

Solving Recurrences and the Master Theorem

CS 4231, Fall 2020

Mihalis Yannakakis

Recurrences

- Arise in the analysis of recursive algorithms that reduce an instance of a problem to smaller instances

$$T(n) = \begin{cases} \text{Function of } T(\text{smaller } n') & \text{if } n > cst. \\ c & \text{if } n \leq cst. \end{cases}$$

- Want to obtain tight asymptotic expression for $T(n)$, i.e.
 $T(n)=\Theta(f(n))$
- Means prove both upper bound $O(f(n))$ and lower bound $\Omega(f(n))$

Methods for Solving Recurrences

- Unfolding (expanding) the recursion
 - if it is not too complicated
- Recursion-tree method
 - Expand the recursion tree
- Substitution (Verification) method
 1. Guess the form of the solution
 2. Use induction to prove it formally
- Master Method
 - explicit solution for a class of common recurrences

Example: Insertion Sort (recursive form)

Method: Reduction to a smaller instance:

1. Sort recursively the first $n-1$ elements using Insertion-Sort
2. Insert the n -th element in the right position

Analysis of Running time

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

Recursive equation / Recurrence

Unfolding (Expansion) of the recursion:

$$\begin{aligned} T(n) &= T(n-1) + cn \\ &= T(n-2) + c(n-1) + cn \\ &= T(n-3) + c(n-2) + c(n-1) + cn \\ &= \dots \\ &= T(i) + c[(i+1) + (i+2) + \dots + (n-1) + n] \\ &= \dots \\ &= T(1) + c[2 + 3 + \dots + (n-1) + n] \\ &= c [1 + 2 + \dots + (n-2) + (n-1) + n] \\ &= cn(n+1)/2 \\ &= \Theta(n^2) \end{aligned}$$

Example: Recurrence of Merge-Sort

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & \text{for } n > 1 \\ \Theta(1) & \text{for } n = 1 \end{cases}$$

Assume for simplicity that n is a power of 2

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

Can solve it by unfolding, or by recursion tree

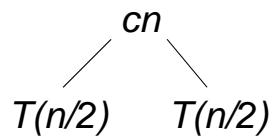
Unfolding the recursion

$$\begin{aligned}T(n) &= 2T(n/2) + cn \\&= 2(2T(n/4) + c(n/2)) + cn \\&= 4T(n/4) + 2cn \\&= 4(2T(n/8) + c(n/4)) + 2cn \\&= 8T(n/8) + 3cn \\&\quad \dots \\&= 2^i \cdot T(n/2^i) + i \cdot cn \\&\quad \dots \\&= n \cdot T(1) + \log n \cdot cn \\&= \Theta(n \log n)\end{aligned}$$

