log_4 n} + \theta(n^{\log_4 3})$

$= O(n^2)$

-

the recursion-tree method

1. solving recurrences
   - expanding the recurrence into a tree
   - summing the cost at each level
   - applying the substitution method
2. another example
   - using a recursion tree

MCS 360 Lecture 39
Introduction to Data Structures
Jan Verschelde, 22 November 2010

solving recurrences

The substitution method for solving recurrences consists of two steps:

1. Guess the form of the solution.
2. Use mathematical induction to find constants in the form and show that the solution works.

In the previous lecture, the focus was on step 2.

Today we introduce the recursion-tree method to generate a guess for the form of the solution to the recurrence.

the recursion-tree method

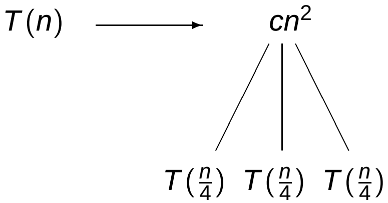
an example

Consider the recurrence relation

$$T(n) = 3T(n/4) + cn^2 \quad \text{for some constant } c.$$

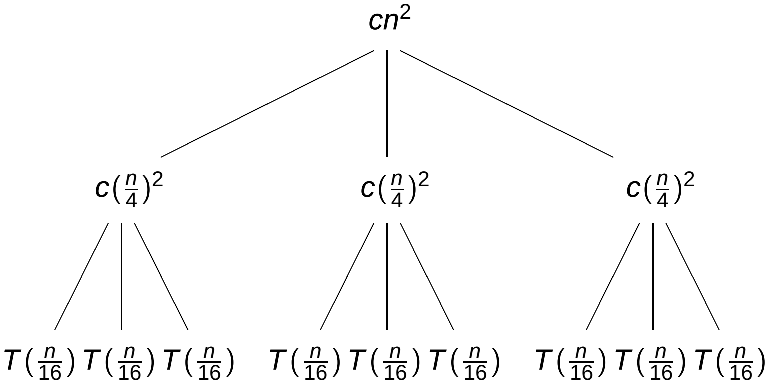
We assume that n is an exact power of 4.

In the recursion-tree method we expand $T(n)$ into a tree:



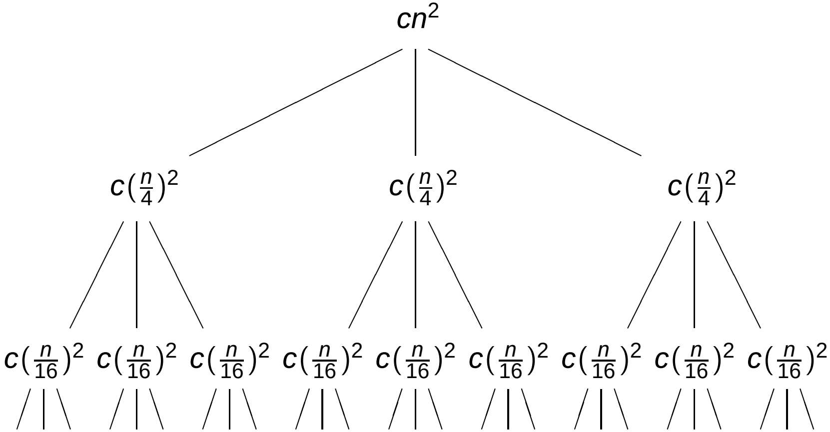
we expand $T(\frac{n}{4})$

Applying $T(n) = 3T(n/4) + cn^2$ to $T(n/4)$ leads to $T(n/4) = 3T(n/16) + c(n/4)^2$, expanding the leaves:



we expand $T(\frac{n}{16})$

Applying $T(n) = 3T(n/4) + cn^2$ to $T(n/16)$ leads to $T(n/16) = 3T(n/64) + c(n/16)^2$, expanding the leaves:



the recursion-tree method

1. solving recurrences
   - expanding the recurrence into a tree
   - summing the cost at each level
   - applying the substitution method

2. another example
   - using a recursion tree

the cost at each level

We sum the cost at each level of the tree:

$$cn^2 \qquad = cn^2$$

$$c\left(\frac{n}{4}\right)^2 + c\left(\frac{n}{4}\right)^2 + c\left(\frac{n}{4}\right)^2 \qquad = \frac{3}{16}cn^2$$

$$c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2 + c\left(\frac{n}{16}\right)^2$$

$$= \left(\frac{3}{16}\right)^2 cn^2$$

adding up the costs

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots \\
&= cn^2 \left(1 + \frac{3}{16} + \left(\frac{3}{16}\right)^2 + \cdots\right)
\end{aligned}
$$

The $\cdots$ disappear if $n = 16$,
or the tree has depth at least 2 if $n \geq 16 = 4^2$.

