Efficiency, Big $O$

Ioan Despi

despi@turing.une.edu.au

University of New England

July 9, 2013
Outline

1. Nested Evaluation of a Polynomial
   - Horner’s algorithm

2. Big $\mathcal{O}$ Notation

3. Examples

4. Simple Features of $\mathcal{O}$
Nested Evaluation of a Polynomial

- A mathematical expression involving powers in one or more variables multiplied by coefficients is called a **monomial**, e.g., $5a^3b^5c$. 

Examples

- $3x^5 - x + 1$ is a polynomial of degree 5,
- $-y^7 + 5y^6 + y^2$ is a polynomial of degree 7,
- constants are polynomials of degree 0 (e.g., $4 = 4x^0$).
A mathematical expression involving powers in one or more variables multiplied by coefficients is called a monomial, e.g., $5a^3b^5c$.

An (algebraic) sum of monomials is called a polynomial.
Nested Evaluation of a Polynomial

- A mathematical expression involving powers in one or more variables multiplied by coefficients is called a **monomial**, e.g., $5a^3b^5c$
- An (algebraic) sum of monomials is called a **polynomial**.
- A **polynomial** of **degree** $n$ in variable $x$ is an expression of the form

$$
\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n,
$$

where $a_0, a_1, \ldots, a_n$ are (real) constants.
Nested Evaluation of a Polynomial

- A mathematical expression involving powers in one or more variables multiplied by coefficients is called a **monomial**, e.g., $5a^3b^5c$.
- An (algebraic) sum of monomials is called a **polynomial**.
- A polynomial of **degree** $n$ in variable $x$ is an expression of the form

$$\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n,$$

where $a_0, a_1, \ldots, a_n$ are (real) constants.
- The highest power in a univariate polynomial is called its **order**, or sometimes its **degree**.
Nested Evaluation of a Polynomial

A mathematical expression involving powers in one or more variables multiplied by coefficients is called a **monomial**, e.g., $5a^3b^5c$.

An (algebraic) sum of monomials is called a **polynomial**.

A polynomial of **degree** $n$ in variable $x$ is an expression of the form

$$
\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n,
$$

where $a_0, a_1, \ldots, a_n$ are (real) constants.

The highest power in a univariate polynomial is called its **order**, or sometimes its **degree**.

Examples
Nested Evaluation of a Polynomial

- A mathematical expression involving powers in one or more variables multiplied by coefficients is called a **monomial**, e.g., \(5a^3b^5c\)
- An (algebraic) sum of monomials is called a **polynomial**.
- A **polynomial** of **degree** \(n\) in variable \(x\) is an expression of the form

\[
\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n,
\]

where \(a_0, a_1, \ldots, a_n\) are (real) constants.

- The highest power in a univariate polynomial is called its **order**, or sometimes its **degree**.
- **Examples**
  - \(3x^5 - x + 1\) is a polynomial of degree 5, and
A mathematical expression involving powers in one or more variables multiplied by coefficients is called a **monomial**, e.g., $5a^3b^5c$.

An (algebraic) sum of monomials is called a **polynomial**.

A **polynomial** of degree $n$ in variable $x$ is an expression of the form

$$
\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n,
$$

where $a_0, a_1, \ldots, a_n$ are (real) constants.

The highest power in a univariate polynomial is called its **order**, or sometimes its **degree**.

**Examples**

- $3x^5 - x + 1$ is a polynomial of degree 5, and
- $-y^7 + 5y^6 + y^2$ is a polynomial of degree 7.
Nested Evaluation of a Polynomial

- A mathematical expression involving powers in one or more variables multiplied by coefficients is called a **monomial**, e.g., $5a^3b^5c$
- An (algebraic) sum of monomials is called a **polynomial**.
- A **polynomial** of degree $n$ in variable $x$ is an expression of the form

$$\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n ,$$

where $a_0, a_1, ..., a_n$ are (real) constants.

