# Week 02 - Lecture 3 Slides

## Dr. Yeganeh Bahoo

Created: 2023-09-17 Sun 11:29

## Table of Contents



## Lecture 3 - algorithm analysis (in a nutshell)

Learning objectives

By the end of this lecture you should be able to:

1. To describe why algorithm analysis is important.
2. Obtain the order of magnitude (aka order of growth) of a function.

### Algorithm Analysis: motivation

- Given two different programs that compute the summation of a number $n$, i.e.:

$$\sum_{i=1}^{n} i$$

  which one is "better"?

```
(defun sum-n (n)
  (do ((i 1 (1+ i))
       (sum 0 (+ i sum)))
      ((> i n) sum)))
```

```
(defun foo (tom)
  (do ((bill 1
      (1+ bill))
      (fred 0 (+ fred bill)))
      ((> bill tom) fred)))
```

## Algorithm analysis

- Comparison of programs depends on the criteria used for the comparison:
  - readability
  - the algorithm itself
- In Algorithm Analysis, as the name implies, we focus on the algorithm.
- In particular, we are concerned with comparing algorithms based upon the amount of **computing resources** that each algorithm uses.

## Computing resources

- Two ways of looking at what "computing resources" an algorithm requires to solve a problem
  - The amount of **memory**
  - The amount of **time**, usually referred as "execution time" or "running time"

Let's then consider this alternative way to compute the summation of an integer $n$:

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

Now, considering the "running time" computing resource, which of the following functions is better?

## Computing resource: running time

Suppose $n$ is a big number, which of the following functions is better?

```
;; 1

(defun sum-n1 (n)
  (do ((i 1 (1+ i))
       (sum 0 (+ i sum)))
      ((> i n) sum)))
```

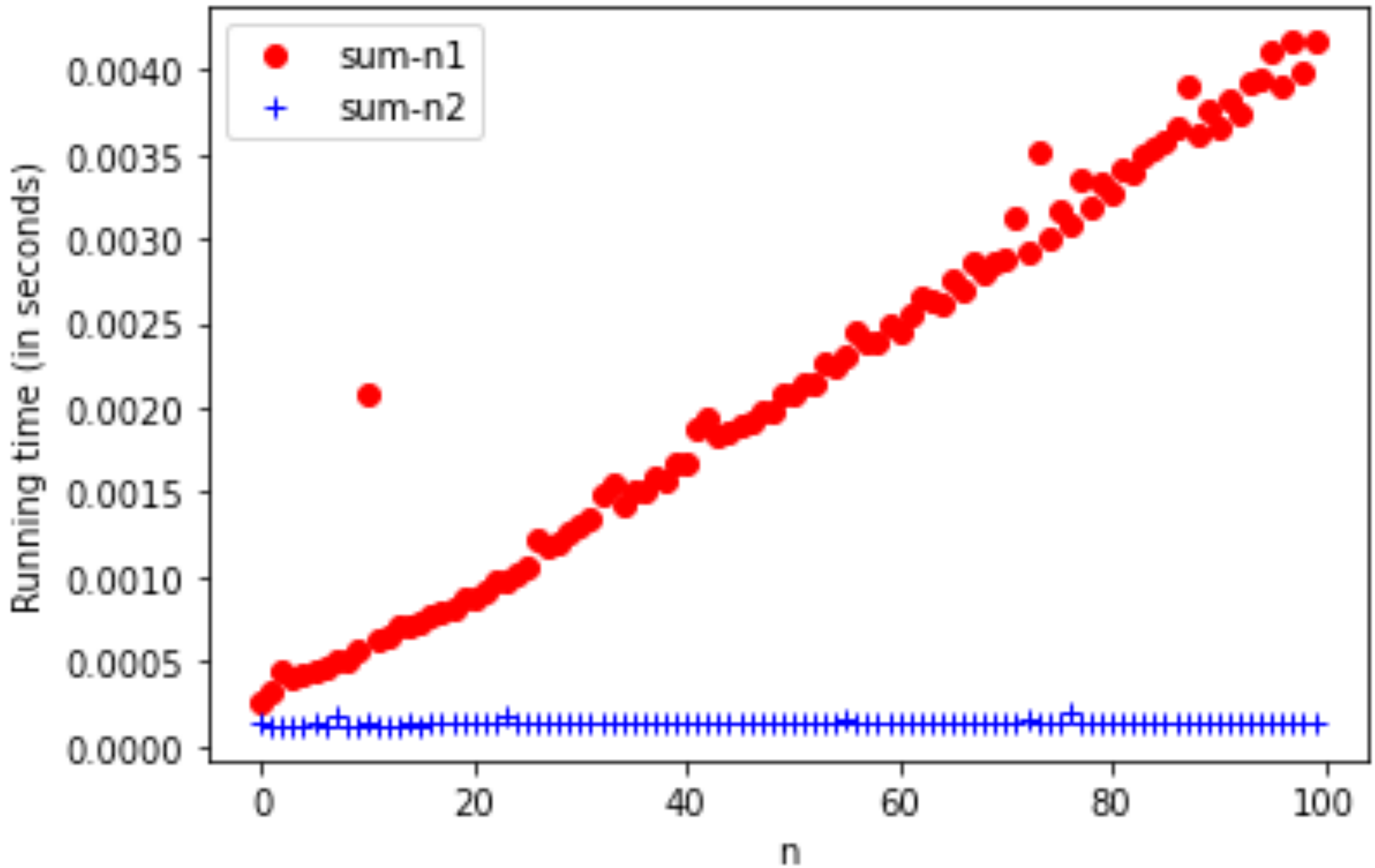
```
;; 2

(defun sum-n2 (n)
  (/ (* n (+ n 1)) 2.0))
```

