COURSE SYLLABUS

CS210 • Data Structures & Complexity

School of Computational Arts & Sciences • Undergraduate • 6 ECTS

Overview

Study core data structures and complexity with hands-on implementation: stacks, queues, trees, hash tables, and graphs. Focus on choosing the right structure, analyzing time/space tradeoffs, and writing performance-aware code with clear invariants.

LOGISTICS

Credits: 6 ECTS
Level: Undergraduate
School: School of Computational Arts & Sciences
Prerequisites: CS101 (or equivalent programming experience)
Tags: algorithms, performance
Meeting time: Weekly lecture + problem-session lab
Instruction mode: Implementation-focused with complexity reasoning

LEARNING OUTCOMES

You will be able to:
- Implement and analyze core data structures (lists, stacks, queues, trees, hash tables)
- Explain time/space complexity and tradeoffs in real code
- Design APIs and tests that make correctness checkable
- Select appropriate data structures for constraints and performance goals
- Analyze time/space complexity and explain trade-offs
- Implement core structures with tests and clear invariants

ASSESSMENT

Components
- Problem sets: 30%
- Implementation labs: 30%
- Midterm (written + code review): 20%
- Final project (data structure + report): 20%

We grade both implementation and explanation. Each major assignment includes a short
complexity note and an invariant list. Performance is measured using controlled inputs
rather than anecdotal timing.

WEEKLY PLAN

Schedule
Week 1: Complexity and measurement
- Big-O as a model
- Benchmarking pitfalls
- Reading profiler output

Week 2: Arrays, lists, and invariants
- Memory layout
- Amortized analysis
- Invariants as documentation

Week 3: Stacks, queues, and parsing
- Monotonic stacks
- Queue implementations
- Small parsing tasks

Week 4: Hash tables
- Hash functions
- Collision strategies
- Load factors and resizing

Week 5: Trees
- BST basics
- Balancing intuition
- Traversal patterns

Week 6: Graphs (intro)
- Representations
- BFS/DFS
- Tradeoffs

Week 7: Testing and verification habits
- Edge cases
- Property-style thinking
- Regression test suites

Week 8: Project build and review
- Code review
- Performance notes
- Write-up and defense

Extended outline
- Asymptotics and cost models: what Big-O hides and what it doesn’t
- Arrays, lists, stacks, queues: invariants and tests
- Hash maps: collisions, resizing, and correctness
- Trees and heaps: ordering, balancing intuition, priority queues
- Graphs: traversal, shortest paths, and failure cases
- Final: build a small library + benchmark note

POLICIES & RESOURCES

- Correctness first: prove with tests and invariants before optimizing.
- No premature cleverness: readability is part of the grade.
- Collaboration: pair on design discussions; code must be your own.

Suggested resources

- Invariant checklist: what must always be true
- Benchmark template: controlled inputs, repeated runs, reporting
- Reference: notes on hashing and tree traversal patterns