Discrete Structures

Syed Faisal Bukhari, PhD
Associate Professor
Department of Data Science (DDS), Faculty of Computing and Information Technology (FCIT), University of the Punjab (PU)

Text book

Discrete Mathematics and Its Application, 7th Edition

Kenneth H. Rosen

References

Chapter 3

1. Discrete Mathematics and Its Application, 7h Edition

by

Kenneth H. Rose

2. Discrete Mathematics with Applications

by

Thomas Koshy

3. Discrete Mathematical Structures, CS 173

by

Cinda Heeren, Siebel Center

These slides contain material from the above resources.

- **Definition** An ***algorithm*** is a **finite sequence** of precise instructions for performing a computation or for solving a problem.

Example Describe an **algorithm** for finding the **maximum (largest)** value in a finite sequence of integers.

1. Set the **temporary maximum** equal to the **first integer** in the sequence. (The temporary maximum will be the largest integer examined at any stage of the procedure.)

2. **Compare** the next integer in the **sequence to the temporary maximum**, and 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 in the sequence.

4. **Stop** when there are **no integers left in the sequence**. The **temporary maximum** at this point is the **largest integer** in the sequence.

Pseudocode

- An **algorithm** can also be described using a **computer language**.
- However, when that is done, only those **instructions permitted in the language can be used**. This often leads to a description of the algorithm that is **complicated and difficult to understand**.
- So, instead of using a **particular computer language** to specify algorithms, a form of **pseudocode**.

Pseudocode

- **Pseudocode** provides an **intermediate step** between an **English language description** of an **algorithm** and an **implementation of this algorithm** in a programming language.

