Discrete Mathematics:
Lecture 9: complexity

complexity of algorithms

```
procedure max(a1, a2, . . . , an: integers)
max := a1
for i :=2 to n
     if max < ai then   max := ai
return max {max is the largest element}
```

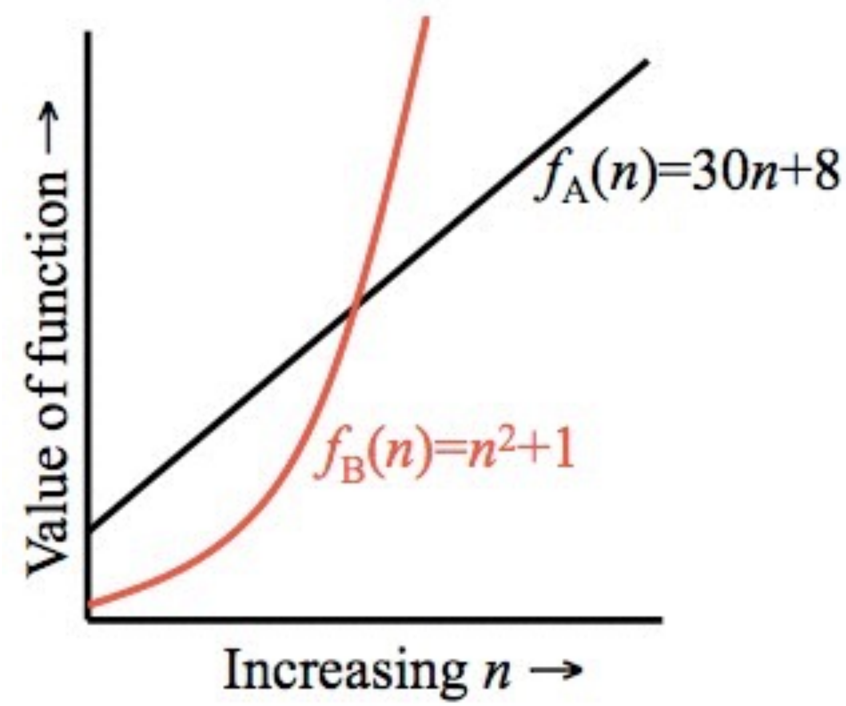
```
procedure matrix multiplication (A, B: matrices)
for i := 1 to m
    for j := 1 to n
         cij := 0
         for q := 1 to k
               cij := cij + aiqbqj
return C { C = [cij] is the product of A and B}
```

complexity of algorithms

we have two programs

c_1: $x^2 + 1$

c_2: $30x + 8$



big-O notation

$f(x)$ is $O(g(x))$ if there are constants C and k such that

$$|f(x)| \leq C\,|g(x)| \quad \text{whenever } x > k$$

- $f(x)$ grows slower than some fixed multiple of $g(x)$ as x grows without bound
- to show that $f(x)$ is $O(g(x))$, we need to find only one pair of constant C and k (witness) such that $|f(x)| \leq C\,|g(x)|$ whenever $x > k$
- if there is a pair of C and k, any pair C' and k', where $C < C'$ and $k < k'$, is also a pair of witness because $|f(x)| \leq C\,|g(x)| \leq C'\,|g(x)|$ whenever $x > k' > k$

