SIGCSE 2026 • ST. LOUIS, MO
Position and Curricula Initiative (PCI)

Is It Time to Remove Data Structures?

A Critical Look at Requirements and Curricular Placement

Albert Lionelle
Khoury College of Computer Sciences
Northeastern University
a.lionelle@northeastern.edu

The Tension

"Why do I need to implement sorting algorithms I'll never write in practice?"

Modern SDKs provide optimized implementations. Industry favors library usage over custom implementations.

"Deep algorithmic understanding is essential for advanced work."

Skills in algorithmic comparison, construction, and formal correctness analysis distinguish CS graduates. Starting with sorting and data structures helps later work.

- The core question:
  - Do students need to *use* data structures or truly *understand* them.
- The problem:
  - Both answers are true
  - Faculty are divided on how to handle it (Recall SIGCSE 2025 Plenary Session on “Future of CS Education”!)

Definitions

- Data Structures / CS2 is poorly defined as a class
  - Another area we can't agree

- Acknowledging previous work
  - Two course sequence CS 1 → CS 2: Data Structures
  - Three course sequence CS 1 → OOP → Data Structures

- Looking at requirements and curricular structure
  - Benefits of three course structure, yet still similar curricular structures

- For a definition, went with Leo Porter et al. work of CS 2: Data Structures

Leo Porter, Daniel Zingaro, Cynthia Lee, Cyntbasichia Taylor, Kevin C. Webb, and Michael Clancy. 2018. Developing course-Level learning goals for data structures in CS2. SIGCSE 2018

What's typically in Data Structures?

Topics mapped to ACM 2023 Knowledge Areas

- Data Structures is Hard…

- While KA overlap is encouraged
- Still **a lot** in a single course
  - Does not include soft-skills
  - Does not include additional tech skills
- Tension – what is the focus?
  - Application
  - Analysis

- Additional issue: Curricular Placement

OOP
FPL-OOP, SDF-Practices, SE-Construction

Basic Data Structures
SDF-DS, AL-Foundational, AL-Complexity, AL-Strategies, MSF-Discrete, FPL-OOP

Recursion
SDF-Fundamentals, AL-Strategies, FPL-Functional, AL-Models, MSF-Discrete

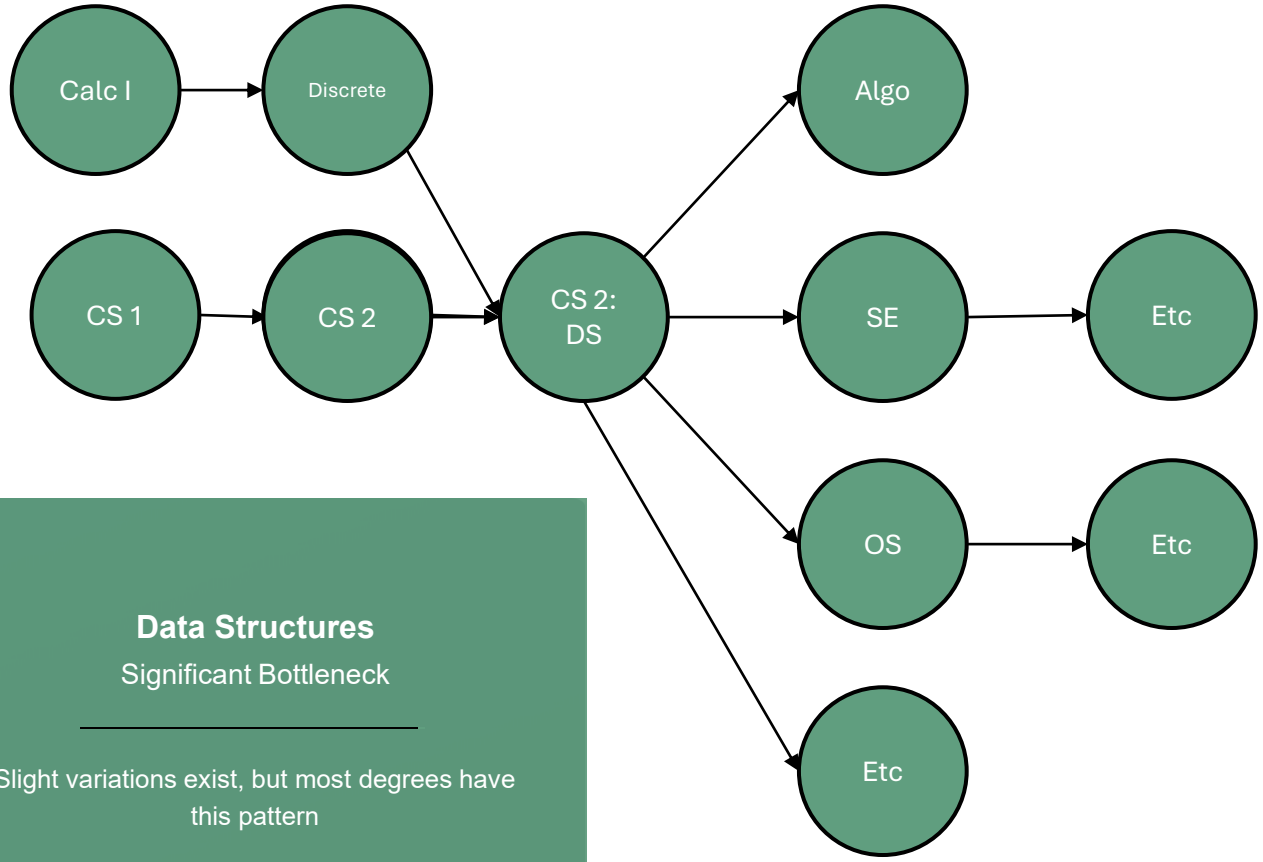
Sorting
SDF-Algorithms, AL-Foundational, AL-Complexity, AL-Strategies

