Discrete Mathematics

Algorithm

Shin Hong

Problems and Algorithms

- There are **key general problems** in many doamins that ask for output with specific properties for a given valid input

- Algorithm as **generalized solution**
  - state precisely the general problem by specifying the input and the desired output, using the appropriate structures
  - specify the steps of a procedure that takes a valid input and produces the desired output.

Algorithms

- An *algorithm* is a finite sequence of precise instructions for deriving solutions of a generalized problem
- Ex. Describe an algorithm for finding the maximum value in a finite sequence of integers.

Solution: perform the following steps:

1. Set the temporary maximum equal to the first integer in the sequence.
2. Compare the next integer in the sequence to the temporary maximum.
   - If it is larger than the temporary maximum, set the temporary maximum equal to this integer.
3. Repeat the previous step if there are more integers. If not, stop.
4. When the algorithm terminates, the temporary maximum is the largest integer in the sequence.

Abu Ja'far Mohammed Ibin Musa Al-Khowarizmi (780-850)

Specifying Algorithms in Pseudocode

- Pseudocode is an intermediate representation between natural language and code using a specific programming language
- The form of pseudocode we use is specified in Appendix 3
  - Similar with C++ and Java.
- Programmers can use the description of an algorithm in pseudocode to construct a program in a particular language
- Pseudocode helps us analyze the time required to solve a problem using an algorithm, independent of the actual programming language used to implement algorithm

Properties of Algorithms

- **Input**: An algorithm has input values from a specified set.
- **Output**: From the input values, the algorithm produces the output values from a specified set. The output values are the solution.
- **Effectiveness**: It must be possible to perform each step of the algorithm correctly and in a finite amount of time.
- **Correctness**: An algorithm should produce the correct output values for each set of input values.
- **Finiteness**: An algorithm should produce the output after a finite number of steps for any input.
- **Generality**: The algorithm should work for all problems of the desired form.

Ex. Finding the Maximum Element in a Finite Sequence

- The algorithm in pseudocode:

```
procedure max(a_1, a_2, …., a_n: integers)
    max := a_1
    for i := 2 to n
            if max < a_i then max := a_i

    return max  {max is the largest element}
```

- Does this algorithm have all the properties?

Efficiency of Algorithm

- What would be a proper criteria to compare the efficiencies of two different algorithms for solving the same generalized problem?
  - Time performance: How long does an algorithm takes to find a solution?
  - Space performance: How much computer memory does this algorithm need to hold for solving a problem?

- Issues
  - Measurement
  - Comparison

Efficiency of Algorithm: Measurement

- Possible ways to measure computation time
  - wallclock time
  - the number of instruction executions

Efficiency of Algorithm: Comparison

- An algorithm executes different numbers of instructions depending on input
  - The number of instructions that an algorithm executes can be modeled as a function from the input domain to natural number
  - Which function is better than the other?

Efficiency of Algorithm: Asymptotic Complexity

- The notion of asymptotic complexity is to consider:
  - performance with larger input is more important
  - for inputs of the same size, worst case performance is more important

- For comparing asymptomtic complexity, the worst-case perofmrnace of an algorithm is modeled as a function over the size of input
  - This provides an upper bound on the number of operations an algorithm uses to solve a problem with input of a particular size

- We analyzes the growth of a worst-case performance functions with respect to input size to model the fundamental complexity of an algorithm
  - Use mathematical notations to compare function growths

Big-*O* Notation (1/3)