big-O notation

show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$

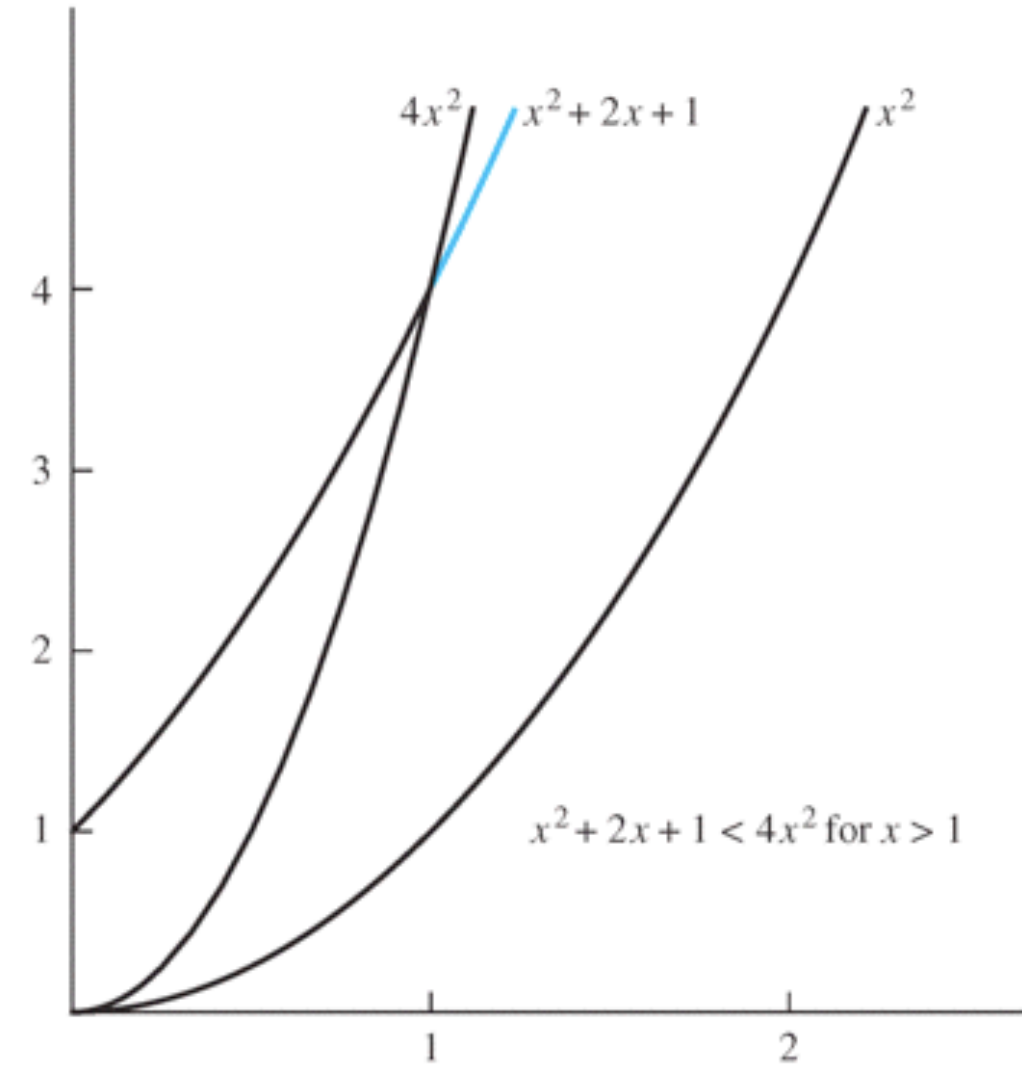
$|f(x)| \leq C\,|g(x)|$ whenever $x > k$

