MA633L-Numerical Analysis

Lecture 3 : Asymptotic Notations and Computation Time

Panchatcharam Mariappan[1]

[1]Associate Professor
Department of Mathematics and Statistics
IIT Tirupati, Tirupati

January 9, 2025

भारतीय प्रौद्योगिकी संस्थान तिरुपति
TIRUPATI

Asymptotic Notations

Big O

Definition 1 (Big O)
Let $\{x_n\}$ and $\{\alpha_n\}$ be two different sequences. We write

$$x_n = O(\alpha_n)$$

if there exist constants C such that

$$|x_n| \leq C|\alpha_n| \quad \forall\, n \geq n_0.$$

If $\alpha_n \neq 0$ for all n, that is,

$$\lim_{n\to\infty} \left|\frac{x_n}{\alpha_n}\right| \leq C$$

Here, we say x_n is big O of α_n.

Little o

Definition 2 (Little o)
We write

$$x_n = o(\alpha_n)$$

if $C = 0$. That is,

$$\lim_{n\to\infty} \left| \frac{x_n}{\alpha_n} \right| = 0$$

Here, we say x_n is little o of α_n.

Let $x_n \to 0$ and $\alpha_n \to 0$.

1. If $x_n = O(\alpha_n)$, then x_n converges to 0 at least as rapidly as α_n.
2. If $x_n = o(\alpha_n)$, then x_n converges to 0 more rapidly than α_n.

Little o and Big O

Example 3

Verify whether the following is true or not?

$$
\begin{aligned}
\frac{n+1}{n^2} &= O\left(\frac{1}{n}\right) \\
\frac{1}{n\ln n} &= o\left(\frac{1}{n}\right) \\
\frac{1}{n} &= o\left(\frac{1}{\ln n}\right) \\
\frac{5}{n} + e^{-n} &= O\left(\frac{1}{n}\right) \\
10\ln(n) + 5(\ln(n))^3 + 7n + 3n^2 + 6n^3 &= O(n^3)
\end{aligned}
$$

Little o and Big O

Example 4

Verify whether the following is true or not?

$$
\begin{aligned}
e^{-n} &= o\left(\frac{1}{n^2}\right) \\
\ln 2 - \sum_{k=1}^{n-1} (-1)^{k-1} \frac{1}{k} &= O\left(\frac{1}{n}\right) \\
e^{x} - \sum_{k=0}^{n-1} x^{k} \frac{1}{k!} &= O\left(\frac{1}{n!}\right) \qquad (|x| \leq 1)
\end{aligned}
$$

Little o and Big O

Definition 5 (Big O)
Let f and g be two real valued functions. We write

$$f(x) = O(g(x)), \quad \text{as} \quad x \to \infty$$

if there exists constant C and r such that

$$|f(x)| \leq C|g(x)| \quad \forall\, x \geq r.$$

Little o and Big O

Definition 6 (Big O)

Let f and g be two real valued functions. We write

$$f(x) = O(g(x)), \quad \text{as} \quad x \to x_0$$

if there exists constant C and r such that

$$|f(x)| \leq C|g(x)| \quad \forall\, x \ \text{ with } \ |x - x_0| < r.$$

Little o and Big O

Definition 7 (Little o)
Let f and g be two real valued functions. In general, we write

$$f(x) = o(g(x)), \quad \text{as} \quad x \to x_0$$

if

$$\lim_{x \to x_0} \left| \frac{f(x)}{g(x)} \right| = 0$$

Little o and Big O

Example 8

Verify whether the following is true or not?

$$
\begin{aligned}
\sin x - x + \frac{x^3}{6} &= O(x^5) \quad (x \to 0) \\
\sqrt{x^2+1} &= O(x) \quad (x \to \infty) \\
e^x - 1 &= O(x^2) \quad (x \to 0) \\
\cot x &= o(x^{-1}) \quad (x \to 0)
\end{aligned}
$$

Ω and ω

Definition 9 (Big Ω)

Let f and g be two real valued functions. In general, we write

$$f(x) = \Omega(g(x)), \quad \text{as} \quad x \to x_0$$

if there exist constants C and r such that

$$|f(x)| \geq C|g(x)| \quad \forall\, x \geq r.$$

Ω and ω

Definition 10 (Little ω)
Let f and g be two real valued functions. In general, we write

$$f(x) = \omega(g(x)), \quad \text{as} \quad x \to x_0$$

$$\text{if} \quad \lim_{x \to x_0} \left| \frac{f(x)}{g(x)} \right| = \infty$$

Θ and $\sim$

Definition 11 (Big Θ)