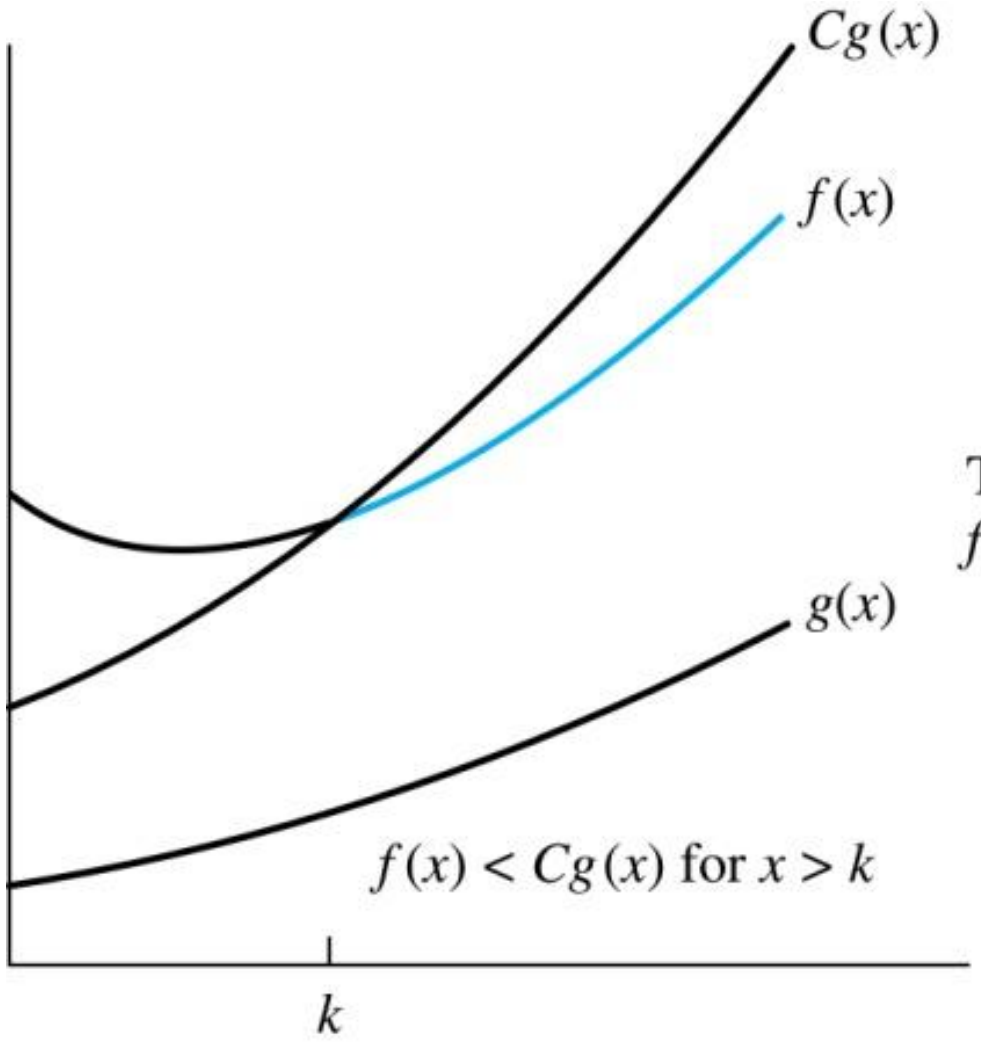
- Let *f* and *g* be functions from the set of real numbers to the set of real numbers. We say that *f*(*x*) is *O*(*g*(*x*)) if there are constants *C* and *k* such that

$$|f(x)| \leq C|g(x)|$$

whenever $x > k$. (illustration on next slide)

- This is read as "*f*(*x*) is big-*O* of *g*(*x*)" or "*g* asymptotically dominates *f*."
- The constants C and k are called *witnesses* to the relationship *f*(*x*) is *O*(*g*(*x*)). Only one pair of witnesses is needed.

Big-*O* Notation (2/3)



$f(x)$ is $O(g(x))$

The part of the graph of $f(x)$ that satisfies $f(x) < Cg(x)$ is shown in color.

Big-*O* Notation (3/3)

- If one pair of witnesses is found, then there are infinitely many pairs
  - We can always make the *k* or the *C* larger and still maintain the inequality $|f(x)| \leq C|g(x)|$
  - Any pair *C*' and *k*′ where *C* < *C*′ and *k* < *k*' is also a pair of witnesses since $|f(x)| \leq C|g(x) \leq C'|g(x)|$ whenever *x* > *k*′ > *k*.

