Recurrence Relations

Md. Faruk Hosen
Department of CIS
Daffodil International University

Need of Recursive Relations

- The following shows the recursive and iterative versions of the factorial function:

Recursive version

```
int factorial (int n)
{
    if (n == 0)
       return 1;
    else
       return  n * factorial (n-1);
}
```

Recursive Call

Iterative version

```
int factorial (int n)
{
    int i, product=1;
    for (i=n; i>1; --i)
         product=product * i;

    return product;
}
```

```
Factorial(n)
{              1
  if n == 0
    return 1
  else
    return n x factorial(n-1)
}         1            1
```

$T(n) = T(n-1) + 3$ if $n > 0$

$T(0) = 1$

$$
\begin{aligned}
T(n) &= T(n-1) + 3 \\
&= T(n-2) + 6 \\
&= T(n-3) + 9 \\
&= T(n-k) + 3k
\end{aligned}
$$

$n - k = 0 \Rightarrow k = n$

$$
\begin{aligned}
\Rightarrow T(n) &= T(0) + 3n \\
&= 3n
\end{aligned}
$$

Recurrence Relations (1/2)

- A recurrence relation is an equation which is defined in terms of itself with smaller value.
- Why are recurrences good things?
  - Many natural functions are easily expressed as recurrences:
    - $a_n = a_{n-1} + 1, a_1 = 1$ --> $a_n = n$ (polynomial)
    - $a_n = 2a_{n-1}, a_1 = 1$ --> $a_n = 2^n$ (exponential)
    - $a_n = na_{n-1}, a_1 = 1$ --> $a_n = n!$ (weird function)
- It is often easy to find a recurrence as the solution of a counting problem

Recurrence Relations (2/2)

- In both, we have general and boundary conditions, with the general condition breaking the **problem into smaller and smaller pieces**.
- The initial or boundary condition terminate the recursion.

Recurrence Equations

- A recurrence equation defines a function, say T(n). The function is defined recursively, that is, the function T(.) appear in its definition. (recall recursive function call). The recurrence equation should have a base case.

For example:

$$T(n) = \begin{cases} T(n-1)+T(n-2), & \text{if } n>1 \\ 1, & \text{if } n=1 \text{ or } n=0. \end{cases}$$

base case

for convenient, we sometime write the recurrence equation as:

$$T(n) = T(n-1)+T(n-2)$$
$$T(0) = T(1) = 1.$$

Recurrence Examples

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

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

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

Methods for Solving Recurrences

- **Iteration method (Backward Substitution Method**
- **Substitution method**
- **Recursion tree method**
- **Master method**

Simplications:

- There are two simplications we apply that won't affect asymptotic analysis
  - Ignore floors and ceilings (justification in text)
  - Assume base cases are constant, i.e., $T(n) = \Theta(1)$ for n small enough

Iteration Method (Backward Substitution)

- Expand the recurrence
- Work some algebra to express as a summation
- Evaluate the summation

Iteration Method

$$\mathbf{T(n) = c + T(n/2)}$$

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

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

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

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

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

Assume $n = 2^k$ $\quad k = \log_2 n$

$T(n) = \underbrace{c + c + \ldots + c}_{k \text{ times}} + T(1)$

$= c\log_2 n + T(1)$

$= \Theta(\lg n)$

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

- s(n) =

  c + s(n-1)

  c + c + s(n-2)

  2c + s(n-2)

  2c + c + s(n-3)

  3c + s(n-3)

  ...

  kc + s(n-k) = ck + s(n-k)
- What if k = n?
  - s(n) = cn + s(0) = cn

Iteration Method

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

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

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

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

$= 16(4T(n/8+n/4)) + 2n + n$ /**simplify**/

$= 64(T(n/8) +4n+2n+n$

$= 4^{\log n} T(1)+ \ldots + 4n + 2n + n$ /** #levels = log n **/

$$= c4^{\log n} + n\sum_{k=0}^{\log n-1} 2^{k}$$ /** convert to summation**/

$$= cn^{\log 4} + n\left(\frac{2^{\log n} - 1}{2 - 1}\right)$$ /** $a^{\log b} = b^{\log a}$ **/

Solving Recurrences: Iteration (convert to summation) (cont.)

$= cn^2 + n(n^{\log 2} - 1)$ /** $2^{\log n} = n^{\log 2}$ **/

$= cn^2 + n(n - 1)$

$= cn^2 + n^2 - n$

$= \Theta(n^2)$

1. $T(n) = T(n-1) + n \qquad n > 1$

$T(n) = 1 \qquad n = 1$

Solution:

$T(n) = T(n-1) + n \qquad .....(1)$

$T(n-1) = T(n-2) + n - 1 \qquad .....(2)$

Substituting (2) in (1) $T(n) = T(n$

$-2) + n - 1 + n \qquad .....(3)$

$T(n-2) = T(n-3) + n - 2 \qquad .....(4)$

Substituting (4) in (3)

$T(n) = T(n-3) + n - 2 + n - 1 + n \qquad .....(5)$

General equation

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

$.....(6)$

$T(1) = 1$

$n - k = 1$

$k = n - 1 \qquad .....(7)$

Substituting (7) in (6)

$T(n) = T(1) + 2 + 3 + \ldots\ldots n - 1 + n$

$= 1 + 2 + 3 + \ldots\ldots + n$

$= n(n+1)/2 \quad T(n) = O(n^2)$

2. $T(n) = T(n-1) + b \quad n > 1$

$T(n) = 1 \quad n = 1$

Solution:

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

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

Substituting (2) in (1)

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

$T(n) = T(n-2) + 2b$ .....(3)

$T(n-2) = T(n-3) + b$ .....(4)

Substituting (4) in (3)

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

General equation $T(n)$

$= T(n-k) + k.b \;\; T(1)$

$= 1$

$n - k = 1 \;\; k = n - 1$

$T(n) = T(1) + (n-1)\,b$

$= 1 + bn - b \;\; T(n) =$

$O(n)$

3. $T(n) = 2\,T(n-1) + b$ $\quad n > 1$

$T(n) = 1$ $\quad n = 1$

Solution:

$T(1) = 1$

$T(2) = 2.T(1) + b$

$= 2 + b$

$= 2^1 + b$

$T(3) = 2T(2) + b$

$= 2(2 + b) + b$

$= 4 + 2b + b$

$= 4 + 3b$

$= 2^2 + (2^2 - 1)b$ $T(4) =$

$2.T(3) + b$

$= 2(4 + 3b) + b$

$= 8 + 7b$

$= 2^3 + (2^3 - 1)b$

General equation

$T(k) = 2^{k-1} + (2^{k-1} - 1)b$

.

