Learn DAA : From B K Sharma

TCS-503: Design and Analysis of Algorithms

Master's Theorem

Unit I: Syllabus

- Introduction:
  - Algorithms
  - Analysis of Algorithms
  - Growth of Functions
  - Master's Theorem
  - Designing of Algorithms

Unit I: Syllabus

- Sorting and Order Statistics
  - Heap Sort
  - Quick Sort
  - Sorting in Linear Time
    - Counting Sort
    - Bucket Sort
    - Radix Sort
  - Medians and Order Statistics

Learn DAA : From B K Sharma

Sorting Algorithms

Bubble Sort

Design approach: incremental

Sorts in place: Yes

Running time: $\Theta(n^2)$

Selection sort

Design approach: incremental

Sorts in place: Yes

Running time: $\Theta(n^2)$

Insertion sort

Design approach: incremental

Sorts in place: Yes

Running time: $\Theta(n^2)$

Merge Sort

Design approach: Divide and Conquer

Sorts in place: No

Running time: Let's See!!!

What is Recurrences?

A recurrence is

an equation

or

inequality

that describes a function

in terms of itself

by using smaller inputs.

What is Recurrences?

A recurrence
can easily describe
the runtime of recursive algorithms.

Linear Search

$$T(n) = \begin{cases} 1 & \text{for } n \leq 1 \\ T(n-1) + 1 & \text{otherwise} \end{cases}$$

Binary Search

$$T(n) = \begin{cases} 1 & \text{for } n \leq 1 \\ T(n/2) + 1 & \text{otherwise} \end{cases}$$

Merge -sort

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

Learn DAA : From B K Sharma

How to Solve Recurrences?

Four methods for solving recurrences:

Iteration method

Substitution method

Recursion Tree Method

and

Master method

Given:

a *divide and conquer* algorithm:

an algorithm that **divides**

the problem of size *n*

into *a* sub-problems,

each of size *n/b*

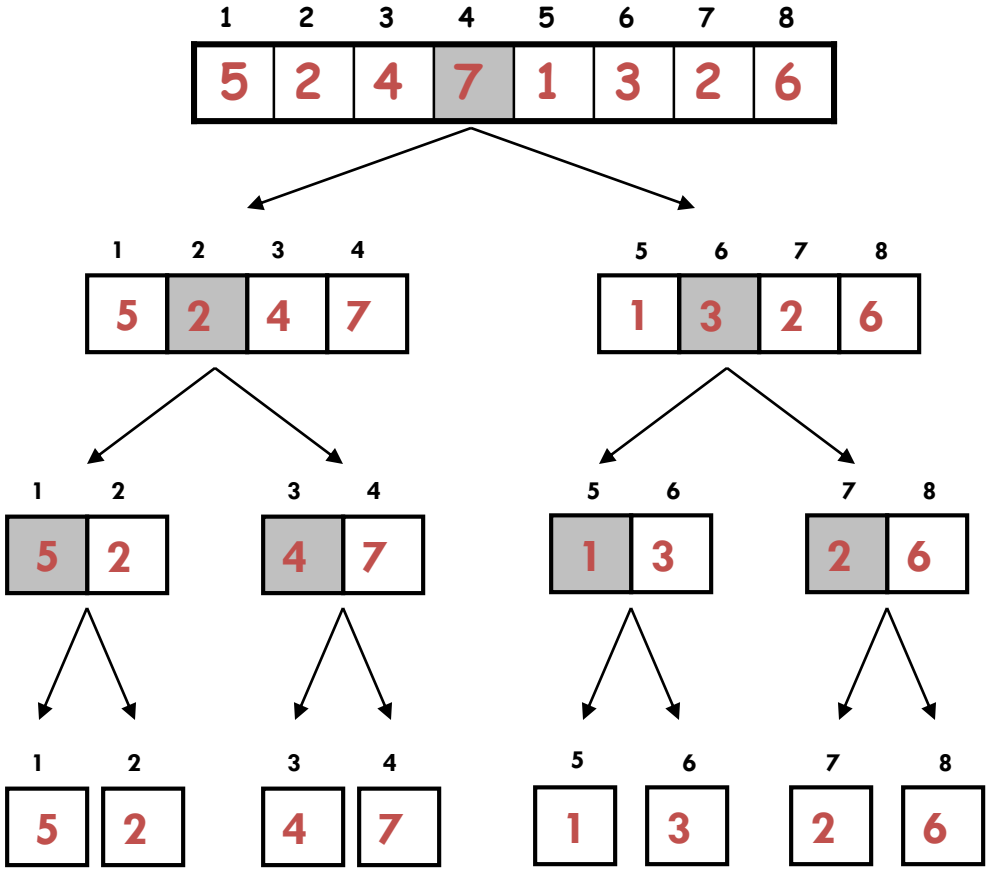
Conquers: Solves the sub-problems recursively