For $n = 4^k$, $k = \log_4(n)$, we have:

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

geometric series

Consider a finite sum first:

$$S_n = 1 + r + r^2 + \cdots + r^n = \sum_{i=0}^{n} r^i.$$

To find an explicit form of the solution we do

$$\begin{array}{rcrcrcrcccrcr} rS_n & = & & & r & + & r^2 & + & \cdots & + & r^n & + & r^{n+1} \\ -S_n & = & 1 & + & r & + & r^2 & + & \cdots & + & r^n & & \\ \hline (r-1)S_n & = & -1 & & & & & & & & & + & r^{n+1} \end{array}$$

So the explicit sum is

$$S_n = \frac{r^{n+1} - 1}{r - 1}.$$

applying the geometric sum

Applying

$$S_n = \sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}$$

to

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

with $r = \frac{3}{16}$ leads to

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

polishing the result

Instead of $T(n) \leq dn^2$ for some constant d, we have

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

Recall

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

To remove the $\log_4(n)$ factor, we consider

$$\begin{aligned} T(n) &\leq cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i \\ &= cn^2 \frac{-1}{\frac{3}{16} - 1} \leq dn^2, \text{ for some constant } d. \end{aligned}$$

the recursion-tree method

verifying the guess

Let us see if $T(n) \leq dn^2$ is good for $T(n) = 3T(n/4) + cn^2$.

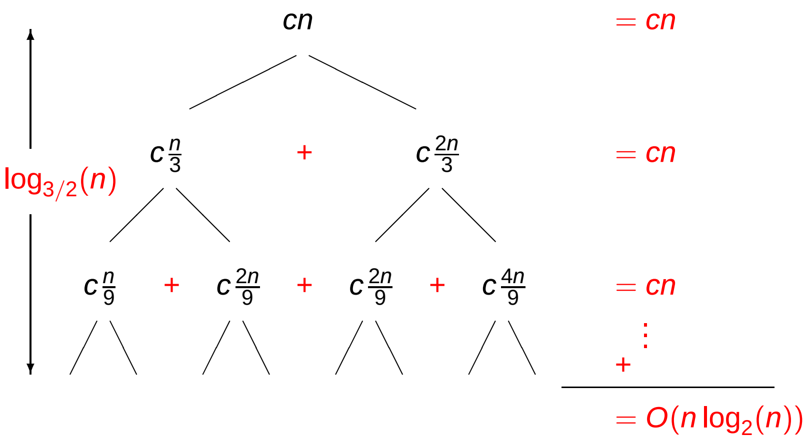
Applying the substitution method:

$$\begin{aligned} T(n) &= 3T(n/4) + cn^2 \\ &\leq 3d\left(\frac{n}{4}\right)^2 + cn^2 \\ &= \left(\frac{3}{16}d + c\right) n^2 \\ &= \frac{3}{16}\left(d + \frac{16}{3}c\right) n^2 \\ &\leq \frac{3}{16}(2d)\,n^2, \quad \text{if } d \geq \frac{16}{3}c \\ &\leq dn^2 \end{aligned}$$

the recursion-tree method

using a recursion tree

Consider $T(n) = T(n/3) + T(2n/3) + cn$.



Summary + Assignments

We covered §4.4 of *Introduction to Algorithms*, 3rd edition by Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson.

Assignments:

1. Consider $T(n) = 3T(n/2) + n$. Use a recursion tree to derive a guess for an asymptotic upper bound for $T(n)$ and verify the guess with the substitution method.
2. Same question as before for $T(n) = T(n/2) + n^2$.
3. Same question as before for $T(n) = 2T(n-1) + 1$.

Last homework collection on Monday 29 November:
#1 of L-30, #1 of L-31, #3 of L-32, #2 of L-33, #1 of L-34.

Final exam on Tuesday 7 December, 8-10AM in TH 216.