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

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

$= 2^n(b + 1)/2 - b$ Let c

$= (b + 1)/2$ $T(n) = c\,2^n$

$- b$

$= O(2^n)$

4. $T(n) = T(n/2) + b$ $\quad n > 1$

$T(1) = 1$ $\quad n = 1$

Solution:

$T(n) = T(n/2) + b$ .....(1)

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

Substituting (2) in (1)

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

$= T(n/4) + 2b$ .....(3)

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

Substituting (4) in (3)

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

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

General equation

$T(n) = T(n/2^k) + kb$ .....(5)

$T(1) = 1$

$n/2^k = 1$

$2^k = n$

$K = \log n$ .....(6)

Substituting (6) in (5)

$T(n) = T(1) + b.\log n$

$= 1 + b \log n$ $T(n) =$

$O(\log n)$

The substitution method

1. Guess a solution
2. Use induction to prove that the solution works

Substitution method

- Guess a solution
  - $T(n) = O(g(n))$
  - Induction goal: apply the definition of the asymptotic notation
    - $T(n) \leq d\, g(n)$, for some $d > 0$ and $n \geq n_0$
  - Induction hypothesis: $T(k) \leq d\, g(k)$ for all $k < n$
- Prove the induction goal
  - Use the **induction hypothesis** to find some values of the constants d and n_0 for which the **induction goal** holds

Example: Binary Search

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

- Guess: $T(n) = O(\lg n)$
  - Induction goal: $T(n) \le d \lg n$, for some d and $n \ge n_0$
  - Induction hypothesis: $T(n/2) \le d \lg(n/2)$
- Proof of induction goal:

$$T(n) = T(n/2) + c \le d \lg(n/2) + c$$

$$= d \lg n - d + c \le d \lg n$$

if: $-d + c \le 0$, $d \ge c$

Example 2

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

- Guess: $T(n) = O(n^2)$
  - Induction goal: $T(n) \le c\, n^2$, for some c and $n \ge n_0$
  - Induction hypothesis: $T(k-1) \le c(k-1)^2$ for all $k < n$
- Proof of induction goal:

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

$$= cn^2 - (2cn - c - n) \le cn^2$$

if: $2cn - c - n \ge 0 \Leftrightarrow c \ge n/(2n-1) \Leftrightarrow c \ge 1/(2 - 1/n)$

  - For $n \ge 1 \Rightarrow 2 - 1/n \ge 1 \Rightarrow$ any $c \ge 1$ will work

Example 3

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

- Guess: $T(n) = O(n\lg n)$
  - Induction goal: $T(n) \leq cn \lg n$, for some c and $n \geq n_0$
  - Induction hypothesis: $T(n/2) \leq cn/2 \lg(n/2)$
- Proof of induction goal:

$$T(n) = 2T(n/2) + n \leq 2c\,(n/2)\lg(n/2) + n$$

$$= cn \lg n - cn + n \leq cn \lg n$$

$$\text{if: } -cn + n \leq 0 \Rightarrow c \geq 1$$

Changing variables

$$T(n) = 2T(\sqrt{n}\ ) + \lg n$$

- Rename: $m = \lg n \Rightarrow n = 2^m$

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

- Rename: $S(m) = T(2^m)$

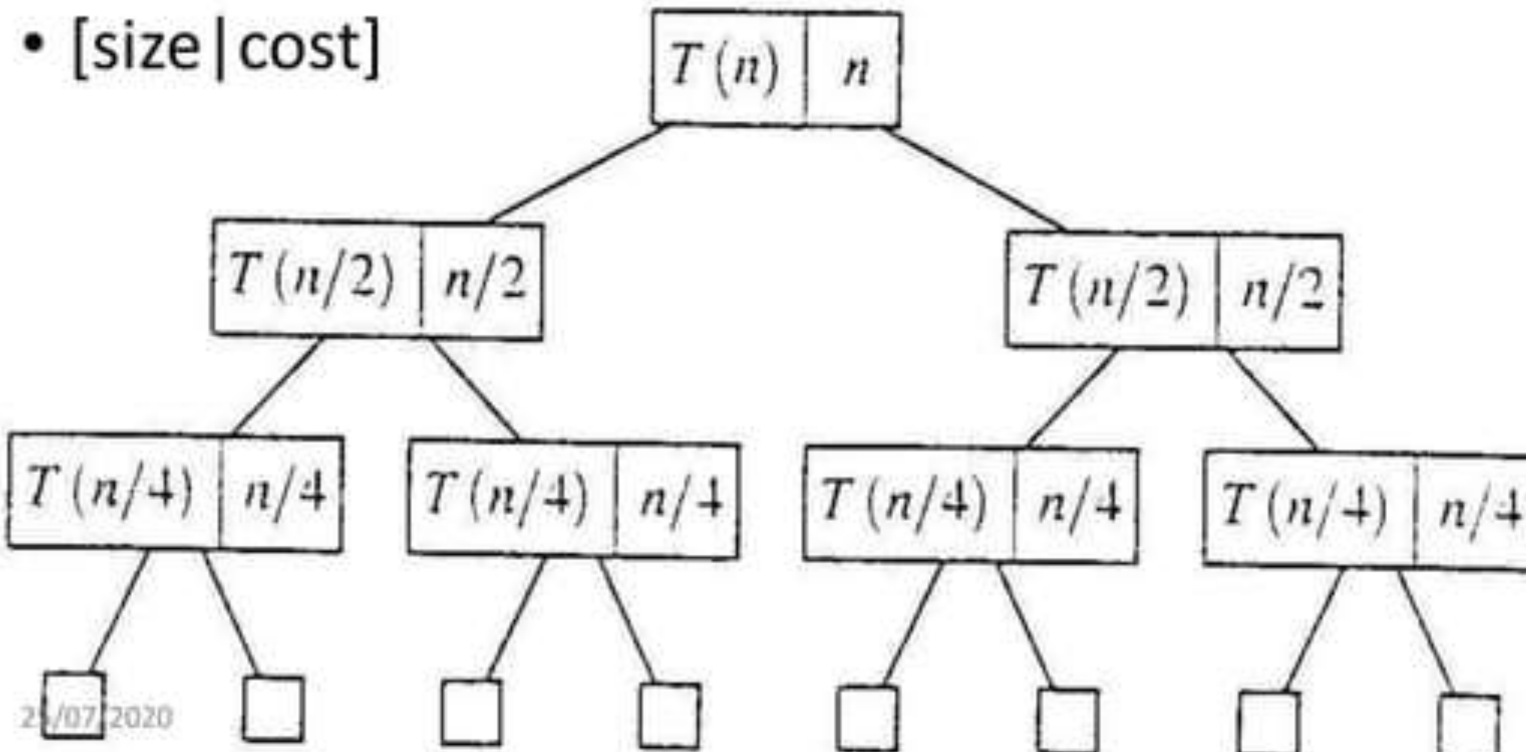
$S(m) = 2S(m/2) + m \Rightarrow S(m) = O(m\lg m)$
(demonstrated before)

$T(n) = T(2^m) = S(m) = O(m\lg m) = O(\lg n \lg\lg n)$

Idea: transform the recurrence to one that you have seen before

Recursion Tree

- Evaluate: $T(n) = T(n/2) + T(n/2) + n$
  - Work copy: $T(k) = T(k/2) + T(k/2) + k$
  - For $k=n/2$, $T(n/2) = T(n/4) + T(n/4) + (n/2)$
- [size|cost]



Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion tree method is good for generating guesses for the substitution method.
- The recursion-tree method can be unreliable.
- The recursion-tree method promotes intuition, however.