and

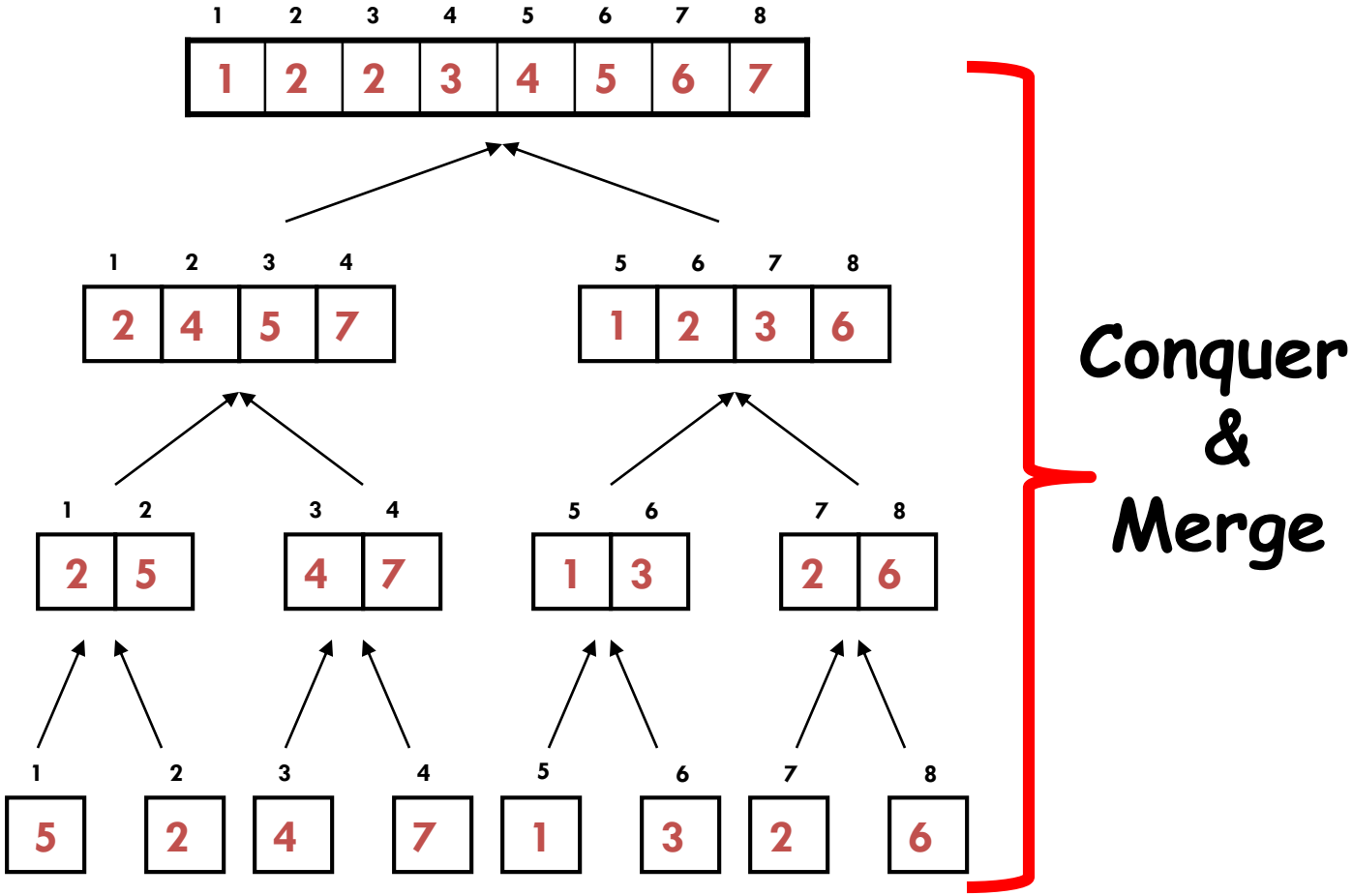
Combine or Merge sub-Problems

Example – Merge sort

Divide



Example – Merge sort



Example – Merge sort

Divide: $q=(p+r)/2$

Division of n elements take constant time.

It does not depends on n.

Conquer: $T(n/2)$ + $T(n/2)$

Combine: Left half subarray n_1 + Right Half Subarray n_2

$n_1 + n_2 = cn$

Learn DAA : From B K Sharma

Example – Merge sort

T(n) = Time to Conquer + Time to Divide + Time to Combine

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

= 2T(n/2) + cn When n > 1

Merge -sort

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

Learn DAA : From B K Sharma

The Master Theorem / Method

gives us

a cookbook method

for solving

running time of

Divide and Conquer Type of Algorithms

if

T(n) has the form

$$T(n) = a\,T(n/b) + f(n)$$

Learn DAA : From B K Sharma

The Master Theorem / Method

$$T(n) = a\,T(n/b) + f(n)$$

Where,

$a \geq 1$ & $b > 1$

n is the size of the problem.

'a' is the number of sub-problems in the recursion

n/b is the size of each sub-problem. (Here it is assumed that all sub-problems are essentially the same size.)

f (n) is the cost of the work done outside the recursive calls, which includes the cost of dividing the problem and the cost of merging the solutions to the sub-problems

f(n) is the work to divide D(n) and combine C(n)

The Master Theorem / Method

$T(n) = a\,T(n/b) + f(n) \quad a \geq 1 \;\&\; b > 1$

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

Understanding the Master Theorem / Method

Compare f(n) with $n^{\log_b a}$

The solution to the recurrence is determined by the larger of the two functions.

Case 1: If f(n) is smaller than $n^{\log_b a}$ then

the solution $T(n) = \Theta(n^{\log_b a})$.

Case 2: If f(n) and $n^{\log_b a}$ are of same size then

the solution is $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n)$.

Case 3: If f(n) is the larger than $n^{\log_b a}$, and *a f(n/b) ≤ c f(n) for large n* then

the solution is $T(n) = \Theta(f(n))$.

