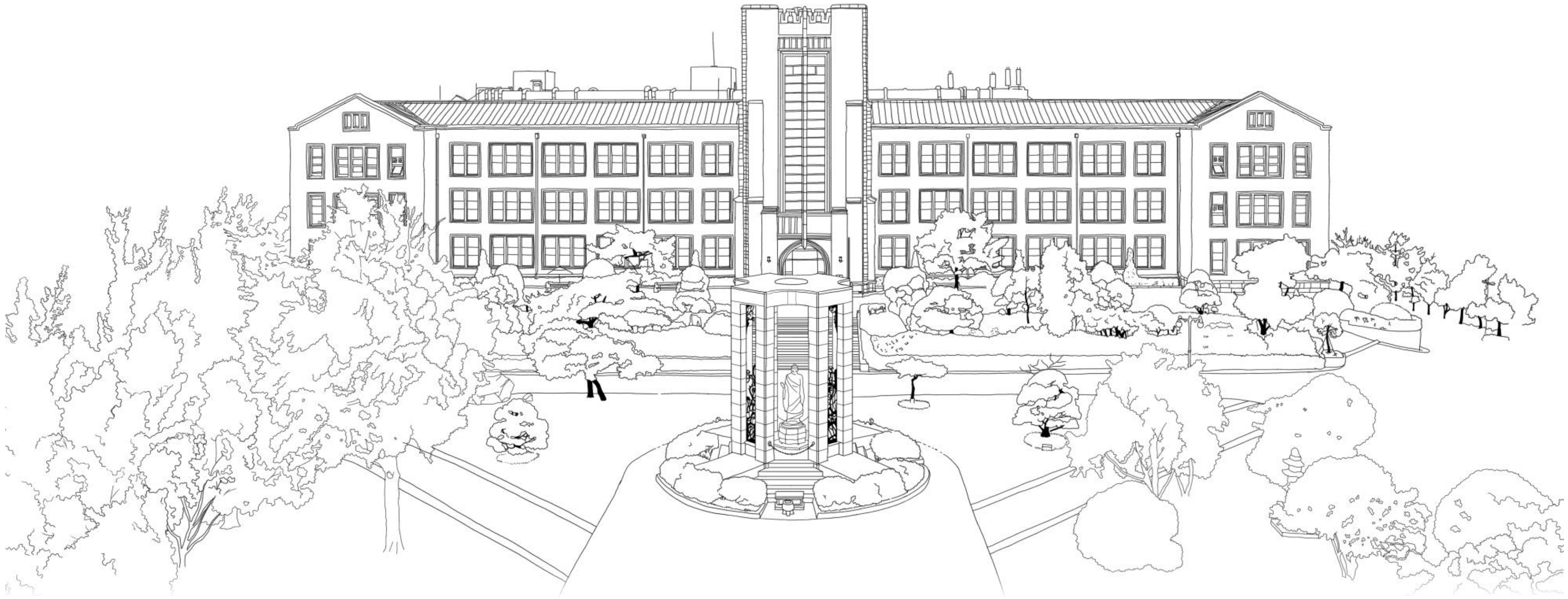


# Introduction to Algorithms

DongYeon Kim
Department of Multimedia Engineering
Dongguk University

BigDataLab
Dept. of Multimedia Engineering at Dongguk University
dongguk UNIVERSITY

# Contents



## ❖ Chapter.1 Foundations

# Divide-and-Conquer

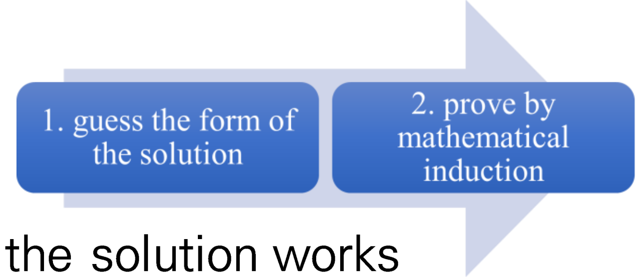


## ❖ The Introduction for solving recurrences

- Introduction
  - ➢ Now that we have seen how recurrences characterize the running times of divide-and-conquer algorithms
    - ✓ Merge sort, Maximum-subarray problem, Strassen's algorithm
    - ✓ Ex) $$\begin{aligned} T(n) &= \Theta(1) + 2T(n/2) + \Theta(n) + \Theta(1) \\ &= 2T(n/2) + \Theta(n) . \end{aligned}$$
  - ➢ From this chapter, learn how to solve recurrences
    - ✓ Substitution method
    - ✓ Recursion-tree method
    - ✓ Master method ★
    - ✓ Ex) $$\begin{aligned} T(n) &= \Theta(1) + 2T(n/2) + \Theta(n) + \Theta(1) \\ &= 2T(n/2) + \Theta(n) . \end{aligned} \Longrightarrow T(n) = \theta(n \log n)$$

