CS 2305: Discrete Mathematics for Computing I

Lecture 20

- KP Bhat

Factors Affecting the Running Time of a Program

1. The size of the input to the program
2. The time complexity of the algorithm underlying the program
3. The quality of code generated by the compiler used to create the object program
4. The nature and speed of the instructions on the machine used to execute the program

- Factors #1 and #2 are the dominant factors
- Factors #3 and #4 are, to a very large extent, independent of factors #1 and #2 and they can be approximated by some constant

Big-*O* Notation [1]

Definition: Let *f* and *g* be functions from the set of integers or 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

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

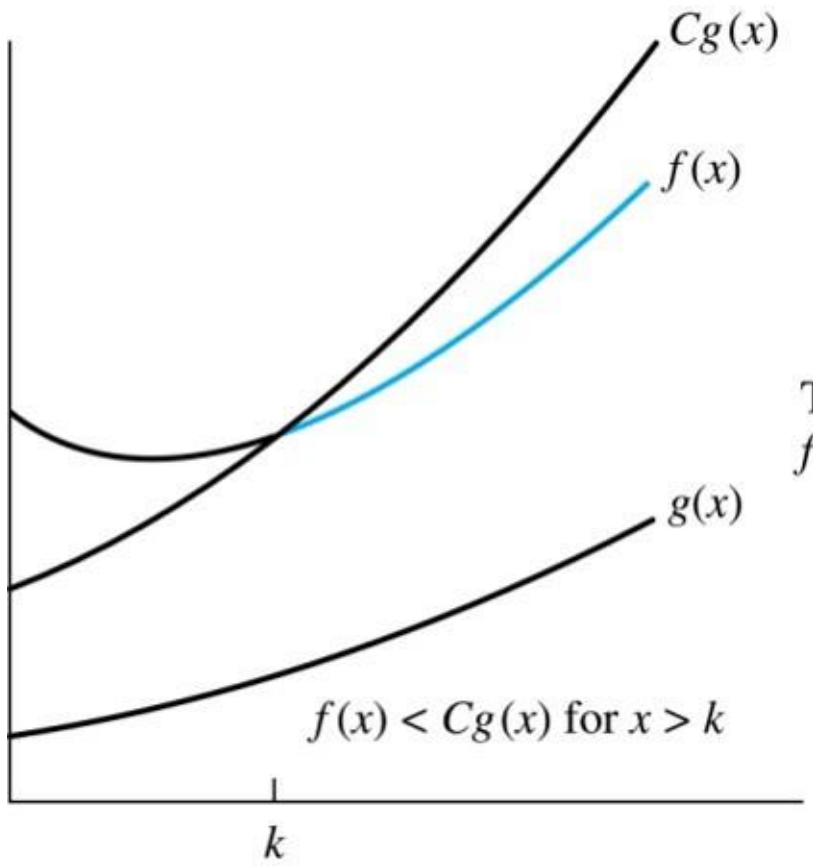
whenever $x > k$. (illustration on next slide)

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.

Note: *f* (*x*) is *O*(*g*(*x*)) says that *f* (*x*) grows slower than some fixed multiple of *g*(*x*) as *x* grows without bound.

Illustration of Big-*O* Notation[1]



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

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

Jump to long description

Some Important Points about Big-*O* Notation

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 $x > k' > k$.

Sometimes in the literature you may see "$f(x) = O(g(x))$" instead of "$f(x)$ is $O(g(x))$."

- Strictly speaking this is an abuse of the equals sign since this notation does not represent a genuine equality.
- It is ok to write $f(x) \in O(g(x))$, because $O(g(x))$ represents the set of functions that are $O(g(x))$.

Usually, we will drop the absolute value sign since we will mostly deal with functions that take on positive values.

Analogy for reasoning often used in Big-O problems

- Consider a weighing scale where on the one side you have an apple, a pear and a banana. On the other side you replace the apple by a bigger apple, the banana by a bigger banana and the pear by a bigger pear
- Obviously the first side of the scale will be lighter than the second side of the scale
- In our case we will look at an expression and replace smaller terms by larger terms. Obviously the original expression will be less than the resultant expression

Using the Definition of Big-*O* Notation [1]

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

.