Understanding the Master Theorem / Method

$$T(n) = a\,T(n/b) + f(n) \quad a \geq 1 \;\&\; b > 1$$

For example

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

$$T(n) = 9T(n/3) + n$$

$$T(n) = 16T(n/4) + n$$

$$T(n) = T(3n/7) + 1$$

$$T(n) = 3T(n/4) + n \lg n$$

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

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

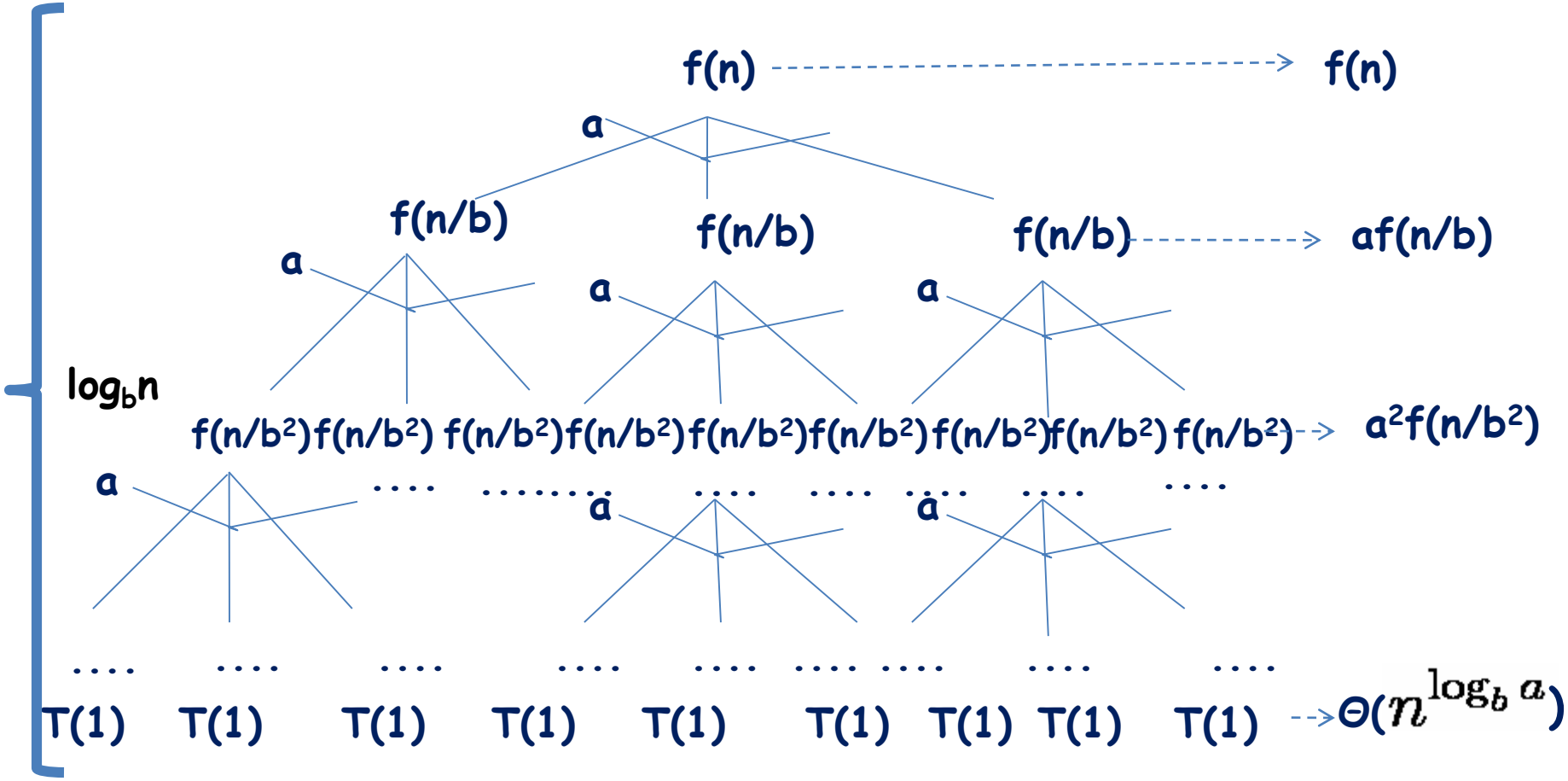
$$T(n) = 4T(n/2) + n^2$$

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

Learn DAA : From B K Sharma

Understanding the Master Theorem / Method

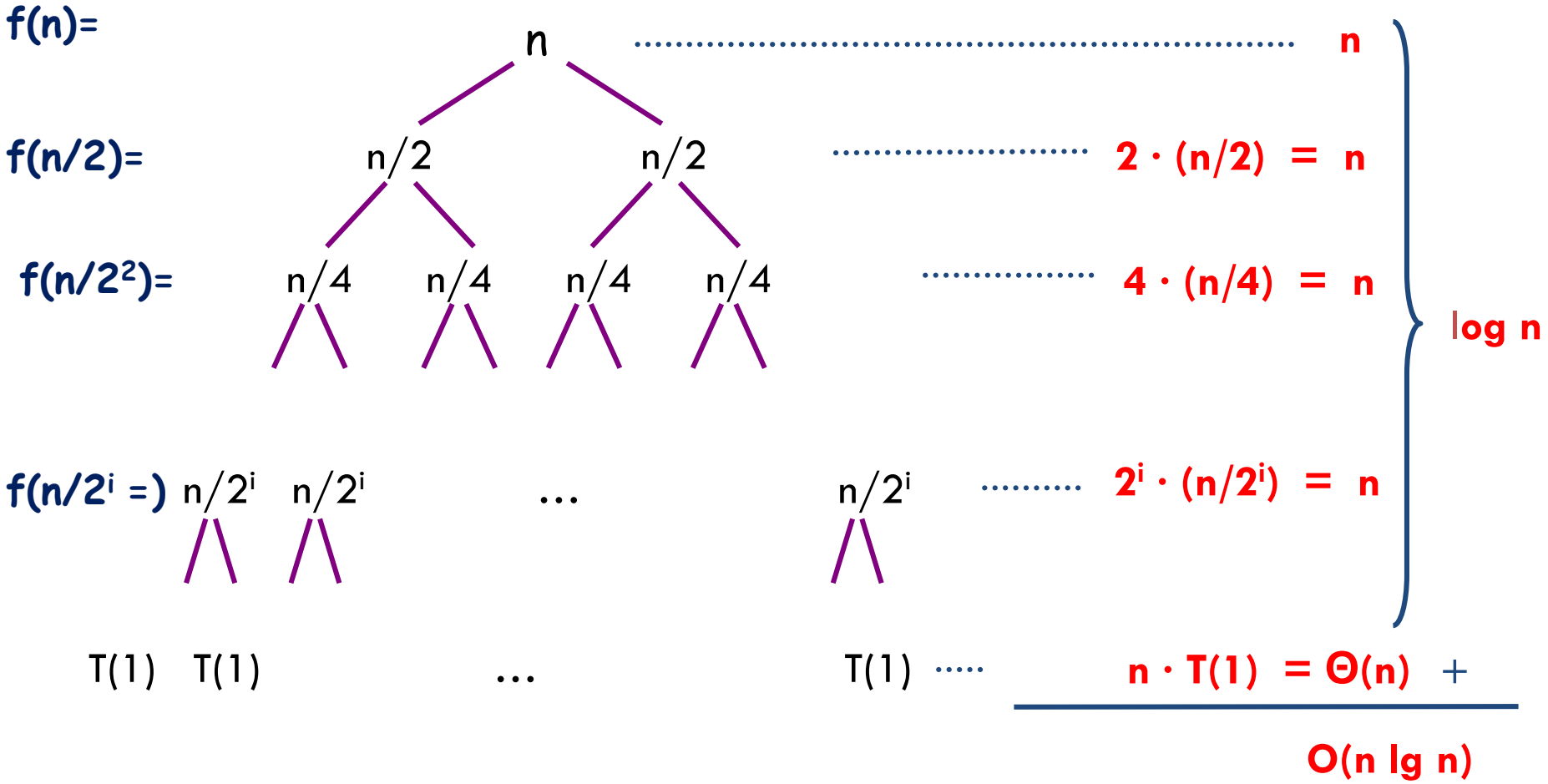
$$T(n)=aT(n/b)+f(n)$$



Learn DAA : From B K Sharma

Understanding the Master Theorem / Method

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



The Master Theorem / Method

$$T(n) = a\,T(n/b) + f(n)$$

The time T(n) might be dominated by:

The cost of the divide/combine of the root

Evenly distributed at all the levels

The cost of the leaves

The master method tells us what the asymptotic running time will be depending on which cost is the highest (dominates).

The Master Theorem / Method

$$T(n) = a\,T(n/b) + f(n)$$

Then based on comparing $f(n)$ and $n^{\log_b a}$ we know, the running time given the following three cases:

- If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$ then $T(n) = \Theta(n^{\log_b a})$ dominates.
- If $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \lg n)$; cost is evenly distributed
- If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ and if $af(n/b) \le cf(n)$ for some constant c < 1 and all sufficiently large n, T(n)= Θ(f(n))

Learn DAA : From B K Sharma

Example 1 : Case 1

$T(n) = 9T(n/3) + n$

Given: $a=9,\ b=3,\ f(n) = n$

Calculate: $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

Compare: $f(n)$ and $n^{\log_b a}$

$$n < n^2$$

$$f(n) < n^{\log_b a}$$

Case 1 applies:

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$$

Example 1 : Case 2

$T(n) = T(3n/7) + 1$

Given: $a = 1,\ b=7/3,\ f(n)=1$

Calculate: $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_{7/3} 1} = n^0 = 1$$

Compare: $f(n)$ and $n^{\log_b a}$

$$1=1$$

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

Case 2 applies:

$$T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n) = \Theta(\lg n)$$

Learn DAA : From B K Sharma

Example 1 : Case 3

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

Given: $a = 2,\ b = 2,\ f(n) = n^2$

Calculate: $n^{\log_b a}$

$n^{\log_b a} = n^{\log_2 2} = n$

Compare: $f(n)$ and $n^{\log_b a}$

$n^2 > n$

$f(n) > n^{\log_b a}$

$2\ f(n/2) = 2.\ (n/2)^2$
$= 2.(n^2/4)$
$= n^2/2$
$= (1/2)\ n^2$
$= (1/2) f(n)$
$\leq c.\ f(n)$
for $c = 1/2$

Now , verify regularity condition:

$a\ f(n/b) \leq c\ f(n)$

Example 1 : Case 3

Now,

$f(n) > n^{\log_b a}$ And Regularity condition

$a\ f(n/b) \leq c\ f(n)$ holds

Case 3 applies:

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

Example 2 :

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

Which case of the master theorem applies?

What is the answer?

Example 2 :

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

Which case of the master theorem applies?

Given: $a = 4,\ b = 2,\ f(n) = n$

Calculate: $n^{\log_b a}$

$n^{\log_b a} = n^{\log_2 4} = n^2$

Compare: $f(n)$ and $n^{\log_b a}$

$n < n^2$

$f(n) < n^{\log_b a}$

Case 1 applies.

What is the answer?

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$

Example 3 :

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

Which case of the master theorem applies?

What is the answer?

Example 3 :

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

Which case of the master theorem applies?

Given: $a = 4,\ b = 2,\ f(n) = n^2$

Calculate: $n^{\log_b a}$

$n^{\log_b a} = n^{\log_2 4} = n^2$

Compare: $f(n)$ and $n^{\log_b a}$

$n^2 = n^2$

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

Case 2 applies:

What is the answer?

$T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n) = \Theta(n^2 \lg n)$

Example 4 :

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

Which case of the master theorem applies?

What is the answer?

Example 4 :

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

Which case of the master theorem applies?

Given: $a = 4,\ b = 2,\ f(n) = n^3$

Calculate: $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

Compare: $f(n)$ and $n^{\log_b a}$

$$n^3 > n^2$$

$$f(n) > n^{\log_b a}$$

Case 3 seems to apply:

Now,

Check the What Condition?

Example 4 :

Now, Check Regularity Condition.

What is Regularity Condition?

$a\ f(n/b) \leq c\ f(n)$ Where, $c < 1$

$4\ f(n/2) = 4\ (n/2)^3 = 4\ (n^3/8) = n^3/2 = (1/2)n^3$

For c=1/2, Regularity Condition holds

Case 3 applies:

What is the answer?

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

Example 5 :

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

Which case of the master theorem applies?

What is the answer?

Example 5 :

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

Given: a = 2, b = 2, f(n) = n

Check Case 1:

$$\text{Is } f(n) = O(n^{\log_b a - \varepsilon})\ ?$$

$$n = O(n^{\log_2 2 - \varepsilon})$$

$$n = O(n^{1-\varepsilon})$$

For any epsilon>0, n is bigger, so case 1 does not work.

Example 5 :

Check case 2:

$$f(n) = \Theta(n^{\log_b a})$$

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

Therefore:

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

Cost is evenly distributed among leaves and upper part of tree.

Learn DAA : From B K Sharma

Understanding Epsilon(ε)

$T(n) = 9T(n/3) + n$

Given: $a=9,\ b=3,\ f(n) = n$

Calculate: $n^{\log_b a}$

$n^{\log_b a} = n^{\log_3 9} = n^2$

Compare: $f(n)$ and $n^{\log_b a}$

$n < n^2$

$f(n) < n^{\log_b a}$

Case 1 works for $f(n)=O(n^{\log_b a-\varepsilon})$

We need to prove this relationship by showing that:

Learn DAA : From B K Sharma

Understanding Epsilon(ε)

$f(n)=O(n^{\log_b a-\varepsilon})$

$n=O(n^{\log_b a-\varepsilon}) \quad =O(n^{2-\varepsilon})$

If $\varepsilon=1$ then $n=O(n)$ and case 1 is satisfied.

