Foundations II: Data Structures and Algorithms

Instructor : Yusu Wang
Topic 1 : Introduction and Asymptotic notation
Course Information

- Course webpage
  
  http://www.cse.ohio-state.edu/~yusu/courses/2331

- Office hours
  
  - Tu/Th 9:30 am – 11:00 am

- Grading policy:
  
  - homework: 20%,
  - two midterms: 40%,
  - final: 40%
Exams

- Midterm 1
  - Sept. 28\textsuperscript{th} : 8 – 10:00pm, Location: JR 251

- Midterm 2
  - Nov. 2\textsuperscript{nd} : 8 – 10:00pm, Location: JR 251

- Final exam:
  - Dec. 11\textsuperscript{th} (Monday), 12:00pm--1:45pm, Room DL369

Mark you calendar.
Contact me within first two full weeks if any schedule conflict for any exam.
Notes

- All homework should be submitted before or in class on the due date. Late homework during the same day receives a 10% penalty. Late homework submitted the next day receives a 30% penalty.

- You may discuss homework with your classmates. It is very important that you write up your solutions individually, and acknowledge any collaboration / discussion with others.
More Information

- **Textbook**
  - Introduction to Algorithms
    - by Cormen, Leiserson, Rivest and Stein
      (third edition)

- **Others**
  - CSE 2331 course notes
    - By Dr. Rephael Wenger (SBX)
  - The Art of Computer Programming
    - By Donald Knuth
Introduction

- What is an algorithm?
  - Step by step strategy to solve problems
  - Should terminate
  - Should return correct answer for any input instance
This Course

- Various issues involved in designing good algorithms
  - What do we mean by “good”?  
    - Algorithm analysis, language used to measure performance  
  - How to design good algorithms  
    - Data structures, algorithm paradigms
  - Fundamental problems  
    - Graph traversal, shortest path etc.
Asymptotic Notation
Sorting problem

- Input: \( A = \langle a_1, a_2, \ldots, a_n \rangle \)
- Output: permutation of \( A \) that is sorted

Example:

- Input: \( \langle 3, 11, 6, 4, 2, 7, 9 \rangle \)
- Output: \( \langle 2, 3, 4, 6, 7, 9, 11 \rangle \)
Insertion Sort

3, 11, 6, 4, 2, 7, 9

3, 6, 11, 4, 2, 7, 9

3, 4, 6, 11, 2, 7, 9

2, 3, 4, 6, 11, 7, 9
Pseudo-code

InsertionSort($A, n$)

for $i = 1$ to $n-1$ do

\[ key = A[i+1] \]

\[ j = i \]

while $j > 0$ and $A[j] > key$ do


\[ j = j - 1 \]

\[ A[j+1] = key \]
Analysis

- Termination
- Correctness
  - beginning of for-loop: if $A[1..i]$ sorted, then
  - end of for-loop: $A[1..i+1]$ sorted.
  - when $i = n-1$, $A[1..i+1]$ is sorted.

- Efficiency: time/space
  - Depends on input size $n$
  - Space: roughly $n$
Running Time

- Depends on input size
- Depends on particular input
  - Best case
  - Worst case

First Idea:
Provide upper bound (as a guarantee)
Make it a function of $n : T(n)$
Running Time

- **Worst case**
  - \( T(n) = \text{max time of algorithm on any input of size } n \)

- **Average case**
  - \( T(n) = \text{expected time of algorithm over all possible inputs of size } n \)
  - Need a statistical distribution model for input
    - E.g, uniform distribution
Insertion Sort

InsertionSort\((A, n)\)

for \(i = 1\) to \(n-1\) do

\[\text{key} = A[i]\]
\[j = i\]

while \(j > 0\) and \(A[j] > \text{key}\) do

\[j = j - 1\]

\[A[j+1] = \text{key}\]

\[T(n) = \sum_{i=2}^{n} (c_1 + c_2 i) = c_3 n^2 + c_4 n + c_5\]
Asymptotic Complexity

- Why shall we care about constants and lower order terms?
- Should focus on the relative growth w.r.t. $n$
  - Especially when $n$ becomes large!
  - E.g: $0.2n^{1.5}$ v.s $500n$

Second Idea:
Ignore all constants and lower order terms
Only order of growth
--- Asymptotic time complexity
\[ f(n) = O(g(n)) \]
Big O Notation

