COT 6405 Introduction to Theory of Algorithms

Topic 5. Master Theorem

Solving the recurrences

- Substitution method
- Recursion tree
- Master method

The Master Theorem

- Given: a *divide-and-conquer* algorithm
  - An algorithm that divides the problem of size *n* into *a* subproblems, each of input size *n/b*
  - Let the cost of each stage (i.e., the work to divide the problem + combine solved subproblems) be described by the function *f*(n)
- Then, the Master Theorem gives us a cookbook for the algorithm's running time

The Master Theorem (Cont'd)

- If T(n) = aT(n/b) + f(n) then

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

The Master Theorem (Cont'd)

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

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

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$.

($f(n)$ is polynomially smaller than $n^{\log_b a}$.)

Solution: $T(n) = \Theta(n^{\log_b a})$.

(Intuitively: cost is dominated by leaves.)

The Master Theorem (Cont'd)

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$.

[This formulation of Case 2 is more general than in Theorem 4.1, and it is given in Exercise 4.6-2]

($f(n)$ is within a polylog factor of $n^{\log_b a}$, but not smaller.)

Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

(Intuitively: cost is $n^{\log_b a} \lg^k n$ at each level, and there are $\Theta(\lg n)$ levels.)

Simple case: $k = 0 \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$.

The Master Theorem (Cont'd)

Case 3: $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$ and $f(n)$ satisfies the regularity condition $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n.
($f(n)$ is polynomially greater than $n^{\log_b a}$.)
Solution: $T(n) = \Theta(f(n))$.
(Intuitively: cost is dominated by root.)

Using the Master Theorem, Case 1

- Solve $T(n) = 9T(n/3) + n$
  - $a=9$, $b=3$, $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = n^2$
  - Since $f(n) = O(n^{2-\varepsilon})$, where $\varepsilon=1$, case 1 applies:

$$\boldsymbol{T}(\boldsymbol{n}) = \Theta\left(\boldsymbol{n}^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right) \text{when } \boldsymbol{f}(\boldsymbol{n}) = \boldsymbol{O}\left(\boldsymbol{n}^{\log_{\boldsymbol{b}} \boldsymbol{a}-\varepsilon}\right)$$

  - Thus the solution is $T(n) = \Theta(n^2)$

Using the Master Theorem, Case 2

- $T(n) = T(2n/3) + 1$
  - a=1, b=3/2, f(n) = 1
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$ → compare with f(n)=1
  - Since $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$, the simple form of case 2 applies:

$$T(n) = \Theta\left(n^{\log_b a} \lg n\right) \text{when } f(n) = \Theta\left(n^{\log_b a}\right)$$

  - Thus the solution is $T(n) = \Theta(\lg n)$

Using the Master Theorem, Case 3

- T(n) = 3T(n/4) + n lg n
  - a=3, b=4, f(n) = nlgn
  - $n^{\log_b a}$= $n^{\log_4 3}$ = n $^{0.793}$ → compare with f(n)=n lg n
  - Since f(n) = $\Omega(n^{0.793+\varepsilon})$, where ε=0.207
    - Also for c=3/4 < 1 , a*f(n/b) <= c*f(n)
      →3(n/4)*lg(n/4) <= (3/4)n lg n
  - case 3 applies:

$$T(n) = \Theta\left(f(n)\right) \text{when } f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$$

  - Thus the solution is T(n) = Θ(n lg n)

Exercises

- T(n) = 5T(n/2) +$\Theta(n^2)$
- T(n) = 27 T(n/3) +$\Theta(n^3 lgn)$
- T(n) = 5T(n/2) +$\Theta(n^3)$

Exercises (cont'd)

- T(n) = 5T(n/2) +$\Theta(n^2)$
- a = 5, b = 2, f(n) = $\Theta(n^2)$
- $n^2 \in O\left(n^{\log_2 5-\varepsilon}\right)$
- Case 1, T(n) = $\Theta(n^{\log_2 5})$

Exercises (cont'd)