Solution: Consider $x > 1$, $x < x^2$ and $1 < x^2$

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

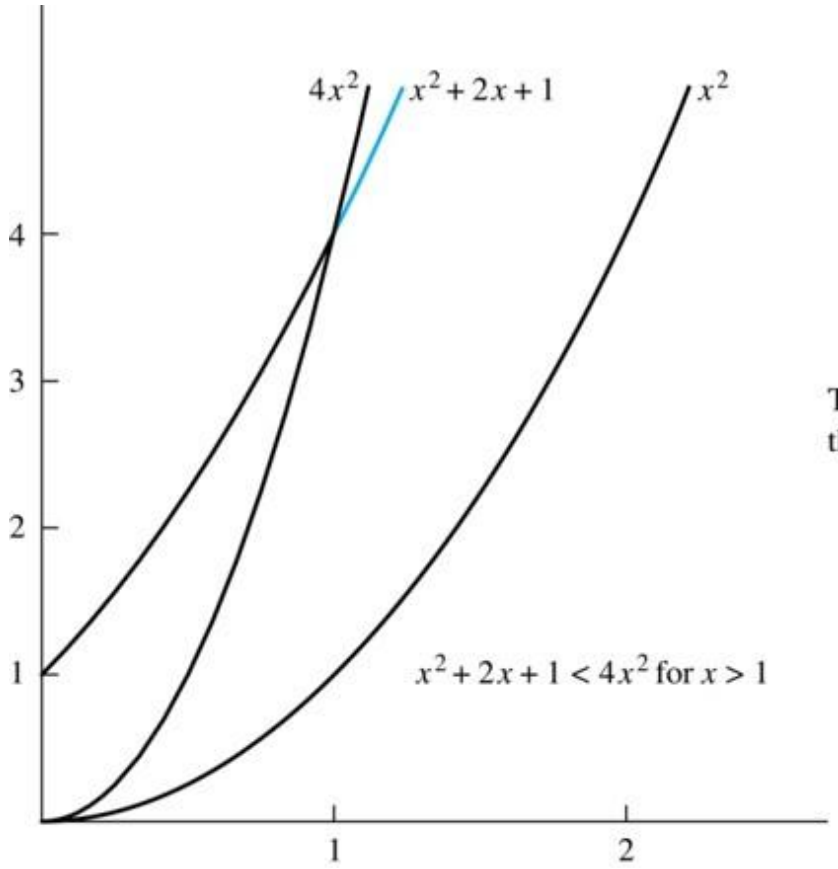
- Can take $C = 4$ and $k = 1$ as witnesses to show that (see graph on next slide)

Alternatively, when $x > 2$, we have $2x \leq x^2$ and $1 < x^2$. Hence, $x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2$ when $x > 2$.

- Can take $C = 3$ and $k = 2$ as witnesses instead.

Illustration of Big-*O* Notation 2

$$f(x) = x^2 + 2x + 1 \text{ is } \quad O(x^2)$$



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 2

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 $O(f(x))$.
We say that the 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, then $f(x)$ is $O(h(x))$.

Note that if $|f(x)| \leq C\,|g(x)|$ for $x > k$ and if $|h(x)| > |g(x)|$ for all x, then $|f(x)| \leq C\,|h(x)|$ if $x > k$. 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 as possible (up to multiplication by a constant, of course).

Using the Definition of Big-*O* Notation 2

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 $n > k$.

We can always choose C to be greater than k because if that is not so we can always choose another constant D > k. Now $\forall n > k$

$$n^2 \leq Cn$$

Then by dividing both sides by n, we have

$n \leq C$ must hold for all $n > k$.

A contradiction!