when $x > 1$, $x^2 > x$ and $x^2 > 1$

$x^2 + 2x + 1 < x^2 + 2x^2 + x^2 = 4x^2$

$|f(x)| \leq 4\,|x^2|$ whenever $x > 1$

thus, $f(x)$ is $O(x^2)$ when $C = 4, k = 1$



big-O notation

show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$

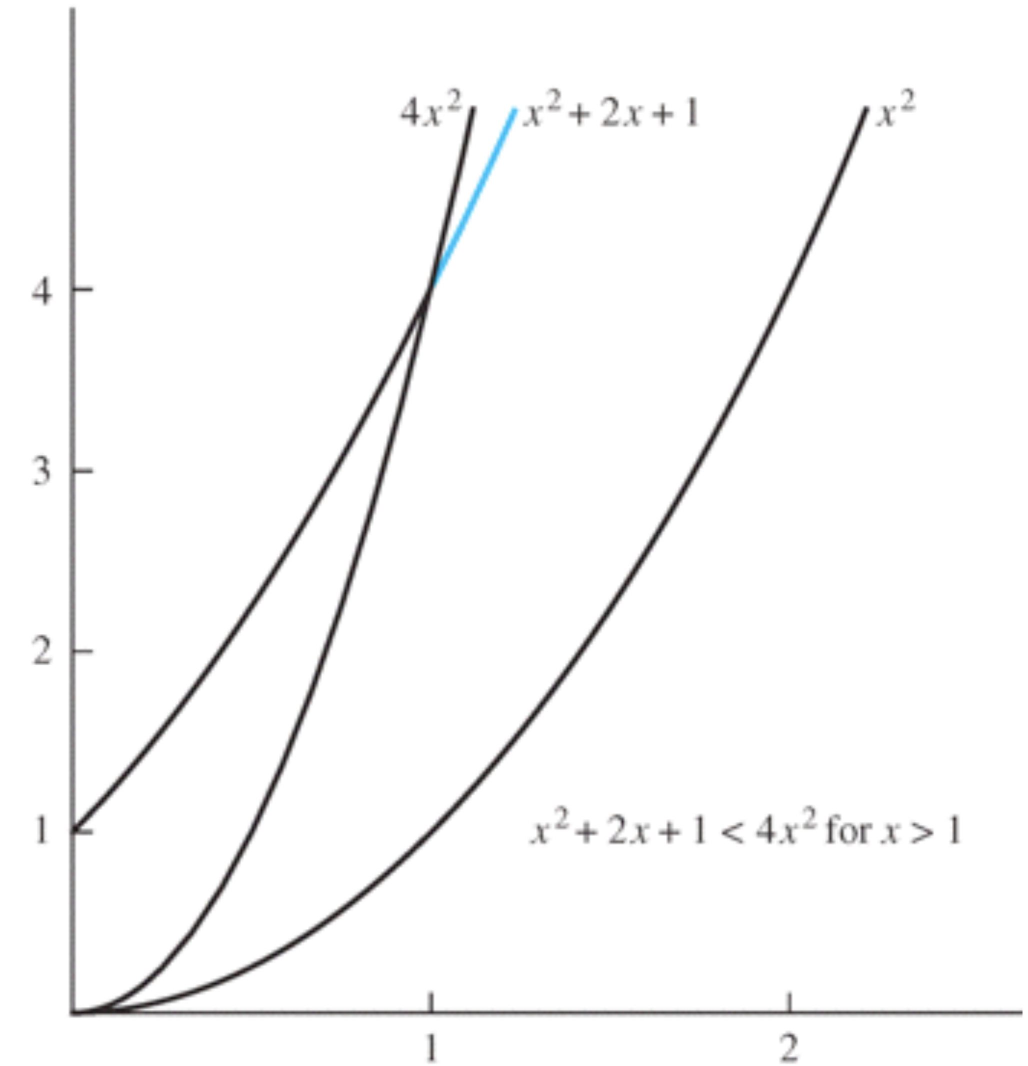
$|f(x)| \leq C\,|g(x)|$ whenever $x > k$