Let f and g be two real valued functions. In general, we write

$$f(x) = \Theta(g(x)), \quad \text{as} \quad x \to x_0$$

if there exist constants C_1, C_2 and r such that

$$C_1|g(x)| \leq |f(x)| \leq C_2|g(x)| \quad \forall\, x \geq r.$$

Θ and $\sim$

Definition 12 (Similar)
Let f and g be two real valued functions. In general, we write

$$f \sim g \quad \text{as} \quad x \to x_0$$

if

$$\text{if} \quad \lim_{x \to x_0} \left| \frac{f(x)}{g(x)} \right| = 1$$

o, O, Ω, ω, Θ and $\sim$

- $f = O(g)$ means that g describes the upper bound for f
- $f = o(g)$ means that g describes the upper bound for f, but f can never be equal to g
- $f = \Omega(g)$ means that g describes the lower bound for f
- $f = \Theta(g)$ means that g describes the exact bound for f
- $f \sim g$ means that f/g approaches 1

o, O, Ω, ω, Θ and $\sim$

- $f = O(g)$ means that f grows no faster than g
- $f = o(g)$ means that f grows slower than g
- $f = \Omega(g)$ means that f grows at least as fast as g
- $f = \omega(g)$ means that f grows faster than g
- $f \sim g$ means that f/g approaches 1

o, O, Ω, ω, Θ and $\sim$

Notation	Comparison	Limit Definition
$f = O(g)$	$f \leq g$	$\lim_{x \to x_0} \left\lvert \frac{f(x)}{g(x)} \right\rvert < \infty$
$f = o(g)$	$f < g$	$\lim_{x \to x_0} \left\lvert \frac{f(x)}{g(x)} \right\rvert = 0$
$f = \Theta(g)$	$f = g$	$\lim_{x \to x_0} \left\lvert \frac{f(x)}{g(x)} \right\rvert \in (0, \infty)$
$f = \Omega(g)$	$f \geq g$	$\lim_{x \to x_0} \left\lvert \frac{f(x)}{g(x)} \right\rvert > 0$
$f = \omega(g)$	$f > g$	$\lim_{x \to x_0} \left\lvert \frac{f(x)}{g(x)} \right\rvert = \infty$
$f \sim g$	$f \sim g$	$\lim_{x \to x_0} \left\lvert \frac{f(x)}{g(x)} \right\rvert = 1$

o, O, Ω, ω, Θ and $\sim$

The following observations are immediate

Theorem 13

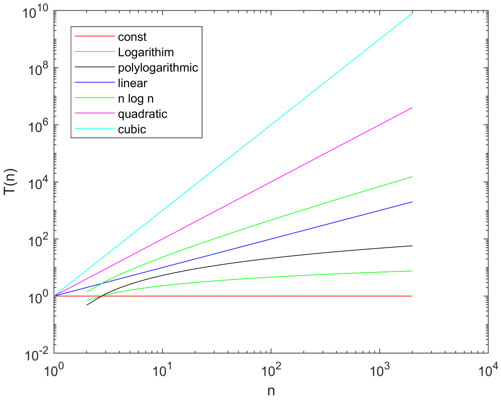
$$
\begin{aligned}
f = O(g) \quad \text{and} \quad f = \Omega(G) &\iff f = \Theta(g) \\
f = O(g) &\iff g = \Omega(f) \\
f = o(g) &\iff g = \omega(f) \\
f = o(g) &\implies f = O(g) \\
f = \omega(g) &\implies f = \Omega(g) \\
f \sim g &\implies f = \Theta(f) \\
(\ln n)^k = o(n^\alpha) \quad &\text{and} \quad n^k = o((1+\alpha)^n), \forall k, \alpha > 0
\end{aligned}
$$

Big O

Notation	Name
$O(1)$	Constant
$O(\log(n))$	logarithmic
$O((\log(n))^c)$	polyalgorithmic
$O(n)$	linear
$O(n^2)$	quadratic
$O(n^c)$	polynomial
$O(c^n)$	exponential

