Overview

- **Solving Recurrence relation**
  - Recursion Tree
  - **Master Method**
  - **Substitution Method**

Divide and Conquer – Recurrence

$$T(n) = \Theta(1) \qquad \text{if } n <= c$$
$$= a\,T(n/b) + D(n) + C(n) \qquad \text{otherwise}$$

- Number of subproblems – ***a***
- Each subproblem size is ***1/b*** the size of the original
- ***D(n)*** – time to divide the problem into subproblems
- ***C(n)*** – time to combine the solutions

Divide and Conquer – Recurrence

$$T(n) = \Theta(1) \qquad \textbf{if } n <= c$$
$$= a\,T(n/b) + f(n) \quad \textbf{otherwise}$$

- Number of subproblems – ***a***
- Each subproblem size is ***1/b*** the size of the original
- ***f(n)*** **:** time to divide the problem into subproblems and to combine the solutions

Solving Recurrence – Substitution Method

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

Substitute the guessed solution for the function when applying the inductive hypothesis to smaller values.

Substitution Method

$T(n) = 2\, T(\lfloor n/2 \rfloor) + n$

1. **Guess solution:** $T(n) = O(n \lg n)$
2. **Prove that** $T(n) <= cn \lg n$ **for some** $c > 0$

Substitution Method

$T(n) = 2\,T(\lfloor n/2 \rfloor) + n$

Guessed solution: $T(n) = O(n \lg n)$

Prove that $T(n) <= cn \lg n$ **for some** $c > 0$

Assume the bound hold for all positive $m < n$**,** in particular for

$m = \lfloor n/2 \rfloor$

$T(\lfloor n/2 \rfloor) <= c\,\lfloor n/2 \rfloor \lg (\lfloor n/2 \rfloor)$

Substitution Method

Assume the bound holds for all positive $m<n$, in particular for $m = \lfloor n/2 \rfloor$. ($T(\lfloor n/2 \rfloor) <= c \lfloor n/2 \rfloor \lg (\lfloor n/2 \rfloor)$)

Substituting,

$$T(n) <= 2 (c \lfloor n/2 \rfloor \lg (\lfloor n/2 \rfloor)) + n$$

$$<= c\ n \lg (n /2)) + n$$

$$= cn \lg n - cn \lg 2 + n \quad = cn \lg n - cn + n$$

For $c >=1$, $cn \lg n - cn + n <= cn \lg n$

Substitution Method

$T(n) <= cn \lg n - cn + n$

For $c >= 1$, $T(n) <= cn \lg n$

Does it hold for $n=1$, $T(1)$ **?**

Substitution Method – Boundary conditions

With T(1) = 1,

$T(n) <= cn \lg n$ yields $T(1) <= c\ 1 \lg 1 = 0$

Choosing $n=1$ as boundary condition for proof is problematic.

Choose $n=2$ or $n=3$ (the bound is to hold only for $n >= n_0$)

Substitution Method – Boundary conditions

From the recurrence , $T(2)= 4,\ T(3)=5$

Prove that

$T(2) <= c\ 2\ lg\ 2$

$T(3) <= c\ 3\ lg\ 3$

What should be value of c?

c >= 2? $n_0 = 2$?

Master Method

$$T(n) = \Theta(1) \quad \text{if } n <= c$$
$$= a\,T(n/b) + f(n) \quad \text{otherwise}$$

