CSE3318: Divide and Conquer

Solving Recurrences-Recursion Tree

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: $\mathbf{T(n)=aT(n/b)+f(n)}$ **Memorize three cases** and use them to determine the running times of algorithms.

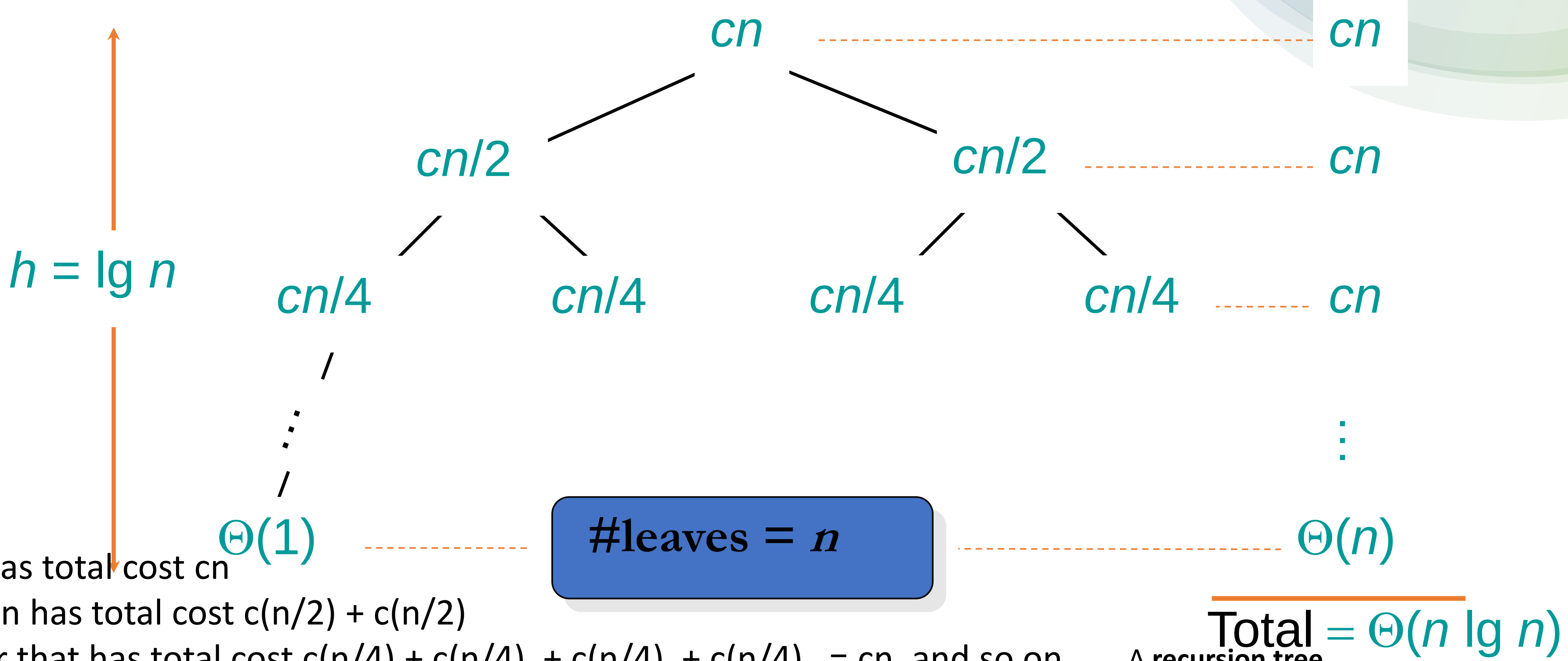
    a subproblems, each of which is **1/b** the size of the original problem, and in which the divide and combine steps together take **f(n)** time . **f(n)** is of the form $\mathbf{O(n^d)}$

Recursion Tree

- **A recursion tree** represents the structure of a recursive algorithm visually, where each node shows the cost of solving a sub-problem.
- **Each node** corresponds to the work done at a particular recursive step, breaking down the problem into smaller parts.
- **Sum of the cost at each level** gives the cost of the entire algorithm.
- **The sum of costs** across all levels of the tree provides an estimate of the total cost of the algorithm.
- **This method** models the cost of a recursive algorithm by organizing computations hierarchically.
- **Using recursion trees** can help generate guesses for solving recurrences with the substitution method.
- **While effective,** the recursion-tree method can at times be unreliable.
- **Despite this,** recursion trees used for developing an intuition for recursive algorithm behavior.

Recursion Tree: Example

Use a **recurrence tree to** analyze the contributions at each level of the tree and determine the total cost. Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.



