ENG 346
Data Structures and Algorithms for Artificial Intelligence
Runtime Complexity of the Algorithms

Dr. Kürşat İnce
kince@gtu.edu.tr

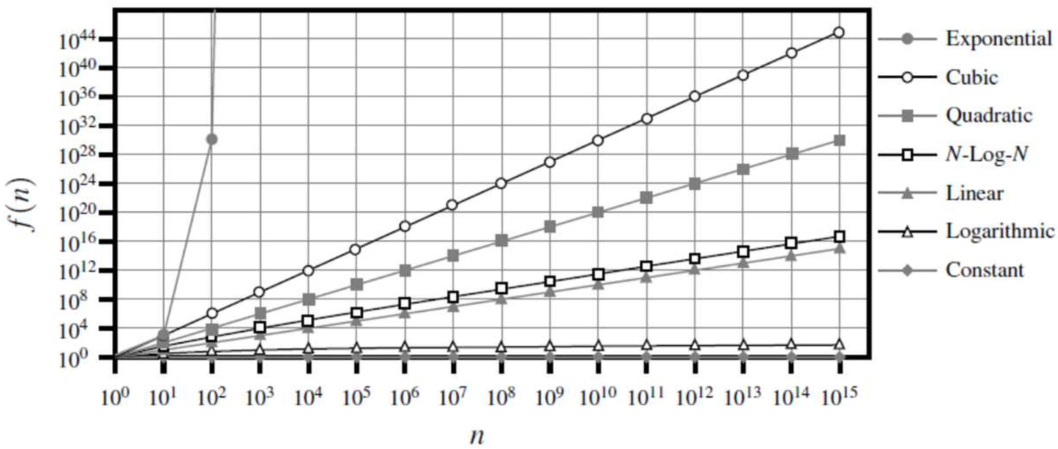
Agenda

- Big O
- Big Ω
- Big Θ
- Algorithm Analysis

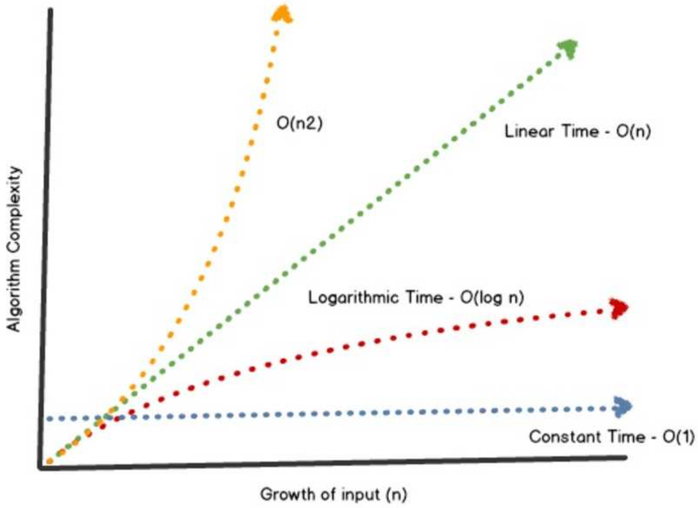
Basics

Name	Function	Relation	Example
Constant Time	f(n) = c	Does not depend on input size.	Accessing array elements.
Logarithmic Time	f(n) = log n	Running time increases logarithmically with the input size.	Binary search.
Linear Time	f(n) = n	Running time increases linearly with the input size.	Iterating through an array or list.
Linearithmic Time	f(n) = n log n	The running time grows slower than O(n^2) but faster than O(n).	Efficient sorting algorithms like quicksort and mergesort.
Quadratic Time	f(n) = n^2	Running time grows proportionally to the square of the input size.	Algorithms with nested loops, such as selection sor tor bubble sort.
Polynomial Time	f(n) = n^k	Running time is a polynomial function of the input size.	Algorithms with “k” nested loops.
Exponential Time	f(n) = 2^n	Running times that grow very rapidly with the input size.	N-P complete problems, such as traveling salesman.

