ALA-TOO INTERNATIONAL UNIVERSITY

ALGORITHMIZATION
AND
PROGRAMMING
PART II

Spring Semester 2025-2026

Lecturer

Nurbekov Mirlan

Office: H 207

Monday–Friday, 08:00–17:00

mirlan.nurbekov@alatoo.edu.kg

***

Bad programmers worry about the code. Good programmers worry about data structures and their relationships.

— Linus Torvalds

FINAL GRADE CALCULATION

MIDTERM ASSESSMENT + FINAL ASSESSMENT

Final Grade = Midterm Assessment (40%) + Final Project (60%)

– – –

Midterm Assessment = accumulated score (attendance + progress + test)

Final Project = individual project evaluation

MIDTERM ASSESSMENT BREAKDOWN

100 POINTS

10% - attendance

10% - active participation in discussions and in-class exercises

20% - weekly project progress

60% - online test (openbook)

FINAL ASSESSMENT

100 POINTS

ONLINE EXAMINATION

FROM CODE TO ALGORITHMIC THINKING

THINKING OVER CODING

IF TWO PROGRAMS GIVE THE SAME CORRECT OUTPUT, ARE THEY EQUALLY GOOD

THE INDUSTRY ANSWER

CORRECTNESS IS ONLY THE MINIMUM

Correct code can still be:

too slow **AND/OR** too memory-hungry **AND/OR** Impossible to scale

Industry rewards **SURVIVABLE** solutions, not just correct ones

WHAT AN ALGORITHM REALLY IS

A STRATEGY UNDER CONSTRAINTS

An algorithm is not just code, it is a decision process

Decisions are limited by:
- Worst-case risk
- Time
- Memory
- Input size
- Reliability

SCALE CHANGES EVERYTHING

INSIGHT

Input size matters more than code style

Algorithms fail when assumptions fail

Most bugs appear only at scale

THREE ALGORITHMIC WORLDS

DIFFERENT STRATEGIES FOR DIFFERENT REALITIES

- 1 - Simple and exhaustive
- 2 - Structured and optimized
- 3 - Fast but approximate

BRUTE-FORCE ALGORITHMS

SIMPLE AND RELIABLE BUT EXHAUSTIVE

The idea is to try every possibility

Easy to implement

Easy to understand

Performance collapses as data grows

Used for:

Small inputs

Prototypes

Learning

BRUTE-FORCE ALGORITHMS

EXAMPLES

Linear Search

Naive String Matching

Exhaustive Subset Search

Brute-force Password Cracking

BRUTE-FORCE ALGORITHMS

CHARACTERISTICS

Simple logic

Easy to implement

No assumptions about data

Performance degrades rapidly with scale

BRUTE-FORCE ALGORITHMS

IN THE REAL WORLD

Password strength checkers (short passwords)

Input validation for small datasets

Game AI at low difficulty levels

Early-stage prototypes and MVPs

STRUCTURED AND OPTIMIZED ALGORITHMS

FAST BECAUSE THEY USE ORDER

It uses structure in data (require preparation)

Much better scalability

But very complex to design

Used for:

Large datasets

Databases

Search systems

STRUCTURED AND OPTIMIZED ALGORITHMS

EXAMPLES

Binary Search

Merge Sort

Quick Sort

Tree Traversals

Hash-based Lookup

STRUCTURED AND OPTIMIZED ALGORITHMS

CHARACTERISTICS

Require ordered or structured data

Much better scalability

Slightly harder to design

Core of high-performance systems

STRUCTURED AND OPTIMIZED ALGORITHMS

IN THE REAL WORLD

Database indexing (B-Trees, Hash Indexes)

File systems (directory trees)

Search engines (sorted indexes)

In-memory caches (HashMaps)

HEURISTIC AND GREEDY ALGORITHMS

FAST BUT NOT ALWAYS PERFECT

The idea is on making local decisions. Often “good enough” for most cases, cuz it is extremely fast but not guaranteed optimal performance.

Used for:

Real-time systems

Navigation

Scheduling

Recommendations

HEURISTIC AND GREEDY ALGORITHMS

EXAMPLES

Greedy Scheduling

Huffman Coding

Nearest Neighbor Search

Heuristic Pathfinding

HEURISTIC AND GREEDY ALGORITHMS

CHARACTERISTICS

Extremely fast

Local decision-making

Not always optimal

Predictable performance

HEURISTIC AND GREEDY ALGORITHMS

IN THE REAL WORLD

GPS navigation (fastest good route)

Task scheduling in operating systems

Network routing decisions

Load balancing systems

A HARD TRUTH

REAL SYSTEMS MIX ALL OF THEM

Trade-offs are everywhere

Optimized algorithms at core

Heuristics for speed

Simple algorithms at edges

WHY NOT ALWAYS USE THE BEST ALGORITHM

ENGINEERING IS ABOUT COMPROMISES

Requirements change

Data grows unexpectedly

Deadlines exist

Simple code is easier to fix

WHAT EXPERIENCED DEVELOPERS KNOW

THE BEST ALGORITHM TODAY MAY FAIL TOMORROW

Over-optimization creates fragile systems

Flexibility often beats perfection

Choosing when to optimize matters

EFFICIENCY WITHOUT FORMULAS

INPUT GROWS AND COST EXPLODES

Small input → everything works

Medium input → slow

Large input → system failure

HARD TRUTH

ALGORITHMS DON'T FAIL - ASSUMPTIONS DO

Wrong assumptions about:

Input size

Data distribution

User behavior

Lead to system collapse

MAIN ALGORITHMIC PARADIGMS

DIFFERENT PROBLEMS NEED DIFFERENT THINKING

Divide and Conquer

Greedy

Dynamic Programming

Graph-based algorithms

NEXT WEEK PREVIEW

COMING SOON

GROWTH

PREDICTION

LIMITS

TRADE-OFFS