## ❖ The substitution method for solving recurrences

- Concept
  - ➢ When we solve the recurrence function, substitute the guessed solution for the function and check if the guessed solution is true or not
  - ➢ The substitution method comprises two steps
    - ✓ Guess the form of the solution
    - ✓ Use mathematical induction to find the constants and show that the solution works



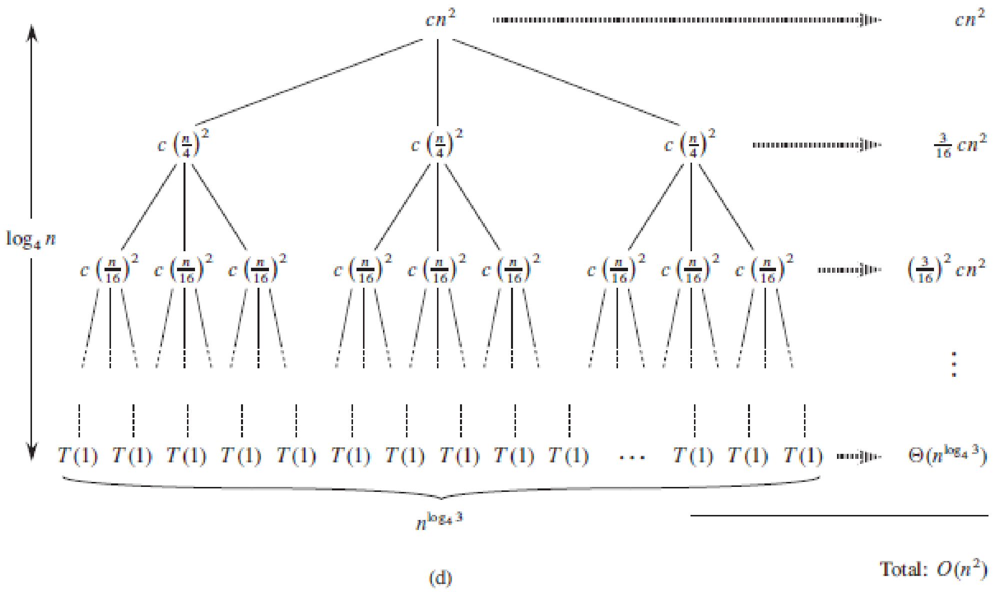

- Property
  - ➢ This method is powerful, but we must be able to guess the form of the answer
  - ➢ We can use this method to establish either upper or lower bounds on a recurrence

## ❖ The substitution method for solving recurrences

- Example
  - Determine the upper bound of the recurrence $T(n) = 2T(\lfloor n/2 \rfloor) + n$
    - ✓ Step 1 : guess the solution
      i. $T(n) = O(n \lg n)$
    - ✓ Step 2 : prove through mathematical induction
      i. Require us to prove that $T(n) \le cn \lg n$ for an appropriate choice of the constant $c > 0$
      ii. Assume that the bound holds for a!l positive $m < n, \left(m = \frac{n}{2}\right)$
      iii. $T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$
      iv.
$$\begin{aligned} T(n) &\le 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\ &\le cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\le cn \lg n \,, \end{aligned}$$
      v. The last step holds as long as $c \ge 1$
      vi. Require us to show that the solution holds for the boundary condition

## ❖ The recursion-tree method for solving recurrences

- Concept
  - ➢ We sum all nodes to determine the total cost of the recursion
  - ➢ Recursion tree is used to generate a good guess, which we can verify by the substitution method




- ✓ Each node represents the cost of a single subproblem
- ✓ Sum the costs within each level of the tree
- ✓ Sum all the per-level costs to determine the total cost of all levels of the recursion