For $n > k$, $n \leq C$

$\therefore k \leq C$

But we chose C to be greater than k

Big-*O* Estimates for Polynomials 1

- A mathematical expression of the following type is known as a polynomial of degree n
  - $a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots\ldots + a_2 x^2 + a_1 x + a_0$
    - more accurately a polynomial in one variable of degree n
- In a polynomial of degree n, the leading term (i.e. $a_n x^n$) dominates its growth
- We will prove the result that a polynomial of degree n is $O(x^n)$
  - We will use the **triangle inequality** as the lemma for this proof
    - if x and y are real numbers, then $|x| + |y| \geq |x + y|$
    - Section 1.8, Exercise 9 (proof by cases)

Big-*O* Estimates for Polynomials 2

Theorem: Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

where $a_0, a_1, \ldots, a_n$ are real numbers with $a_n \neq 0$. Then $f(x)$ is $O(x^n)$.

Proof:

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

Uses triangle inequality, an exercise in Section 1.8.

Assuming $x > 1$

Take $C = |a_n| + |a_{n-1}| + \cdots + |a_0|$ and $k = 1$. Then $f(x)$ is $O(x^n)$.

The leading term $a_n x^n$ of a polynomial dominates its growth.

Big-*O* Estimates for some Important Functions[1]

Example: Use big-*O* notation to estimate the sum of the first *n* positive integers.

Solution: $1+2+\cdots+n \le n+n+\cdots n = n^2$

$1+2+\ldots+n$ is $O\left(n^2\right)$ taking $C=1$ and $k=1$.

Example: Use big-*O* notation to estimate the factorial function $f\left(n\right)=n!=1\times 2\times\cdots\times n$.

Solution:

$n!=1\times 2\times\cdots\times n \le n\times n\times\cdots\times n = n^n$

$n!$ is $O\left(n^n\right)$ taking $C=1$ and $k=1$.

Big-*O* Estimates for some Important Functions 2

Example: Use big-*O* notation to estimate log *n*!

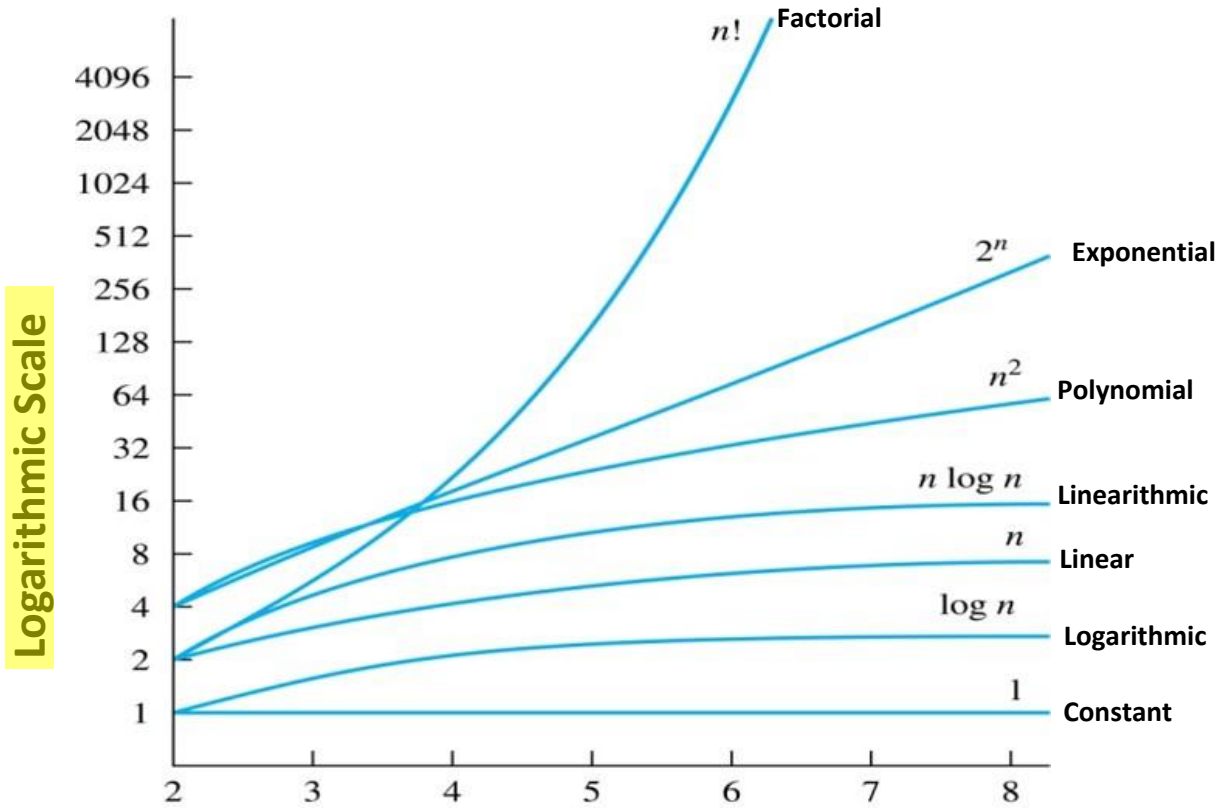
Solution: Given that $n! \leq n^n$ (previous slide)