- The highest power in a univariate polynomial is called its **order**, or sometimes its **degree**.
- Examples
  - $3x^5 - x + 1$ is a polynomial of degree 5, and
  - $-y^7 + 5y^6 + y^2$ is a polynomial of degree 7
  - Constants are polynomials of degree 0 (e.g., $4 = 4x^0$)
Nested Evaluation of a Polynomial

- The sum of two polynomials is obtained by adding together the coefficients sharing the same powers of variables (i.e., the same terms).
Nested Evaluation of a Polynomial

- The sum of two polynomials is obtained by adding together the coefficients sharing the same powers of variables (i.e., the same terms).
  - For example,
    
    $$(ax^2 + bx + c) + (dx + e) = ax^2 + (b + d)x + (c + e)$$

    and has order $\leq$ to the maximum order of the original two polynomials.
The sum of two polynomials is obtained by adding together the coefficients sharing the same powers of variables (i.e., the same terms).

For example,

\[(ax^2 + bx + c) + (dx + e) = ax^2 + (b + d)x + (c + e)\]

and has order \(\leq\) to the maximum order of the original two polynomials.

The product of two polynomials is obtained by multiplying term by term and combining the results.
The sum of two polynomials is obtained by adding together the coefficients sharing the same powers of variables (i.e., the same terms).

For example,

\[(ax^2 + bx + c) + (dx + e) = ax^2 + (b + d)x + (c + e)\]

and has order \( \leq \) to the maximum order of the original two polynomials.

The product of two polynomials is obtained by multiplying term by term and combining the results.

For example

\[(ax^2 + bx + c)(dx + e) = ax^2(dx + e) + bx(dx + e) + c(dx + e)\]

\[= adx^3 + (ae + bd)x^2 + (be + cd)x + ce\]

and has order = to the sum of the orders of the two original polynomials.
Nested Evaluation of a Polynomial

- The sum of two polynomials is obtained by adding together the coefficients sharing the same powers of variables (i.e., the same terms).
  - For example,
    \[(ax^2 + bx + c) + (dx + e) = ax^2 + (b + d)x + (c + e)\]
    and has order ≤ to the maximum order of the original two polynomials.

- The product of two polynomials is obtained by multiplying term by term and combining the results.
  - For example
    \[(ax^2 + bx + c)(dx + e) = ax^2(dx + e) + bx(dx + e) + c(dx + e)\]
    \[= adx^3 + (ae + bd)x^2 + (be + cd)x + ce\]
    and has order = to the sum of the orders of the two original polynomials.

- A polynomial quotient
  \[R(x) = \frac{P(x)}{Q(x)}\]
  of two polynomials \(P(x)\) and \(Q(x)\) is known as a rational function. The process of performing such a division is called long division.
Nested Evaluation of a Polynomial

- To evaluate a polynomial means to find its numerical value for given numerical values of its variables.

△ the direct evaluation (brute force, term by term) of polynomial $f(x) = 2x^3 + 9x^2 + 5x - 1$ requires a total of 9 operations (6 multiplications + 3 additions/subtractions) while

△ the evaluation of the same polynomial in the following form $f(x) = x[(2x + 9) + 5] - 1$ requires only 6 operations (3 multiplications plus 3 additions/subtractions).

This method of evaluation is essentially Horner’s algorithm, also called the nested evaluation.
Nested Evaluation of a Polynomial

- To evaluate a polynomial means to find its numerical value for given numerical values of its variables.
- Many ways to do the evaluation, some better than others.
To evaluate a polynomial means to find its numerical value for given numerical values of its variables.

Many ways to do the evaluation, some better than others.

For example,
To evaluate a polynomial means to find its numerical value for given numerical values of its variables.

Many ways to do the evaluation, some better than others.