- \( f(n) \in O(g(n)) \) if and only if \( \exists c > 0, \text{ and } n_0 \), so that
  \[
  0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0
  \]
- \( f(n) = O(g(n)) \) means \( f(n) \in O(g(n)) \) (i.e., at most)
  - We say that \( g(n) \) is an *asymptotic upper bound* for \( f(n) \).
- It also means: \[
\lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c
\]
- E.g, \( f(n) = O(n^2) \) if there exists \( c, n_0 > 0 \) such that \( f(n) \leq cn^2 \), for all \( n \geq n_0 \).
More Examples

- \( 5n^2 + 6n + 8 = O(n^3) \) ?
- \( 100n^2 - 1000n = O(n^2) \) ?
- \( \sqrt{6n^3 + 7n^2 + 3n} = O(n^{1.5}) \) ?
- \( 2^n = O(n^2) \) ?
- \( 3\lg n = O(n) \) ?
- \( 3^n = O(2^n) \) ?
- \( 3^n = O(2^{2n}) \)
- \( \log_3 n = O(\log_2 n) \) ?
- \( \frac{n}{\lg n} = O(n) \)
Omega $\Omega$ Notation

- $f(n) \in \Omega(g(n))$ if and only if $\exists c > 0$, and $n_0$, so that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.

- $f(n) = \Omega(g(n))$ means $f(n) \in \Omega(g(n))$ (i.e., at least).
  - We say $g(n)$ is an *asymptotic lower bound* of $f(n)$.

- It also means:
  \[
  \lim_{n \to \infty} \frac{f(n)}{g(n)} \geq c
  \]
Illustration

\[ f(n) = \Omega(g(n)) \]
More Examples

- $5n^2 + 6n + 8 = \Omega(n^3)$?
- $5n^2 + 6n + 8 = \Omega(n^2)$?
- $\sqrt{6n^3 + 7n^2 + 3n} = \Omega(n^{1.5})$?
- $2^n = \Omega(n^2)$?
- $3\log n = \Omega(n)$?
- $3^n = \Omega(2^n)$?
- $\log_3 n = \Omega(\log_2 n)$?
- $2^{\log_3 n} = \Omega(2^{\log_2 n})$
Theta $\Theta$ Notation

- Combine lower and upper bound
- Means tight: of the same order
- $f(n) \in \Theta(g(n))$ if and only if $\exists c_1, c_2 > 0$, and $n_0$, such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for any $n \geq n_0$
- $f(n) = \Theta(g(n))$ means $f(n) \in \Theta(g(n))$
  - We say $g(n)$ is an asymptotically tight bound for $f(n)$.
- It also means:
  - $c_1 \leq \lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c_2$
\[ f(n) = O(g(n)) \quad f(n) = \Omega(g(n)) \quad f(n) = \Theta(g(n)) \]
Remarks -- 1

- \( f(n) = \Theta(g(n)) \) if and only if \( f(n) = O(g(n)) \), and \( f(n) = \Omega(g(n)) \).
  - Insertion sort algorithm as described earlier has time complexity \( \Theta(n^2) \).

- Transpose symmetry:
  - If \( f(n) = O(g(n)) \), if and only if \( g(n) = \Omega(f(n)) \)

- Transitivity:
  - If \( f(n) = O(g(n)) \) and \( g(n) = O(h(n)) \), then \( f(n) = O(h(n)) \).
  - Same for \( \Omega \) and \( \Theta \)
Remarks -- 2

- It is convenient to think that
  - Big-O is smaller than or equal to
  - Big-Ω is larger than or equal to
  - Big-Θ is equal

- However,
  - These relations are modulo constant factor scaling
  - Not every pair of functions have such relations

If \( f(n) \notin O(g(n)) \), this does not imply that \( f(n) = \Omega(g(n)) \).
Remarks -- 3

- Provide a unified language to measure the performance of algorithms
  - Give us intuitive idea how fast we shall expect the alg.
  - Can now compare various algorithms for the same problem

- Constants hidden!
  - $O(n \ lg \ lg \ n)$, $2n \ lg \ n$

- $T(n) = n^2 + O(n)$
  means $T(n) = n^2 + h(n)$, and $h(n) = O(n)$
Pitfalls

- \( O(n) + O(n) = O(n) \)  
  
- \( T(n) = n + \sum_{i=1}^{k} O(n) \)  
  
  \( = n + O(n) \)

\( k \) should not depend on \( n \)!