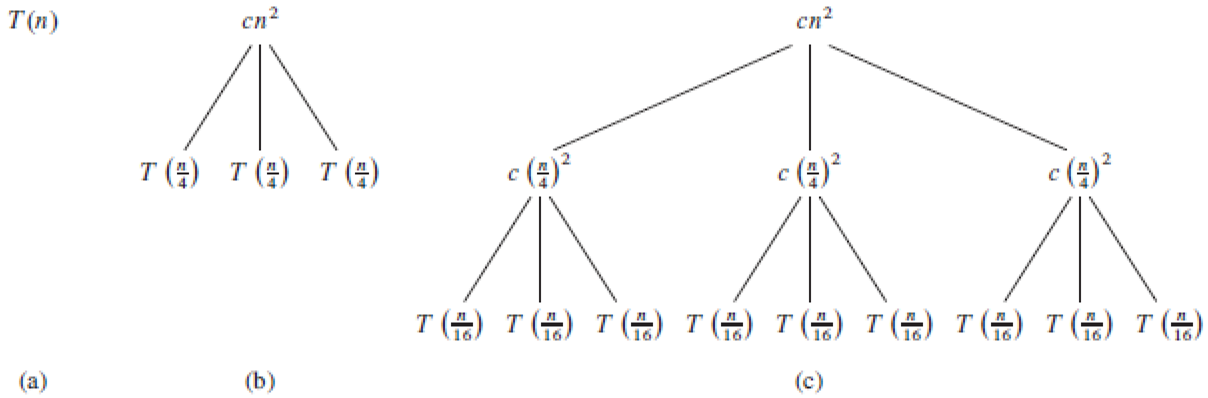
## ❖ The recursion-tree method for solving recurrences

- Concept (cont'd)
  - We can often tolerate a small amount of "slopness"
    - ✓ Because we can verify a good guess by the substitution method later
    - ✓ $T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2) \quad \rightarrow \quad T(n) = 3T(n/4) + cn^2$

- Make a good guess for the recurrence (example)
  - $\boldsymbol{T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)}$
    - ✓ $T(n) = 3T(n/4) + cn^2$ (sloppiness that we can tolerate)
    - ✓ Assume that n is an exact power of 4
      1. Another example of tolerable sloppiness
      2. All subproblem sizes are integers

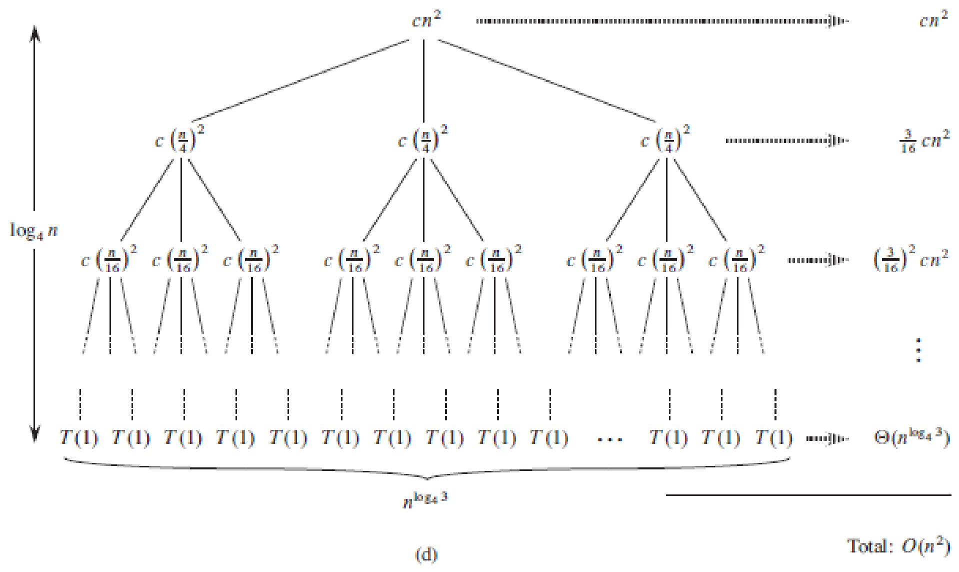
## ❖ The recursion–tree method for solving recurrences

- **Make a good guess for the recurrence (cont'd)**
  - ➢ $\boldsymbol{T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)}$
    - ✓ Part (a)
      - i. initial state of the recurrence
    - ✓ Part (b)
      - i. The $cn^2$ term at the root represents the cost at the top level of recursion
      - ii. The three subtrees of the root represent the costs incurred by the subproblems of size $n/4$
    - ✓ Part (c)
      - i. The cost for each of the three children of the root is $c(\frac{n}{4})^2$
      - ii. The three subtrees of the root represent the costs incurred by the subproblems of size $n/16$



## ❖ The recursion-tree method for solving recurrences

- **Make a good guess for the recurrence (cont'd)**
  - ➢ $\boldsymbol{T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)}$