Growth Rates



Growth Rates



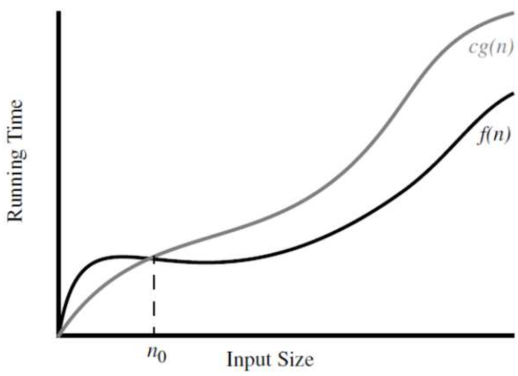
Definitions: Big O

- Upper-bound of a function f(n)

- Let f(n) and g(n) be functions mapping positive integers to positive real numbers. We say that f(n) is O(g(n)) if
  - there is a real constant c > 0 and
  - an integer constant n0 ≥ 1 such that

    f(n) ≤ c g(n), for n ≥ n0.

- f(n) is O(g(n))



Definitions: Big Ω

- Lower-bound of a function f(n)

- Let f(n) and g(n) be functions mapping positive integers to positive real numbers. We say that f (n) is Ω(g(n)), pronounced " f (n) is big-Omega of g(n)," if g(n) is O(f (n)),
  - there is a real constant $c > 0$ and
  - an integer constant $n0 \geq 1$ such that

    $f(n) \geq cg(n)$, for $n \geq n0$.

- f(n) is Ω(g(n))

Definitions: Big Θ

- Two functions grow at the same rate, up to constant factors. We say that f (n) is Θ(g(n)), pronounced " f (n) is big-Theta of g(n)," if
  - f (n) is O(g(n)) and
  - f (n) is Ω(g(n)) and
  - there are real constants $c1 > 0$ and $c2 > 0$, and an integer constant $n0 \geq 1$ such that

    $c1g(n) \leq f(n) \leq c2g(n)$, for $n \geq n0$.

- f(n) is Θ(g(n))

Examples:

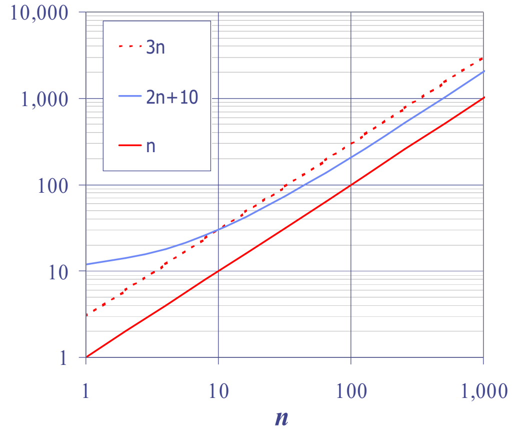
2n + 10 is O(n)

$2n + 10 \le cn$

$(c - 2)\, n \ge 10$

$n \ge 10/(c - 2)$

Pick c = 3 and n0 = 10



Examples:

- 7n-2 is O(n)
  - need c > 0 and n0 ≥ 1 such that 7n-2 ≤ c n for n ≥ n0
  - this is true for c = 7 and n0 = 1
- $3n^3 + 20n^2 + 5$ is $O(n^3)$
  - need c > 0 and $n_0 \ge 1$ such that $3n^3 + 20n^2 + 5 \le c\, n^3$ for $n \ge n_0$
  - this is true for c = 4 and n0 = 21
- 3 log n + 5 is O(log n)
  - need c > 0 and n0 ≥ 1 such that 3 log n + 5 ≤ c log n for n ≥ n0
  - this is true for c = 8 and n0 = 2

Big O Rules

- Simplifications:

- If is f(n) a polynomial of degree d, then f(n) is O(n^d), i.e.,
  - Drop lower-order terms
  - Drop constant factors
- Use the smallest possible class of functions
  - Say “2n is O(n)” instead of “2n is O(n^2)”
- Use the simplest expression of the class
  - Say “3n + 5 is O(n)” instead of “3n + 5 is O(3n)”

Exercises

- Book: R-3.1