Advanced Algorithm Design (AAD)

Prof. G. Pucci

Classes

Mondays 14:30 - 15:15, 15:20 - 16:05 Ee
Tuesdays 12:30 - 13:15, 13:20 - 14:05 Pe
Wednesdays 10:30 - 11:15, 11:20 - 12:05 Pe

5 min BREAK

Office Hours (DEI/G, room 414)

Mondays: 11:30 - 13:00
compulsory reservation by (same) monday morning
via mail: geppino.pucci@unipd.it

MOODLE: registration is compulsory! Follow regularly for announcements, slides, etc.

Password: AAD2425

WEBPAGE: www.dei.unipd.it/~geppo/AAD

Surf to get to know the sections

Advanced Algorithm Design (AAD)

First Semester, 2024/25
Laurea Magistrale in Computer Engineering

- Instructor ← Info about me, office hours
- Syllabus ← High-level description of course topics
- Textbooks
- Lectures Diary A.A. 2024/25 Lecture-by-lecture - This year
- Lectures Diary A.A. 2023/24 Lecture-by-lecture - Last year
- Exam Rules
- Exam Sessions ← Dates/rooms already fixed!
- Online Material ← Supplementary material

INFO ALSO ACCESSIBLE FROM MOODLE

Objective: Integrate and expand the knowledge acquired in an introductory course on algorithms
The material is mainly theoretical but has enormous practical impact

Topics

- NP-Completeness: more concrete approach (non determinism + complexity)
  - REDUCTION TECHNIQUES
- Approximation algorithms for hard problems (solutions of guaranteed quality)
- Computational aspects of number theory (divisibility, modular arithmetic, primality) and cryptographic applications of hardness
- Randomization: general techniques for designing randomized algorithms

IMPORTANT: Emphasis on rigorous proof techniques, general paradigms but will ALWAYS discuss the practical relevance of the material

(MAIN) TEXTBOOK

- CLRS (CORMEN, LEISERSON, RIVEST, STEIN) (Introduction to Algorithms 3rd ed)
- Abundant supplementary material (see web page)

EXAM: written, two parts

Part 1: theory questions on covered material (proofs, definitions, analyses) 30%

Part 2: creative exercises apply techniques to new problems (lots of exercise sessions during lectures) 70%

Voluntary TEAMWORK: groups of 2-3 students. Read and understand advanced material, prepare a short written report plus class-presentation (with slides)

EVALUATION: Exemption from Part 1 and 30% of final grade

IMPORTANT: Embark into teamwork only if strongly motivated

→ team creation + assignment first weeks of december 2023

→ presentation must be done in January (last week before finals)

PREREQUISITES
(DISCRETE MATH & ALGORITHMS)

- Asymptotic notation: O, Ω, Θ
- Summations, basic probability (distributions, averages, variance random variables)
- Divide-and-conquer: design (recursion), analysis (recurrences)
- Elements of Dynamic Programming and Greedy

If these topics are not in your background you should acquire them by yourself (use CLRS: Chaps 3, 4, 15, 16, App A, B, C)

I will be glad to help students with difficulties (use office hours / emails)

END (course intro)

COMPLEXITY CLASSES AND THEORY OF INTRACTABILITY (NP-COMPLETENESS)

Objective: CATEGORIZE computational problems according to the complexity of the algorithms for their solution

RECAP: Computational Problem

$\Pi \subseteq \mathcal{I} \times \mathcal{S}$ (relation)

$(i,s) \in \Pi$ [$i \Pi s$] $\leftrightarrow$ s is a solution to i

one-to-many!

Algorithm that solves Π: implements

$A_\Pi : \mathcal{I} \to \mathcal{S}$

$\forall i \in \mathcal{I}: \; i \; \Pi \; A_\Pi(i) \in \mathcal{S}$

We want to study upper/lower bounds (on algorithms) for Π:

- $\exists A_\Pi$ solving $\Pi : T_A(n) = O(f(n))$ — UPPER BOUND
- $\forall A_\Pi$ solving $\Pi : T_A(n) = \Omega(f(n))$ — LOWER BOUND (HARDER!)

The order of $f(n)$ captures the difficulty of Π

PROBLEM CATEGORIES

1. Problems admitting polynomial algorithms:
   $\exists A_\pi : T_{A_\pi}(n) = O(n^k), k \geq 0$

TRACTABLE

2. Problems admitting an exponential lower bound:
   $\exists c > 1 : \forall A_\pi : T_{A_\pi}(n) : \Omega(c^n)$

PROVABLY INTRACTABLE

3. Problems not admitting any algorithm (e.g. HALTING PROBLEM)

UNDECIDABLE / UNSOLVABLE

Π_{HALT} : $I = \{\langle M, x\rangle$, M Turing Machine, $x \in \{0,1\}^*\}$

$S = \{yes, no\}$

$\langle M, x\rangle \, \Pi_{HALT}$ yes $\Leftrightarrow$ $M(x)$ halts

4. Intractable problems:
neither an exponential l.s.
or a polynomial u.s.
are known!

- Problems of huge practical interest (economics, electronics, AI, learning, biology, logistics)

- Strong structural properties:
a polynomial algorithm
for any single problem yields
polynomial algorithms for all!

NP-Completeness theory

We'll deal with classes of problems:
we need to standardize computational
problems as much as possible!

