CSE3318: Divide and Conquer Solving Recurrences by Dr. Bhanu Jain

All slides are based on: ***Introduction to Algorithms***, by Thomas H. Cormen, Charles E. Leiserson, Ronald E. Rivest, Clifford Stein,3rd edition (CLRS)

Recurrences

What is a recurrence?

- An equation or inequality that describes a function in terms of its value(s) at smaller inputs. They are used in mathematics/computer science to model problems where a solution depends on smaller subproblems, (like in divide-and-conquer algorithms)

Key Components of a Recurrence

1. **Base Case(s):** Defines the value of the function for the smallest input(s), providing a starting point for the recurrence.
2. **Recursive Relation:** Expresses how the function depends on its value(s) for smaller input(s). **Examples:**

   a. **Fibonacci Numbers:**
      - **Recurrence:** $F(n)=F(n-1)+F(n-2)$ **Base Case(s):** $F(0)=0$ $F(1)=1$

   b. **Factorial:**
      - **Recurrence:** $n!=n\cdot(n-1)!$ **Base Case**: $0!=1$

   c. **Merge Sort Time Complexity:**
      - **Recurrence**: $T(n)=2T(n/2)+O(n)$ **Base Case**: $T(1)=O(1)$

Recurrences

- **Purpose of Recurrences**
  - **Algorithm Analysis:** Recurrences are used to analyze the time complexity of recursive algorithms.
  - **Problem Modeling:** They model processes that involve repeated computation or self-similar structures.
  - **Mathematical Computation:** Recurrences provide a way to compute sequences or functions iteratively or recursively.
- **Solving Recurrences**
  - To solve a recurrence means to find a closed-form expression (non-recursive) for the function. Methods include:
    - **Substitution Method:** Guess a bound and verify the solution by using mathematical induction (by substitution.)
    - **Recursion Tree:** Visualize the recurrence as a tree and sum contributions at each level.
    - **Master Theorem:** A shortcut for solving divide-and-conquer recurrences of the form: **$T(n)=aT(n/b)+f(n)$** **Memorize three cases** and use them to determine the running times of algorithms.

    a subproblems, each of which is **1/b** the size of the original problem, and in which the divide and combine steps together take **f(n)** time . **f(n)** is of the form **$O(n^d)$**

When we state and solve recurrences, we often omit floors, ceilings, and boundary condition
Two problems of sizes $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$ are written as 2 problems of size n/2 as the running time of an algorithm on a constant-sized input is a constant and hence ignored. (Example Merge Sort)

Master Method

- The **Master Method** is a tool to figure out how much time a recursive algorithm will take. Recursive algorithms are like a process where a big problem is broken into smaller parts, each part is solved, and then the solutions are combined to solve the whole problem.
- Think of it like organizing a party:
    1. **Breaking the task into smaller ones:** Divide tasks: buying food, setting up decorations, arranging seating etc.
    2. **Doing the smaller tasks:** Each task (shopping, decorating, etc.) is worked on individually.
    3. **Combining everything:** Once the smaller tasks are done, bring everything together to make the party happen.

- The Master Method helps identify where most of the effort goes:
    1. Does dividing the problem into smaller parts take the most time?
    2. Does solving all the smaller tasks take the most time?
    3. Or does combining everything at the end take the most time?
- By figuring this out, you can estimate the total effort needed for the process.

Master Method

- Master Theorem for Divide-and-Conquer Recurrences. Consider a recurrence of the form:

$T(n) = a\, T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$

2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.

3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \leq c f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the “regularity condition” holds.

Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$

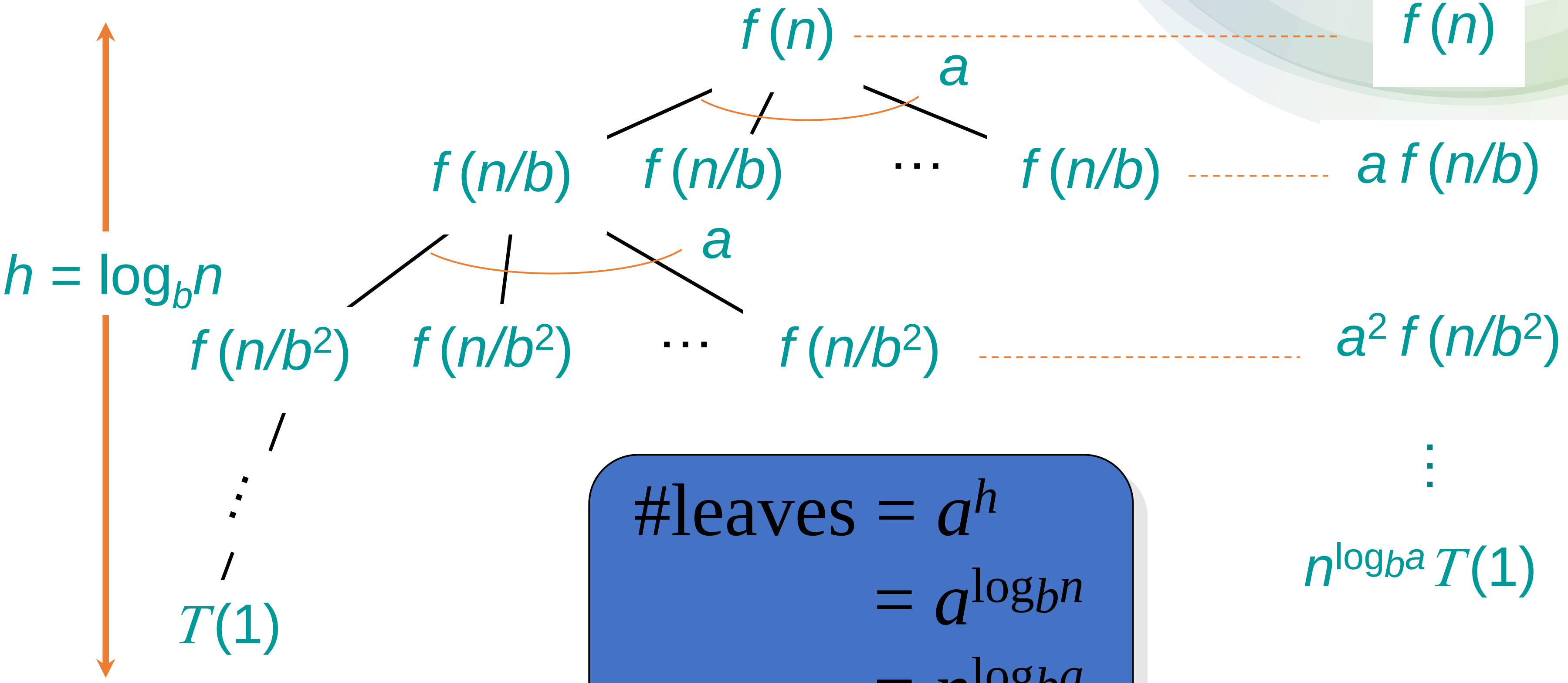
2. If $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.

3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

Recursion tree:



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

$$a^{\log_b c} = c^{\log_b a}$$
$$a, b, c > 0;\ b \neq 1$$

$a^{\log_b c} = c^{\log_b a}$

a, b, c>0; b ≠1

Check with data:

assume a=4 b=2 c=8

$4^{\log_2 8} = 8^{\log_2 4} \Rightarrow 4^3 = 8^2 \Rightarrow 64=64$

Application of Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$

2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.

3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the “regularity condition” holds.

Ex.1 $T(n) = 4T(n/2) + n$

$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n$.

CASE 1: $f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1$.

$\therefore\ T(n) = \Theta(n^2)$.