Recursion Tree

- To evaluate the total cost of the recursion tree

  Sum all the non-recursive costs of all nodes

  = Sum (rowSum(cost of all nodes at the same depth))

- Determine the maximum depth of the recursion tree:

  For our example, at tree depth d
  the size parameter is $n/(2^d)$

  the size parameter converging to base case, i.e. case 1

  such that, $n/(2^d) = 1$,

  $d = \lg(n)$

  The row Sum for each row is n

- Therefore, the total cost, $T(n) = n \lg(n)$

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$$T(n)$$

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$$
\begin{array}{ccc}
 & n^2 & \\
 \swarrow & & \searrow \\
T(n/4) & & T(n/2)
\end{array}
$$

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

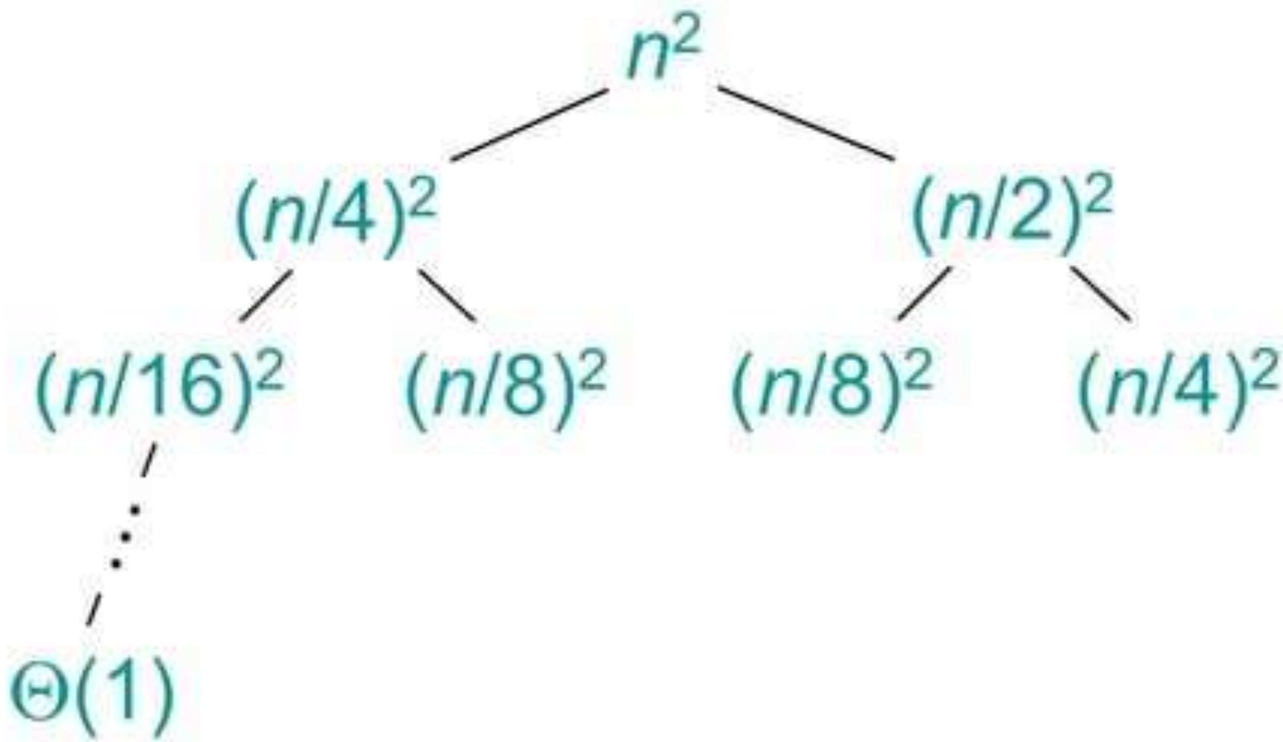
$$n^2$$

$$(n/4)^2 \qquad\qquad (n/2)^2$$

$$T(n/16) \quad T(n/8) \quad T(n/8) \quad T(n/4)$$

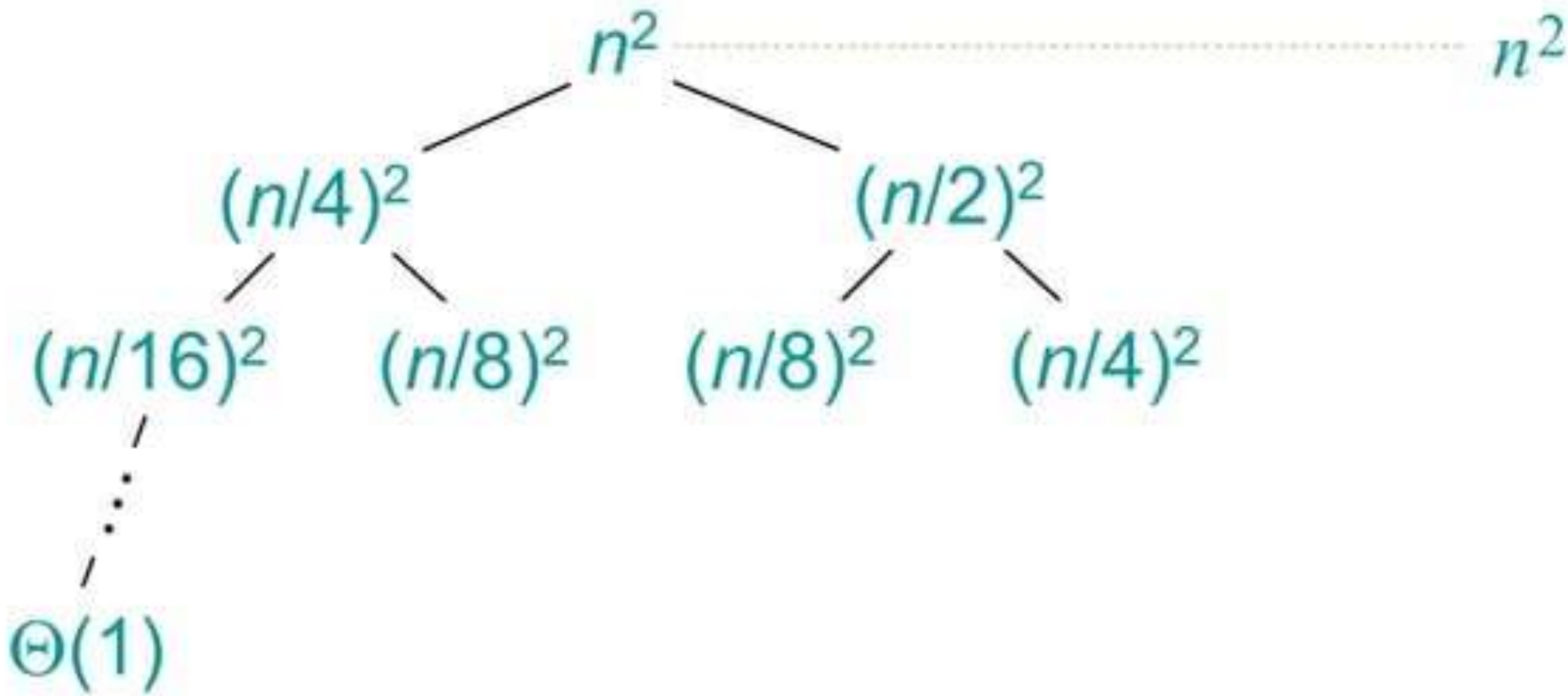
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