For example,

- the direct evaluation (brute force, term by term) of polynomial

\[ f(x) = 2x^3 + 9x^2 + 5x - 1 \]

requires a total of 9 operations (6 multiplications + 3 additions/subtractions) while
Nested Evaluation of a Polynomial

- To evaluate a polynomial means to find its numerical value for given numerical values of its variables.
- Many ways to do the evaluation, some better than others.
- For example,
  - the direct evaluation (brute force, term by term) of polynomial
    $$f(x) = 2x^3 + 9x^2 + 5x - 1$$
    requires a total of 9 operations (6 multiplications + 3 additions/subtractions) while
  - the evaluation of the same polynomial in the following form
    $$f(x) = x [x (2x + 9) + 5] - 1$$
    requires only 6 operations (3 multiplications plus 3 additions/subtractions).
Nested Evaluation of a Polynomial

- To evaluate a polynomial means to find its numerical value for given numerical values of its variables.
- Many ways to do the evaluation, some better than others.
- For example,
  - the direct evaluation (brute force, term by term) of polynomial
    \[ f(x) = 2x^3 + 9x^2 + 5x - 1 \]
    requires a total of 9 operations (6 multiplications + 3 additions/subtractions) while
  - the evaluation of the same polynomial in the following form
    \[ f(x) = x \left[ x (2x + 9) + 5 \right] - 1 \]
    requires only 6 operations (3 multiplications plus 3 additions/subtractions).
- This method of evaluation is essentially Horner’s algorithm, also called the nested evaluation.
Horner’s algorithm

- Given an $n$–th order polynomial

\[ P(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 , \]

we can rewrite $P(x)$ as
Horner’s algorithm

- Given an $n$–th order polynomial

\[ P(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0, \]

we can rewrite $P(x)$ as
Horner’s algorithm

- Given an \( n \)-th order polynomial

\[
P(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 ,
\]

we can rewrite \( P(x) \) as

\[
P(x) = x (a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_1) + a_0
\]

\[
= x (x (a_n x^{n-2} + \cdots + a_2) + a_1) + a_0 
\]

\[
= \cdots 
\]

\[
= x \left[ x \left[ \cdots x (x (a_n x + a_{n-1}) + a_{n-2}) + \cdots \right] + a_1 \right] + a_0 
\]

- The above telescoping form requires no more than \( n \) multiplications and \( n \) additions/subtractions.
Horner’s algorithm

- Given an \( n \)-th order polynomial

\[
P(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,
\]

we can rewrite \( P(x) \) as

\[
P(x) = x(a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_1) + a_0
\]

\[
= x(x(a_n x^{n-2} + \cdots + a_2) + a_1) + a_0
\]

\[
= \cdots
\]

\[
= x \left[ x \left[ \cdots x(x(a_n x + a_{n-1}) + a_{n-2}) + \cdots \right] + a_1 \right] + a_0
\]

- The above telescoping form requires no more than \( n \) multiplications and \( n \) additions/subtractions.

- It is obviously more efficient than the direct term by term evaluation.
We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.
We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.

The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms (AA)**.
We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.

The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms (AA)**.

- It is the process of finding estimates for the time and space needed to execute the algorithm.

The big $\mathcal{O}$ notation will serve well in this regard.
We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.

The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms** (AA).

- It is the process of finding estimates for the time and space needed to execute the algorithm.
  - The time is expressed as the number of steps and
Big $\mathcal{O}$ Notation

- We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.
- The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms (AA)**.
  - It is the process of finding estimates for the time and space needed to execute the algorithm.
    - The time is expressed as the number of steps and
    - The space is expressed as the number of memory cells.

In other words, AA deals with functions that define the quantity of some resource consumed by a particular algorithm. We call such a function the **complexity of the algorithm** (sometimes the **cost function** of the algorithm).

