# Complexity of Algorithms

Cunsheng Ding

HKUST, Hong Kong

October 9, 2015

# Contents

# Objectives of This Lecture

- Introduce the $O$-notation and $\Theta$-notation.
- Introduce computer algorithms.
- Define the time and space complexity of algorithms.
- Analyse the time complexity of some algorithms.

# The $O$-Notation

This lecture introduces the $O$-notation and $\Theta$-notation, which are fundamental in algorithmics.

## Definition 1

Let $f$ and $g$ be real-valued functions that are defined on the same set of nonnegative real numbers. Then $f$ is of **order at most** $g$, written $f(x)$ is $O(g(x))$, if and only if there exist a positive real number $M$ and a real number $x_0$ such that for all $x$ in the common domain of $f$ and $g$,

$$|f(x)| \leq M \cdot |g(x)|, \text{ where } x > x_0$$

The sentence "$f(x)$ is $O(g(x))$" is also read "$f$ of $x$ is big-oh of $g$ of $x$" or "$g$ is a big-oh approximation for $f$".

# The $O$-Notation

## Example 2

$17x^6 + 20x^3 - x^2 + 8$ is $O(x^6)$.

## Proof.

Let $x \geq 1$. Then

$$|17x^6 + 20x^3 - x^2 + 8| \leq 17|x^6| + 20|x^6| + |x^6| + 8|x^6| = 46|x^6|$$

Let $x_0 = 1$ and $M = 46$. Then

$$|17x^6 + 20x^3 - x^2 + 8| \leq M|x^6| \text{ where } x > x_0$$

Hence by definition, $17x^6 + 20x^3 - x^2 + 8$ is $O(x^6)$. $\square$

# The Θ-Notation

**Definition 3**

Let $f$ and $g$ be two functions defined on the same set of nonnegative real numbers. Then $\underline{f \text{ is of order } g}$, written $\underline{f(x) \text{ is } \Theta(g(x))}$, if and only if

- $f(x)$ is $O(g(x))$ and
- $g(x)$ is $O(f(x))$.

# The $\Theta$-Notation

## Example 4

$17x^6 + 20x^3 - x^2 + 8$ is $\Theta(x^6)$.

## Proof.

It was proved in Example 2 that $17x^6 + 20x^3 - x^2 + 8$ is $O(x^6)$.
Let $x > 1$. Note that

$$20x^3 - x^2 + 8 \geq 20x^3 - x^3 + 8 \geq 19x^3 + 8 > 0.$$

Hence for any $x > 1$, we have

$$x^6 < 17x^6 < 17x^6 + 20x^3 - x^2 + 8.$$

By definition, $x^6$ is $O(17x^6 + 20x^3 - x^2 + 8)$.
The desired conclusion then follows from the definition of the $\Theta$-notation. $\square$

# The Order of a Polynomial

## Proposition 5

*If $a_0, a_1, a_2, \ldots, a_n$ are real numbers and $a_n \neq 0$, then*

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \text{ is } \Theta(x^n).$$

## Proof.

Let $M = |a_n| + |a_{n-1}| + \cdots + |a_0|$. Define $x_0 = 1$. Then for $x > x_0$,

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

By definition, $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ is $O(x^n)$.
It is left as an exercise to prove that $x^n$ is
$O(a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0)$. □

# The Order of a Function of Integer Variable

## Definition 6

Let $f(n)$ and $g(n)$ be functions defined on sets of integers. If there exist a positive integer $M$ and $n_0$ such that

$$|f(n)| \leq M|g(n)|, \quad n > n_0$$

then $f(n)$ is $O(g(n))$.

# Orders for Functions of Integral Variable

## Definition 7

Let $f(n)$ and $g(n)$ be functions defined on sets of integers. Then $\underline{f \text{ is of order } g}$, written $\underline{f(n) \text{ is } \Theta(g(n))}$, if and only if

- $f(n)$ is $O(g(n))$ and
- $g(n)$ is $O(f(n))$.

**Example 8**

Prove that $1^2 + 2^2 + \cdots + n^2$ is $\Theta(n^3)$.

**Proof.**

It is known that

$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} \leq n^3.$$

By definition, $1^2 + 2^2 + \cdots + n^2$ is $O(n^3)$.
On the other hand,

$$n^3 < n(n+1)(2n+1) = 6(1^2 + 2^2 + \cdots + n^2).$$

By definition, $n^3$ is $O(1^2 + 2^2 + \cdots + n^2)$. The desired conclusion follows. $\square$

# The Order of a Function of Integer Variable

## Proposition 9

*Let $f, g, f_1, g_1$ be functions from the set of natural numbers to the set of real numbers.*

1. *If $f$ is $O(g)$, then $f+g$ is $O(g)$.*
2. *If $f$ is $O(f_1)$ and $g$ is $O(g_1)$, then $fg$ is $O(f_1 g_1)$.*

