CS 341 - Algorithms **Winter 2018**

Lecture 2: January 11, 2018

Lecturer: Bin Ma *Notes By: Harsh Mistry*

2.1 Recursion

Example 2.1 *Consider Merge Sort*

- *If* $n \leq 3$ *sort* A *with trivial algorithm and return*
- *Merge-sort(*$A[1..n/2]$*)*
- *Merger-sort(*$A[n/2+1..n]$*)*
- $A \leftarrow$ *Merge(*$A[1..n/2], A[n/2+1..n]$*)*

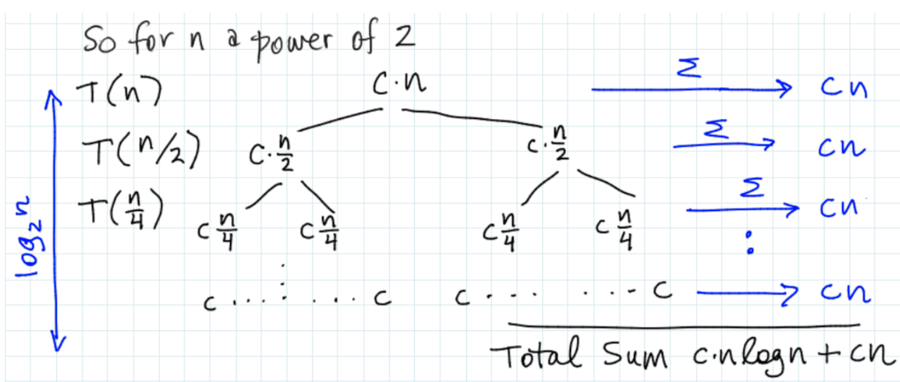
Time complexity *: The merge takes* $O(n)$ *time. So,* $T(n) = 2T(\frac{n}{2}) + cn$*. This uses a recurrence relation to define* $T(n)$*. But we prefer a closed-form simple function for the time complexity. So, we could unroll*

$$T(n) = 2T(\frac{n}{2}) + cn = 4T(\frac{n}{4}) + 2cn = \ldots = 2^k T(2^{-k}n) + ckn$$

Therefore, when $k = \log n$, $T(n) = 2^{\log n} T(1) + cn \log n = O(nlogn)$

2.1.1 Solving recurrences

- Method 1 : By Unrolling (Refer to `Example 2.1`)
- Method 2 : Use a recurrence tree



- Method 3 : By guess and Verify :
  Guess $T(n) \leq cn \log_2 n$. Prove by induction. For n = 1, trivial.
  Assume it is true for $n < m$. Now we prove its true for $n = m$

$$T(n) \leq 2T(\frac{n}{2}) + cn \leq 2c \cdot \frac{n}{2} \cdot \log_2(\frac{n}{2}) + cn = cn(\log_2 n)$$

- Method 4 : Master Theorem
  Let $a > 1, b > 2, c \geq 0$ be constants. Let $T(n)$ be defined on nonnegative integers by recurrence

  $$T(n) = a \cdot T(\frac{n}{b}) + n^c$$

  Then :

  $$T(n) = \begin{cases} \Theta(n^c), \text{if } c > \log_b a \\ \Theta(n^c \cdot \log n), \text{ if } c = \log_b a \\ \Theta(n^{\log_b a}), \text{if } c < \log_b a \end{cases}$$

  Note that the three cases correspond to the fix-up step dominates; balance; and small problems dominate, respectively. Roughly speaking, you want to keep and small but large.

  Sketch proof of case $c > \log_b a$. We prove $T(n) \leq \gamma \cdot n^c$ for some constant γ to be determined later. By induction

  $$T(n) = a \cdot T(\frac{n}{b}) + n^c \leq (a \cdot \gamma \cdot b^{-c} + 1) \cdot n^c$$

  We only need to prove there is a γ such that $a \cdot \gamma \cdot b^{-c} + 1 = \gamma$. Let $\gamma = \frac{1}{1 - a \cdot b^{-c}}$ QED.

  This process must be repeated for each case