- The magnitude rather than the precise value is sufficient or is significant for these functions.
- We are interested in comparing our functions with well-known ones (polynomial, exponential, logarithmic, etc.) to find their upper or lower bounds.
- The big $\mathcal{O}$ notation will serve well in this regard.
Big $\mathcal{O}$ Notation

- We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.
- The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms (AA)**.
  - It is the process of finding estimates for the time and space needed to execute the algorithm.
    - The time is expressed as the number of steps and
    - The space is expressed as the number of memory cells.
- In other words, AA deals with functions that define the quantity of some resource consumed by a particular algorithm.
Big $\mathcal{O}$ Notation

- We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.
- The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms** (AA).
  - It is the process of finding estimates for the time and space needed to execute the algorithm.
    - The time is expressed as the number of steps and
    - The space is expressed as the number of memory cells.
- In other words, AA deals with functions that define the quantity of some resource consumed by a particular algorithm.
- We call such a function the **complexity of the algorithm** (sometimes the **cost function** of the algorithm).
We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.

The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms (AA)**.

- It is the process of finding estimates for the time and space needed to execute the algorithm.
  - The time is expressed as the number of steps and
  - The space is expressed as the number of memory cells.

In other words, AA deals with functions that define the quantity of some resource consumed by a particular algorithm.

We call such a function the **complexity of the algorithm** (sometimes the **cost function** of the algorithm).

- The magnitude rather than the precise value is sufficient or is significant for these functions.
Big $\mathcal{O}$ Notation

- We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.
- The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms** (AA).
  - It is the process of finding estimates for the time and space needed to execute the algorithm.
    - The time is expressed as the number of steps and
    - The space is expressed as the number of memory cells.
- In other words, AA deals with functions that define the quantity of some resource consumed by a particular algorithm.
- We call such a function the **complexity of the algorithm** (sometimes the **cost function** of the algorithm).
  - The magnitude rather than the precise value is sufficient or is significant for these functions.
- We are interested in comparing our functions with well-known ones (polynomial, exponential, logarithmic, etc.) to find their upper or lower bounds.
Big $\mathcal{O}$ Notation

- We devise algorithms to solve problems and we choose the best one to be implemented in a programming language.
- The performance of algorithms is checked in a branch of computer science called **Analysis of Algorithms** (AA).
  - It is the process of finding estimates for the time and space needed to execute the algorithm.
    - The time is expressed as the number of steps and
    - The space is expressed as the number of memory cells.
- In other words, AA deals with functions that define the quantity of some resource consumed by a particular algorithm.
- We call such a function the **complexity of the algorithm** (sometimes the **cost function** of the algorithm).
  - The magnitude rather than the precise value is sufficient or is significant for these functions.
- We are interested in comparing our functions with well-known ones (polynomial, exponential, logarithmic, etc.) to find their upper or lower bounds.
- The big $\mathcal{O}$ notation will serve well in this regard.
**Big $\mathcal{O}$ Notation**

**Definition. (Big $\mathcal{O}$)**

Let $D \subseteq \mathbb{R}$ and $f: D \rightarrow \mathbb{R}$ and $g: D \rightarrow \mathbb{R}$ be two (real-valued) functions. We say $f(x)$ is $\mathcal{O}(g(x))$ ("big oh"), and we write $f(x) = \mathcal{O}(g(x))$, if $\exists M \in \mathbb{R}$ and $\exists C > 0$ such that $\forall x \in D$,

$$|f(x)| \leq C|g(x)|, \quad \text{whenever } x \geq M.$$

*Notice that this is just a notation, that is*
Big $\mathcal{O}$ Notation

Definition. (Big $\mathcal{O}$)

Let $D \subseteq \mathbb{R}$ and $f: D \to \mathbb{R}$ and $g: D \to \mathbb{R}$ be two (real-valued) functions. We say $f(x)$ is $\mathcal{O}(g(x))$ ("big oh"), and we write $f(x) = \mathcal{O}(g(x))$, if $\exists M \in \mathbb{R}$ and $\exists C > 0$ such that $\forall x \in D$,

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

whenever $x \geq M$.