Therefore,

$$T(n) = \Theta(n^{\log_b a})= \Theta(n^{\log_3 9})=\Theta(n^2)$$

In this case, the cost of leaves has dominated the runtime.

Learn DAA : From B K Sharma

Understanding Epsilon(ε)

T (n)=3T (n/4) + n lg n

Given: a=3, b=4, f(n) = nlgn

Is f (n) = $\Omega(n^{\log_4 3+\varepsilon})$? Let us Check:

$$n \lg n = \Omega(n^{\log_4 3+\varepsilon}) = \Omega(n^{0.79+\varepsilon})$$

YES, if ε =0.21, then $n \lg n = \Omega(n)$

Is af (n/b) ≤cf (n) ? for c<1? Let us Check:

Understanding Epsilon(ε)

3 $f(n/4) \le cf(n)$

3 $(n/4)\lg(n/4) \le cn\lg n)$

3 $(n/4)[\lg n - \lg 4] \le cn\lg n$

3 $(n/4)[\lg n - 2] \le cn\lg n$

Yes, if $c=3/4$, then $3(n/4)[\lg n - 2] \le (3/4)n\lg n$

Therefore,

$T(n) = \Theta(f(n)) = \Theta(n \lg n)$

Learn DAA : From B K Sharma

Understanding Epsilon(ε)

$T(n)=4T(n/2) + n^2/\log n$

Given: $a=4$, $b=2$, $f(n)=n^2/\log n$

$$n^{\log_b a}=n^{\log_2 4}=n^2$$

Try Case 1:

$$Is\ f(n) = O(n^{\log_b a-\varepsilon})\ ?$$

$$n^2/\log n = O(n^{\log_2 4-\varepsilon}) = O(n^{2-\varepsilon})$$

NO, for epsilon>0, f(n) is larger.

Understanding Epsilon(ε)

Try Case 2:

Is $f(n)=\Theta(n^{\log_b a})$?

$n^2/\log n = \Theta(n^{\log_2 4}) = \Theta(n^2)$

No, grows smaller than n^2.

Try Case 3:

Is $f(n)= \Omega(n^{\log_b a+\varepsilon})$?

$n^2/\log n = \Omega(n^{\log_2 4+\varepsilon}) = \Theta(n^{2+\varepsilon})$

NO, for epsilon > 0, f(n) is smaller, not bigger.

Master method does not work for this recurrence relation!

When does Master Theorem Apply?

In Case 1, $f(n)=O(n^{\log_b a-\varepsilon})$

This can be written as,

$$f(n) < n^{\log_b a} \times n^{-\varepsilon}$$

$$f(n) < n^{\log_b a} \times 1/ n^{\varepsilon}$$

$$f(n) \times n^{\varepsilon} < n^{\log_b a}$$

In Case 3, $f(n)= \Omega (n^{\log_b a+ \varepsilon})$

This can be written as,

$$f(n) > n^{\log_b a+ \varepsilon}$$

$$f(n) > n^{\log_b a} \times n^{\varepsilon}$$

$$f(n)/n^{\varepsilon} > n^{\log_b a}$$

When does Master Theorem Apply?

If f(n) is polynomially smaller than $n^{\log_b a}$, by a factor of n^{ε} for some constant $\varepsilon > 0$, then $n^{\log_b a}$ dominates, and the runtime is $T(n) = \Theta(n^{\log_b a})$.

If f(n) is instead polynomially larger than $n^{\log_b a}$, by a factor of n^{ε} for some constant $\varepsilon > 0$, then f(n) dominates, and the runtime is $T(n) = \Theta(f(n))$.

Finally, if f(n) and $n^{\log_b a}$ are asymptotically the same, then $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n)$.

When does Master Theorem not Apply?