Finding the Maximum Element in a Finite Sequence

Algorithm 1 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}
```

Properties of Algorithms

There are several properties that algorithms generally share. They are useful to keep in mind when algorithms are described.

1. **Input.** An algorithm has **input values** from a **specified set**.

2. **Output.** From each **set of input values** an algorithm produces **output values** from a **specified set**. The output values are the solution to the problem.

3. **Definiteness.** The steps of an algorithm must be **defined precisely**.

Properties of Algorithms

4. **Correctness.** An algorithm should produce the **correct output values** for each set of **input values**.

5. **Finiteness.** An algorithm should produce the desired output after a **finite (but perhaps large) number of steps** for any input in the set.

6. **Effectiveness.** It must be possible to perform each step of an algorithm exactly and in a **finite amount of time**.

7. **Generality.** The procedure should be **applicable for all problems** of the desired form, not just for a particular set of input values.

Example Show that Algorithm 1 for finding the maximum element in a finite sequence of integers has all the properties listed.

Finding the Maximum Element in a Finite Sequence

Algorithm 1 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

1. The **input** to Algorithm 1 is a sequence of **integers**.

2. The **output** is the **largest integer** in the sequence.

3. Each step of the algorithm is **precisely defined**, because only **assignments, a finite loop,** and **conditional statements** occur.

4. To show that the algorithm is **correct**, we must show that when the **algorithm terminates**, the value of the variable ***max*** **equals the maximum** of the terms of the sequence.

4. Cont. To see this, note that the **initial value of *max*** is the first term of the sequence; as successive terms of the sequence are examined, ***max* is updated** to the value of a term if the term exceeds the maximum of the terms previously examined. This (informal) argument shows that when all the terms have been examined, ***max* equals the value of the largest term**. (A rigorous proof of this requires techniques developed in Section 5.1.)

5. The algorithm uses a **finite number** of steps, because it **terminates after all the integers** in the sequence have been examined.

7. The algorithm can be carried out in a **finite amount of time** because each step is either a **comparison or an assignment**, there are a **finite number of these steps**, and each of these two operations takes a finite amount of time.

8. Finally, Algorithm 1 is **general,** because it can be used to find the maximum of **any finite sequence of integers**.

Big-*O* notation

- **Big-*O*** notation is used extensively to estimate the number of **operations an algorithm** uses as its **input grows**.

- With the help of this notation, we can determine whether it is practical to use a **particular algorithm** to solve a problem as the **size of the input increases**.

- Furthermore, using **big-*O* notation**, we can **compare two algorithms** to determine which is **more efficient** as the size of the **input grows**.