- ✓ Subproblem size for a node at depth $i$ : $\frac{n}{4^i}$
- ✓ The tree has $\log_4 n + 1$ levels
  - i. Bottom node size $= \frac{n}{4^i} = 1$
  - ii. $i = \log_4 n$
- ✓ The number of nodes at depth $i = 3^i$
- ✓ The cost of each node at level $i = c(\frac{n}{4^i})^2$
- ✓ The total cost of level $i = 3^i c(\frac{n}{4^i})^2 = cn^2(\frac{3}{16})^i$

- ✓ The bottom level has $3^{\log_4 n} = n^{\log_4 3}$ nodes, each contributing cost $T(1)$ is a constant
- ✓ A total cost of bottom level $= n^{\log_4 3} T(1) = \theta(n^{\log_4 3})$

# Divide–and–Conquer



## ❖ The recursion–tree method for solving recurrences

- **Make a good guess for the recurrence (cont'd)**
  - ➢ $\boldsymbol{T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)}$
    - ✓ Add up the costs over all levels to determine the cost for the entire tree

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3}) \qquad \text{(by equation (A.5))} .
\end{aligned}
$$

Root node cost

Last level total cost

Sum of geometric sequence

total cost of each level(1~ $\log_4 n - 1$)

    - ✓ This formula looks somewhat messy → we can take advantage of small amounts of sloppiness
    - ✓ Use an infinite decreasing geometric series as an upper bound

## ❖ The recursion-tree method for solving recurrences

- **Make a good guess for the recurrence (cont'd)**
  - ➢ $\boldsymbol{T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)}$
    - ✓ Take advantage of small amounts of sloppiness & use infinite decreasing geometric series

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) .
\end{aligned}
$$

i. Line 1~2 : use infinite decreasing geometric series as an upper bound

ii. Line 4~5 : take advantage of small amounts of sloppiness

- ➢ We have derived a guess of $\boldsymbol{T(n) = O(n^2)}$ for our original recurrence $\boldsymbol{T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)}$
- ➢ Now we can use the substitution method to verify that our guess was correct

## ❖ The recursion-tree method for solving recurrences

- **Make a good guess for the recurrence (cont'd)**

➢ $\boldsymbol{T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)}$

✓ Use the substitution method to verify our guess

✓ $T(n) = O(n^2)$ is an upper bound for the recurrence $T(n) = 3T(\lfloor n/4 \rfloor) + \theta(n^2)$

✓ We want to show that $T(n) \le \mathrm{d}n^2$ for some constant $\mathrm{d} > 0$

$$\begin{aligned} T(n) &\le 3T(\lfloor n/4 \rfloor) + cn^2 \\ &\le 3d\lfloor n/4 \rfloor^2 + cn^2 \\ &\le 3d(n/4)^2 + cn^2 \\ &= \frac{3}{16}dn^2 + cn^2 \\ &\le dn^2 , \end{aligned}$$

i. Line 1~2 : $T(n) = O(n^2)$ is an upper bound

ii. Line 2~3 : Floor function means upper bound

iii. Line 4~5 : holds as long as $d \ge \left(\frac{16}{13}\right) c$

✓ So, there exists some constant $\mathrm{d} > 0$, our guess is correct!!

## ❖ The master method for solving recurrences

- Introduction
  - ➢ The master method provides a "cookbook" method for solving recurrences of the form $\boldsymbol{T(n) = aT(n/b) + f(n)}$
    - ✓ Where $a \geq 1$ $and$ $b > 1$ are constants
    - ✓ $f(n)$ is an asymptotically positive function
  - ➢ The recurrence describes the running time of algorithms
    - ✓ Divide the problem of size $n$ into $a$ subproblems
    - ✓ Each of size $n/b$
    - ✓ $a$ subproblems are solved recursively, each in time $T(n/b)$
    - ✓ $f(n)$ : divide and combine steps in divide and conquer algorithm
    - ✓ $T(\lfloor n/b \rfloor)$ or $T(\lceil n/b \rceil)$ will not affect the asymptotic behavior of the recurrence, so replace the terms $T(n/b)$

# ❖ The master method for solving recurrences

- The master theorem
  - To use the master method, you will need to memorize three cases

*Theorem 4.1 (Master theorem)*
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret $n/b$ to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, 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 $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$, then $T(n) = \Theta(f(n))$. ∎

- ✓ compare the function $f(n)$ with the function $n^{\log_b a}$
- ✓ The larger of the two functions determines the solution to the recurrence
- ✓ Case 1
  - i. The function $n^{\log_b a}$ is the larger
  - ii. The solution is $T(n) = \theta(n^{\log_b a})$
  - iii. The function $f(n)$ must be *polynomially* smaller → $f(n)$ must be asymptotically smaller than $n^{\log_b a}$ by a factor of $n^{\epsilon}$
- ✓ Case 2
  - i. The two functions are the same size
  - ii. We multiply by a logarithmic factor
  - iii. The solution is $T(n) = \theta\left(n^{\log_b a} \lg n\right) = \theta(f(n) \lg n)$