- **Three cases**
- **Based on *a*, *b* and *f(n)***

Master Method

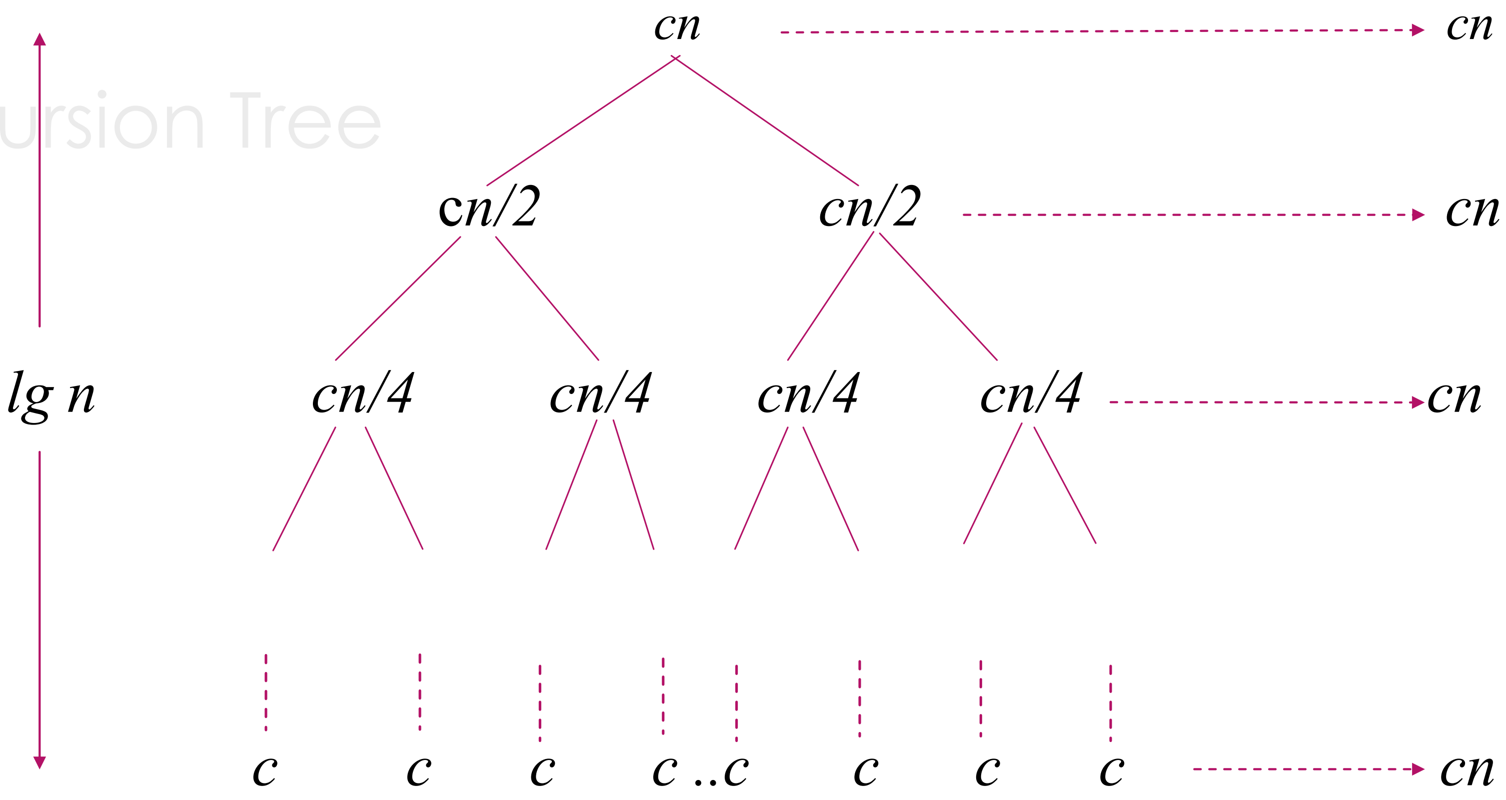
$T(n) = \Theta(1)$ if $n <= c$

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

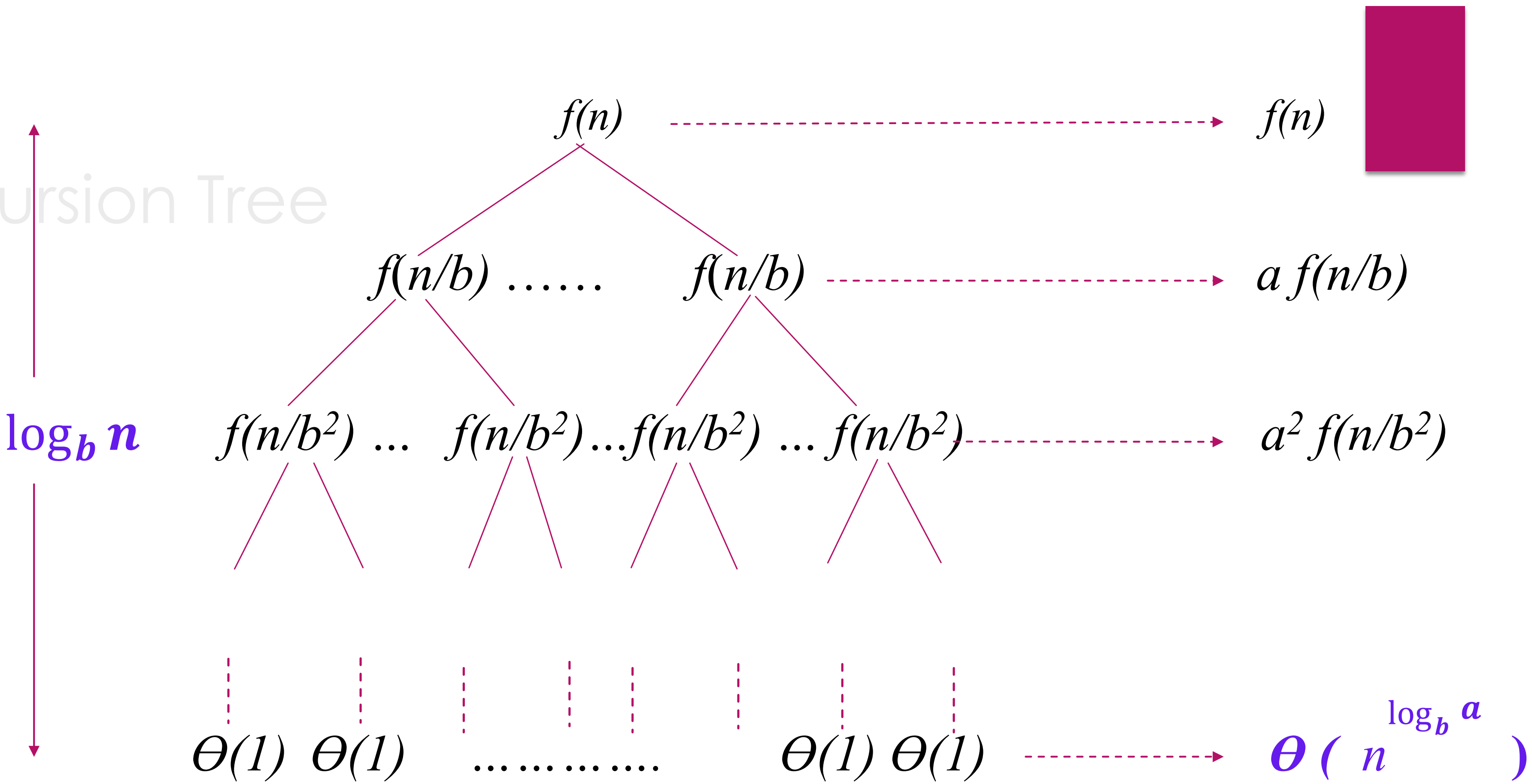
$\Theta(n^{\log_b a})$

- Three cases
- Based on a, b and $f(n)$

Recursion Tree



Recursion Tree



Total Cost = ?

Master Method

$T(n) = \Theta(1)$ if $n <= c$

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

Recursion tree height : $\log_b n$

Number of leaves : $n^{\log_b a}$

Total cost at leaf level = $\Theta(\ n^{\log_b a}\ )$

Master Theorem

Let a>=1 and b>1 be constants, let f(n) be a function and 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 $\lfloor n/b \rfloor$ ***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 $a f(n/b) \leq c . f(n)$ for some constant

$c < 1$ and all sufficiently large n, then

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

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 $a f(n/b) \le c . f(n)$ for some constant

$c < 1$ and all sufficiently large n, then

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

Master Method

$T(n) = \Theta(1)$ if $n <= c$

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

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

Larger of the two determines the solution to the recurrence.

Case 1 : $n^{\log_b a}$ is larger

Case 3 : $f(n)$ is larger

Case 2 : same size

Master Theorem: Case 1- Example

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