Application of Master Theorem

$T(n) = a\, T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$

2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.

3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a\, f(n/b) \leq c\, f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

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

$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n^2$.

CASE 2: $f(n) = \Theta(n^2)$.

$\therefore\ T(n) = \Theta(n^2 \lg n)$.

Application of Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$

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

   Both recursion and f(n) contribute equally.

3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

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

$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n^3$.

CASE 3: $f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$

and $4(n/2)^3 \leq cn^3$ (reg. cond.) for $c = ½<1$

$\therefore\ T(n) = \Theta(n^3)$.

Application of Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$
2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.
3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

Ex.4 $T(n) = 9T(n/3)+n;$

$a=9, b=3, f(n) = n$

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

$f(n)=O(n^{\log_3 9-\varepsilon})$ for $\varepsilon=1$

By case 1, $T(n) = \Theta(n^2)$.

Application of Master Theorem

$T(n) = a\, T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$
2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.
3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \leq c f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

Ex.5 $T(n) = T(2n/3)+1$

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

$n^{\log_b a} = n^{\log_{3/2} 1} = \Theta(n^0) = \Theta(1)$

By case 2, $T(n) = \Theta(\lg n)$.

https://www.gigacalculator.com/calculators/log-calculator.php $\log_{1.5} 1 = 0$

Application of Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$
2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.
3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \leq c f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

Ex.6 $T(n) = 3T(n/4) + n\lg n$;

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

$n^{\log_b a} = n^{\log_4 3} = \Theta(n^{0.793})$

$f(n) = \Omega(n^{\log_4 3 + \varepsilon})$ for $\varepsilon \approx 0.2$

Moreover, for large n, the "regularity" holds for c=3/4.

$af(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n\lg n = cf(n)$

By case 3, $T(n) = \Theta(f(n)) = \Theta(n\lg n)$.

https://www.gigacalculator.com/calculators/log-calculator.php $\log_4 3 = 0.792481$

Application of Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$

2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.

3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \leq c f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

Ex.7. $T(n) = 2T(n/4) + n^{0.5}$;

$a=2, b=4, f(n) = n^{0.5}$

$n^{\log_b a} = n^{0.5}$ and $f(n) = n^{0.5}$

By case 2, $T(n) = \Theta(f(n)) = \Theta(n^{0.5} \lg n)$.

https://www.gigacalculator.com/calculators/log-calculator.php $\log_4 2 = .5$

Application of Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$
2. If $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the “regularity condition” holds.

Ex.8. $T(n) = 7T(n/2) + n^2$;

$a=7, b=2, f(n) = n^2$

$n^{\log_b a} = n^{\log_2 7 - \varepsilon}$ and $f(n) = n^2$

$2 < \lg 7 < 3$

By case 1, $T(n) = \Theta(f(n)) = \Theta(n^{\log_2 7})$.

https://www.gigacalculator.com/calculators/log-calculator.php $\log_2 7 = 2.8073$

Application of Master Theorem

$T(n) = a\,T(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. **Then:**

1. **If** $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$. Then, $T(n) = \Theta(n^{\log_b a})$

   Recursion dominates as $f(n)$ grows slower than $n^{\log_b a}$
2. **If** $f(n) = \Theta(n^{\log_b a})$ Then, $T(n) = \Theta(n^{\log_b a} \lg n)$.

   Both recursion and f(n) contribute equally.
3. **If** $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ ***and*** $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \leq c f(n)$ for some constant $c < 1$. Then, $T(n) = \Theta(f(n))$

   Work outside recursion dominates, provided the "regularity condition" holds.

Ex.9 $T(n) = 16T(n/4) + n^2$

$a = 16$, $b = 4 \Rightarrow n^{\log_b a} = n^{\log_4 16} = n^2$; $f(n) = n^2$.

CASE 2: $f(n) = \Theta(n^2)$.

$\therefore\ T(n) = \Theta(n^2 \lg n)$.

The END!