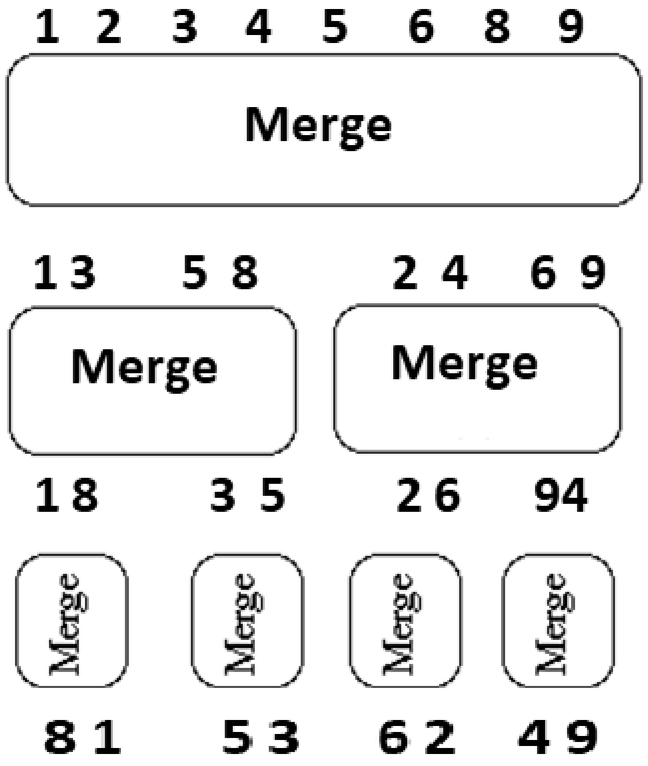
The top level has total cost cn
Next level down has total cost c(n/2) + c(n/2)
Next level after that has total cost c(n/4) + c(n/4) + c(n/4) + c(n/4) = cn, and so on.
In general, the level i below the top has 2^i nodes, each contributing a cost of $c(n/2^i)$, so that the ith level below the top has total cost $2^i c(n/2^i)$ = cn.
The bottom level has n nodes, each contributing a cost of c, for a total cost of cn

A **recursion tree**

- Each node represents the cost of a certain recursive sub-problem.
- Sum of the cost at each level gives the cost of the entire algorithm.

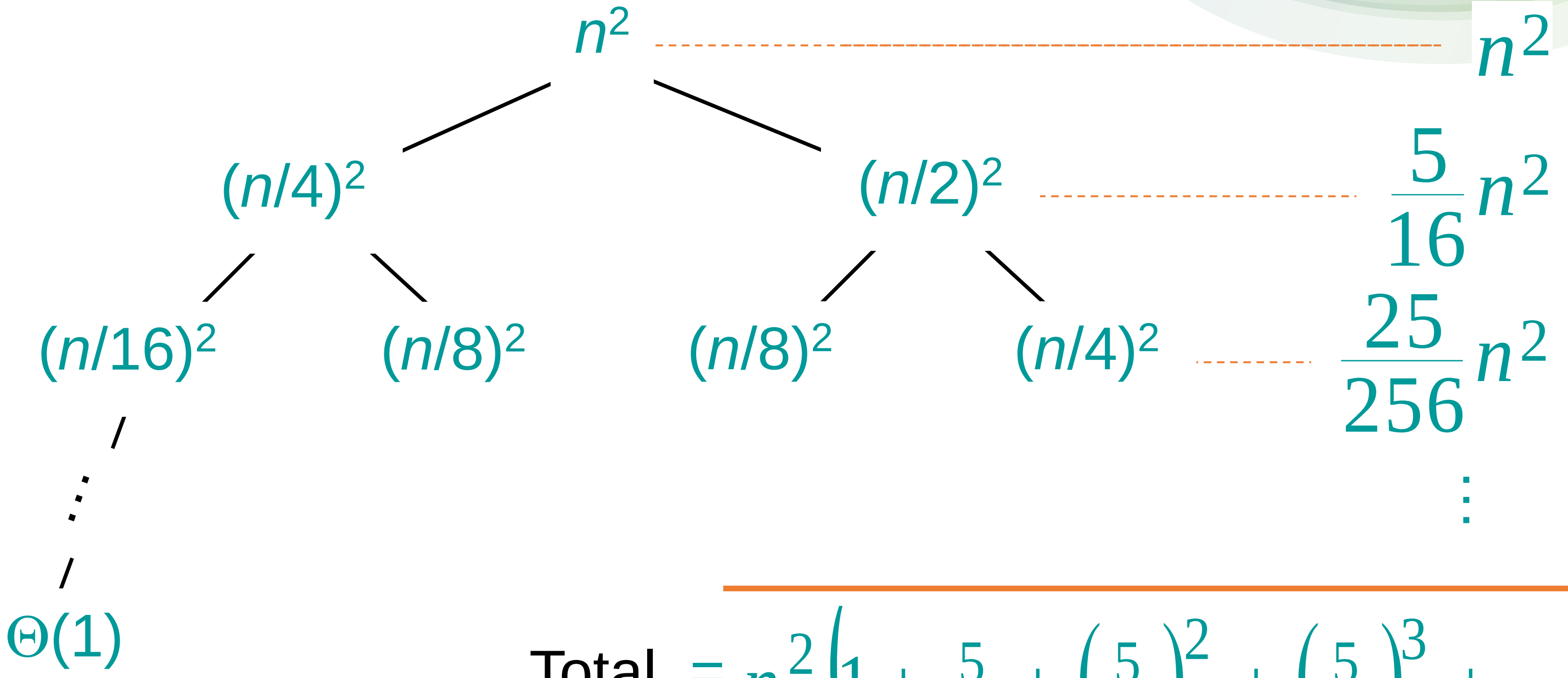
Recursion Tree: Example

- $\Theta(n \lg n)$ grows more slowly than $\Theta(n^2)$.
- Therefore, merge sort asymptotically beats insertion sort in the worst case.



Recursion Tree: Example

Use a recurrence tree to analyze the contributions at each level of the tree and determine the total cost. **T(n)=T(n/4)+T(n/2)+n²**



$$\text{Total} = n^2\left(1+\frac{5}{16}+\left(\frac{5}{16}\right)^2+\left(\frac{5}{16}\right)^3+\cdots\right)$$
$$= \Theta(n^2)$$

geometric series

The END!