CIT 596

Recurrence Relations

Penn Engineering

SOLVING RECURRENCE RELATIONS

- Example: `mergesort` recurrence

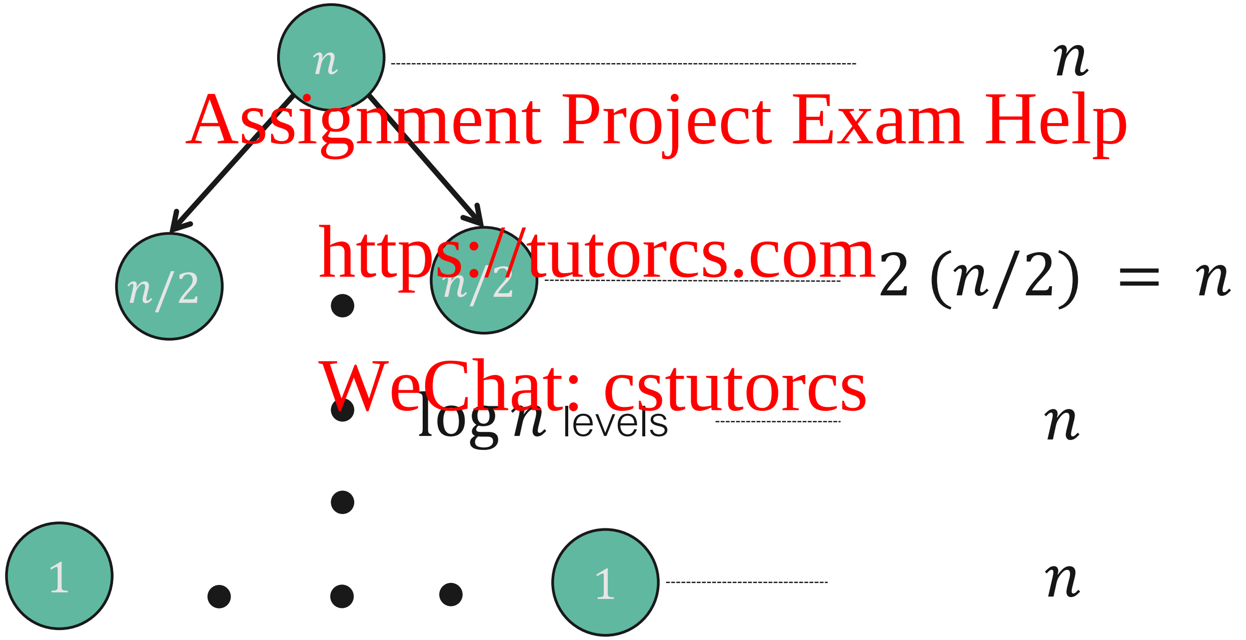
$$T(n) \leq 2T(n/2) + n, T(1) = 0$$

- We can represent the quantities involved by a tree:

  - Each node is labeled with the size of the problem being solved
  - Each level is labeled with the number of steps (excluding recursive calls)
  - Total running time of the algorithm is the sum of all the steps in each level

- *In the next slide, we will see this example as a tree*

RECURSION TREE METHOD



ANOTHER EXAMPLE

- $T(n) \leq 2T(n/2) + n^2$

- Without drawing a tree, let's do the same kind of analysis
  - Top level: $\boxed{n^2}$

  - Second level: $\boxed{2(\frac{n}{2})^2 = \frac{n^2}{2}}$
  - Third level: $\boxed{4(\frac{n}{4})^2 = \frac{n^2}{4}}$
- Levels are contributing terms in a geometric series with terms decreasing by half. What is the sum? $O(n^2)$

THREE MAIN CASES

- For recurrences of form:

$$T(n) \le aT\left(\frac{n}{b}\right) + n^c$$

Contribution from levels can be:

1. Decreasing geometric series — Solution: root level
2. Increasing geometric series — Solution: leaf level
3. Equal for all levels — Multiply by number of levels

- Neatly captured by Master Theorem

THREE MAIN CASES (CONTINUED)

Let us understand these cases a little more…

- Root level contributes n^c
- Second level?
  - …There are a nodes, each contributing $\boxed{(\frac{n}{b})^c}$
  - Total is $\boxed{\frac{a}{b^c} n^c}$
- $\frac{a}{b^c}$ is the ratio between successive levels

- The 3 cases correspond to this ratio being $< 1, > 1, = 1$ respectively.

NUMBER OF LEAVES

- If solution is contribution from leaf level:
  - Each leaf contributes 1 (if problem is small enough then we can solve it in constant time)

  - Number of leaves?
    - Number of levels: $\log_b n$
    - Number of nodes: multiplied by a at each level

    - Number of leaves: $a^{\log_b n} = n^{\log_b a}$

- Total contribution of leaves is $n^{\log_b a}$

MASTER THEOREM

- Theorem: the recurrence $T(n) \leq aT\left(\frac{n}{b}\right) + n^c$ has solution

$$T(n) = \begin{cases} O(n^c) & \text{if } a < b^c \\ O(n^{\log_b a}) & \text{if } a > b^c \\ O(n^c \log n) & \text{if } a = b^c \end{cases}$$

- Example…

`mergesort` recurrence: $a = 2, b = 2, c = 1: a = b^c$

Solution: $T(n) = O(n \log n)$

MASTER THEOREM (CONTINUED)

- This is a very useful theorem for understanding the running time of standard divide and conquer algorithms

- We will see more examples in the next segment