Operations Research

Safia Kedad-Sidhoum

safia.kedad_sidhoum@cnam.fr

Cnam

2025-2026

Quick Overview and Organization (1/2)

- Welcome to USEEN3 : Operations Research !
- Academic Team : Safia Kedad-Sidhoum and Daniel Porumbel.
- Objectives and program :
  - Modeling combinatorial optimization problems
  - Linear programming
  - Integer linear programming
  - Shortest path algorithms
  - Minimum spanning tree algorithms
  - Network flow algorithms
  - Applications : Routing and traffic, Network design,sudo snap install typora Network connectivity and reliability, Energy consumption

Quick Overview and Organization (2/2)

- Lesson 1 - Tue. 07/10
  Introduction to networks models / computational complexity
- Lesson 2-3 - Tue. 14/10 & Tue. 21/10
  Graph search algorithms - Application : Network accessibility
- Lesson 4 - Tue. 28/10
  Shortest paths (Dijkstra) - Application : Routing tables
- Lesson 5 - Tue. 4/11
  Shortest paths (Bellman/Ford)
- Lesson 6 - Tue. 18/11
  Minimum spanning trees - Application : Network design
- Lesson 7 - Tue. 25/11
  Linear Programming
- Lesson 8-9 - Tue. 02/12 & Tue. 09/12
  Integer Programming
- Lesson 10 - Tue. 16/12 :
  Maximum flow problem - Min-cost flow - Application : Network flow routing

  Exam Tue. 6/01/2026

References

Books...

- The Algorithms Illuminated Book Series, Tim Roughgarden, 2017.
- Algorithm Design, Eva Tardos and Jon Klein, 2005.
- Network Flows : Theory, Algorithms, and Applications, Ravindra Ahuja, Thomas Magnanti, James Orlin, 1993

Basics of Algorithm Analysis
Why Study Algorithms ?

- Important for all other branches of computer science.
  - Routing protocols in communication networks piggyback on classical shortest path algorithms.
  - Public-key cryptography relies on efficient number-theoretic algorithms.
  - Computer graphics requires the computational primitives supplied by geometric algorithms.
  - Database indices rely on balanced search tree data structures.
  - Computational biology uses dynamic programming algorithms to measure genome similarity.
- Driver of technological innovation (PageRank algorithm for computing the relevance of various Web pages to a given search query).
- Lens on other sciences.

Guiding Principles for the Analysis of Algorithms

- Worst-Case Analysis : gives a running time bound that is valid even for the “worst” inputs
- “Big-picture” analysis : balance predictive power with mathematical tractability by ignoring constant factors and lower-order terms
- Asymptotic Analysis : focus on the rate of growth of an algorithm’s running time, as the input size n grows large

Algorithm efficiency ?

What is a Fast Algorithm ?
A fast algorithm is an algorithm whose worst-case running time grows slowly with the input size

Asymptotic notation provides the basic vocabulary for discussing the design and analysis of algorithms.

Asymptotic notation

- High level idea : Suppress constant factors (too system-dependent) and lower-order terms (irrelevant for large inputs)

Examples (Simple)

- Searching an array
- Searching two arrays
- Checking for a Common Element
- Checking for Duplicates

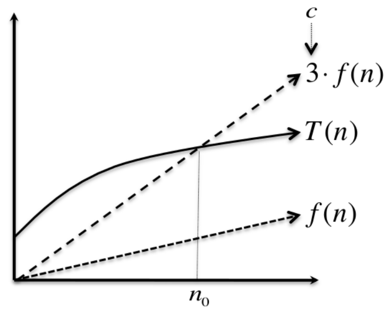
Codes + Quizz + Discussions

O notation (1/2)

It concerns functions $T(n)$ defined on the positive integers $n = 1, 2, \ldots$.

Definition (Intuitive)

$T(n) = O(f(n))$ if and only if $T(n)$ is eventually bounded above by a constant multiple of $f(n)$.



$T(n)$ will denote a bound on the worst-case running time of an algorithm, as a function of the size n of the input.

O notation (2/2)

Definition (Mathematical)

$T(n) = O(f(n))$ if and only if there exist positive constants c and n_0 such that $T(n) \leq cf(n)$ for all $n \geq n_0$

For all $n \geq n_0$ expresses that the inequality only needs to hold eventually, once n is sufficiently large (with the constant n_0 specifying how large)

Remark : c and n_0 are constants, they cannot depend on n.

Property

Suppose $T(n) = a_k n^k + \cdots + a_1 n + a_0$, where $k \geq 0$ is a nonnegative integer and the a_i's are real numbers (positive or negative). Then $T(n) = O(n^k)$

Proof.

Ω and Θ notations (1/2)

Definition (Ω)

$T(n) = \Omega(f(n))$ if and only if there exist positive constants c and n_0 such that $T(n) \geq cf(n)$ for all $n \geq n_0$

Definition (Θ)

$T(n) = \Theta(f(n))$ if and only if there exist positive constants c_1, c_2 and n_0 such that $c_1 f(n) \leq T(n) \leq c_2 f(n)$ for all $n \geq n_0$

Algorithm designers often use O notation even when Θ notation would be more accurate. This course will follow that line as we generally focus on upper bounds about how long our algorithms could possibly run.

Example (Quizz)

Ω and Θ notations (2/2)

Example (Additional examples)

- Adding a Constant to an Exponent
- Multiplying an Exponent by a Constant
- Maximum vs. Sum

Running Time

The running times of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second.
Very long : exceeds 10^{25} years

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	4 sec
$n = 30$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	18 min	10^{25} years
$n = 50$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	11 min	36 years	very long
$n = 100$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	$<$ 1 sec	$<$ 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	$<$ 1 sec	$<$ 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	$<$ 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long