Ch8: Recurrences

305234

Algorithm Analysis and Design

Jiraporn Pooksook
Naresuan University

What is Recurrences?

- A recurrence is an equation or inequality that that describes a function in terms of its value on smaller inputs.
- Example:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

We can write T(n) in terms of $\Theta(n \lg n)$

Methods for solving Recurrences

- **Substitution method**
  - Guess the form of the solution and use mathematical induction to find constants to prove the solution.
- **Recursion-tree method**
  - Each node represents the cost of a single subproblem. We sum the costs within each level of the tree to obtain a set of per-level costs, and sum all the per-level costs to determine the total cost.
- **Master method**
  - Provides a “cookbook” method for solving recurrences of the form $T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

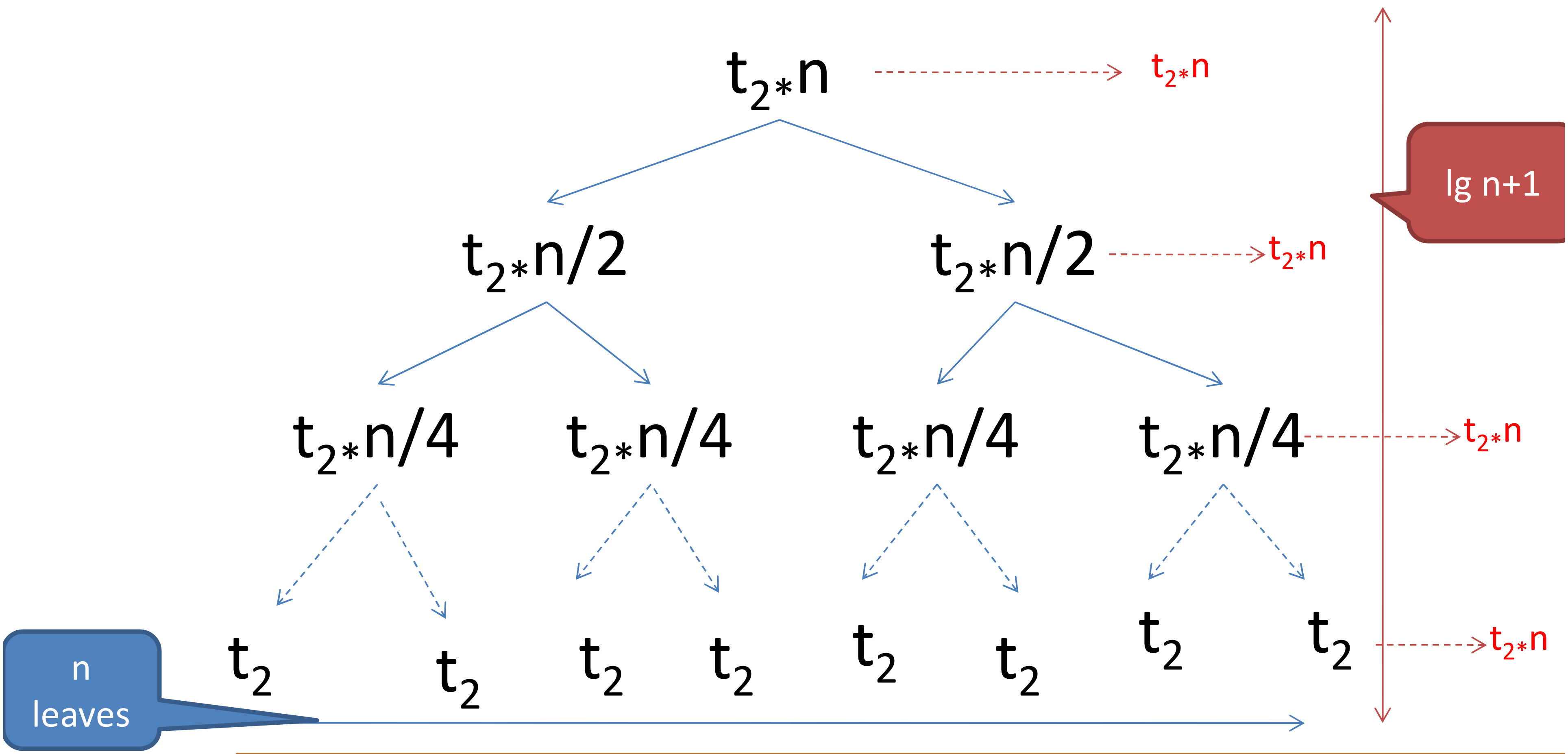
Example: Substitution of Merge-Sort

- Let the recurrence $T(n) = 2T(\lfloor n/2 \rfloor) + n$
- We guess that the solution is T(n) = O(n lg n)
- Must prove that T(n) ≤ cn lg n for choosing a constant c > 0.
- Start by assuming that this bound, cn lg n, holds for $\lfloor n/2 \rfloor$
- Then substitute into the recurrence yields:

$$
\begin{aligned}
T(n) &\le 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\
&\le cn \lg(n/2) + n \\
&= cn \lg n - cn \lg 2 + n \\
&= cn \lg n - cn + n \\
&\le cn \lg n
\end{aligned}
$$

- Where the last step holds as long as c >= 1.