when $x > 2$, $x^2 > x$ and $x^2 > 1$

$x^2 + 2x + 1 < x^2 + 2x^2 + x^2 = 4x^2$

$|f(x)| \leq 4\,|x^2|$ whenever $x > 2$

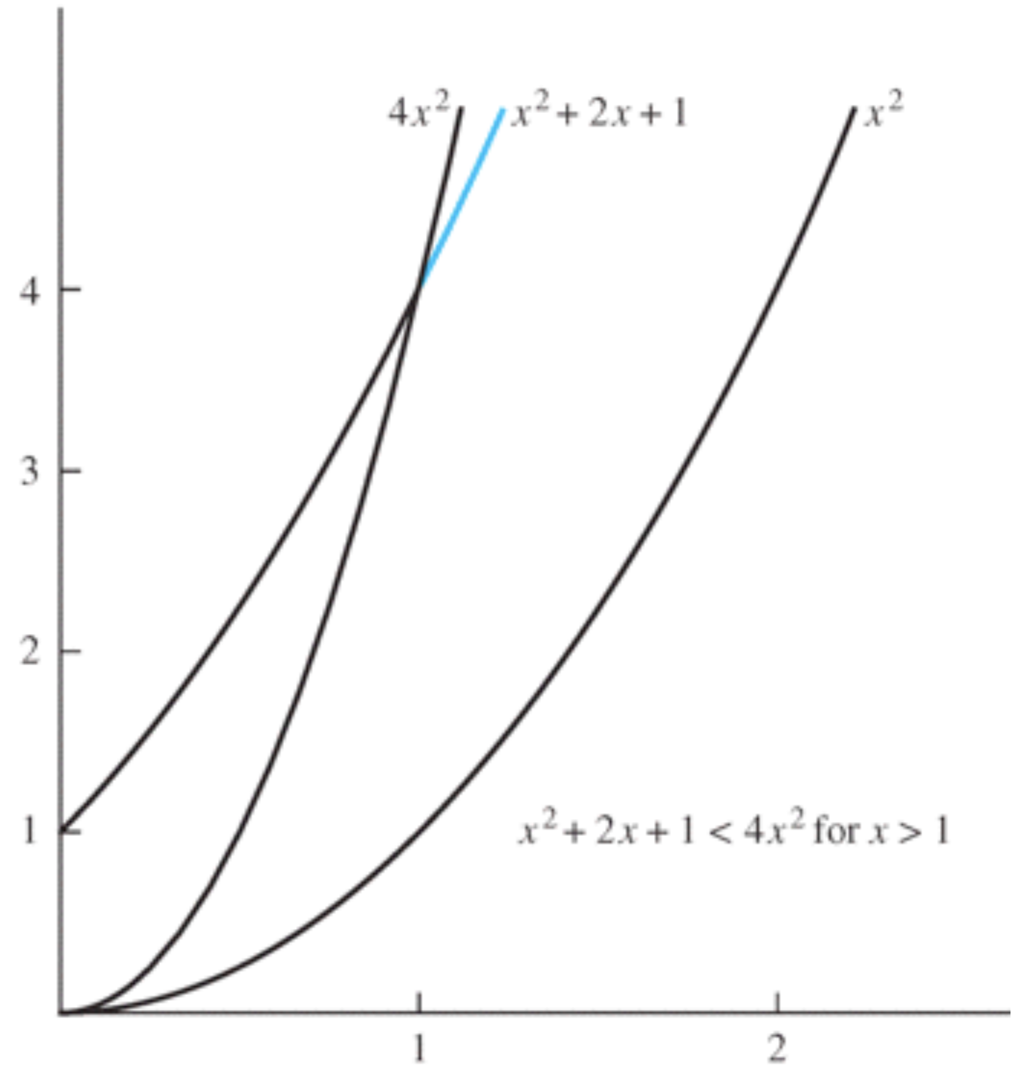
thus, $f(x)$ is $O(x^2)$ when $C = 4, k = 2$



big-O notation

show that $f(x) = x^2 + 2x + 1$ is $O(x^3)$

$|f(x)| \leq C\,|g(x)|$ whenever $x > k$



when $x > 1$, $x^3 > x^2$, $x^3 > x$ and $x^3 > 1$

$x^2 + 2x + 1 < x^3 + 2x^3 + x^3 = 4x^3$

$|f(x)| \leq 4\,|x^3|$ whenever $x > 1$

thus, $f(x)$ is $O(x^3)$ when $C = 4, k = 3$

big-O notation

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

$f(x) = x^2 + 2x + 1$ is $O(x^3)$

if $|f(x)| \leq C\,|g(x)|$ when $x > k$, and $|h(x)| > |g(x)|$ for all $x > k$,

$|f(x)| \leq C\,|h(x)|$ when $x > k$

big-O notation

show that $f(x) = x^2$ is not $O(x)$

$|f(x)| \leq C\,|g(x)|$ whenever $x > k$

use a proof by contradiction

suppose that there are a pair of C and k for which $x^2 \leq Cx$ whenever $x > k$

when $x > 0$, $x \leq C$

$x \leq C$ cannot hold for all x with $x > k$ since x can be arbitrarily large

big-O notation

let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$,

where $a_0, a_1, \ldots, a_{n-1}, a_n$ are real numbers.

then $f(x)$ is $O(x^n)$

$|f(x)| \leq C\,|x^n|$ whenever $x > k$

when $x > 1$

$$
\begin{aligned}
|f(x)| &= |a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0| \\
&\leq |a_n| x^n + |a_{n-1}| x^{n-1} + \ldots + |a_1| x + |a_0| \\
&= x^n(|a_n| + |a_{n-1}|/x + \ldots + |a_1|/x^{n-1} + |a_0|/x^n) \\
&\leq x^n(|a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|)
\end{aligned}
$$