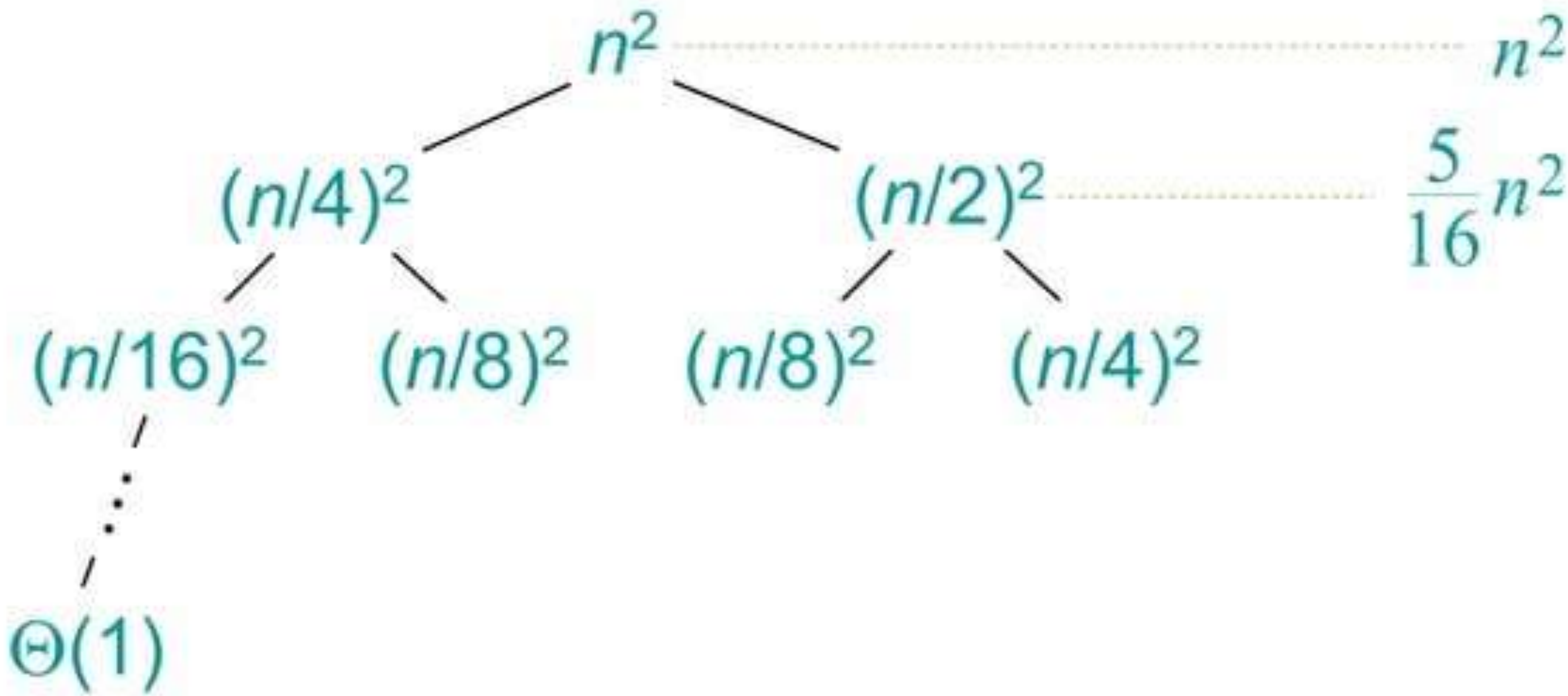
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



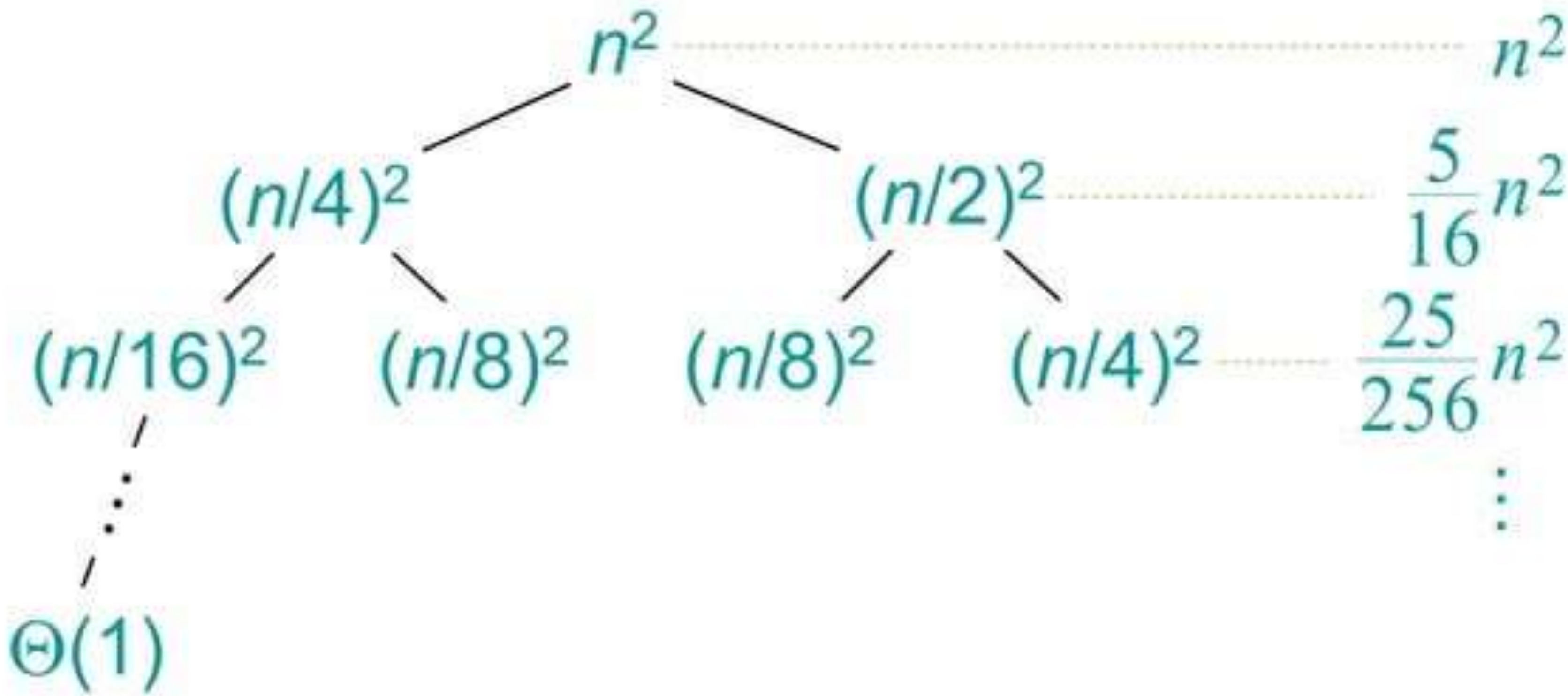
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