- T(n) = 27 T(n/3) +$\Theta(n^3 lgn)$
- a = 27, b = 3, f(n) = $\Theta(n^3 lgn)$
- $n^{\log_3 27}$ = n^3
- Case 2: k = 1, and f(n) = Θ ($n^{\log_3 27}$lg$n$)
- T(n) = Θ ($n^3 lg^2 n$)

Exercises (cont'd)

- T(n) = 5T(n/2) +$\Theta(n^3)$
- a = 5, b = 2, f(n) = $\Theta(n^3)$
- $n^3 \in \Omega\left(n^{\log_2 5+\varepsilon}\right)$
- Case 3, check the regularity condition
  - a f(n/b) = 5$(\frac{n}{2})^3$ = (5/8) $n^3 \leq cn^3$ for c = 5/8 <1
- T(n) = $\Theta(n^3)$

Limitation of the Master Theorem

- Master Theorem does not apply to all f(n)!
  - Gap between Case 1 and 2: f(n) is not polynomially smaller than $n^{\log_b a}$
  - Gap between Case 2 and 3: f(n) is not polynomially larger than $n^{\log_b a}$
  - The regularity condition in Case 3

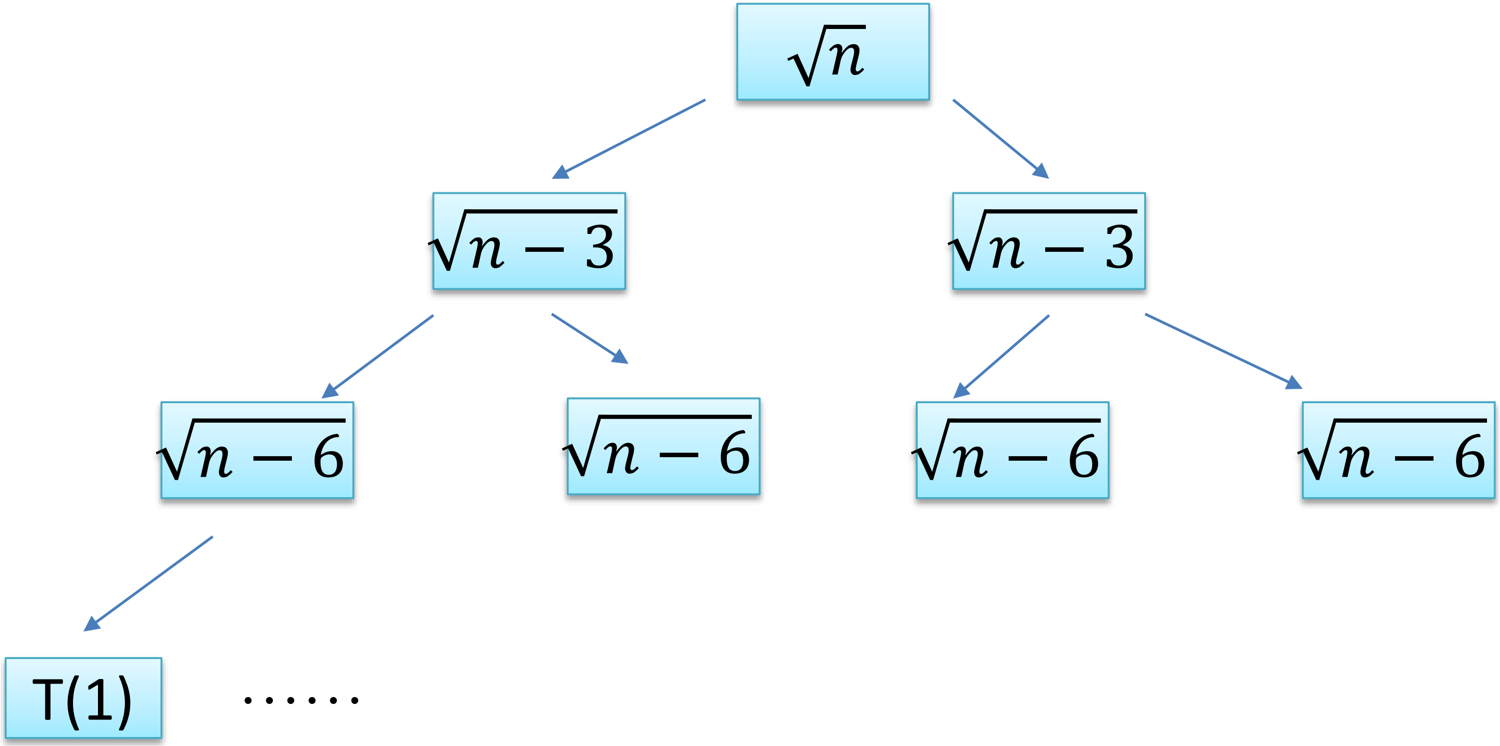
$T(n) = 27T(n/3) + \Theta(n^3/\lg n)$
$n^{\log_3 27} = n^3$ vs. $n^3/\lg n = n^3 \lg^{-1} n \neq \Theta(n^3 \lg^k n)$ for any $k \geq 0$.
Cannot use the master method.