$|f(x)| \leq Cx^n$, where $C = |a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|$ whenever $x > 1$

big-O notation

what is the big-O notation for estimating the sum of the first n positive integers

$1 + 2 + 3 + 4 + \ldots + n \leq n + n + \ldots + n = n^2$

$f(n) \leq n^2$

$f(n)$ is $O(n^2)$, $C = 1$ and $k = 1$

insertion sort

procedure insertion_sort ($a_1, a_2, \ldots, a_n$: real numbers, $n \geq 2$)

for $j := 2$ to n

    $i := 1$

    while ($a_j > a_i$)

        $i := i + 1$

    $m := a_j$

    for $k := 0$ to $j - i - 1$

        $a_{j-k} := a_{j-k-1}$

    ai := m

return {$a_1, \ldots a_n$ is in increasing order}

big-O notation

what is the big-O notation for factorial function and the logarithm of the factorial function

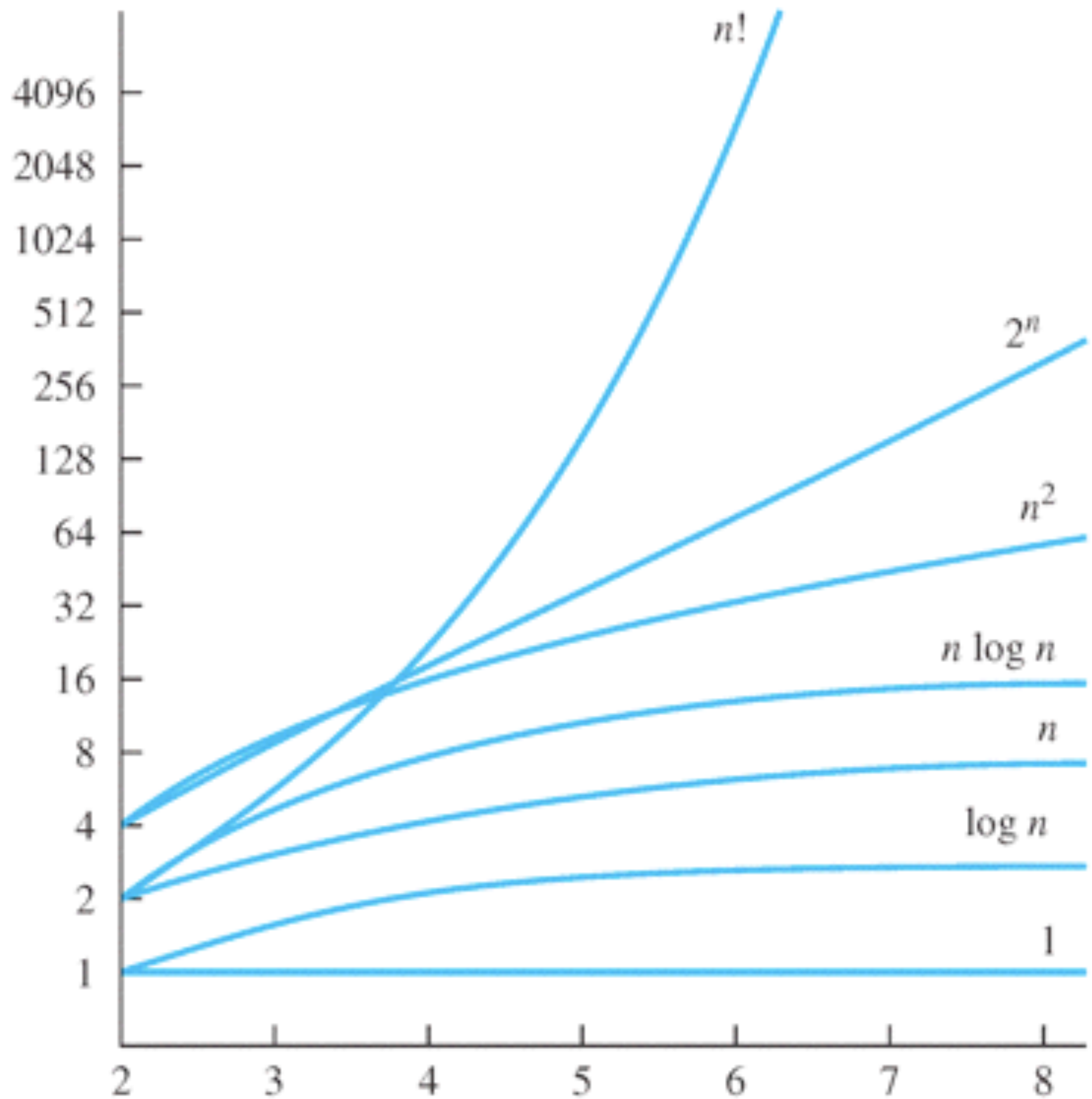
$n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \; ... \; \cdot n \leq n \cdot n \cdot ... \cdot n = n^n$