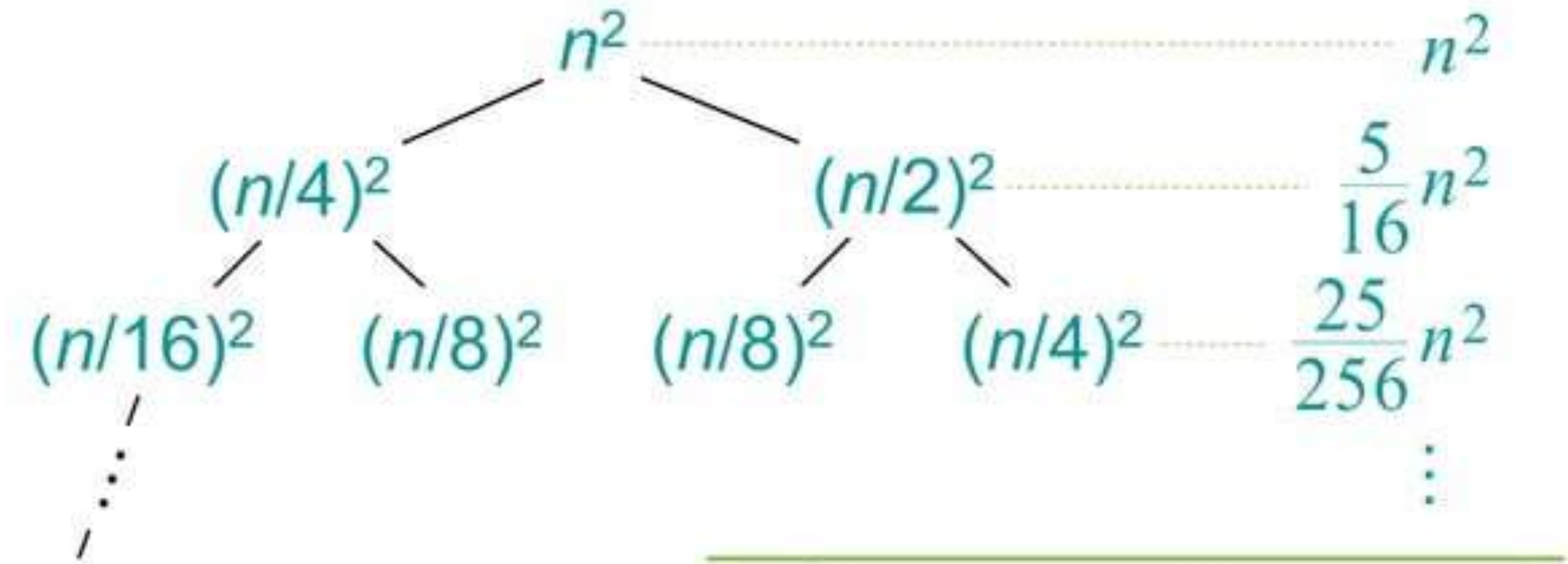
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



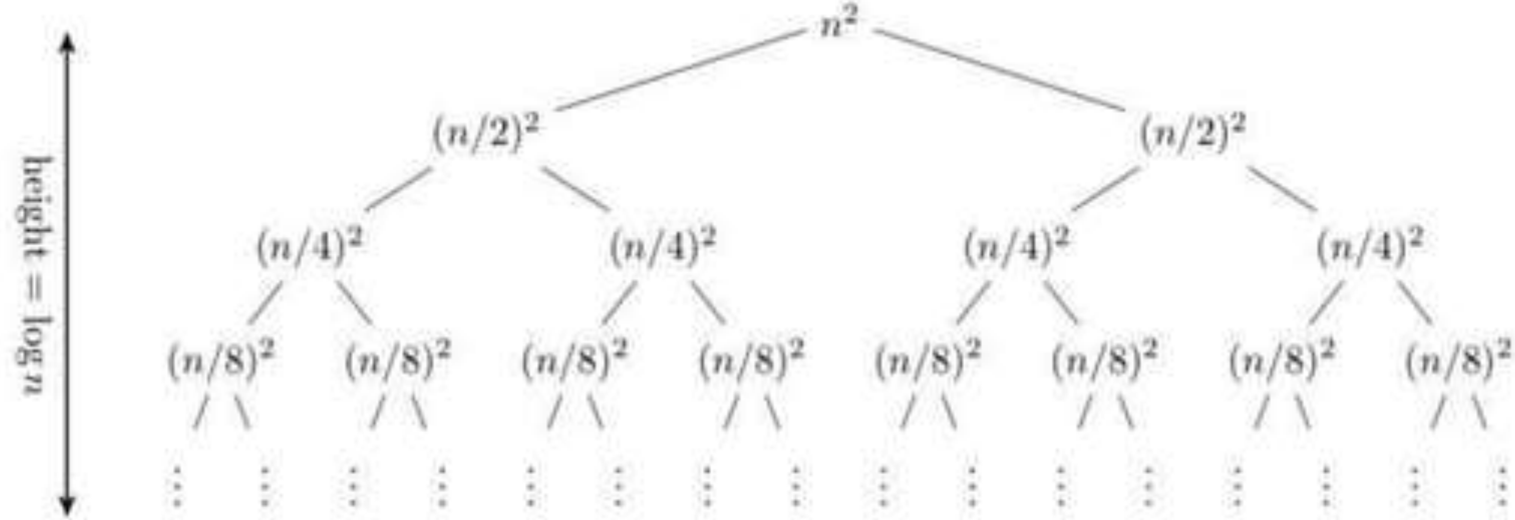
$\Theta(1)$

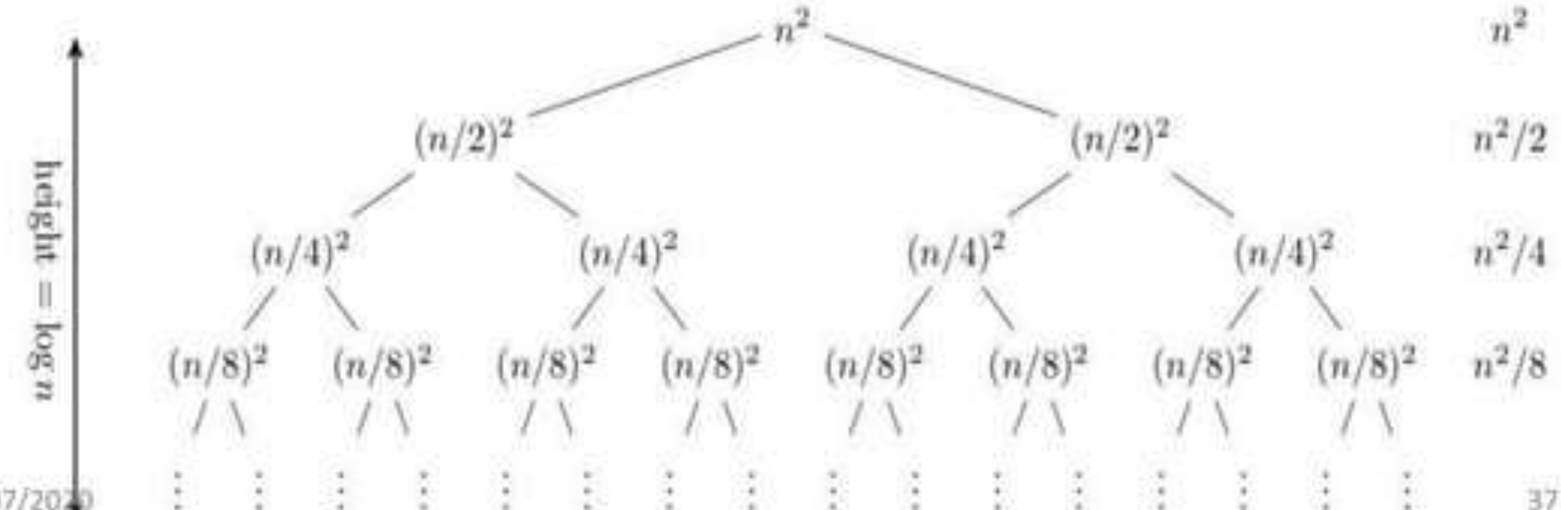
$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2)$$ *geometric series*