Using the Definition of Big-*O* Notation

Example: Show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$

Using the Definition of Big-*O* Notation

Example: Show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$

Solution: Since when $x > 1$, $x < x^2$ and $1 < x^2$

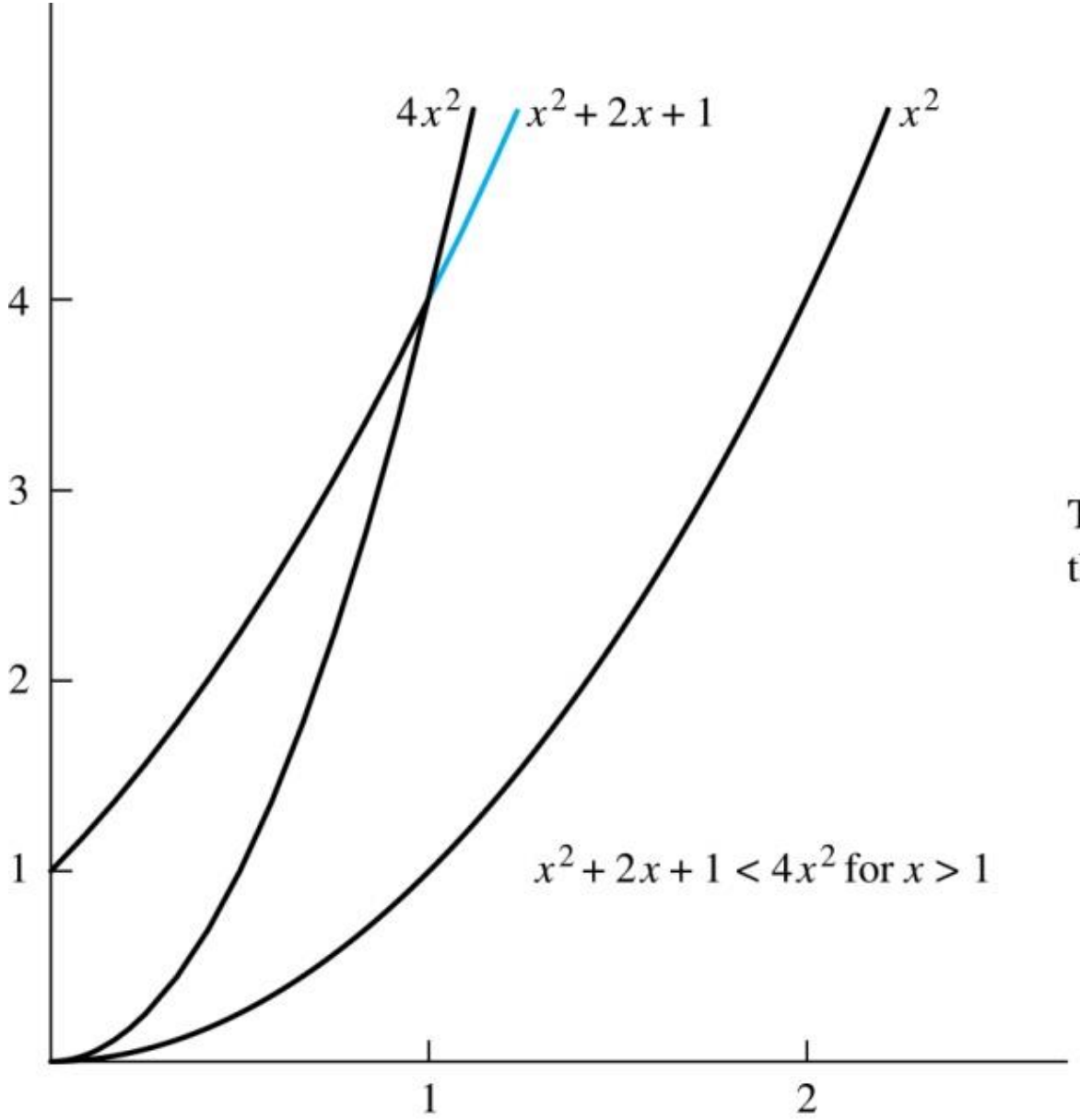
$$0 \leq x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$$

- Can take $C = 4$ and $k = 1$ as witnesses to show that $f(x)$ is $O(x^2)$

- Alternatively, when $x > 2$, we have $2x \leq x^2$ and $1 < x^2$. Hence, when $x > 2$.
  - Can take $C = 3$ and $k = 2$ as witnesses instead.

$$0 \leq x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2$$

Illustration of Big-*O* Notation



$$f(x) = x^2 + 2x + 1$$

is $O(x^2)$

The part of the graph of $f(x) = x^2 + 2x + 1$ that satisfies $f(x) < 4x^2$ is shown in blue.