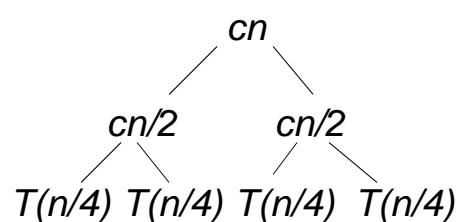
Recursion Tree

$T(n)$

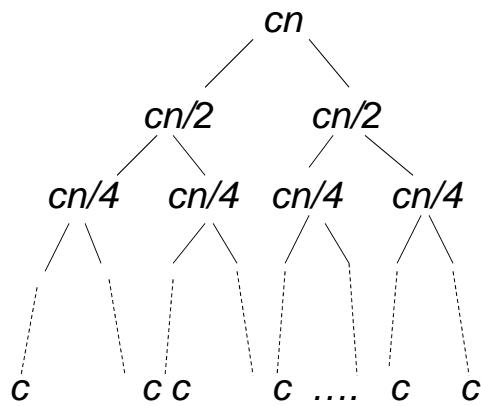
Recursion Tree



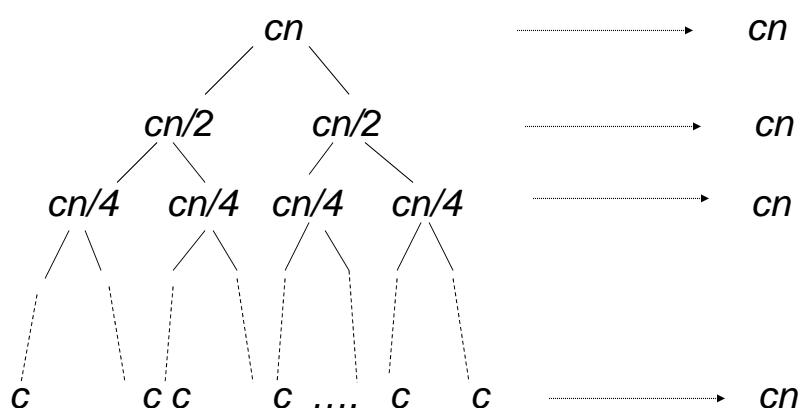
Recursion Tree



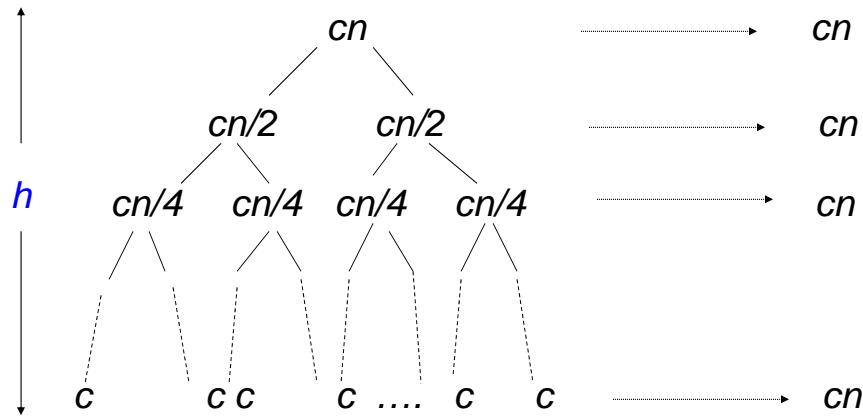
Recursion Tree



Recursion Tree

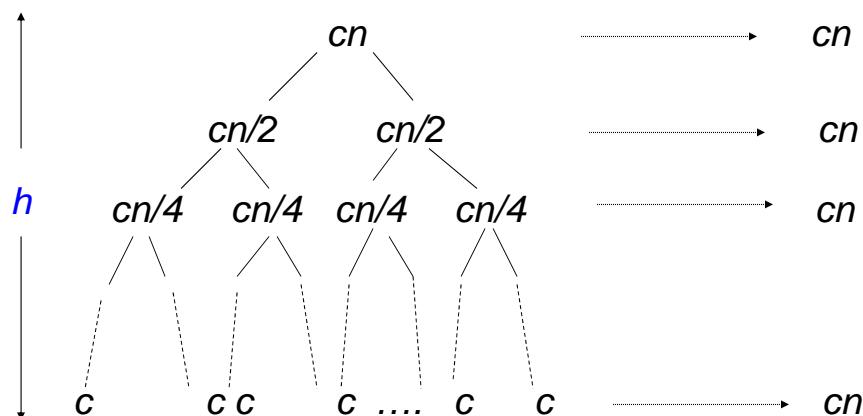


Recursion Tree



#levels: $h = \log n + 1$

Recursion Tree



#levels: $h = \log n + 1$

Total: $cn(\log n + 1)$

What if n is not a power of 2?

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn & \text{for } n > 1 \\ c & \text{for } n = 1 \end{cases}$$

$T(n)$ is monotonic in n , i.e.,
 $n \leq m \Rightarrow T(n) \leq T(m)$

Formal Proof: By induction on n

Basis: $n = 1$: $T(1) = c \leq T(m)$

Induction step: $n > 1$

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn \\ &\leq T(\lfloor m/2 \rfloor) + T(\lceil m/2 \rceil) + cm \\ &= T(m) \end{aligned}$$

What if n is not a power of 2?