- **Example**

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





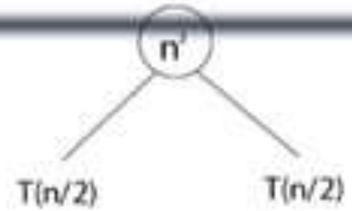
$$T(n) = n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \ldots\ldots \log n \text{ times.}$$

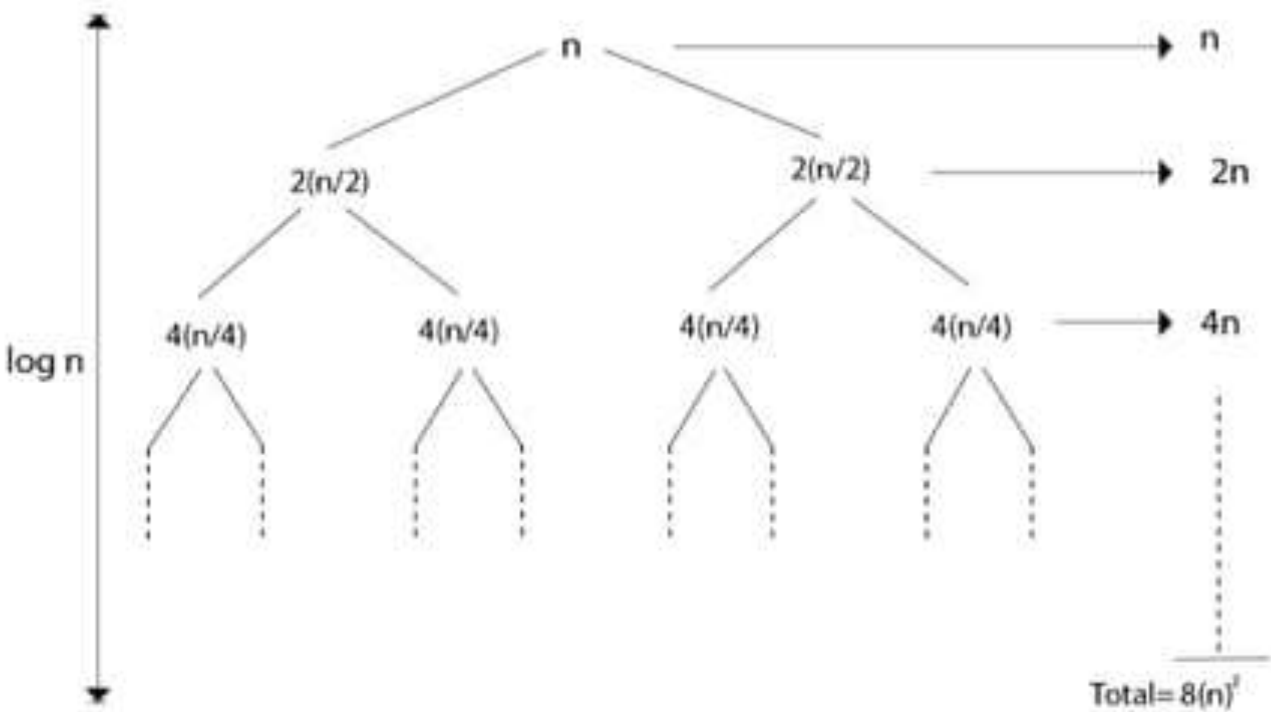
$$\leq n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2^i}\right)$$

$$\leq n^2 \left(\frac{1}{1-\frac{1}{2}}\right) \leq 2n^2$$

$$\mathbf{T\ (n) = \theta} n^2$$

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





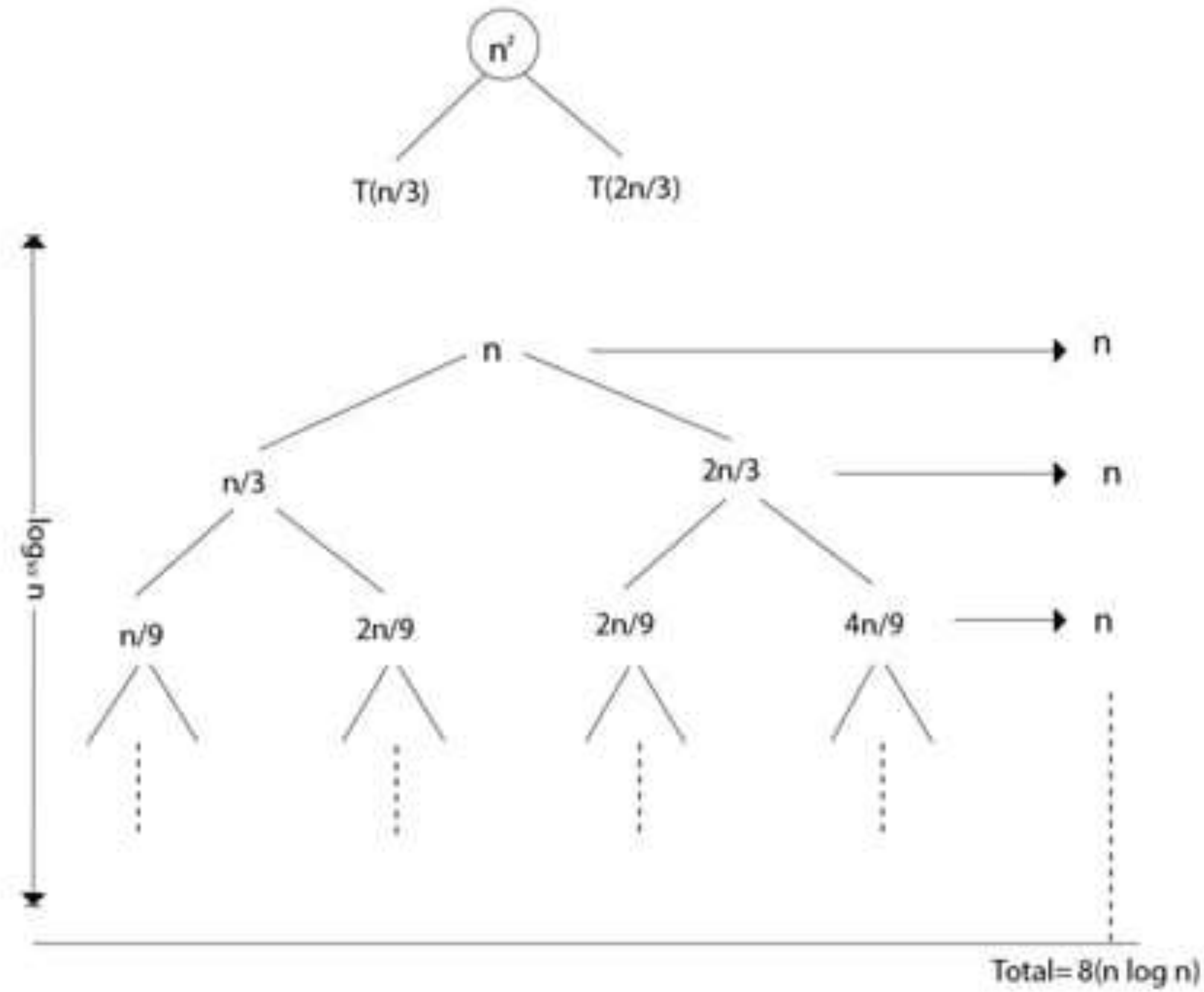
We have $n + 2n + 4n + \ldots\ldots \log_2 n$ times