Algorithm Analysis
SDF-Algorithms, AL-Complexity, AL-Strategies, AL-Foundational

Advanced Structures
AL-Foundational, AL-Complexity, MSF-Discrete, SDF-DS, AL-Strategies

Curricular Graph

Typical Prerequisite Mappings – a problem



Data Structures

Significant Bottleneck

Slight variations exist, but most degrees have this pattern

Potential Issues

- DS – Foundation of all courses?
- Large class sections
- High Attrition
- Frustration (both staff and students)
- Makes minors + interdisciplinary degrees more difficult

Curricular Bottleneck

Analysis of 75 CS degree programs, Data Structures – Major Bottleneck

COURSE CENTRALITY RANKING – Top Groups (75 randomly sampled R1 schools)

Course	Count	%
CS 2: Data Structures	45	**60.0%**
CS 1	12	16.0%
Mathematics	8	10.7%

Impact: When students struggle in high-centrality courses, progress is blocked. Students often conclude the major isn't for them — increasing attrition.

60%

of programs have Data Structures as the **most central course**

(others have it as second highest)

Highest centrality = most prerequisite chains pass through this course

Data Structures is Hard…

But it doesn't have to be.

A Suggestion: Replace one overburdened course with two focused courses

1

OOP with Data Structures

Practical application / SE focus

2

Data Structures & Algorithms

Mathematical foundations and proofs

Key: These courses should NOT be forced prerequisites of each other

COURSE 1

OOP with Data Structures

Focus: Practical application, SDK usage, software engineering principles

14 WEEK CURRICULUM

Intro to OOP	Constructors
Unit Testing / TDD	Encapsulation
Inheritance	Polymorphism
Interfaces	Exception Handling
Collections & Lists	Generics
Trees	Hash Tables
Streams	Advanced Testing

PREREQUISITE
Only CS1

KEY ADVANTAGE
Minimal prerequisites enable earlier placement. Opens pathways to HCI, web dev, mobile, etc.

BIG O COVERAGE
Focused introduction through practical lens — swapping data structures to observe performance impact.

Additional Skills:

- Design focused / easier transition into SE
- Source control
- Code reviews
- Collaborative programming
- Technical communication
- Gen ai?

COURSE 2

Data Structures and Algorithms

Focus: Mathematical rigor, experimental analysis, formal proofs

14 WEEK CURRICULUM

Memory Fundamentals	Algorithm Analysis
Math for Proofs	Quadratic Sorts
Divide & Conquer	Advanced D&C
Arrays & Lists	Sequential Advanced
Hash Tables	Trees & BSTs
Heaps & Priority Q	Graphs
Greedy Algorithms	Advanced Integration

PREREQUISITES
Discrete Math + CS1

EXPERIMENT-DRIVEN
Students conduct empirical analysis — comparing theoretical bounds against actual execution time.