The Growth of Functions

- **Big-O Notation:** 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 $x > k$.

  [This is read as **"f (x) is big-oh of g(x)."**]
- The constants **C** and **k** in the definition of big-O notation are called **witnesses** to the relationship **f (x)** is **O(g(x)).**

Remark: Intuitively, the definition that ***f(x) is O(g(x))*** says that ***f(x)*** grows **slower** that some fixed multiple of ***g(x)*** as ***x* grows without bound**.

The Growth of Functions

To establish that **f(x)** is **O(g(x))** we need **only one pair of witnesses** to this relationship. That is, to show that **f(x) is O(g(x)),** we need find only **one pair of constants C** and **k**, the witnesses, such that

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

Note: C and k are constants, whereas k is a positive real number

The Growth of Functions

Note that when there is one pair of witnesses to the relationship **f(x)** is **O(g(x)),** there are **infinitely many pairs of witnesses**.

To see this, note that if **C** and **k** are **one pair of witnesses**, then any pair C' and $\mathbf{k}'$, where **C <** $\mathbf{C}'$and **k <** $\mathbf{k}'$, is also a **pair of witnesses**, because

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

Working with the definition of Big-*O* Notation

A useful approach for finding a pair of witnesses is to first select a value of **k** for which the size of $|f(x)|$ can be readily estimated when **$x > k$** and to see whether we can use this estimate to find a value of C for which

$|f(x)| \leq C|g(x)|$ for **$x > k$**.

Big-*O* Notation

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

Solution:

$f(x) = x^2 + 2x + 1\,x^0$

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

We observe that we can readily estimate the size of $f(x)$ when $x > 1$

$\because x < x^2$ and $x^0 < x^2$ when $x > 1$.

x	$x < x^2$	$1 < x^2$
1.1	$1.1 < 1.1^2$ (True)	$1 < 1.1^2$ (True)
⋮	⋮	⋮
2	$2 < 2^2$ (True)	$1 < 2^2$ (True)
3	$3 < 3^2$ (True)	$1 < 3^2$ (True)
⋮	⋮	⋮

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

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

$\Rightarrow x^2 + 2x + 1 \leq Cg(x)$ when $x > 1$

Consequently, we can take **$C = 4$** and **$k = 1$** as witnesses to show that $f(x)$ is $O(x^2)$

Note: C and k are constants, whereas k is a positive real number and C is a real number.

Checking the witness **C = 4** and **k = 1** **for the inequality** $x^2 + 2x + 1 \leq Cx^2$

x values	Inequality
1.1	$x^2 + 2x + 1 \leq 4x^2$ $\Rightarrow 1.1^2 + 2(1.1) + 1 \leq 4(1.1)^2$ $\Rightarrow 4.41 \leq 4.84$ (true)
⋮	⋮
2	$x^2 + 2x + 1 \leq 4x^2$ $\Rightarrow 2^2 + 2(2) + 1 \leq 4(2)^2$ $\Rightarrow 9 \leq 16$ (true)
3	$\Rightarrow 3^2 + 2(3) + 1 \leq 4(3)^2$ $\Rightarrow 15 \leq 36$ (true)
⋮	⋮

Big-*O* Notation

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

Alternative solution

f (x) = $x^2 + 2x + 1x^0$

$|f(x)| \le C|g(x)|$ whenever $x > k$.

We observe that we can also readily estimate the size of f (x) when $x > 2$

$\because 2x \le x^2$ and $1x^0 \le x^2$ when **$x > 2$**

x	$2x \le x^2$	$1 \le x^2$
2.1	$2\times 2.1 \le 2.1^2$ (True)	$1 \le 2.1^2$ (True)
⋮	⋮	⋮