It can be an arbitrarily large constant …
Yet Another Notation

- Small \( o \) notation:
- \( f(n) = o(g(n)) \) means for any \( c > 0 \), \( \exists n_0 \) s.t. for all \( n \geq n_0 \), \( 0 \leq f(n) < cg(n) \). In other words,
  \[
  \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0
  \]

- Examples:
  - Is \( n - \lg n = o(n) \)?
  - Is \( n = o(n \lg n) \)?
Yet Another Notation

- **Small $\omega$ notation:**
  - $f(n) = \omega(g(n))$ means for any $c > 0$, $\exists n_0$ s.t. for all $n \geq n_0$, $0 \leq cg(n) < f(n)$. In other words,
  
  $$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \quad \text{or} \quad \lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$$

- **Examples:**
  - Is $n - \lg n = \omega(n)$?
  - Is $n = \omega(n / \lg n)$?
Review some basics
- Chapter 3.2 of CLRS
- See the board
Recall Asymptotic Notation

\[ f(n) \in O(g(n)) \] if there exists \( c, n_0 > 0 \) such that:
\[ f(n) \leq cg(n) \quad \text{for all } n \geq n_0. \]

\[ f(n) \in \Omega(g(n)) \] if there exists \( c, n_0 > 0 \) such that:
\[ f(n) \geq cg(n) \quad \text{for all } n \geq n_0. \]

\[ f(n) \in \Theta(g(n)) \] if there exists \( c_1, c_2, n_0 > 0 \) such that:
\[ c_1g(n) \leq f(n) \leq c_2g(n) \quad \text{for all } n \geq n_0. \]
Another View

\[ f(n) \in O(g(n)) \] if there exists \( c > 0 \) such that:

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c.
\]

\[ f(n) \in \Omega(g(n)) \] if there exists \( c > 0 \) such that:

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} \geq c.
\]

\[ f(n) \in \Theta(g(n)) \] if there exists \( c_1, c_2 > 0 \) such that:

\[
c_1 \leq \lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c_2.
\]
Some Special Cases

If \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \), then

\[ f(n) \in O(g(n)) \text{ but } f(n) \notin \Theta(g(n)). \]

If \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \), then

\[ f(n) \in \Omega(g(n)) \text{ but } f(n) \notin \Theta(g(n)). \]

If \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = c > 0 \ (c \neq \infty) \), then

\[ f(n) \in \Theta(g(n)). \]
More Examples

- $n^3$ vs. $(1.0001)^n$
- $n^3$ vs. $(1.0001)^{\frac{n}{2}}$
- $n^a$ vs. $b^n$
- $\ln(n^4)$ vs. $\log_3 2n$
- $2\lg n$ vs. $\sqrt{n}$
- $\lg n$ vs. $n^\epsilon$
- $n \lg n$ vs. $n \sqrt{n \lg n}$
- $n$ vs. $\log_2 3^n$
- $n$ vs. $2^{\log_2 n + 4}$
- $n \log_2 n$ vs. $2^{2 \log_2 n}$
More Examples

- \( \lg n^2 \) vs. \( (\lg n)^2 - 100 \lg n \)
- \( 10^{10} \) vs. \( \ln n \)
- \( 3^{\log_2 n} \) vs. \( 0.5n \)
- \( \lg \sqrt{2n^2 + n \lg n} \) vs. \( \lg (n^3 - 100n) \)
- \( n - \sqrt{n \lg n} \) vs. \( \sqrt{2n^2 + n \lg n} \)

In general, we can safely ignore low order terms.
Hierarchy

- $\Theta(n^n)$
- $\Theta(3^n)$
- $\Theta(2^n)$
- $\Theta(n^3)$
- $\Theta(n^2)$
- $\Theta(n \log(n))$
- $\Theta(n)$
- $\Theta(n^{0.5})$
- $\Theta(n^{0.1})$
- $\Theta((\log(n))^2)$
- $\Theta(\log(n))$
- $\Theta(1)$
More Example

Rank the following functions by order of growth:

- $2^{n-4}$, $(n + 3) \log(3n^2 - 4n)$, $n\sqrt{n} + 5$, $n \ln n$, $3 \cdot 2^n$, $\frac{4n}{\log n}$, $\sqrt{n + \log n}$
Summary

- Algorithms are useful

- Example: sorting problem
  - Insertion sort

- Asymptotic complexity
  - Focus on the growth of complexity w.r.t input size