$$= n\,(1 + 2 + 4 + \ldots\ldots \log_2 n \text{ times})$$

$$= n\frac{(2^{\log_2 n} - 1)}{(2-1)} = \frac{n(n-1)}{1} = n^2 - n = \theta(n^2)$$

$\mathbf{T(n) = \theta(n^2)}$

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



$$n \longrightarrow \frac{2}{3}n \longrightarrow \left(\frac{2}{3}\right) n \longrightarrow \ldots 1$$

Since $\left(\frac{2}{3}\right)$ n=1 when i=$\log_{\frac{3}{2}}$ n.

Thus the height of the tree is $\log_{\frac{3}{2}}$ n.

$$T(n) = n + n + n + \ldots\ldots + \log_{\frac{3}{2}} n \text{ times.} = \boldsymbol{\theta(n \log n)}$$

Let k th steps, it moves up to $n/3^k$

It moves until 1

The total sum up to kth step =kn

If $n/3^k$ = 1

n = 3k

k= logn

Total time taken =nlogn

The Master Method

- Based on the Master theorem.
- "Cookbook" approach for solving recurrences of the form

  $$T(n) = aT(n/b) + f(n)$$

  - $a \geq 1$, $b > 1$ are constants.
  - $f(n)$ is asymptotically positive.
  - n/b may not be an integer, but we ignore floors and ceilings.
- Requires memorization of three cases.

The Master Theorem

Theorem 4.1

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and
Let $T(n)$ be defined on nonnegative integers by the recurrence
$T(n) = aT(n/b) + f(n)$, where we can replace n/b by $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.
$T(n)$ can be bounded asymptotically in three cases:

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 $\varepsilon > 0$, and if, for some constant $c < 1$ and all sufficiently large n, we have $a \cdot f(n/b) \leq c\, f(n)$, then $T(n) = \Theta(f(n))$.

Master Method – Examples

- $T(n) = 16T(n/4)+n$
  - $a = 16$, $b = 4$, $n^{\log_b a} = n^{\log_4 16} = n^2$.
  - $f(n) = n = O(n^{\log_b a-\varepsilon}) = O(n^{2-\varepsilon})$, where $\varepsilon = 1 \Rightarrow$ **Case 1.**
  - Hence, $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$.

- $T(n) = T(3n/7) + 1$
  - $a = 1$, $b=7/3$, and $n^{\log_b a} = n^{\log_{7/3} 1} = n^0 = 1$
  - $f(n) = 1 = \Theta(n^{\log_b a}) \Rightarrow$ **Case 2.**
  - Therefore, $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(\lg n)$

Master Method – Examples

- $T(n) = 3T(n/4) + n \lg n$
  - $a = 3$, $b=4$, thus $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$
  - $f(n) = n \lg n = \Omega(n^{\log_4 3 + \varepsilon})$ where $\varepsilon \approx 0.2 \Rightarrow$ **Case 3**.
  - Therefore, $T(n) = \Theta(f(n)) = \Theta(n \lg n)$.

- $T(n) = 2T(n/2) + n \lg n$
  - $a = 2$, $b=2$, $f(n) = n \lg n$, and $n^{\log_b a} = n^{\log_2 2} = n$
  - $f(n)$ is asymptotically larger than $n^{\log_b a}$, but not polynomially larger. The ratio $\lg n$ is asymptotically less than n^{ε} for any positive ε. Thus, the Master Theorem ***doesn't*** apply here.

Master Method – Examples

- $T(n) = 9T(n/3) + n$
  - $a=9$, $b=3$, $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
  - Since $f(n) = n$, $f(n) < n^{\log_b a}$
  - **Case 1 applies:**

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a-\varepsilon}\right) \\ \Theta\left(n^{\log_b a}\log n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right) \text{AND} \\ & af(n/b) < cf(n) \text{ for large } n \end{cases} \quad \begin{matrix} \varepsilon > 0 \\ c < 1 \end{matrix}$$

$$T(n) = \Theta\left(n^{\log_b a}\right) \text{when } f(n) = O\left(n^{\log_b a-\varepsilon}\right)$$

- Thus the solution is $T(n) = \Theta(n^2)$

Problems on The Master Method

- $T(n) = 4T(n/2) + n^2$
  - $a = 4$, $b = 2$, $f(n) = n^2$
  - $n^{\log_b a} = n^2$
  - *Since,* $f(n) = n^2$
  - *Thus,* $f(n) = n^{\log_b a}$

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{AND} \\ & af(n/b) < cf(n) \text{ for large } n \end{cases} \quad \begin{matrix} \varepsilon > 0 \\ c < 1 \end{matrix}$$

- **Case 2 applies**:

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

- Thus the solution is $T(n) = \Theta(n^2 \log n)$.

Master Method – Examples

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{AND} \\ & af(n/b) < cf(n) \text{ for large } n \end{cases} \quad \begin{matrix} \varepsilon > 0 \\ c < 1 \end{matrix}$$

- ***Ex.*** $T(n) = 4T(n/2) + n^3$
  - $a = 4$, $b = 2$, $f(n) = n^3$ $n^{\log_b a}$
  - $= n^2$; $f(n) = n^3$.

Since, $f(n) = n^3$ *Thus,* $f(n) > n^{\log_b a}$

  -
  - **Case 3 applies**:

$$f(n) = \Omega(n^3)$$

  - ***and*** $4(n/2)^3 \le cn^3$ (regulatory condition) for $c = 1/2$.

$$T(n) = \Theta(n^3).$$

Master Method – Examples

1. $T(n) = 4T(n/2) + n \qquad n > 1$

   $T(n) = 1 \qquad n = 1$

   From the above recurrence relation we obtain

   $a = 4$, $b = 2$, $c = 1$, $d = 1$, $f(n) = n$

   $\log_b a = \log_2 4 = \log_2 2^2 = 2 \log_2 2 = 2$

   $n^{\log_b a} = n^2$

Master Method – Examples

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

$n = O(n^2)$ It will fall in Case 1. So that

$T(n) = \theta(n^2)$

2. $T(n) = 4T(n/2) + n^2$ $n > 1$ $T(n) = 1$ $n = 1$

From the above recurrence relation we obtain $a = 4, b = 2, c = 1, d = 1, f(n) = n^2$

$n^{\log_b a} = n^2$

$f(n) = \theta(n^2)$ $n^2 = \theta(n^2)$

It will fall in case 2.