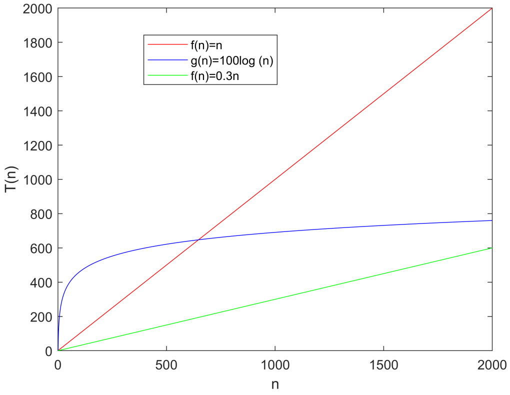
Note that $O(n^c)$ and $O(c^n)$ are different, $O(c^n)$ grows much faster. A function that grows faster than any power of n is called superpolynomial and that grows slower than an exponential function c^n is called subexponential.

Big O Graph



const
Logarithim
polylogarithmic
linear
n log n
quadratic
cubic
T(n)
n
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
10^{0}
10^{-2}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}

Big O Graph



2000
1800
1600
1400
1200
1000
800
600
400
200
0
T(n)
0
500
1000
1500
2000
n
f(n)=n
g(n)=100log (n)
f(n)=0.3n

Computation Time

Algorithm and Computer Program

Definition 14 (Algorithm)
An algorithm is a list of unambiguous rules that specify successive steps to solve a problem.

Definition 15 (Computer Program)
The computer program is clearly specified sequence of computer instructions implementing algorithm.

Elementary Operations

Definition 16 (Elementary Operations)
Most modern computers and languages build complex programs from ordinary arithmetic and logical operations such as standard unary and binary operations (negation, addition, subtraction, multiplication, division, modulo operations, assignment), boolean operations, binary comparisons ($=, >, <, \geq, \leq$), branching operations. These operations are called as elementary operations.

Running Time

Definition 17 (Running Time)
The running time or computing time of an algorithm is the number of its elementary operations. It is denoted by $T(n)$.

For this course, let us consider only addition, subtraction, multiplication and division as elementary operations, for simplicity. Also, let us ignore the indexing sum.

FLOPS

Definition 18 (Floating Point Operation)
A floating point operation means that the arithmetic mathematical computation is accomplished in floating point numbers that may include addition, subtraction, multiplication or division.

Definition 19 (FLOPS)
FLoating point **O**perations **P**er **S**econd (FLOPS)

FLOPS

- A major factor in the comparison of the computational power of different systems
- In particular, it is an important factor where numerical calculations are a key point
- It measures the number of operations that a system can execute in terms of one second computation power
- One of the major measure to assess whether a computer is supercomputer or not

Running Time : Sum of Elements of an array

Example 20

Let a denote an array or list of integers where the sum

$$s = \sum_{i=0}^{n-1} a[i]$$

is required. To get the sum s, we need to repeat n times the same elementary operations. Therefore, the running time $T_1(n)$ is proportional to or linear in n. That is $T_1(n) = cn$. This algorithm is called linear algorithm. The unknown factor c depends on a particular computer, programming language, compiler, OS etc.

Running Time: $T_1(n)$

Algorithm 1: Linear Sum

Input: array, $a[0, 1, \cdots, n-1]$

Output: s

```
s ← 0
for i ← 0 to n − 1 do
    s ← s + a[i]
```

In the above algorithm, suppose $T_1(1)$ is given to you, then you can compute $T_1(1000) = 1000T_1(1) = 10T_1(100)$. If per addition, it takes 1s, then $T_1(1) = 1$, then $T_1(1000) = 1000s$.

Sum of Elements of Subarrays

Example 21

Now, let us compute the sum of each subarray of some m. That is,

$$s_j = \sum_{k=0}^{m-1} a[j+k], j = 0, 1, 2, \cdots, n-m$$

1. How many subarrays are there in this sum?
2. Prove or disprove $T_2(n) = cm(n-m+1)$.
3. Also, if $m = \frac{n}{2}$, prove or disprove $T_2(n) = 0.25cn^2 + 0.5cn = O(n^2)$.