## Proof.

We now prove only the first part. By definition, there is a positive integer $n_1$ and $M_1$ such that

$$|f(n)| \leq M_1 |g(n)| \text{ for all } n > n_1$$

Hence for all $n > n_1$

$$|f(n)+g(n)| \leq |f(n)| + |g(n)| \leq (M_1+1)|g(n)|.$$

By definition, $f+g$ is $O(g)$. □

# The Order of a Function of Integer Variable

## The Second Part of Proposition 9

Let $f, g, f_1, g_1$ be functions from the set of natural numbers to the set of real numbers. If $f$ is $O(f_1)$ and $g$ is $O(g_1)$, then $fg$ is $O(f_1 g_1)$.

## Proof.

By definition, there are positive integers $n_1, n_2, M_1$ and $M_2$ such that

$$|f(n)| \ \leq \ M_1 |f_1(n)| \text{ for all } n > n_1$$

and

$$|g(n)| \ \leq \ M_2 |g_1(n)| \text{ for all } n > n_2$$

Define $n_0 = max\{n_1, n_2\}$ and $M = M_1 M_2$. Then we have for $n > n_0$,

$$|f(n)g(n)| \leq M_1 M_2 |f_1(n) g_1(n)| = M |f_1(n) g_1(n)|$$

By definition, $fg$ is $O(f_1 g_1)$. □

# The Order of a Function of Integer Variable

**Example 10**

By Proposition 9

1. $n+1$ is $O(n^2)$ implies that $n^2+n+1$ is $O(n^2)$.
2. $n^2+n+1$ is $O(n^2)$ and $(n+1)$ is $O(n)$ implies that $(n+1)(n^2+n+1)$ is $O(n^3)$.

# Algorithms

**Definition 11**

An algorithm is a procedure that is used to solve some problem.

**Example 12**

The following procedure is an algorithm for calculating the sum of $n$ given numbers $a_1, a_2, \ldots, a_n$.

Step 1: *Set* $S = 0$;
Step 2: *For* $i = 1$ *to* $n$, *replace* $S$ *by* $S + a_i$;
Step 3: *Output* $S$.

The value of $S$ output at Step 3 is the desired sum.

# The Time and Space Complexity

Algorithms are used to solve problems and each algorithm takes time and memory space.

**Definition 13**

The number of **machine operations** needed in an algorithm is the time complexity of the algorithm, and amount of memory needed is the space complexity of the algorithm.

**Example 14**

Consider the algorithm of Example 12. Step 1 and Step 3 take one machine operation respectively. Step 2 takes $2n$ operations ($n$ additions and $n$ replacements). So altogether this algorithm takes $2n+2$ operations. Note that $2(n+1)$ is $O(n)$. So the time complexity of this algorithm is $O(n)$.

# Horner's Algorithm and Its Complexity

## Example 15

Consider the evaluation of

$$f(x) = 1 + 2x + 3x^2 + 4x^3$$

Direct computation takes 3 additions and 6 multiplications. Another way is:

$$f(x) = 1 + x(2 + x(3 + 4x))$$

So it takes 3 additions and 3 multiplications. Generally, Horner's algorithm for evaluating a polynomial

$$a_0 + a_1 x + \cdots + a_n x^n$$

can be formulated as follows:

Step 1: Set $S = a_n$;

Step 2: For $i = 1$ to $n$, replace $S$ by $a_{n-i} + Sx$;

Step 3: Output S.

The final value of $S$ output at Step 3 is the desired value of $a_0 + a_1 x + \cdots + a_n x^n$.
The number of operations needed in this algorithm is $1 + 3n + 1 = 3n + 2$.
So the time complexity of this algorithm is $O(n)$.

# The Time and Space Complexity

## Example 16

Determine the time complexity of the following algorithm:

```
for i := 1 to n
    for j := 1 to n
        a := 2*n + i*j
    next j
next i
```

## Solution

In the second loop, computing $a := 2 \cdot n + i \cdot j$ takes 4 operations (two multiplications, one addition, and one replacement). For each $i$, it takes $n \times 4 = 4n$ operations to complete the second loop. So it takes $n \times 4n = 4n^2$ operations to complete the two loops. Hence the time complexity of this algorithm is $O(n^2)$.

# The Time and Space Complexity

## Example 17

Determine the time complexity of the following algorithm:

```
S := 0
for i := 1 to n
    for j := 1 to i
        S := S + i*j
    next j
next i
```

## Solution

Computing $S := S + j \cdot i$ takes 3 operations. For each $i$, completing the second loop takes $3i$ operations. So altogether it takes

$$1 + \sum_{i=1}^{n} 3i = 1 + 3\frac{n(n+1)}{2}$$

operations. hence the complexity of this algorithm is $O(n^2)$.