$n!$ is $O(n^n)$

$\log n! \;\; \leq \;\; \log n^n \; = n \log n$

$\log n!$ is $O(n \log n)$

big-O notation



1, log n, n, n log n, n^2, 2^n, n!

growth of combinations of functions

when $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$

$(f_1 + f_2)(x)$ is $O(\max(|g_1(x)|, |g_2(x)|))$

$$
\begin{aligned}
|(f_1 + f_2)(x)| &= |f_1(x) + f_2(x)| \\
&\le |f_1(x)| + |f_2(x)| \\
&\le C_1|g_1(x)| + C_2|g_2(x)| \\
&\le C_1|g(x)| + C_2|g(x)| && g(x) = \max(|g_1(x)|, |g_2(x)|) \\
&= (C_1 + C_2)\,|g(x)| \\
&= C\,|g(x)| && C = C_1 + C_2
\end{aligned}
$$

whenever $x > k$, $k = \max(k_1, k_2)$

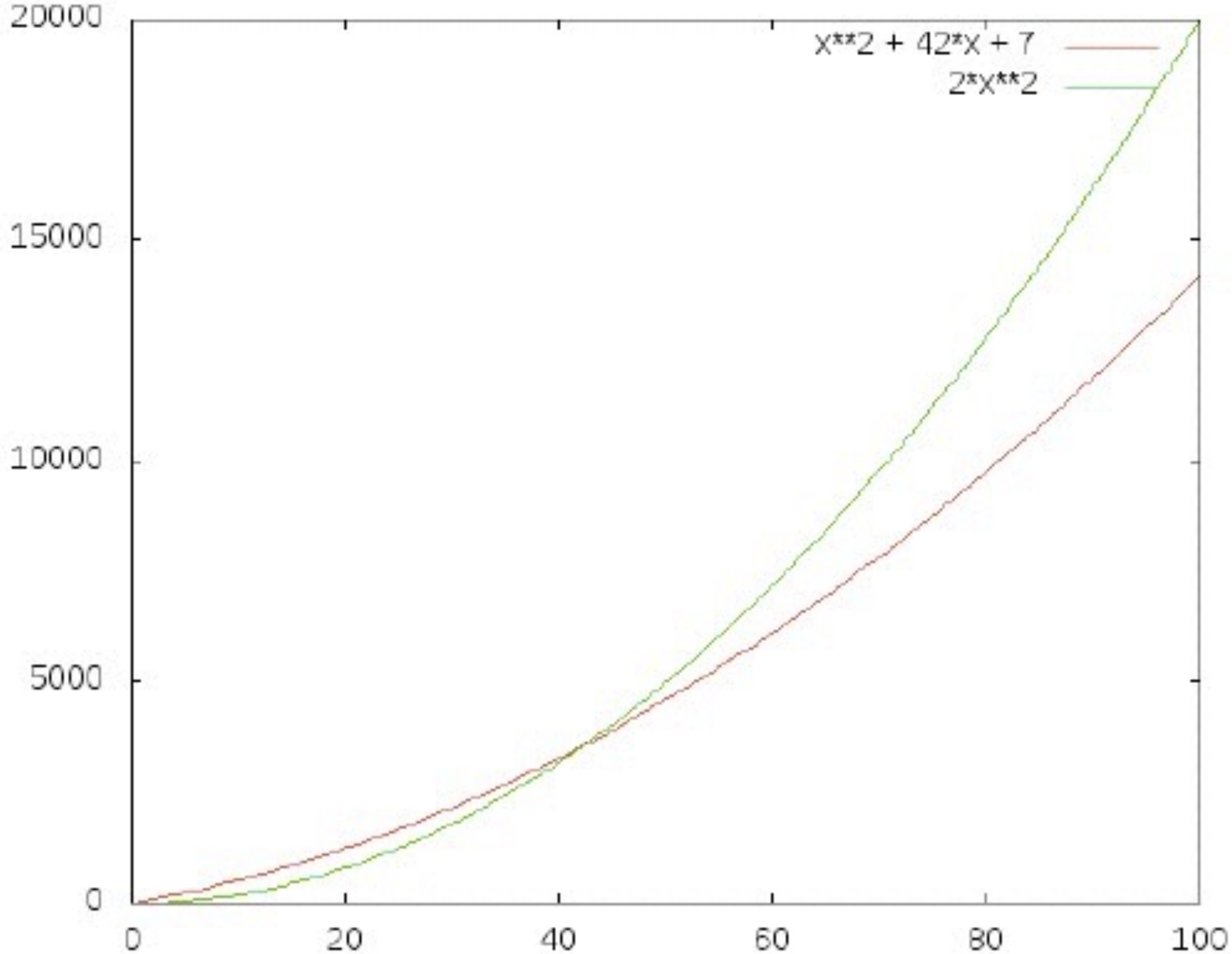
growth of combinations of functions

when $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$

$(f_1\ f_2)(x)$ is $O(g_1(x)\ g_2(x))$

$$
\begin{aligned}
|(f_1\ f_2)(x)| &= |f_1(x)||f_2(x)| \\
&\leq C_1|g_1(x)|\ C_2|g_2(x)| \\
&= C_1C_2|(g_1g_2)(x)| \\
&= C|(g_1g_2)(x)|, \qquad C = C_1C_2 \quad k = \max(k_1, k_2)
\end{aligned}
$$

growth of combinations of functions



$$n^2 + 42n + 7 \leq 2n^2 \text{ for all } n \geq 50$$

growth of combinations of functions

$f(n) = 3n \log(n!) + (n^2 + 3) \log n$, where n is a positive integer

$3n \log(n!) + (n^2 + 3) \log n$ is $O(n^2 \log n) + O(n^2 \log n)$, which is $O(n^2 \log n)$

$O(n)$ $O(n \log n)$

$(n^2 + 3) < 2n^2$ when $n > 2$

thus, $O(n^2)$

big-Ω (big-omega) notation

