UNIVERSITY OF REGINA

CS310-002
DISCRETE COMPUTATIONAL STRUCTURES

ANDRÉ E. **DOS SANTOS**
dossantos@cs.uregina.ca
andreeds.github.io

CS310-002
DISCRETE COMPUTATIONAL STRUCTURES

BASIC STRUCTURES ALGORITHMS

ANDRÉ E. **DOS SANTOS**
dossantos@cs.uregina.ca
andreeds.github.io

THE GROWTH OF FUNCTIONS

PROPERTIES OF ALGORITHMS

INPUT

OUTPUT

DEFINITENESS

CORRECTNESS

FINITENESS

EFFECTIVENESS

GENERALITY

Definition

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

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

Example

Rank the following functions by order of growth

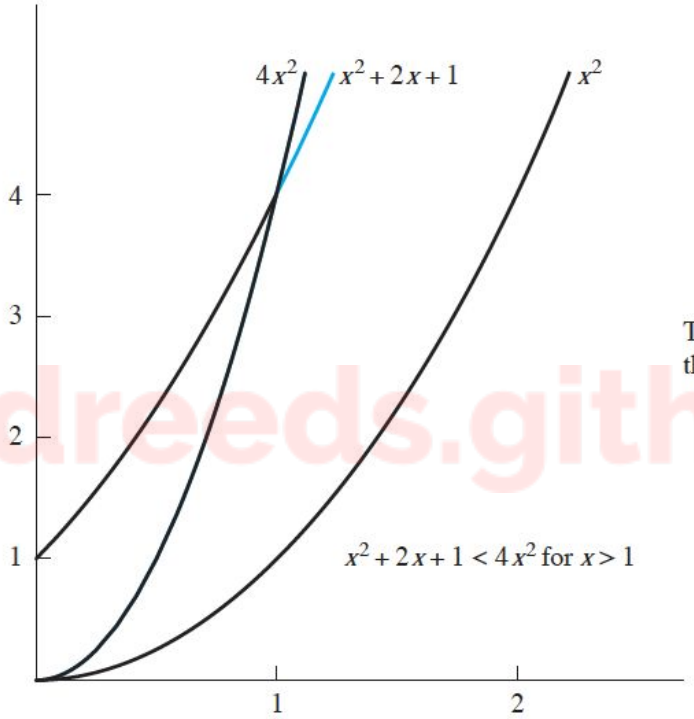
$100n\ log(n)$, $n^{1.01}$, 2^n, $2^{log(n)}$, $log(log(n))$, $log^2 n$, $n\ log(n!)$

Example:

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

The Function $x^2 + 2x + 1$ is $O(x^2)$

p206



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

TERMINOLOGY

p226

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

Definition

f(x) is **Ω(g(x))** if there are constants **C** and **k** such that

$$|f(x)| \geq C|g(x)| \text{ whenever } x > k$$

Definition

f(x) is **Θ(g(x))** if **f(x)** is ***O*(g(x))** and **f(x)** is **Ω(g(x))**

Worst-case time complexity

The greatest amount of time required for an algorithm to solve a problem of a given size

Example:

What is the worst-case complexity of the bubble sort in terms of the number of comparisons made?

procedure *bubblesort*($a_1, \ldots, a_n$: real numbers with $n \geq 2$)
for $i := 1$ **to** $n - 1$
  for $j := 1$ **to** $n - i$
    if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}
{$a_1, \ldots, a_n$ is in increasing order}

Worst-case space complexity

The greatest amount of memory space required for an algorithm to solve a problem of a given size

Average-case time complexity

the average amount of time required for an algorithm to solve a problem of a given size

TERMINOLOGY

p226

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

Upper bound

The *maximum* time a program can take to produce outputs

worst-case scenario

Lower bound

The *minimum* time a program can take to produce outputs

best-case scenario

expressed in terms of the size of the inputs

WHAT MAKES P VS. NP SO HARD?

P ≠ EXPTIME, TIME HIERARCHY, BAKER-GILL-SOLOVAY

https://www.youtube.com/watch?v=XV6f7XYUMg8

P VERSUS *NP*

Problems for which a *solution* can be checked in *polynomial time* are said to belong to the class *NP*

tractable problems are said to belong to class *P*

Solvable

a problem is solvable if it can be solved by an algorithm

e.g., sorting problem, searching problem, MSS problem, etc.

Unsolvable

a problem is unsolvable if no algorithm exists for solving it

Question: What problem is unsolvable?

Halting problem

deciding whether a computational procedure halts on an input

Theorem (Turing, 1936). The halting problem is unsolvable.

IMPOSSIBLE PROGRAMS
THE HALTING PROBLEM

https://www.youtube.com/watch?v=wGLQiHXHWNk

REVIEW QUESTIONS
ALGORITHMS

List these functions so that each function is big-O of the next function in the list: $(\log n)^3$, $n^3/1000000$, $\sqrt{n}$, $100n + 101$, 3^n, $n!$, $2^n n^2$.

Determine the worst-case complexity

```
procedure palindrome check(a1a2 ... an: string)
answer := true
for i := 1 to ⌊n/2⌋
  if ai ≠ an+1−i then answer := false
return answer
```

Define what it means for a problem to be tractable and what it means for a problem to be solvable.