- Notice that this is just a notation, that is
  - the equal sign in the expression does not really denote mathematical equality, and
### Definition. (Big $\mathcal{O}$)

Let $D \subseteq \mathbb{R}$ and $f: D \rightarrow \mathbb{R}$ and $g: D \rightarrow \mathbb{R}$ be two (real-valued) functions. We say $f(x)$ is $\mathcal{O}(g(x))$ ("big oh"), and we write $f(x) = \mathcal{O}(g(x))$, if $\exists M \in \mathbb{R}$ and $\exists C > 0$ such that $\forall x \in D$,

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

whenever $x \geq M$.

- Notice that this is just a notation, that is
  - the equal sign in the expression does not really denote mathematical equality, and
  - $\mathcal{O}(\cdot)$ does not really mean that $\mathcal{O}$ is a mathematical function.
Big $\mathcal{O}$ Notation

**Definition. (Big $\mathcal{O}$)**

Let $D \subseteq \mathbb{R}$ and $f: D \to \mathbb{R}$ and $g: D \to \mathbb{R}$ be two (real-valued) functions. We say $f(x)$ is $\mathcal{O}(g(x))$ ("big oh"), and we write $f(x) = \mathcal{O}(g(x))$, if $\exists M \in \mathbb{R}$ and $\exists C > 0$ such that $\forall x \in D$,

$$|f(x)| \leq C|g(x)|, \quad \text{whenever } x \geq M.$$ 

- Notice that this is just a notation, that is
  - the equal sign in the expression does not really denote mathematical equality, and
  - $\mathcal{O}(\cdot)$ does not really mean that $\mathcal{O}$ is a mathematical function
- The value $M$ and the value $C > 0$ themselves are not important:
Let $D \subseteq \mathbb{R}$ and $f: D \to \mathbb{R}$ and $g: D \to \mathbb{R}$ be two (real-valued) functions. We say $f(x)$ is $\mathcal{O}(g(x))$ (“big oh”), and we write $f(x) = \mathcal{O}(g(x))$, if $\exists M \in \mathbb{R}$ and $\exists C > 0$ such that $\forall x \in D,$

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

whenever $x \geq M$.

- Notice that this is just a notation, that is
  - the equal sign in the expression does not really denote mathematical equality, and
  - $\mathcal{O}(\cdot)$ does not really mean that $\mathcal{O}$ is a mathematical function

- The value $M$ and the value $C > 0$ themselves are not important:
  - the significance lies in the existence of such $M$ and $C$, rather than the magnitude of these values.
The diagram below (see Figure 1) depicts an intuitive interpretation of the above inequality: the curve $y = |f(x)|$ will always be below the curve $y = C|g(x)|$ after $x$ has passed the mark $x = M$. 

**Figure :** Big O
We note that if a property is to be established from first principles, then the property has to be derived or proved from the basic definitions.

In other words we need to find a $C > 0$ and an $M$ such that $|F(n)| \leq C |G(n)|$ holds whenever $n \geq M$.

In establishing big $\mathcal{O}$ properties, one typically has to make use of the triangle inequalities (see Preliminary Mathematics):

$$|a| - |b| \leq |a + b| \leq |a| + |b|$$

$$|x_1| - |x_2| - \cdots - |x_n| \leq |x_1| + |x_2| + \cdots + |x_n|$$
We note that if a property is to be established from **first principles**, then the property has to be derived or proved from the basic **definitions**. Hence if we are to show, for instance, \( F(n) = \mathcal{O}(G(n)) \) from first principles for some given functions \( F(n) \) and \( G(n) \), then we have to derive it from the very definition of big \( \mathcal{O} \).
Big $\mathcal{O}$ Notation

- We note that if a property is to be established from **first principles**, then the property has to be derived or proved from the basic **definitions**.
- Hence if we are to show, for instance, $F(n) = \mathcal{O}(G(n))$ from first principles for some given functions $F(n)$ and $G(n)$, then we have to derive it from the very definition of big $\mathcal{O}$.
- In other words we need to find a $C > 0$ and an $M$ such that $|F(n)| \leq C|G(n)|$ holds whenever $n \geq M$. 