n lies between two successive powers of 2:

$$2^k \leq n \leq 2^{k+1}, \quad k = \lfloor \log n \rfloor$$

$T(n)$ is monotonic in $n \Rightarrow$

$$T(2^k) \leq T(n) \leq T(2^{k+1})$$

$$c2^k(k+1) \leq T(n) \leq c2^{k+1}(k+2)$$

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

Substitution method - example

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

Guess: $T(n) = \Theta(n)$

Verify upper bound ($T(n)=O(n)$)
and lower bound ($T(n) = \Omega(n)$)

Substitution method - example

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

Upper Bound:

Guess: $T(n) \leq cn$
and show it inductively

Verify inductive step:

$$\begin{aligned} T(n) &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn+1 \end{aligned}$$

Induction does not go through.

Substitution method - example

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

Guess: $T(n) \leq cn - b$

Verify inductive step:

$$\begin{aligned} T(n) &\leq c\lfloor n/2 \rfloor - b + c\lceil n/2 \rceil - b + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b, \quad \text{holds provided } b \geq 1 \end{aligned}$$

Choose c, b large enough to handle also the boundary condition:

For example, if $T(1)=3$, then let $b=1, c=4$

Master Method

- Applies to class of recurrences

$$T(n) = aT(n/b) + f(n), \text{ where constants } a \geq 1, b > 1$$

- Arise often in divide and conquer

- Divide the given instance of size n into a subinstances of size n/b
- Conquer recursively the subinstances
- Combine the solutions for the subinstances

$$f(n) = \text{time of divide and combine steps}$$

Master Method: $T(n) = aT(n/b) + f(n)$

$$1. \quad f(n) = O(n^{\log_b a - \varepsilon}), \quad \varepsilon > 0 \quad \Rightarrow \quad \Theta(n^{\log_b a})$$

$$2. \quad f(n) = \Theta(n^{\log_b a}) \quad \Rightarrow \quad \Theta(n^{\log_b a} \log n)$$

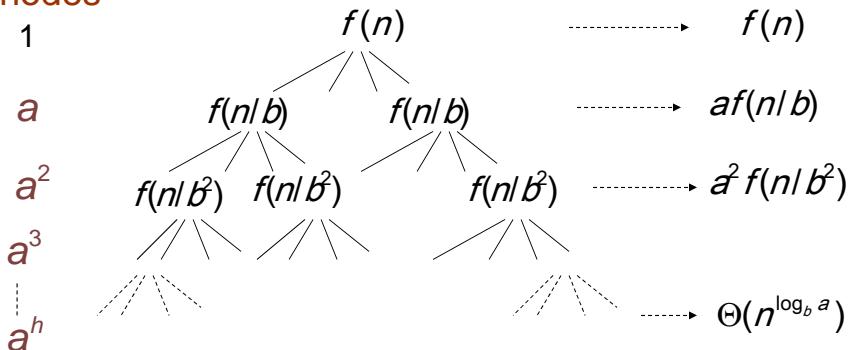
$$f(n) = \Theta(n^{\log_b a} \log^k n) \quad \Rightarrow \quad \Theta(n^{\log_b a} \log^{k+1} n)$$

$$3. \quad \left. \begin{array}{l} f(n) = \Omega(n^{\log_b a + \varepsilon}), \quad \varepsilon > 0 \\ af(n/b) \leq cf(n), \text{ some } c < 1 \end{array} \right\} \Rightarrow \quad \Theta(f(n))$$

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

Recursion tree

#nodes



$$\text{Height } h = \log_b n$$

$$\#\text{leaves} = a^{\log_b n} = n^{\log_b a}$$

$$\text{Total: } \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i)$$

Which term dominates?

