中国科学院大学
University of Chinese Academy of Sciences

CS101

Algorithmic Thinking

Asymptotic Notations for Algorithmic Complexity

zxu@ict.ac.cn
zhangjialin@ict.ac.cn

Outline

- What is algorithmic thinking
  - Knuth's characterization
  - Sorting: problem and algorithms
  - Asymptotic notations
- Divide-and-conquer paradigm
- Other interesting paradigms
- P vs. NP

These slides acknowledge sources for additional data not cited in the textbook

1.3 Asymptotic notations

- Big O notation
  - Bubble-sort
    - Running time $O(n^2)$

Input: An array A of length n to be sorted, e.g., A=[6, 2, 4, 1, 5, 9].
Output: A sorted array A, e.g., A=[1, 2, 4, 5, 6, 9].
Steps:

```
for i = 1 to n-1            // for each round
    for j = 1 to n-i        // compare every adjacent pair
        if A [j] > A [j + 1] then exchange A [j] with A [j + 1];
```

  - Quick-sort
    - Average running time $O(n \log n)$
    - Is $o(n^2)$

```
p, r = n, 1
QuickSort(A, p, r)
  If p < r
    1. q = Partition(A, p, r)
    2. QuickSort(A, p, q-1)
    3. QuickSort(A, q+1, r)
```

Big-O notation simplifies analysis

- How many basic steps (operations) does the bubble sort algorithm need?
- Bubble sort terminates within $\sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2}$ steps
  - outer loop executes $n-1$ iterations
  - inner loop has $n-i$ comparisons and at most $n-i$ swaps
- Ignore constants and lower-order terms
  - Bubble sort terminates within $O(n^2)$ steps, or needs $O(n^2)$ operations
  - Much simpler: $O(n^2)$ vs. $\frac{n^2}{2} - \frac{n}{2}$

Input: An array A of length n to be sorted, e.g., A=[6, 2, 4, 1, 5, 9].
Output: A sorted array A, e.g., A=[1, 2, 4, 5, 6, 9].
Steps:

```
for i = 1 to n-1            // for each round
    for j = 1 to n-i        // compare every adjacent pair
        if A [j] > A [j + 1] then exchange A [j] with A [j + 1];
```

Little-o and big-O notations

- $f(n) = o(g(n))$: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
  - $n^{1.58} = o(n^2)$
  - $n^{1000} = o(2^n)$
  - $(\log n)^{200} = o(n)$
- $f(n) = O(g(n))$: exist constant $c > 0$, for sufficiently large n, we have $f(n) \leq cg(n)$
  - $n^{1.58} = O(n^2)$
  - $10n^{1.58} = O(n^{1.58})$
  - $10^{1000}n = O(n)$

$\Omega(\cdot), \Theta(\cdot)$ notations

- $f(\mathrm{n}) = \Omega(\mathrm{g}(\mathrm{n}))$: exist constant $c > 0$, for sufficiently large n, we have $f(n) \geq cg(n)$
  - $n^2 = \Omega(n^{1.58})$
- $f(n) = \Theta(g(n))$
  - $f(\mathrm{n}) = O\big(g(\mathrm{n})\big)$ and $f(\mathrm{n}) = \Omega(\mathrm{g}(\mathrm{n}))$:
  - $10n^2 - 20n + 45 = \Theta\big(n^2\big)$
- They are all one-way equality
  - Not mathematical equality
- Thinking problem:
  - Are $2^{\Theta(n)}$ and $\Theta(2^n)$ the same thing?

Example: O(.) and Ω(.) for Fibonacci computing

- How much time (how many operations) for F(n)?

```
func fibonacci(n int) int {
        if n == 0 || n == 1 {
                return n
        }
        return fibonacci(n-1) + fibonacci(n-2)
}
```

- From the code, we have
  - T(n) = T(n-1) + T(n-2); T(1)=T(0)=1
    - 2*T(n-2) < T(n) < 2*T(n-1), when n > 2
  - T(n) < 2*T(n-1) < 4*T(n-2) < 8*T(n-3) … < 2^n
  - T(n) > 2*T(n-2) > 4*T(n-4) > 8*T(n-6) … > $2^{n/2}$
  - **Upper bound: T(n) = O(2^n)**
  - **Lower bound: T(n) = Ω($2^{n/2}$)**

Put it together: o, O, Ω, Θ at a glance

Common assumption: when n is sufficiently large

- $f(n) = o(g(n))$, strictly upper bound
  - $n^{1.58} = \mathbf{o}(n^2)$, $n^{1.58} \neq \mathbf{o}(n^{1.58})$, $n^2 \neq \mathbf{o}(n^{1.58})$
- $f(n) = O(g(n))$, upper bound
  - $n^{1.58} = \mathbf{O}(n^2)$, $n^{1.58} = \mathbf{O}(n^{1.58})$, $n^2 \neq \mathbf{O}(n^{1.58})$
- $f(n) = \Omega(g(n))$, lower bound
  - $n^{1.58} \neq \mathbf{\Omega}(n^2)$, $n^{1.58} = \mathbf{\Omega}(n^{1.58})$, $n^2 = \mathbf{\Omega}(n^{1.58})$
- $f(n) = \Theta(g(n))$, tight bound
  - $n^{1.58} \neq \mathbf{\Theta}(n^2)$, $n^{1.58} = \mathbf{\Theta}(n^{1.58})$, $n^2 \neq \mathbf{\Theta}(n^{1.58})$

Put it together: o, O, Ω, Θ at a glance

Common assumption: when n is sufficiently large

- $f(n) = o(g(n))$, strictly upper bound
  - $n^{1.58} = \mathbf{o}(n^2)$, $n^{1.58}$ **??** $\mathbf{o}(n^{1.58})$, $n^2 \neq \mathbf{o}(n^{1.58})$
- $f(n) = O(g(n))$, upper bound
  - $n^{1.58}$ **??** $\mathbf{O}(n^2)$, $n^{1.58} = \mathbf{O}(n^{1.58})$, $n^2 \neq \mathbf{O}(n^{1.58})$
- $f(n) = \Omega(g(n))$, lower bound
  - $n^{1.58}$ **??** $\mathbf{\Omega}(n^2)$, $n^{1.58} = \mathbf{\Omega}(n^{1.58})$, $n^2 = \mathbf{\Omega}(n^{1.58})$
- $f(n) = \Theta(g(n))$, tight bound
  - $n^{1.58}$ **??** $\mathbf{\Theta}(n^2)$, $n^{1.58} = \mathbf{\Theta}(n^{1.58})$, $n^2 = \mathbf{\Theta}(n^{1.58})$