In establishing big $\mathcal{O}$ properties, one typically has to make use of the triangle inequalities (see Preliminary Mathematics): 

\[|a| - |b| \leq |a + b| \leq |a| + |b|\]

\[|x_1 - x_2 - \cdots - x_n| \leq |x_1| + |x_2| + \cdots + |x_n|\]
Big $\mathcal{O}$ Notation

- We note that if a property is to be established from **first principles**, then the property has to be derived or proved from the basic **definitions**.

- Hence if we are to show, for instance, $F(n) = \mathcal{O}(G(n))$ from first principles for some given functions $F(n)$ and $G(n)$, then we have to derive it from the very definition of big $\mathcal{O}$.

- In other words we need to find a $C > 0$ and an $M$ such that $|F(n)| \leq C|G(n)|$ holds whenever $n \geq M$.

- In establishing big $\mathcal{O}$ properties, one typically has to make use of the **triangle inequalities** (see Preliminary Mathematics):
Big $\mathcal{O}$ Notation

- We note that if a property is to be established from **first principles**, then the property has to be derived or proved from the basic definitions.
- Hence if we are to show, for instance, $F(n) = \mathcal{O}(G(n))$ from first principles for some given functions $F(n)$ and $G(n)$, then we have to derive it from the very definition of big $\mathcal{O}$.
- In other words we need to find a $C > 0$ and an $M$ such that $|F(n)| \leq C|G(n)|$ holds whenever $n \geq M$.
- In establishing big $\mathcal{O}$ properties, one typically has to make use of the **triangle inequalities** (see Preliminary Mathematics):
We note that if a property is to be established from **first principles**, then the property has to be derived or proved from the basic **definitions**.

Hence if we are to show, for instance, \( F(n) = \mathcal{O}(G(n)) \) from first principles for some given functions \( F(n) \) and \( G(n) \), then we have to derive it from the very definition of big \( \mathcal{O} \).

In other words we need to find a \( C > 0 \) and an \( M \) such that \( |F(n)| \leq C|G(n)| \) holds whenever \( n \geq M \).

In establishing big \( \mathcal{O} \) properties, one typically has to make use of the **triangle inequalities** (see Preliminary Mathematics):

\[
|a| - |b| \leq |a + b| \leq |a| + |b|
\]

\[
|x_1| - |x_2| - \cdots - |x_n| \leq |x_1 + x_2 + \cdots + x_n| \leq |x_1| + |x_2| + \cdots + |x_n|
\]
1. Let \( f : \mathbb{N} \rightarrow \mathbb{R} \) be given by \( f(n) = 2n - 3 \). Show \( f(n) = \mathcal{O}(n) \).
Examples

1. Let $f : \mathbb{N} \rightarrow \mathbb{R}$ be given by $f(n) = 2n - 3$. Show $f(n) = \mathcal{O}(n)$.

Solution. Observe

$$|f(n)| = |2n - 3| \leq |2n| + |-3| \leq 2|n| + 3 \quad \text{if } n \geq 3 \leq 2n + n = 3n$$

i.e. $|f(n)| \leq 3|n|$, $\forall n \geq 3$,

where we have made use of the triangle inequality

$$|2n - 3| = |(2n) + (-3)| \leq |(2n)| + |(-3)| = |2n| + |3| ,$$

By taking $C = 3$ and $M = 3$ (and $D = \mathbb{N}$), we see $f(n)$ is $\mathcal{O}(n)$.

We note that there are many different yet all valid choices of $C$ and $M$. For example,

$$|f(n)| \leq |2n| + |3| \quad \text{if } n \geq 1 \leq 2n + 3n = 5n , \quad \text{i.e. } |f(n)| \leq 5n, \quad \forall n \geq 1$$

implies we can choose $C = 5$ and $M = 1$ in the definition of $f(n) = \mathcal{O}(n)$. 
2. Show \( f(x) = \frac{3\sqrt{x}(2x + 5)}{|x| + 1} \) is \( O(\sqrt{x}) \) for \( x \in \mathbb{R}^+ \).