## ❖ The master method for solving recurrences

- The master theorem (cont'd)
  - ➢ To use the master method, you will need to memorize three cases

*Theorem 4.1 (Master theorem)*
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret $n/b$ to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, 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 $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$, then $T(n) = \Theta(f(n))$. ∎

✓ Case 3
- i. The function $f(n)$ is the larger
- ii. The solution is $T(n) = \theta(f(n))$
- iii. The function $f(n)$ must be *polynomially* larger → $f(n)$ must be asymptotically larger than $n^{\log_b a}$ by a factor of $n^{\epsilon}$
- iv. Must satisfy the "regularity" condition that $af(n/b) \leq cf(n)$

✓ Three cases do not cover all the possibilities for $f(n)$
✓ 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
✓ The regularity condition in case 3 fails to hold
→ Cannot use the master method to solve the recurrence!!!

## ❖ The master method for solving recurrences

- Using the master method
  - ➢ To use the master method, we simply determine which case of the master theorem applies and write down the answer
  - ➢ $\boldsymbol{T(n) = 9T\left(\frac{n}{3}\right) + n}$
    - ✓ $a = 9, b = 3, f(n) = n$
    - ✓ $f(n) = O(n^{\log_3 9-\epsilon})$, where $\epsilon = 1$
    - ✓ $n^{\log_b a} = n^{\log_3 9} = \theta(n^2)$
    - ✓ We can apply case 1 of the mater theorem and conclude that the solution is $T(n) = \theta(n^2)$
  - ➢ $\boldsymbol{T(n) = T\left(\frac{2n}{3}\right) + 1}$
    - ✓ $a = 1, b = \frac{3}{2}, f(n) = 1$
    - ✓ $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
    - ✓ $f(n) = \theta\left(n^{\log_b a}\right) = \theta(1)$
    - ✓ We can apply case 2 of the mater theorem and conclude that the solution is $T(n) = \theta(\lg n)$

## ❖ The master method for solving recurrences

- Using the master method (cont'd)
  - $\boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + n\,lg\,n}$
    - ✓ $a = 2, b = 2, f(n) = n\,lg\,n$
    - ✓ $n^{log_b a} = n^{log_2 2} = n$
    - ✓ We might mistakenly think that case 3 should apply, since $f(n)$ is asymptotically larger than $n$
    - ✓ The problem is that it is not *polynomially* larger
      1. The ratio $f(n)/n^{log_b a} = n\,lg\,n/n = lg\,n$
      2. It is asymptotically less than $n^{\epsilon}$ for any positive constant $\epsilon$
    - ✓ The recurrence falls into the gap between case 2 and case 3
  - $\boldsymbol{T(n) = 7T(n/2) + \theta(n^2)}$ **(Strassen's algorithm)**
    - ✓ $a = 7, b = 2, f(n) = \theta(n^2)$
    - ✓ $n^{\log_b a} = n^{\log_2 7} \rightarrow 2.80 < lg\,7 < 2.81$
    - ✓ $f(n) = O\left(n^{lg\,7-\epsilon}\right)$ for $\epsilon = 0.8$
    - ✓ We can apply case 1 of the mater theorem and conclude that the solution is $T(n) = \theta(n^{\lg_2 7})$

# Thank You!

## ❖ The substitution method for solving recurrences

- Example (cont'd)
  - ➢ Show the solution holds for base case
    - ✓ For $n = 1$
      1. $T(n) \leq cn \lg n \rightarrow T(1) \leq c \lg 1 = 0$
      2. It is odds with $T(1) = 1 \leq 0$
      3. The base case of our inductive proof fails to hold.
    - ✓ we can overcome this obstacle by only proving $T(n) \leq cn \lg n$ for $n \geq n_0$ where $n_0$ is a constant that we get to choose
      1. Induction proves the statement for $n \geq 2,\ n_0 = 2$
      2. Base cases for induction proof $n = 2\ and\ n = 3$

| base case | recurrence relation | induction proof [ $T(n) \leq cn \lg n$ ] |
|---|---|---|
| $n = 2$ | $T(2) = 2T(1) + 2 = 4$ | $4 \leq (2)(2)\lg(2)$ |
| $n = 3$ | $T(3) = 2T(1) + 3 = 5$ | $5 \leq (2)(3)\lg(3)$ |

$\}\ c = 2$