$a = 9, b = 3$

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

$f(n) = n = O(n) = O(n^{2-1}), \varepsilon = 1$

Case 1 of Master Theorem

Solution: $T(n) = \Theta(n^2)$

Master Theorem: Case2 Example

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

$a = 1,\ b = 3/2\ ,\ f(n) = 1$

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

$f(n) = \Theta(1)$

Case 2 of Master Theorem

Solution: $\boldsymbol{T(n) = \Theta(lgn)}$

Master Theorem: Merge Sort

$T(n) = 2T(n/2) + \Theta(\mathrm{n})$

$a=2,\ b=2\ ,\ f(n) = \Theta(\mathrm{n})$

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

$f(n) = \Theta(\mathrm{n})$

Case 2 of Master Theorem

Solution: ***$T(n) = \Theta(n \lg n)$***

Master Theorem: Case3 Example

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

$a=3,\ b=4\ ,\ f(n) = n \lg n$

$n^{\log_b a} = O(n^{0.793})$

$f(n) = \Omega(n^{\log_b a + \varepsilon})$ for $\varepsilon \approx 0.2$

Show that the regularity condition holds.

$a\, f(n/b) = 3(n/4) \lg(n/4) <= (3/4)\, n \lg n = c\, f(n)$ for $c=3/4$

Case 3 of Master Theorem

Solution: $\boldsymbol{T(n) = \Theta(n \lg n)}$

Master Method - Limitations

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

$a=2,\ b=2,\ f(n) = n \lg n$

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

Case 3 ?

Falls into the gap between case 2 and case 3.

Master method does not apply.

Reference

T H Cormen, C E Leiserson, R L Rivest, C Stein *Introduction to Algorithms,* 3rd ed., PHI, 2010