The set of positive real numbers is defined as \( \mathbb{R}^+ = \{x \in \mathbb{R} \mid x > 0\} \).
Examples

2. Show $f(x) = \frac{3\sqrt{x}(2x + 5)}{|x| + 1}$ is $O(\sqrt{x})$ for $x \in \mathbb{R}^+$. The set of positive real numbers is defined as $\mathbb{R}^+ = \{x \in \mathbb{R} | x > 0\}$.

**Solution.** The inequality

$$|f(x)| = \left| \frac{3\sqrt{x}(2x + 5)}{|x| + 1} \right| \quad x > 0 \leq \frac{3\sqrt{x}(2x + 5)}{x} = 6\sqrt{x} + \frac{15}{\sqrt{x}}$$

assume $x \geq 1$

$$\leq 6\sqrt{x} + 15 \leq 6\sqrt{x} + 15\sqrt{x} = 21\sqrt{x},$$

i.e., $|f(x)| \leq 21\sqrt{x}$ for $x \geq 1$, gives immediately $f(x) = O(\sqrt{x})$ (by choosing $C = 21$ and $M = 1$ in the definition).

Elementary mathematics have been used here:

- if $a < b$ then $1/a > 1/b$
- if $x > 0$ then $x + 1 > x$ so $\frac{1}{x+1} < \frac{1}{x}$
- $\frac{a+b}{c} = \frac{a}{c} + \frac{b}{c}$, etc.
Examples

3. Let \( f : \mathbb{N} \to \mathbb{R} \), \( f(n) = \frac{1}{2}n(n + 5) \). Prove \( f(n) = \mathcal{O}(n^2) \).
Examples

3. Let $f : \mathbb{N} \to \mathbb{R}$, $f(n) = \frac{1}{2}n(n + 5)$. Prove $f(n) = \mathcal{O}(n^2)$.

Solution. We need to find constant $M$ and positive constant $C$ such that $|f(n)| \leq Cn^2$ for all $n \geq M$ ($|n^2| = n^2$). There are different ways to achieve this, and we give below the 2 most obvious ones.

(a) $|f(n)| = \frac{1}{2}n(n + 5 \times 1) \leq \frac{1}{2}n(n + 5n)$ if assume $n \geq 1$

i.e., $|f(n)| \leq 3n^2$

We take in this case $M = 1$ and $C = 3$, so $f(n) = \mathcal{O}(n^2)$.

(b) $|f(n)| = \frac{1}{2}n(n + 5) = \frac{1}{2}n^2 + \frac{5}{2}n$

$\leq \frac{1}{2}n^2 + \frac{1}{2}n^2$ if assume $n \geq 5$

i.e., $|f(n)| \leq n^2$

We take in this case $M = 5$ and $C = 1$, so $f(n) = \mathcal{O}(n^2)$.
Simple Features of $\mathcal{O}$

Let $f$ and $g$ be mappings from $D \subseteq \mathbb{R}$ to $\mathbb{R}$, $f, g : D \rightarrow \mathbb{R}$, then

- $f(x) = \mathcal{O}(f(x))$, $|f(x)| = \mathcal{O}(f(x))$, $c \cdot f(x) = \mathcal{O}(f(x))$

where $c$ is any constant. For example, $n^2 = \mathcal{O}(n^2)$. 
Simple Features of $\mathcal{O}$

Let $f$ and $g$ be mappings from $D \subseteq \mathbb{R}$ to $\mathbb{R}$, $f, g : D \to \mathbb{R}$, then

- $f(x) = \mathcal{O}(f(x))$, $|f(x)| = \mathcal{O}(f(x))$, $c \cdot f(x) = \mathcal{O}(f(x))$ where $c$ is any constant. For example, $n^2 = \mathcal{O}(n^2)$.

