Algorithms-Design and Analysis(Stanford) Notes

zxm

目录

1 Divide and Conquer 分而治之

1. **DIVIDE** into smaller sub-problems
2. **CONQUER** via recursive calls
3. **COMBINE** solutions of sub-problems into one for the original problem

1.1 Master Method

- Cool feature: a "black-box" method for solving recurrences
- Determine the upper bound of **running time** for most of the D&C algos
- **Assumption**: all sub-problems have equal size
    - unbalanced sub-problems?
    - more than one recurrence?
- **Recurrence format**:
    - base case: $T(n) \leq C$ (a constant), for all sufficiently small n
    - for all larger n, $T(n) \leq aT(\frac{n}{b}) + O(n^d)$
        * a: # of recurrence calls (e.g., # of sub-problems), $a \geq 1$
        * b: input size shrinkage factor, $b > 1$, $T(\frac{n}{b})$ is the time required to solve each sub-problem
        * d: exponent in running time of the combine step, $d \geq 0$
        * constants a, b, d independent of n
- **Three Cases**:
    - Case 1: base of logarithm does not matter
    - Case 3: base of logarithm matters

$$T(n) = \begin{cases} O(n^d \log n), & a = b^d \text{ (case 1)} \\ O(n^d), & a < b^d \text{ (case 2)} \\ O(n^{\log_b a}), & \text{otherwise (case 3)} \end{cases}$$

If $T(n) = aT(\frac{n}{b}) + \Theta(n^d)$, then (with similar proof)

$$T(n) = \begin{cases} \Theta(n^d \log n), & a = b^d \text{ (case 1)} \\ \Theta(n^d), & a < b^d \text{ (case 2)} \\ \Theta(n^{\log_b a}), & \text{otherwise (case 3)} \end{cases}$$