$f(x)$ is $\Omega(g(x))$ if there are constants C and k such that

$|f(x)| \geq C\,|g(x)|$ whenever $x > k$

big-Ω (big-omega) notation

$f(x) = 8x^3 + 5x^2 + 7$ is $\Omega(x^3)$

$|f(x)| \geq C\,|g(x)|$ whenever $x > k$

$f(x) = 8x^3 + 5x^2 + 7 \geq 8x^3$ for all positive real number x

big-Ω (big-omega) notation

$f(x) = 8x^3 + 5x^2 + 7$ is $\Omega(x^2)$

$|f(x)| \geq C\,|g(x)|$ whenever $x > k$

$f(x) = 8x^3 + 5x^2 + 7 \geq 5x^2$ for all positive real number x

big-Θ(big-theta) notation

$f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$

- $f(x)$ and $g(x)$ are of the same order
- when $f(x)$ is $\Theta(g(x))$, $g(x)$ is $\Theta(f(x))$
- $C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$

big-Θ(big-theta) notation

$f(x) = 3x^2 + 8x \log x$ is $\Theta(x^2)$

$0 \leq 8x \log x \leq 8x^2$

$3x^2 + 8x \log x \leq 3x^2 + 8x^2 = 11x^2$ for $x > 1$

since $3x^2 + 8x \log x$ is $O(x^2)$ and x^2 is $O(3x^2 + 8x \log x)$,

$3x^2 + 8x \log x$ is $\Theta(x^2)$

time complexity of an algorithm

the time complexity of an algorithm can be expressed in terms of the number of operations used by the algorithm

procedure max($a_1, a_2, \ldots, a_n$: integers)

max := a_1

for i :=2 to n $\longleftarrow$ $i \leq n$

max < a_i $\longrightarrow$ if max < a_i then max := a_i

return max {max is the largest element}

when the number of comparisons are used as the measure of the time complexity of the algorithm,