Limitations (cont’d)

- Situations that don’t look anything like that of the Master Theorem
- $T(n) = 2T(n-3) + \sqrt{n}$

What to do when it doesn't apply

- The recursion-tree method



Cont'd

- The sub-problem size for a node at depth *i* is $n - 3i$
- The sub-problem size hits T(1), when $n - 3i$= 1, or $i = (n - 1)/3$
- Thus, tree has 1+ (n-1)/3 levels (*i* = 0,1,..., (n-1)/3)

Cont'd

- Each node at level *i* has a cost of $\sqrt{n-3i}$
- Each level has 2^i nodes
  - Level 0: 1, level 1: 2, level 2:4, level 3: 8….
- Thus, the total cost of level *i* is $2^i\sqrt{n-3i}$

Cont'd

- The bottom level has $2^{(n-1)/3}$ nodes, each costing T(1)
- Assume T(1) = c_0. The total cost of the bottom level will be $c_0 2^{(n-1)/3}$

Cont'd

- We add up the costs over all levels to determine the total cost for the entire tree:

$$\mathrm{T}(n) = \sum_{i=0}^{\frac{n-1}{3}-1} 2^i\sqrt{n-3i} + c_0 2^{(n-1)/3}$$

$$\leq \sum_{i=0}^{\frac{n-1}{3}-1} 2^i\sqrt{n} + c_0 2^{(n-1)/3}$$

$$= -\sqrt{n}(1\text{-}2^{(n-1)/3}) + c_0 2^{(n-1)/3}$$

$$= \sqrt{n}2^{(n-1)/3} + c_0 2^{(n-1)/3} - \sqrt{n}$$

$$= \mathrm{O}(\sqrt{n}2^{n/3})$$

Processing floors and ceilings

- T(n) = 2T($\lfloor n/2 \rfloor$) + n has the solution of T(n) = $\Theta(nlgn)$
- T(n) = 2T($\lfloor n/2 \rfloor$) + n

  $\leq 2\mathrm{T}\left(\frac{n}{2}\right) + n$ - > O(nlgn)
- T(n) = 2T($\lfloor n/2 \rfloor$) + n

  $\geq 2\mathrm{T}\left(\frac{n}{2} - 1\right) + n$

  = $2\mathrm{T}\left(\frac{n-2}{2}\right) + (n$ -2) +2 - > Ω(nlgn)

Processing floors and ceilings (cont'd)

- T(n) = 2T($\lceil n/2 \rceil$) + n has the solution of T(n) = $\Theta(nlgn)$
- T(n) = 2T($\lceil n/2 \rceil$) + n

$$\leq 2\mathrm{T}\left(\frac{n}{2} + 1\right) + n$$

$$= 2\mathrm{T}\left(\frac{n+2}{2}\right) + (n + 2) - 2 \; - > \Omega(\mathrm{nlgn})$$

- T(n) = 2T($\lceil n/2 \rceil$) + n

$$\geq 2\mathrm{T}\left(\frac{n}{2}\right) + n \;\; -> O(nlgn)$$