Announcements

C++ Bootcamp next week
Watch all videos on edX
PS #1 is out
Gradescope
Office Hours
Textbook

Previously ...

Analysis of Algorithms
running times (empirical analysis)
examples

Today ...

Analysis of Algorithms
running times (mathematical models)
big O notation
examples

Math Review

Summations
Powers
Logarithms
Proof Techniques
Basic Probability and Combinatorics

Time Complexity

Mathematical Model

High-level analysis — no need to implement
running time \( T(n) \) is a function of the input size \( n \)
Independent of HW/SW
Based on counts of elementary operations
additions, multiplications, comparisons, etc.
exact definition not important
must be ‘relevant’ to the problem

Growth Rate

Changes in HW/SW
affect \( T(n) \) by a constant factor
do not alter the growth rate of \( T(n) \)

Interest on execution times with large inputs

growth of \( T(n) \) as \( n \to \infty \)
### Order of Growth

- $T(n) = \Theta(1)$
- $T(n) = \Theta(\log n)$
- $T(n) = \Theta(n)$
- $T(n) = \Theta(n \log n)$
- $T(n) = \Theta(n^2)$
- $T(n) = \Theta(2^n)$

### Asymptotic Performance

For large values of $n$, an $O(n^2)$ algorithm always beats an $O(n^3)$ algorithm.

In practice, we shouldn’t completely ignore asymptotically slower algorithms.

### Big O Notation

$T(n) = O(f(n))$ if there are positive constants $c$ and $n_0$ such that $T(n) \leq c \cdot f(n)$ when $n \geq n_0$.

**Example 1**

$7n - 2$ is $O(n)$

Proof 1: A possible choice is $c = 7$ and $n_0 = 1$.

**Example 2**

$20n^3 + 10n \log n + 5$ is $O(n^3)$

Proof 2: $c = 35$ and $n_0 = 1$.

**Example 3**

$3 \log n + \log \log n$ is $O(\log n)$

Proof 3: $c = 4$ and $n_0 = 2$.

**Example 4**

$2^{100}$ is $O(1)$

Proof 4: $c = 2^{100}$ and $n_0 = 1$.

Tight bound is preferred.

### Big Omega

$T(n) = \Omega(f(n))$ if there are positive constants $c$ and $n_0$ such that $T(n) \geq c \cdot f(n)$ when $n \geq n_0$.

**Example 1**

$3 \log n + \log \log n$ is $\Omega(\log n)$

Proof 1: $3 \log n + \log \log n \geq 3 \log n$, for $n_0 = 2$.

**Example 2**

$3 \log n + \log \log n$ is $\Theta(\log n)$

Proof 2: $3 \log n + \log \log n$ is $\Omega(\log n)$ and $O(\log n)$

### Big Theta

$T(n) = \Theta(f(n))$ if and only if $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$.

**Example 1**

$3 \log n + \log \log n$ is $\Omega(\log n)$

Proof 1: $3 \log n + \log \log n \geq 3 \log n$, for $n_0 = 2$.

**Example 2**

$3 \log n + \log \log n$ is $\Theta(\log n)$

Proof 2: $3 \log n + \log \log n$ is $\Omega(\log n)$ and $O(\log n)$

### Examples

- $10^2 + 3000n + 10$
- $21 \log n$
- $500 \log n + n^4$
- $\sqrt{n} + \log n^{50}$
- $4^n + n^{5000}$
- $3000n^3 + 3n^{3.5}$
- $2^2 + n!$
for (int i = 0; i < n; i++) {
    // O(1) operations
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        // O(1) operations
    }
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n*n; j++) {
        // O(1) operations
    }
}

T(n) =

for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        // O(1) operations
    }
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        for (int k = 0; k < n; k++) {
            // O(1) operations
        }
    }
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < i*i; j++) {
        for (int k = 0; k < j; k++) {
            // O(1) operations
        }
    }
}

Case Study

Maximum Subarray Problem

Given an array of n integers, find the subarray A[j:k] that maximizes the sum

\[ s_{j,k} = a_j + a_{j+1} + \cdots + a_k = \sum_{i=j}^{k} a_i \]

Brute Force

\[ T(n) = O(n^3) \]

Using Prefix Sums

\[ T(n) = O(n^2) \]

Using Maximum Suffix Sums

\[ T(n) = O(n) \]