Divide and Conquer: Solving Recurrences

Solving Recurrences

1. Divide and conquer examples:
    - MergeSort
        - $T(n) = 2T(n/2) + O(n) = O(n\log(n))$
    - Integer Multiplication
        - $T(n) = 4T(n/2) + O(n) = O(n^2)$
    - Improved Integer Multiplication
        - $T(n) = 3T(n/2) + O(n) = O(n^{\log_2 3})$
    - Median
        - $T(n) = T(3n/4) + O(n) = O(n)$

Example 1

1. $T(n) = 4T(n/2) + O(n)$
    - For some constant $c > 0$, $T(n) = 4T(n/2) + cn$, $T(1) <= c$
        - $T(n) <= cn + 4T(n/2)$
        - $T(n) <= cn + 4[4T(n/4) + cn/2]$
        - $T(n) <= cn(1 + 4/2) + 4^2T(n/4)$
        - $T(n) <= cn(1 + 4/2) + 4^2[4T(n/8) + cn/4]$
        - $T(n) <= cn(1 + 4/2 + (4/2)^2) + 4^3T(n/2^3)$

Expanding Out

1. $T(n) <= cn(1 + 4/2 + (4/2)^2) + 4^3T(n/2^3)$
    - $T(n) <= cn(1 + (4/2) + (4/2)^2 + \ldots + (4/2)^{i-1}) + 4^iT(n/2^i)$
        - let $i = \log_2 n$ then $n/2^i = 1$
    - $T(n) <= cn(1 + (4/2) + (4/2)^2 + \ldots + (4/2)^{\log_2 n-1}) + 4^{\log_2 n}T(1)$
        - $cn = O(n)$
        - $((4/2)^{\log_2 n}) = O(n^2/n) = O(n)$
        - $4^{\log_2 n} = O(n^2)$
    - Total is $O(n^2)$

Geometric Series

For constant $\alpha > 0$,

$$\sum_{j=0}^{k} \alpha^j = 1 + \alpha + \alpha^2 + \cdots + \alpha^k = \begin{cases} O(\alpha^k) & \text{if } \alpha > 1 \\ O(k) & \text{if } \alpha = 1 \\ O(1) & \text{if } \alpha < 1 \end{cases}$$

Geometric Series

Manipulating Polynomials

1. $4^{\log_2 n} = n^2$
2. $3^{\log_2 n} = n^c$
    - $3 = 2^{\log_2 3}$
    - $3^{\log_2 n} = (2^{\log_2 3})^{\log_2 n} = 2^{\log_2 3 * \log_2 n}$
    - $2^{\log_2 3 * \log_2 n} = (2^{\log_2 n})^{\log_2 3} = n^{\log_2 3}$
    - $c = \log_2 3$

Example 2

1. $T(n) = 3T(n/2) + O(n)$
    - $T(n) <= cn + 3T(n/2)$
    - $T(n) <= cn(1 + (3/2) + (3/2)^2 + \ldots + (3/2)^{i-1}) + 3^i T(n/2^i)$
        - let $i = \log_2 n$
    - $T(n) <= cn(1 + (3/2) + (3/2)^2 + \ldots + (3/2)^{\log_2 n-1}) + 3^{\log_2 n} T(1)$
        - $cn = O(n)$
        - $(1 + (3/2) + (3/2)^2 + \ldots + (3/2)^{\log_2 n-1}) = O((3/2)^{\log_2 n}) = O(3^{\log_2 n})$
        - $3^{\log_2 n} T(1) = O(3^{\log_2 n})$
    - Total is $O(n^{\log_2 3})$

General Recurrence

1. Constants $a > 0$, $b > 1$
    - $T(n) = aT(n/b) + O(n)$
    - $T(n) = cn(1 + (a/b) + (a/b)^2 + \ldots + (a/b)^{\log_b n-1}) + a^{\log_b n} T(1)$
        - If $a > b$: $O(n^{\log_b a})$
        - If $a = b$: $O(n\log(n))$
        - If $a < b$: $O(n)$
    - This is how the Master Theorem is derived