ABSTRACT DECISION PROBLEMS

$\Pi_D \subseteq I \times J$, $J = \{yes, no\}$

Formulation of an astract decision problem:

- description of the generic instance
- question on the istance.
  Yes/no answer is the solution

EXAMPLE. Membership of integer key in a set

SEARCH:

{ INSTANCE: $\langle S, n \rangle$: $S \subseteq \mathbb{N}$, finite, $n \in \mathbb{N}$

QUESTION: $n \in S$?

OBS: A decision problem is a function

$\Pi_D : I \longrightarrow \{si, no\}$

- positive/negative instances

Restriction to decision problems: excessive?

In practice we usually solve more complex optimization problems!

OPTIMIZATION PROBLEM

$\Pi_{opt} \subseteq \mathcal{I} \times \mathcal{I}$ $\quad c : \mathcal{I} \to \mathbb{N} \cup \{+\infty\}$

$\forall i \in \mathcal{I}$: determine $s^* \in \mathcal{S}(i) = \{s : i \,\Pi\, s\}$: ← FEASIBLE SOLUTIONS

$c(s^*) = \begin{matrix} \min \\ \max \end{matrix} \{c(s) : s \in \mathcal{S}(i)\}$

EXAMPLE: SHORTEST-UNWEIGHTED PATH (SUP)

$\mathcal{I} = \langle G=(V,E), u, z \rangle : V \subseteq \mathbb{N}$, finite, $E \subseteq V \times V$, $u, z \in V$

$\mathcal{S} = \mathbb{N}^*$ (sequences of naturals : paths)

- $i \,\Pi\, s$ if $(s = \langle s_1, s_2, \dots, s_k \rangle, s_i \in V) \wedge$
  $\|$ $\quad (s_1 = u, s_k = z) \wedge$
  $\langle G=(V,E), u, z \rangle$ $\quad (s_i, s_{i+1}) \in E, \; 1 \le i < k$
  $\|$
- $i \,\Pi\, \varepsilon$ if $\nexists$ path from u to z in G

2. $c(\langle s_1, \dots, s_k \rangle) = k-1, \quad k \ge 1$
   $c(\varepsilon) = +\infty$

Minimization problem

Am optimization problem can be easily associated to a decision problem

using the following standard technique: augment instance with a natural value representing an upper (MIN) or lower (MAX) limitation to the objective function, for all feasible solutions.

EXAMPLE: K-UNWEIGHTED-PATH (KUP)

- I: $\langle G=(V,E), u, z, K \rangle$: $V \subseteq \mathbb{N}$, finite, $E \subseteq V \times V$, $u, z \in V$, $K \in \mathbb{N}$
- D: ∃ path from u to z of length $\leq K$?

NOTE: The decision problem is not fully equivalent but it holds that:

A polynomial algorithm for KUP can be used as a subroutine to solve SUP in polynomial time!

Let $A_{KVP}(G=(V,E), u, z, k)$ be a polynomial time algorithm for KVP

We first show that A_{KVP} can be employed to determine the length of the shortest path between u and z as follows:

```
P_LENGTH(<G=(V,E), u, z>)
    n ← |V|
    for k ← 0 to n do
        if A_KVP(<G=(V,E), u, z, k>) = yes
            then return k
    return +∞
```

Clearly, P_LENGTH(G, u, z) returns the smallest value of k such that $\exists$ a path from u to z ($+\infty$ if no path exists)

Since $|V| = n \leq |\langle G, u, z\rangle|$ (SIZE OF THE INSTANCE) and

$$|\langle G, u, z, k\rangle| = \Omega(|\langle G, u, z\rangle|)$$

If $T_{KVP}(|\langle G, u, z, k\rangle|) = O(|\langle G, u, z, k\rangle|^c)$

then $T_{P\_LENGTH}(|\langle G, u, z\rangle|) = O(|\langle G, u, z\rangle|^{c+1})$

Once the length k^* of the shortest path is established, it is easy to determine the actual path through $|E|$ calls to A_{KUP}

Algorithm idea: progressively eliminate edges which are not necessary to guarantee the existence of a shortest path. To check this, we use A_{KUP} as an ORACLE

$A_{SUP}(<G=(V,E), u, z>)$

$k^* \leftarrow$ P_LENGTH $(<G, u, z>)$ {see above}

if $(k^* = +\infty)$ then return "no path"

* $E = \{e_1, e_2, \ldots, e_{|E|}\}$ *

$E' \leftarrow E$

for $i \leftarrow 1$ to $|E|$ do

if $A_{KUP}(<G'=(V, E'-\{e_i\}), u, z, k^*>) = $ 'yes'

then $E' \leftarrow E' - \{e_i\}$

{} shortest path $\pi^* \subseteq E'$ not using e_i }

* E' contains only the edges of a s.p. π^* *

return π^*

Correctness: if $k^* = +\infty$, A_{SUP} is correct.
Otherwise, by contradiction, if after the for loop, E' does not contain only the edges of a single shortest path π^*:
$\exists e_{\bar{i}} \in E'$ in E' not used by π^*
BUT at iteration $\bar{i}$ we would have:
$A_{KWP}(\langle G' = (V, E' - \{e_{\bar{i}}\}), u, z, k^* \rangle) =$ yes!
thus $\bar{e} = e_{\bar{i}}$ would have been removed!

→ We have obtained a poly-time algorithm A_{SUP} for SUP!

(EXERCISE: determine complexity)

Morale: restriction to decision problems is without loss of generality (w.l.o.g.) if we are only interested in determining existence of poly-time algorithms