## Running time

```
(defun sum-n1 (n)
  (do ((i 1 (1+ i))
       (sum 0 (+ i sum)))
      ((> i n) sum)))
```

```
(defun sum-n2 (n) (/ (* n (+ n 1)) 2))
```




**Algorithm analysis: conclusions so far**

- The times recorded for SUM-N2 are shorter than SUM-N1.
- They are very consistent no matter what the value of $n$.
- But what does this benchmark technique tell us?
  - apparently solutions involving a loop over $n$ take longer as we increase $n$
  - but would we get the same result if we run the same function in a different computer?

    - the actual execution time does not really provide a useful measurement.
- We need a characterization of execution time **of algorithms** that is independent of the program or computer.

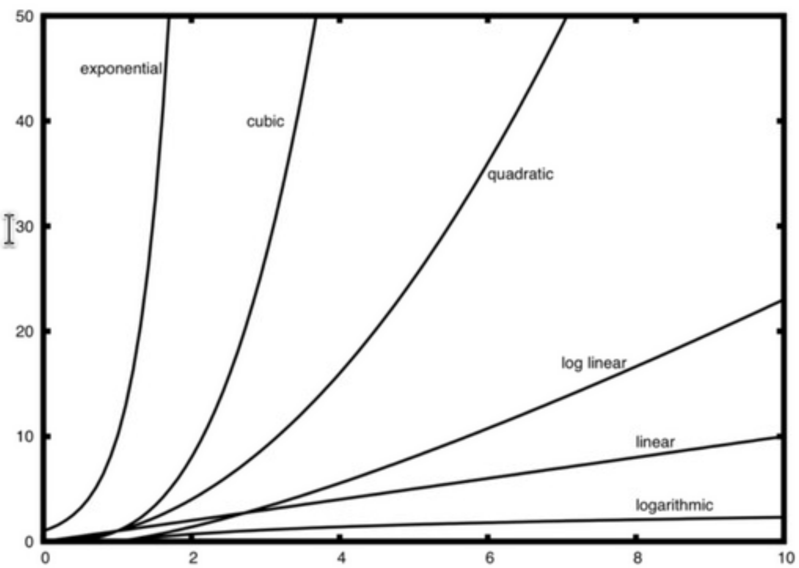
## Big-O notation

- In general, the running time of an algorithm grows with the size of the input.
- The notion for **input size** depends on the problem being studied (e.g.?)
- We can use a function $T(n)$ to represent the **running time** ("the number of steps") of an algorithm on a input of size $n$. E.g.:

$$T(n) = 5n^2 + 27n + 1005$$

- The **Order of Magnitude** function, $O(f(n))$ describes the part $f(n)$ of $T(n)$ that increases the fastest when $n$ grows.
- Notice:
    - when $n$ gets larger, the term $n^2$ becomes the most important
    - Therefore the running time $T(n)$ above has an order of magnitude $O(n^2)$.

## Common big-O functions




| f(n) | Name |
|---|---|
| 1 | Constant |
| $\log n$ | Logarithmic |
| $n$ | Linear |
| $n \log n$ | Log Linear |
| $n^2$ | Quadratic |
| $n^3$ | Cubic |
| $2^n$ | Exponential |

## Obtaining the order of magnitude

Steps: Given a function $T(n)$

1. Identify which term of $T(n)$ increases its value the fastest when $n$ grows
2. Remove the other terms from $T(n)$

Example:

| | | | |
|---|---|---|---|
| $T(n)$ | $36\overbrace{n^2 \log n} + 3 \log n^3 + 4$ | | $3n \log n^3 + 360 \overbrace{n^2} + 40$ |
| $O(T(n))$ | $n^2 \log n$ | > | $n^2$ |

Example:

| | | | |
|---|---|---|---|
| $T(n)$ | $\frac{n^{10}}{n^8+10} \log n^3 + 35 \overbrace{n^3} + 40$ | | $36n^2 \log n + \overbrace{3^n} + 4$ |
| $O(T(n))$ | $n^3$ | < | $3^n$ |

## Exercise

Provide the order of magnitude (aka Order of growth, or Big O) for the following functions and indicate which one has larger order of growth.

| | | | |
|---|---|---|---|
| $T(n)$ | $36n^{11} \log n + 78$ | | $43 \log n^{20} + 68$ |
| $O(T(n))$ | ... | ... | ... |

### Solution

| | | | |
|---|---|---|---|
| $T(n)$ | $36 \overbrace{n^{11} \log n} + 78$ | | $43 \overbrace{\log n^{20}} + 68$ |
| $O(T(n))$ | $n^{11} \log n$ | > | $\log n$ |