Big-*O* Notation

- When both $f(x) = x^2 + 2x + 1$ and $g(x) = x^2$ are such that $f(x)$ is $O(g(x))$ and $g(x)$ is $O(f(x))$, two functions are of the same order
- If $f(x)$ is $O(g(x))$ and *h(x)* is larger than *g(x)* for all positive real numbers, $f(x)$ is $O\big(h(x)\big)$
- If $|f(x)| \leq C|g(x)|$ for *k* < *x* and if $|g(x)| < |h(x)|$ for all *x*, $|f(x)| \leq C|h(x)|$ if *k* < *x*. Hence, $f(x)$ is $O(h(x))$
- For many applications, the goal is to select the function g(*x*) in *O(g(x))* as small (tight) as possible (up to multiplication by a constant, of course)

Using the Definition of Big-*O* Notation

- **Example**: Show that $7x^2$ is $O(x^3)$.
- **Solution**: When $x > 7$, $7x^2 < x^3$. Take $C = 1$ and $k = 7$ as witnesses to establish that $7x^2$ is $O(x^3)$

- **Example**: Show that n^2 is not $O(n)$
- **Solution**: Suppose there are constants C and k for which $n^2 \leq Cn$, whenever $n > k$. Then (by dividing both sides of $n^2 \leq Cn$) by n, then $n \leq C$ must hold for all $n > k$. A contradiction!

Big-*O* Estimates for Polynomials

Example: Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_o$ where $a_0, a_1, \ldots, a_n$ are real numbers with $a_n \neq 0$. Then $f(x)$ is $O(x^n)$.

Proof: $|f(x)| = |a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_1|$

$\leq |a_n| x^n + |a_{n-1}| x^{n-1} + \cdots + |a_1| x^1 + |a_1|$

$= x^n (|a_n| + |a_{n-1}| /x + \cdots + |a_1| / x^{n-1} + |a_1| / x^n)$

$\leq x^n (|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_1|)$

- Take $C = |a_n| + |a_{n-1}| + \cdots + |a_1| + |a_1|$ and $k = 1$. Then $f(x)$ is $O(x^n)$.
- The leading term $a_n x^n$ of a polynomial dominates its growth.

Big-Theta Notation

- **Definition**: Let *f* and *g* be functions from the set of integers (or real numbers) to the set of real numbers.

$$f(x)\ is\ \Theta(g(x))\ \text{iff}\ f(x)\ is\ O(g(x))\ \text{and}\ g(x)\ is\ O(f(x))$$

- We say that "f is big-Theta of *g*(*x*)" and also that "*f*(*x*) is of *order g*(*x*)" and also that "*f*(*x*) and *g*(*x*) are of the *same order*."

- $f(x)\ \text{is}\ \Theta(g(x))$ if and only if there exists constants C_1 , C_2 and *k* such that $C_1 g(x) < f(x) < C_2\, g(x)$ if $k < x$

Big-Theta Notation

- **Example**: Show that $f(x) = 3x^2 + 8x \log x$ is $\Theta(x^2)$
- **Solution**:
  - $3x^2 + 8x \log x \leq 11x^2$ for $1 < x$, because $0 \leq 8x \log x \leq 8x^2$,
  - x^2 is clearly $O(3x^2 + 8x \log x)$, hence, $3x^2 + 8x \log x$ is $\Theta(x^2)$.

Complexity Analysis of Algorithms

- **Example**: Describe the time complexity of the algorithm for finding the maximum element in a finite sequence.

```
procedure max(a_1, a_2, …., a_n: integers)
    max := a_1
    for i := 2 to n
        if max < a_i then max := a_i
    return max  {max is the largest element}
```

Solution: Count the number of comparisons.

- The $max < a_i$ comparison is made $n - 2$ times.
- Each time i is incremented, a test is made to see if $i \leq n$.
- One last comparison determines that $i > n$.
- Exactly $2(n-1) + 1 = 2n - 1$ comparisons are made.

Hence, the time complexity of the algorithm is $\Theta(n)$.