Note that the master theorem does not provide a solution for all f.

For example:

$$T(n)=3T(n/3) + n \lg n$$

In this case, $f(n)=n\lg n$ and $n^{\log_b a}=n$.

While f is larger than n, it is larger only by a logarithmic factor; it is not the case that $f(n)=O(n^{\log_b a - \varepsilon})$ for some $\varepsilon>0$.

Therefore, the master theorem makes no claim about the solution to this recurrence.

When Master Theorem Does not Apply?

$T(n)=8T(n/2) + n^3/\log n$

In this case, $f(n)=n^3/\log n$ and $n^{\log_b a}=n^3$

f(n) is smaller than $n^{\log_b a}$ but by less than a logarithmic factor.

Therefore, the master theorem makes no claim about the solution to the recurrence.

Inadmissible equations

The following equations cannot be solved using the master theorem:

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

a is not a constant; the number of sub-problems should be fixed.

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

non-polynomial difference between f(n) and $n^{\log_b a}$

The difference between $f(n)$ and $n^{\log_b a}$ can be expressed with the ratio $\frac{f(n)}{n^{\log_b a}} = \frac{\frac{n}{\log n}}{n^{log_2 2}} = \frac{n}{n \log n} = \frac{1}{\log n}$

Inadmissible equations

It is clear that $\frac{1}{\log n} < n^{\epsilon}$ for any constant.

Therefore, the difference is not polynomial and the Master Theorem does not apply.

Inadmissible equations

3. $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

a<1 cannot have less than one sub problem

4. $T(n) = 64T\left(\frac{n}{8}\right) - n^2 \log n$

f(n) which is the combination time is not positive.

5. $T(n) = T\left(\frac{n}{2}\right) + n(2 - \cos n)$

case 3 but regularity violation.

Master Theorem: Changing Variables

$T(n) = T(\sqrt{n}) + 1$

Let $n = 2^m \Rightarrow \lg n = \lg 2^m = m$

Then

$T(2^m) = T(2^{m/2}) + 1$

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

$S(m) = S(m/2) + 1$

$f(n)=1, \quad a=1, \quad b=1$

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$\Rightarrow \quad f(n) = n^{\log_b a}$

Case 2 applies:

$S(m) = \Theta(\log m)$

Changing back from $S(m)$ to $T(n)$, we obtain

$T(n) = T(2^m)$

$= S(m)$

$= \Theta(\lg m)$

$= \Theta(\lg \lg n)$.

Simplified Master Theorem

Let $a \geq 1$ and $b > 1$ be constants and let $T(n)$ be the recurrence

$$T(n) = a\ T(n/b) + c\ n^k$$

defined for $n \geq 0$.

1. If $a > b^k$, then $T(n) = \Theta(\ n^{\log_b a}\ )$.
2. If $a = b^k$, then $T(n) = \Theta(\ n^k \lg n\ )$.
3. If $a < b^k$, then $T(n) = \Theta(\ n^k\ )$.

Learn DAA : From B K Sharma

Self Exercises

4-1 *Recurrence examples*

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 2T(n/2) + n^3$.

b. $T(n) = T(9n/10) + n$.

c. $T(n) = 16T(n/4) + n^2$.

d. $T(n) = 7T(n/3) + n^2$.

e. $T(n) = 7T(n/2) + n^2$.

f. $T(n) = 2T(n/4) + \sqrt{n}$.

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

h. $T(n) = T(\sqrt{n}) + 1$.

Learn DAA : From B K Sharma

Self Exercises

4-4 More recurrence examples

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 3T(n/2) + n \lg n$.

b. $T(n) = 5T(n/5) + n/\lg n$.

c. $T(n) = 4T(n/2) + n^2\sqrt{n}$.

d. $T(n) = 3T(n/3 + 5) + n/2$.

e. $T(n) = 2T(n/2) + n/\lg n$.

f. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.

g. $T(n) = T(n-1) + 1/n$.

h. $T(n) = T(n-1) + \lg n$.

i. $T(n) = T(n-2) + 2\lg n$.

j. $T(n) = \sqrt{n}T(\sqrt{n}) + n$.