- If $g(x) = \mathcal{O}(f(x))$, then $f(x) + g(x) = \mathcal{O}(f(x))$.
  This is because $g(x) = \mathcal{O}(f(x))$ implies the existence of constants $C > 0$ and $M$ such that $|g(x)| \leq C|f(x)|$, $\forall x \geq M$. Hence

  $$|f(x) + g(x)| \leq |f(x)| + |g(x)| \leq (C + 1)|f(x)|, \quad x \geq M$$

  which means $f(x) + g(x) = \mathcal{O}(f(x))$. For example, $n^3 + 4n = \mathcal{O}(n^3)$. 
Simple Features of $\mathcal{O}$

Let $f$ and $g$ be mappings from $D \subseteq \mathbb{R}$ to $\mathbb{R}$, $f, g : D \rightarrow \mathbb{R}$, then

- $f(x) = \mathcal{O}(f(x))$, $|f(x)| = \mathcal{O}(f(x))$, $c \cdot f(x) = \mathcal{O}(f(x))$
  where $c$ is any constant. For example, $n^2 = \mathcal{O}(n^2)$.

- If $g(x) = \mathcal{O}(f(x))$, then $f(x) + g(x) = \mathcal{O}(f(x))$.
  This is because $g(x) = \mathcal{O}(f(x))$ implies the existence of constants $C > 0$ and $M$ such that $|g(x)| \leq C|f(x)|$, $\forall x \geq M$. Hence

  $$|f(x) + g(x)| \leq |f(x)| + |g(x)| \leq (C + 1)|f(x)|, \quad x \geq M$$

  which means $f(x) + g(x) = \mathcal{O}(f(x))$. For example, $n^3 + 4n = \mathcal{O}(n^3)$.

- If $g(x) = \mathcal{O}(f(x))$, then $f(x) + \mathcal{O}(g(x)) = \mathcal{O}(f(x))$.
  Since $f(x) + \mathcal{O}(g(x))$ represents a quantity $f(x) + h(x)$ such that $h(x) = \mathcal{O}(g(x))$, we see from $f(x) + h(x) = \mathcal{O}(f(x))$ that
  $f(x) + \mathcal{O}(g(x)) = \mathcal{O}(f(x))$. For example,
  $$n^3 + 2n^2 + 5n = n^3 + (2n^2 + 5n) = n^3 + \mathcal{O}(n^2) = \mathcal{O}(n^3).$$
Simple Features of $\mathcal{O}$

- If $f, g : \mathbb{N} \rightarrow \mathbb{R}$, $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) + f_2(n) = O(max(g_1(n), g_2(n)))$
Simple Features of $\mathcal{O}$

- If $f, g : \mathbb{N} \rightarrow \mathbb{R}$, $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) + f_2(n) = O(max(g_1(n), g_2(n)))$
Simple Features of $\mathcal{O}$

- If $f, g : \mathbb{N} \to \mathbb{R}$, $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) + f_2(n) = O(max(g_1(n), g_2(n)))$

**Solution.**

By definition of 'big oh', there are two integers, say $M_1$ and $M_2$ and two constants $C_1$ and $C_2$ such that

- $|f_1(n)| \leq C_1|g_1(n)|$, for $n \geq M_1$
- $|f_2(n)| \leq C_2|g_2(n)|$, for $n \geq M_2$

Let $M_0 = \max(M_1, M_2)$ and let $C_0 = 2 \max(C_1, C_2)$ and consider the sum $f_1(n) + f_2(n)$ for $n \geq M_0$:

$$|f_1(n) + f_2(n)| \leq |f_1(n)| + |f_2(n)|$$
$$\leq C_1|g_1(n)| + C_2|g_2(n)|$$
$$\leq C_0(g_1(n) + g_2(n))/2$$
$$\leq C_0 \max(g_1(n), g_2(n))$$

Therefore $f_1(n) + f_2(n) = O(max(g_1(n), g_2(n)))$. 