Example: Recursion Tree of Merge-Sort



มีทั้งหมด lgn+1 level และแต่ละ level มี cost = $t_{2*}n$ ดังนั้น จะได้ว่า total cost = $t_{2*}n \lg n + t_{2*}n$ ซึ่งคือ $\Theta(n \lg n)$

The Master Method

- Provides a "cookbook" method for solving recurrences of the form

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

  where a ≥ 1 and b >1 are constants and f(n) is an asymptotically positive function.

- This recurrence form describes the running time of an algorithm that divides a problem of size n into a subproblems, each of size n/b.
- The master method are used in 3 cases:

  f(n) < $n^{\log_b a}$, f(n) = $n^{\log_b a}$, and f(n) > $n^{\log_b a}$

Master Theorem

- Let a ≥ 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) can be bounded asymptotically as follows.

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

Example: Master method (case1)

- Let $T(n) = 9T(n/3) + n$
- Determine which case of the master theorem applies:
- We have a=9, b=3, f(n)= n
- Thus we have $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
- Since $f(n) = O(n^{\log_3 9 - \varepsilon})$, where ε = 1, we can apply case 1 of the master theorem and conclude that the solution is

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

ข้อสังเกต $n < n^2$,
case 1

เหมือนกับโชว์ให้เห็นว่า
f(n)=n อยู่ใน
$O(n^{2-e})$ เมื่อสมมติให้
e คือค่าคงที่

Example: Master method (case 2)

- Let $T(n) = T(2n/3) + 1$
- Determine which case of the master theorem applies:
- We have a=1, b=3/2, f(n)= 1
- Thus we have $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
- Since $f(n) = \Theta(n^0) = \Theta(1)$ we can apply case 2 of the master theorem and conclude that the solution is $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(\lg n)$

ข้อสังเกต $1 = n^0$, case 2

Example: Master method (case 3)

- Let $T(n) = 3T(n/4) + n\lg n$
- Determine which case of the master theorem applies:
- We have a=3, b=4, f(n)= n lg n
- Thus we have $n^{\log_b a} = n^{\log_4 3}$
- Since $f(n) = \Omega(n^{\log_4 3 + \varepsilon})$ where $\varepsilon \approx 0.2$ we can apply case 3 of the master theorem.
- We show that: 3(n/4)lg(n/4) <= (3/4)n lg n , for c=3/4
- Following case 3, we conclude that the solution is

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

ข้อสังเกต n lg n> $n^{0.793}$, case 3

Practice: Master method

- Let $T(n) = 4T(n/3) + 5n$
- Let $T(n) = 3T(n/3) + 5n$
- Let $T(n) = 2T(n/3) + 5n$

Practice: Master method

- Let $T(n) = 4T(n/3) + 5n$
- We have a=4, b=3, f(n)=5n

$$n^{\log_b a} = n^{\log_3 4} = n^{1.26}$$

- We guess it might be case 3:
- Since $f(n) \notin \Omega(n^{\log_3 4+\varepsilon})$, where ε = 0.2.
- Hence we check for case 1:
- Since $f(n) = O(n^{\log_3 4-\varepsilon})$, where ε = 1, ($n^{\log_3 4-1} = n$). we can apply case 1 of the master theorem and conclude that the solution is

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_3 4})$$

พิสูจน์ big omega,
สมการคือ
0<= $cn^{>1.26}$ <=5n
เลือก c<5 แต่ว่า พอ n มากขึ้น ก็จะยิ่งทำให้สมการไม่เป็นจริง ดังนั้น พิสูจน์ว่าไม่จริง เพราะหาค่า n น้อยสุดไม่ได้

พิสูจน์ big oh,
สมการคือ
0<=5n <= cn
เลือก n>=1, c>5 ก็จะทำให้สมการเป็นจริง

Practice: Master method

- Let $$T(n) = 3T(n/3) + 5n$$
- We have a=3, b=3, f(n)=5n

$$n^{\log_b a} = n^{\log_3 3} = n^1$$

- We guess it might be case 2:
- Since $f(n) = \Theta(n^{\log_3 3})$, ($n^{\log_3 3} = n$) , hence we can apply case 2 of the master theorem and conclude that the solution is

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

Practice: Master method

- Let $T(n) = 2T(n/3) + 5n$
- We have a=2, b=3, f(n)=5n

$$n^{\log_b a} = n^{\log_3 2} = n^{0.63}$$

- We guess it might be case 3:
- Since $f(n) = \Omega(n^{\log_3 2+\varepsilon})$, where ε =1,($n^{\log_3 3} = n$) , and 2(5n/3) <= (2/3)5n , for c=2/3
- hence we can apply case 3 of the master theorem and conclude that the solution is

$$T(n) = \Theta(f(n)) = \Theta(5n)$$