Case 1: $f(n) \ll n^{\log_b a}$

$f(n) = O(n^{\log_b a - \varepsilon}) \Rightarrow \exists c > 0 \text{ s.t. } f(n) \leq cn^{\log_b a - \varepsilon} \text{ for large } n$

$$a^i f(n/b^i) \leq c a^i (n/b^i)^{\log_b a - \varepsilon} = c n^{\log_b a - \varepsilon} a^i \frac{b^{i\varepsilon}}{b^{i\log_b a}} = c n^{\log_b a - \varepsilon} b^{i\varepsilon}$$

$$\begin{aligned} \sum_{i=0}^{\log_b n-1} a^i f(n/b^i) &\leq c n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n-1} b^{i\varepsilon} = c n^{\log_b a - \varepsilon} \frac{b^{\varepsilon \log_b n} - 1}{b^\varepsilon - 1} \\ &= c n^{\log_b a - \varepsilon} \frac{n^\varepsilon - 1}{b^\varepsilon - 1} = O(n^{\log_b a}) \end{aligned}$$

Total: $\Theta(n^{\log_b a})$ (first term dominates)

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

Upper bound:

$$\begin{aligned} \sum_{i=0}^{\log_b n-1} a^i f(n/b^i) &\leq \sum_{i=0}^{\log_b n-1} a^i c (n/b^i)^{\log_b a} = \sum_{i=0}^{\log_b n-1} c a^i \frac{n^{\log_b a}}{b^{i\log_b a}} \\ &= \sum_{i=0}^{\log_b n-1} c n^{\log_b a} = O(n^{\log_b a} \log n) \end{aligned}$$

Lower bound:

similar (reverse inequality)

Total: $\Theta(n^{\log_b a} \log n)$

Case 3: $f(n) \gg n^{\log_b a}$

$af(n/b) \leq cf(n)$ for some $c < 1 \Rightarrow$
 $a^i f(n/b^i) \leq c^i f(n)$ (proof by induction on i) \Rightarrow

$$\sum_{i=0}^{\log_b n-1} a^i f(n/b^i) \leq \sum_{i=0}^{\log_b n-1} c^i f(n) = f(n) \sum_{i=0}^{\log_b n-1} c^i$$
$$\leq f(n) \sum_{i=0}^{\infty} c^i = \frac{1}{1-c} f(n) = O(f(n)) \quad \text{since } c < 1$$

The first term is $\Theta(n^{\log_b a}) = O(f(n))$

Total: $\Theta(f(n))$

Master Method

- Same result applies to the floor and ceiling variants:

$$T(n) = aT(\lfloor n/b \rfloor) + f(n)$$

$$T(n) = aT(\lceil n/b \rceil) + f(n)$$

Examples

1. Merge Sort: $T(n) = 2T(n/2) + \Theta(n)$

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

Case 2 $\Rightarrow T(n) = \Theta(n \log n)$

2. Binary search: $T(n) = T(n/2) + 1$

$$a=1, b=2, f(n)=1, n^{\log_b a} = n^0 = 1$$

Case 2 $\Rightarrow T(n) = \Theta(\log n)$

Examples ctd.

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

$$a=2, b=2, f(n)=1, n^{\log_b a} = n$$

Case 1 $\Rightarrow T(n) = \Theta(n)$

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

$$a=3, b=2, f(n)=n^2, n^{\log_b a} = n^{\log 3} \approx n^{1.58}$$

$$af(n/b) = 3(n/2)^2 = (3/4)n^2 = (3/4)f(n)$$

Case 3 $\Rightarrow T(n) = \Theta(n^2)$

Changing variables

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

$$m = \log n$$

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

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

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

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

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

Sometimes the master method does not apply: in-between cases

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

$$a = 4, b = 2, f(n) = n^2/\log n, n^{\log_b a} = n^2$$

- $f(n) = O(n^{\log_b a})$ but not $O(n^{\log_b a - \varepsilon})$ for any constant $\varepsilon > 0$
because $\frac{f(n)}{n^{\log_b a - \varepsilon}} = \frac{n^2/\log n}{n^2/n^\varepsilon} = \frac{n^\varepsilon}{\log n} \rightarrow \infty$
 \Rightarrow not Case 1

- $f(n)$ not $\Theta(n^{\log_b a})$ because $\frac{f(n)}{n^{\log_b a}} = \frac{1}{\log n} \rightarrow 0$
 \Rightarrow not Case 2 (nor 3)

Solution: $\Theta(n^2 \log \log n)$

Master Theorem Summary

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

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