Running Time: $T_2(n)$

Algorithm 2: Quadratic Algorithms: Slow Sum

Input: array, $a[0, 1, \cdots, 2m-1]$
Output: $s[0, 1, \cdots, m]$

$s \leftarrow 0$
for $i \leftarrow 0$ **to** m **do**
    $s[i] \leftarrow 0$
    for $j \leftarrow 0$ **to** $m-1$ **do**
        $s[i] \leftarrow s[i] + a[i+j]$

Running Time: $T_3(n)$

Algorithm 3: Quadratic algorithms: Fast Sum

Input: array, $a[0, 1, \cdots, 2m-1]$

Output: $s[0, 1, \cdots, m]$

$s[0] \leftarrow 0$

for $j \leftarrow 0$ **to** $m-1$ **do**

  $s[0] \leftarrow s[0] + a[j]$

for $i \leftarrow 1$ **to** m **do**

  $s[i] \leftarrow s[i-1] + a[i+m-1] - a[i-1]$

Running Time: T_1, T_2, T_3

n	$T_1(n)$	**Minutes**	$T_2(n)$	**Minutes**	$T_3(n)$	**Minutes**
100						
500						
1000						
5000						
50000						

Table 1: $T_1(n), T_2(n), T_3(n)$

Norm of a Vector

Example 22

Let $x = (x_1, x_2, \cdots, x_n)$ be a vector in $\mathbb{R}^n$.

$$\|\mathbf{x}\|_2^2 = \sum_{i=1}^{n} |x_i|^2 \tag{1}$$

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{2}$$

$$\|\mathbf{x}\|_p^p = \sum_{i=1}^{n} |x_i|^p \tag{3}$$

Compute the time requirement for each of the above sum when $n = 10^{12}$.

Exercise

Exercise 1: Medium

1. Compute the $T_1(10^{12})$, $T_2(10^{12})$ and $T_3(10^{12})$ without doing the array sum operation.
2. Estimate the value of c from the table
3. Let a denote an array or list of real number and

$$s(x) = \sum_{i=0}^{n-1} a_i x^i$$

is required. Let $T_4(n)$ be the running time to compute $s(x)$. Compute the $T_4(10^{12})$ without doing the array sum operation.

Exercise

Exercise 2: Medium

1. Compute the $T_1(10^{12})$, $T_2(10^{12})$ and $T_3(10^{12})$ without doing the array sum operation.
2. Estimate the value of c from the table
3. Let a denote an array or list of real number and

$$s(x) = \sum_{i=0}^{n-1} a_i x^i$$

is required. Let $T_4(n)$ be the running time to compute $s(x)$. Compute the $T_4(10^{12})$ without doing the array sum operation.

Exercise

Exercise 3: Medium

4. Let $S_1(x)$ be the Taylor polynomial for $\sin(x)$

$$S_1(x) = \sin x = \sum_{i=0}^{n-1} (-1)^i \frac{x^{2i+1}}{(2i+1)!}$$

5. For given n and x, compute $S_1(x)$
6. Compare the value of $S_1(x)$ and $\sin x$ value obtained using the Python numpy library

Exercise

Exercise 4: Medium

7. Let $T_5(n, x)$ be running time for $S_1(x)$. Fill out the following table

x	n	$S_1(x)$	np.sin(x)	$T_5(n, x)$	Seconds
$\pi/3$	20				
$\pi/4$	20				
$\pi/6$	20				
$\pi/2$	20				
$\pi/3$	50				
$\pi/3$	100				

Table 2: $\sin(x)$

Exercise

Exercise 5: Medium

8. Repeat exercise 7 for the Taylor polynomial corresponding to $\cos(x), e^x, \tan^{-1}(x), \cos^{-1}(x), Li_2(x)$ [make changes in x, if necessary]
9. Convert the table entries to years. How many years will take for $T_i(n), i = 1, 2, \cdots, 8$ if $n = 10^6$?
10. Suppose your computer can perform 10^3 [kFLOPS] operations (+,-,*,/) per second, how many years will it take for finding the sum using algorithm 2 and algorithm 3, when $n = 10^9$?
11. Repeat exercise 10, if computer can do 10^6 operations [MFLOPS], 10^9 operations [GFLOPS] and 10^{12} operations [TFLOPS].

Thanks

Doubts and Suggestions

panch.m@iittp.ac.in

MA633L-Numerical Analysis

Lecture 3 : Asymptotic Notations and Computation Time

Panchatcharam Mariappan[1]

[1]Associate Professor
Department of Mathematics and Statistics
IIT Tirupati, Tirupati

January 9, 2025

भारतीय प्रौद्योगिकी संस्थान तिरुपति
TIRUPATI