then $log(n!) \leq log(n^n)$

$log(n!) \leq nlog(n)$

Hence, log(*n*!) is $O(n \cdot \log(n))$ taking $C = 1$ and $k = 1$.

Display of Growth of Functions



Note the difference in behavior of functions as *n* gets larger

Background Information: Sum and Product of Functions

- $(f_1 + f_2)(x) = f_1(x) + f_2(x)$
- $(f_1 f_2)(x) = f_1(x) * f_2(x)$

Examples: Let f_1 and f_2 be functions from **R** to **R** such that $f_1(x) = x^2$ and $f_2(x) = x - x^2$.

$(f_1 + f_2)(x) = f_1(x) + f_2(x) = x^2 + (x - x^2) = x$

$(f_1 f_2)(x) = f_1(x) * f_2(x) = x^2(x - x^2) = x^3 - x^4$

Combinations of Functions 1

- The number of steps used by a computer to solve a problem with input of a specified size, using an algorithm that is composed of subprocedures, is the sum of the number of steps used by the subprocedures
- The big-O estimate of the algorithm is the sum of the big-O estimates of the subprocedures of the algorithm

Combinations of Functions 2

If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$ then

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

- See next slide for proof

If $f_1(x)$ and $f_2(x)$ are both $O(g(x))$ then

$(f_1 + f_2)(x)$ is $O(g(x))$.

- See text for argument

If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$ then

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

- See text for argument

Combinations of Functions 3

If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$ then

$$(f_1 + f_2)(x) \text{ is } O(\max(|g_1(x)|, |g_2(x)|)).$$

- By the definition of big-*O* notation, there are constants C_1, C_2, k_1, k_2 such that $|f_1(x)| \le C_1 |g_1(x)|$ when $x > k_1$ and $|f_2(x)| \le C_2 |g_2(x)|$ when $x > k_2$.

$$\begin{aligned} |(f_1 + f_2)(x)| &= |f_1(x) + f_2(x)| && \text{by the triangle inequality } |a + b| \le |a| + |b| \\ &\le |f_1(x)| + |f_2(x)| \\ |f_1(x)| + |f_2(x)| &\le C_1 |g_1(x)| + C_2 |g_2(x)| \\ &\le C_1 |g(x)| + C_2 |g(x)| && \text{where } g(x) = \max(|g_1(x)|, |g_2(x)|) \\ &= (C_1 + C_2) |g(x)| \\ &= C |g(x)| && \text{where } C = C_1 + C_2 \end{aligned}$$

- Therefore $|(f_1 + f_2)(x)| \le C |g(x)|$ whenever $x > k$, where $k = \max(k_1, k_2)$.

Combinations of Functions 4

Example: Give a big-*O* estimate for
$f(n) = 3n \log(n!) + (n^2 + 3) \log n$, where n is a positive integer.
Solution:
$(f_1 f_2)(x)$ is $O(g_1(x)g_2(x))$
$3n$ is $O(n)$; $\log(n!)$ is $O(n \log n)$
$\therefore$ $3n \log(n!)$ is $O(n^2 \log n)$

$(n^2 + 3)$ is $O(n^2)$
$\therefore$ $(n^2 + 3) \log n$ is $O(n^2 \log n)$

$(f_1 + f_2)(x)$ is $O(\max(|g_1(x)|, |g_2(x)|))$
$\therefore = 3n \log(n!) + (n^2 + 3) \log n$ is $O(n^2 \log n)$