$2\,(n - 1) + 1 = 2n - 1$

thus, $\Theta(n)$

worst-case complexity

procedure **linear search**(x: integer, $a_1, a_2, \ldots, a_n$: integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ then location := i
else location := 0
return location {location is the subscript of the term that equals x, or 0 if x is not found}

$x = a_i$: $2i + 1$ comparisons ($2i$($i \leq n$ and $x \neq a_i$) + $i \leq n$)

x does not exist: $2n + 2$ comparisons ($2n$($i \leq n$ and $x \neq a_i$) + $i \leq n$ + $i \leq n$)

linear search requires $\Theta(x)$ comparisons in the worst case

average-case complexity

procedure **linear search**(x: integer, $a_1, a_2, \ldots, a_n$: integers)
i := 1
while ($i \leq n$ and $x \neq a_i$)
    i := i + 1
if $i \leq n$ then location := i
else location := 0
return location {location is the subscript of the term that equals x, or 0 if x is not found}

when assuming that x is in the list

$x = a_1$: 3 comparisons ($i \leq n$, $x \neq a_i$, $i \leq n$)

$x = a_2$: 5 comparisons

$i \leq n$: $2i + 1$ comparisons ($2i$($i \leq n$ and $x \neq a_i$) + $i \leq n$)

$(3 + 5 + \ldots + (2n+1))/n = (2(1 + 2 + 3 + \ldots + n) + n)/n$

$= (2 (n(n+1)/2) + n)/n = (n+1) + 1 = n + 2$, which is $\Theta(n)$ in average case

time complexity of matrix multiplication

$$
\begin{bmatrix}
a_{11} & a_{12} & \dots & a_{1k} \\
a_{21} & a_{22} & \dots & a_{2k} \\
\vdots & \vdots & & \vdots \\
a_{i1} & a_{i2} & \dots & a_{ik} \\
\vdots & \vdots & & \vdots \\
a_{m1} & a_{m2} & \dots & a_{mk}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} & \dots & b_{1j} & \dots & b_{1n} \\
b_{21} & b_{22} & \dots & b_{2j} & \dots & b_{2n} \\
\vdots & \vdots & & \vdots & & \vdots \\
b_{k1} & b_{k2} & \dots & b_{kj} & \dots & b_{kn}
\end{bmatrix}
=
\begin{bmatrix}
c_{11} & c_{12} & \dots & c_{1n} \\
c_{21} & c_{22} & \dots & c_{2n} \\
\vdots & \vdots & c_{ij} & \vdots \\
c_{m1} & c_{m2} & \dots & c_{mn}
\end{bmatrix}
$$

A: m x k matrix, B: k x n matrix, AB = [c_{ij}] (m x n matrix)

$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{ik}b_{kj}$

time complexity of matrix multiplication

A: m x k matrix, B: k x n matrix, C = AB: m x n matrix

procedure matrix multiplication (A, B: matrices)
for $i := 1$ to m
    for $j := 1$ to n
        $c_{ij} := a_{i1}b_{1j}$ $\qquad$ $c_{ij} := 0$?
        for $q := 2$ to k
            $c_{ij} := c_{ij} + a_{iq}b_{qj}$
return C { $C = [c_{ij}]$ is the product of A and B}

when A: n x n matrix, B: n x n matrix

n multiplications and n-1 additions for each entry

n^3 multiplications and $n^2(n-1)$ additions in total

complexity of algorithms

complexity	terminology
$\Theta(1)$	constant complexity
$\Theta(\log n)$	logarithmic complexity
$\Theta(n)$	linear complexity
$\Theta(n \log n)$	linearithmic complexity
$\Theta(n^b)$	polynomial complexity
$\Theta(b^n)$, where $b > 1$	exponential complexity
$\Theta(n!)$	factorial complexity

tractable vs. intractable

- a problem with at most polynomial time complexity is considered tractable.
- P is the set of all tractable problems
- a problem that has complexity greater than polynomial is considered intractable.
- NP is the set of problems for which there exists a tractable algorithm for checking a proposed solution to tell if it is correct