3	$2\times 3 \le 3^2$ (True)	$1 \le 3^2$ (True)
4	$2\times 4 \le 4^2$ (True)	$1 < 4^2$ (True)
⋮	⋮	⋮

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

$\Rightarrow x^2 + 2x + 1 \le 3x^2$

$\Rightarrow x^2 + 2x + 1 \le Cg(x)$ Consequently, we can also **take C = 3** and **k = 2** as witnesses to show that f (x) is $O(x^2)$

Note: C and k are constants, whereas k is a positive real number and C is a real number.

Checking the witness **C = 3** and **k = 2** **for the inequality** $x^2 + 2x + 1 \le Cx^2$

x values	Inequality
2.1	$x^2 + 2x + 1 \le 3x^2$ $\Rightarrow 2.1^2 + 2(2.1) + 1 \le 3(2.1)^2$ $\Rightarrow 9.61 \le 13.23$ (true)
⋮	⋮
3	$\Rightarrow 3^2 + 2(3) + 1 \le 3(3)^2$ $\Rightarrow 15 \le 27$ (true)
4	$\Rightarrow 4^2 + 2(4) + 1 \le 3(4)^2$ $\Rightarrow 25 \le 48$ (true)
⋮	⋮

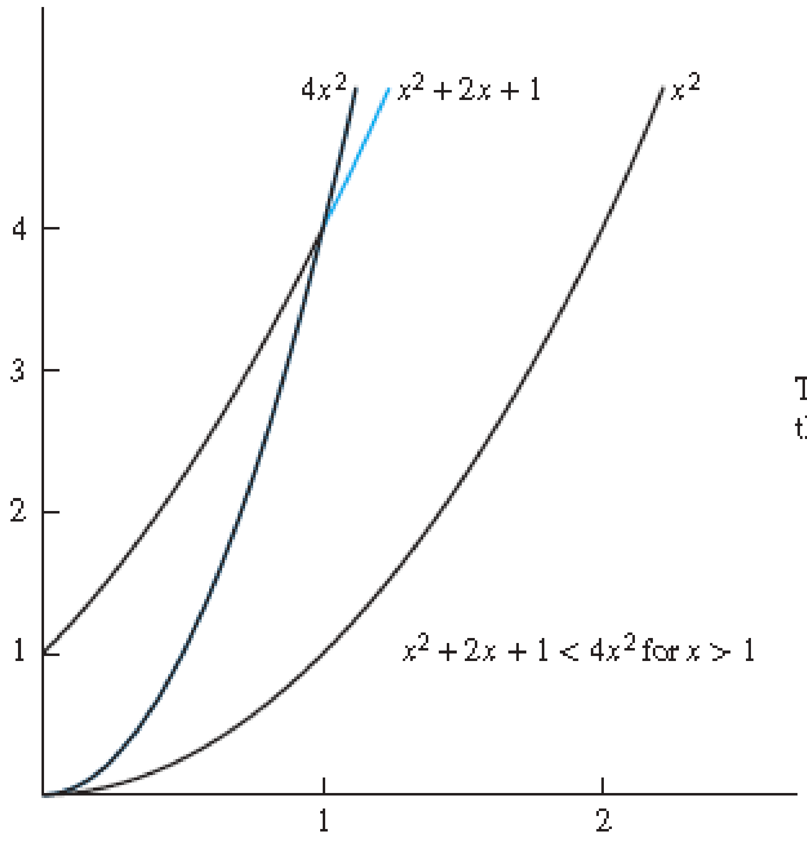
Big-*O* Notation

Note that when there is **one pair of witnesses** to the relationship f (x) is O(g(x)), **there are infinitely many pairs of witnesses**.

To see this, note that if C and k are one pair of witnesses, then any pair C' and k' , where C < C' and k < k' is also a pair of witnesses, because

|f (x)| ≤ C|g(x)| ≤ C' |g(x)| whenever **x > k' > k**.

Big-*O* Notation



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

FIGURE 1 The function $x^2 + 2x + 1$ is $O(x^2)$.

Big-*O* Notation

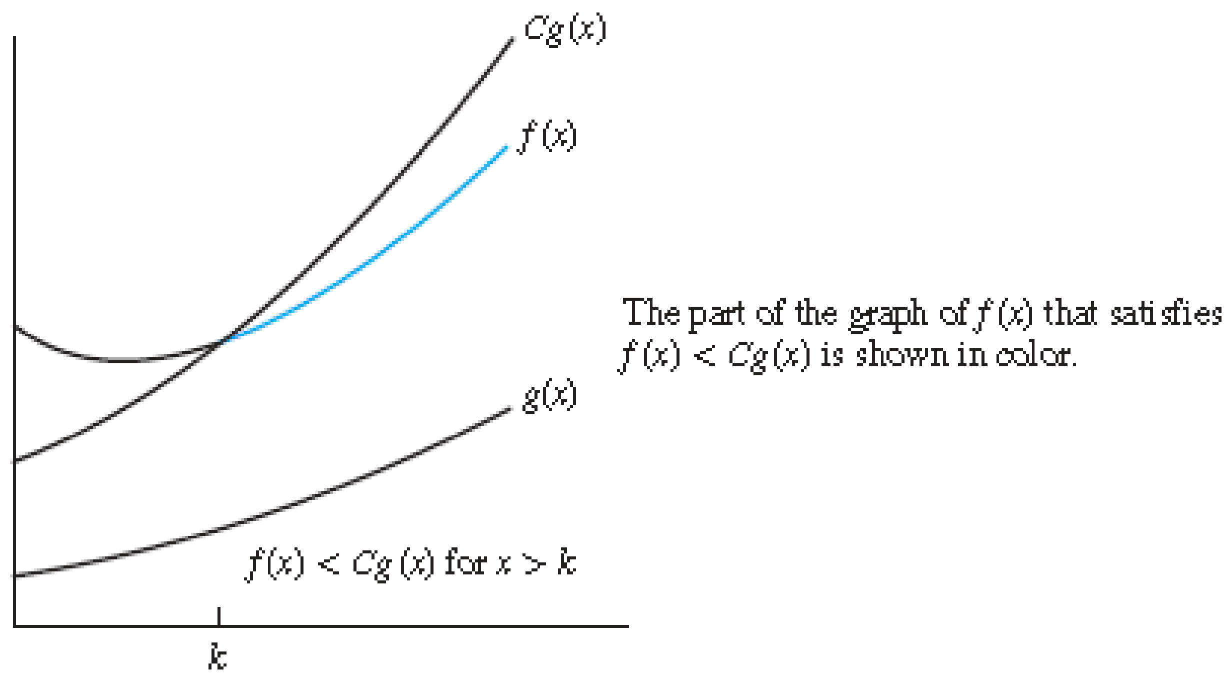

$Cg(x)$
$f(x)$
The part of the graph of $f(x)$ that satisfies $f(x) < Cg(x)$ is shown in color.
$g(x)$
$f(x) < Cg(x)$ for $x > k$
k

FIGURE 2 **The function $f(x)$ is $O(g(x))$.**

Big-*O* Notation

Remark: The fact that **f (x)** is **O(g(x))** is sometimes written

f(x) = O(g(x)).

However, the **equals sign** in this notation does not represent a **genuine equality**. Rather, this notation tells us that an inequality holds relating the values of the functions f and g for sufficiently large numbers in the domains of these functions.

However, it is acceptable to write **f(x) ∈ O(g(x))** because **O(g(x))** represents the **set of functions** that are **O(g(x)).**

Big-*O* Notation

Example Show that $f(x) = 50x^3 - 6x + 23$ is $O(x^3)$.

Solution:

$50x^3 - 6x + 23$

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

$|f(x)| = | 50x^3 - 6x + 23 |$

$\leq | 50x^3| + |- 6x | + |23| = 50x^3 + 6x + 23 x^0$

We observe that we can readily estimate the size of f (x) when x > 1

$\because x < x^3$ and $x^0 < x^3$ when **$x > 1$**.

$0 \leq 50x^3 + 6x + 23 \leq 50x^3 + 6x^3 + 23x^3 = 79x^3$

$\Rightarrow 50x^3 + 6x + 23 \leq \mathbf{79x^3}$

$\Rightarrow 50x^3 + 6x + 23 \leq Cg(x)$ when $x > 1$

Consequently, we can take C = 79 and k = 1 as witnesses to show that f (x) is $O(x^3)$

Note: C and k are constants, whereas k is a positive real number and C is a real number

Checking the witness **C = 79** and **k = 1** **for the inequality** $50x^3 + 6x + 23 \leq C\,x^3$

x values	Inequality
1.1	$50x^3 + 6x + 23 \leq 79\,x^3$ $\Rightarrow 50(1.1)^3 + 6(1.1) + 23 \leq 79(1.1)^3$ $\Rightarrow 96.15 \leq 105.149$ (true)
⋮	⋮
3	$\Rightarrow 3^2 + 2(3) + 1 \leq 3(3)^2$ $\Rightarrow 15 \leq 27$ (true)
4	$\Rightarrow 4^2 + 2(4) + 1 \leq 3(4)^2$ $\Rightarrow 25 \leq 48$ (true)
⋮	⋮

Suggested Readings

Chapter 3

3.1 Algorithms

3.2 The Growth of Functions