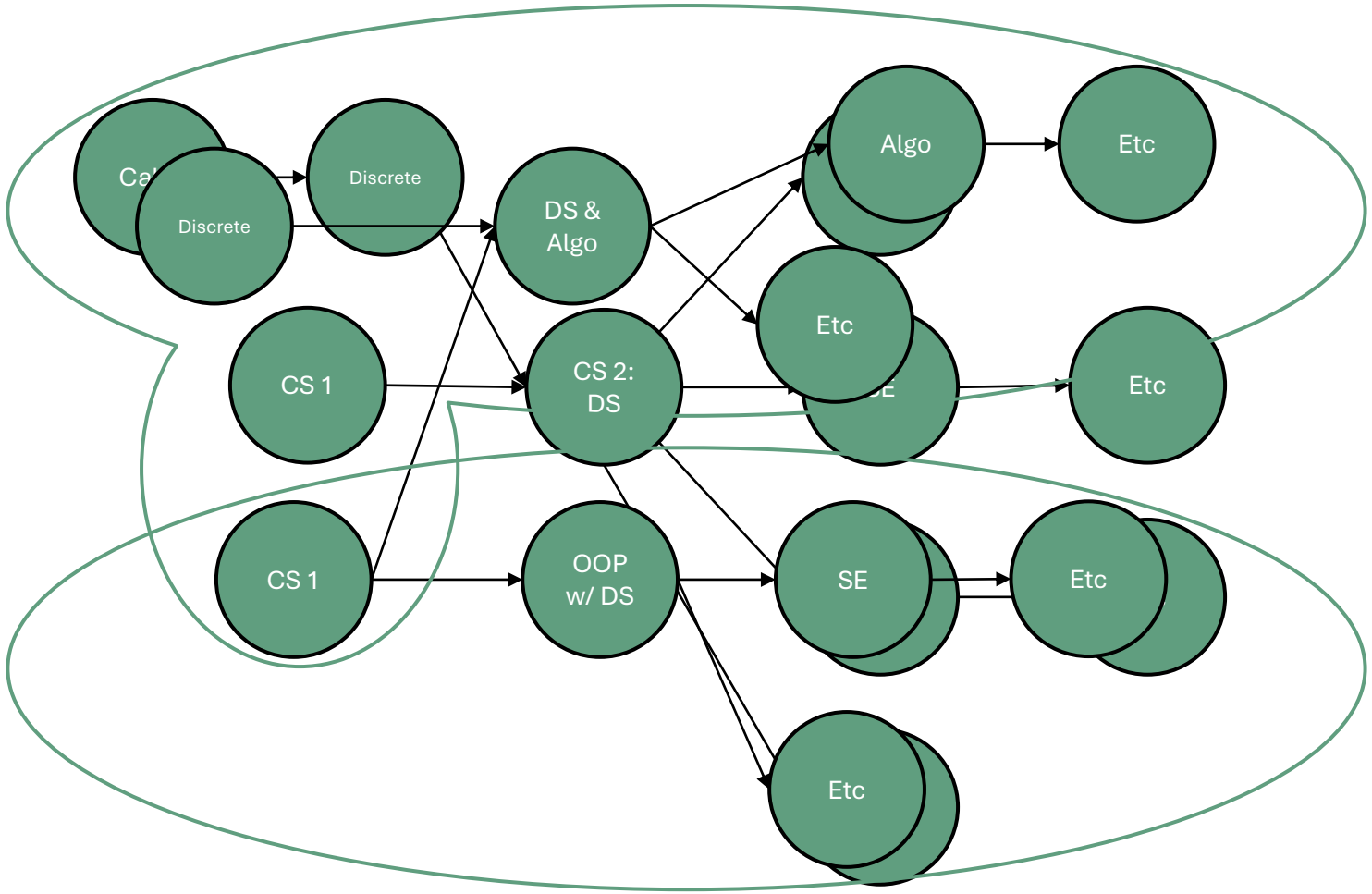
FINAL PROJECT
Independent paper on uncovered data structure: theory, implementation, empirical evaluation.

Additional Skills:

- Technical writing
- Data visualization
- Research methodology
- Interview prep

Updated Curricular Graph

Eliminates the single bottleneck, creates flexible progression



Improvements

- Distinct Pathways
  - Minors
  - Interdisciplinary majors
- Reduced / Removed bottleneck
- Stronger Algo Background
- Provides room for soft skills

Overall

- A solution, others exist (e.g. spiral designs)

- This is not new!
  - Calling attention to the design pattern
  - Supported by curricular metrics

- Curricular Complexity – Helps Develop Good Design Patterns
  - Research is showing less complexity is better
  - As a community, uncover curricular design patterns

- Most importantly
  - We need more research
  - We need more discussion

Thank you and Discussion

Presentation Slides

Full Paper

Northeastern University
Khoury College of Computer Sciences

Northeastern
Center for Inclusive Computing