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

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

$T(n) = 1$ $n = 1$

From the above recurrence relation we obtain

$a = 4, b = 2, c = 1, d = 1, f(n) = n^3$

$n^{\log_b a} = n^3$

$f(n) = \Omega(n^{\log_b a + \epsilon})$

Recurrence Relation :— ① Substitution ② Recursion Tree ③ Master Method

Ex: C program.

$T(n)$ —	void Test (int n)
1 —	{ if (n > 0)
1 —	{ printf ("%d", n)
$T(n-1)$ —	} Test (n-1);
	}

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

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1)+1 & n > 0 \end{cases}$$

stitution method :-

the above recurrence relation.

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

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

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

substitute

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

Continue for k times

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

Assume $n-k=0$

$\therefore n = k$

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

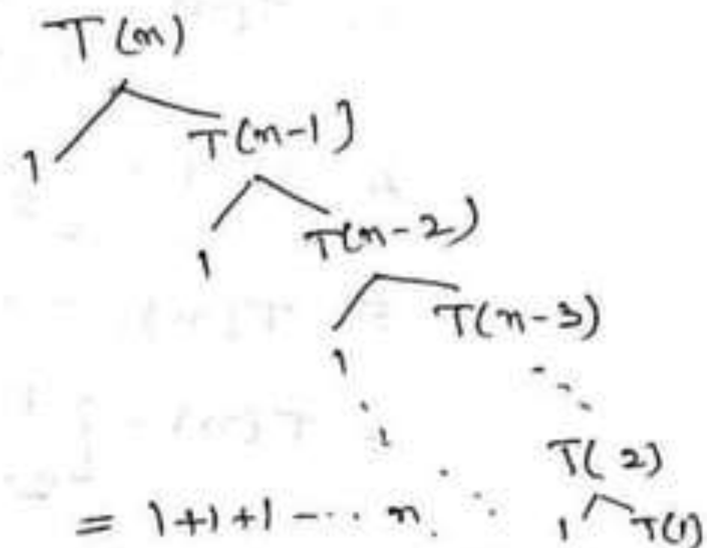
$= T(0)+n$

$T(n) = 1+n$

$= O(n)$

$\boxed{\text{Substitution method}}$.

Recursion Tree method



$= 1+1+1 \cdots n.$

$= \underline{\underline{n}} \quad O(n).$

Examples

(2) $T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + c & n > 1 \end{cases}$

Ex: $T(n)$ — void Test(int n)

{ if (n>1)

1 — { print("%d", n)

$T(\frac{n}{2})$ — } test(n/2);

}

Substitution method

$T(n) = T(\frac{n}{2}) + c$

$T(\frac{n}{2}) = T(\frac{n}{2^2}) + c$

$T(\frac{n}{4}) = T(\frac{n}{2^3}) + c$

Substitute.

$T(n) = T(\frac{n}{2^3}) + 3c$.

$k = 3 \quad \therefore 2^k = n$.

$= 1 + kc$

$= 1 + c\log_2 n \quad \therefore O(\log_2 n)$.

Recursion Tree method

$T(n)$

$1 \quad T(\frac{n}{2})$

$1 \quad T(\frac{n}{4})$

$1 \quad T(\frac{n}{2^2})$

$T(\frac{n}{2^k})$

$\therefore 2^k = n$

$= 1 + 1 + 1 + \dots k$

$= k + 1$

$= O(\log_2 n)$.

Examples

Problems on Iterative substitution
Recursion Tree method.

1. $T(n) = \begin{cases} 1 & n=0 \\ 2T(n-1)+1 & n>0 \end{cases}$

2. $T(n) = \begin{cases} 1 & n=1 \\ 2T(\frac{n}{2})+4n & n>1 \end{cases}$

3. $T(n) = \begin{cases} 2 & n=1 \\ 2T(\frac{n}{2})+7 & n>1 \end{cases}$

4. $T(n) = \begin{cases} 2 & n=1 \\ 2T(\frac{n}{2})+8 & n>1 \end{cases}$

5. $T(n) = 5T(\frac{n}{5}) + \frac{n}{\log n}$

6. $T(n) = \begin{cases} 1 & n\le 4 \\ 2T(\sqrt{n})+\log n & n>4 \end{cases}$

solution for $T(n) = \begin{cases} 1 & n\le 4 \\ 2T(\sqrt{n}) + \log n \end{cases}$

Substitution method.

$T(n) = 2T(\sqrt{n}) + \log n$

$T(n^{\frac{1}{2}}) = 2T(n^{\frac{1}{4}}) + \log\sqrt{\frac{n}{2}}$

$T(n^{\frac{1}{4}}) = 2T(n^{\frac{1}{8}}) + \frac{1}{4}\log n$

Substitute

$T(n) = 2^3 T(n^{\frac{1}{2^3}}) + \frac{1}{2^2}\log n$

$= 2^3 T(n$

Master Method – Examples

Master Method

$T(n) = a\,T(\frac{n}{b}) + f(n)$. where $a \geq 1$, $b > 1$
$f(n)$ should be (+)ve.

Cases: I if $f(n) < n^{\log_b^a}$ then $T(n) = \Theta(n^{\log_b^a})$

II If $f(n) = n^{\log_b^a}$ then $T(n) = \Theta((n^{\log_b^a}) . \log n)$

III If $f(n) > n^{\log_b^a}$ then $T(n) = \Theta(f(n))$

Example: $T(n) = 4T(\frac{n}{2}) + n$.

given $a = 4$ $b = 2$ $f(n) = n$ $\therefore n^{\log_b^a} = n^{\log_2^4} = n^2$

then $n < n^2 \Rightarrow$ case I then $T(n) = \Theta(n^2)$.

Example (i) $T(n) = 2T(\frac{n}{2}) + n$. Ans = $T(n) = \Theta(n \log n)$.

(ii) $T(n) = 8T(\frac{n}{4}) + n^2$ — case III

(iii) $T(n) = 4T(\frac{n}{2}) + n^3$ — case-III

(iv) $T(n) = 3T(\frac{2n}{6}) + n^3$.

{ sol: $T(n) = 3T(\frac{2n/2}{6/2}) + n^3$

$T(n) = 3T(\frac{n}{3}) + n^3$ $a = 3$ $b = 3$ }.

(v) $T(n) = 16T(\frac{n}{8}) + n^2$

Some Common Recurrence Relation

Recurrence Relation	Complexity	Problem
$T(n) = T(n/2) + c$	$O(\log n)$	Binary Search
$T(n) = 2T(n-1) + c$	$O(2^n)$	Tower of Hanoi
$T(n) = T(n-1) + c$	$O(n)$	Linear Search
$T(n) = 2T(n/2) + n$	$O(n\log n)$	Merge Sort
$T(n) = T(n-1) + n$	$O(n^2)$	Selection Sort, Insertion Sort
$T(n) = T(n-1)+T(n-2) + c$	$O(2^n)$	Fibonacci Series