The Next 700 Algorithms Education Papers

Seth Poulsen

Presentation Roadmap

The title is a play on “The Next 700 Programming Languages” by P.J. Landin (1966)

- What has been done so far?
- What will happen in the future?

Presentation Roadmap

1. *Teaching Algorithm Design: A Literature Review* with Jonathan Liu, Erica Goodwin, Hongxuan Chen, Grace Williams, Yael Gertner, Diana Franklin, TOCE 2025 (slides from Jon)
2. *How Do Students Select and Algorithm Design Technique?* with Dip Kiran Pradhan Newar and Peter Fowles, ACE 2026 (slides from Dip)
3. What's Next?

Presentation Roadmap

1. *Teaching Algorithm Design: A Literature Review* with Jonathan Liu, Erica Goodwin, Hongxuan Chen, Grace Williams, Yael Gertner, Diana Franklin, TOCE 2025 (slides from Jon)
2. *How Do Students Select and Algorithm Design Technique?* with Dip Kiran Pradhan Newar and Peter Fowles, ACE 2026 (slides from Dip)
3. What's Next?

Methods Overview

01

Problem Formulation

Which topics and papers are relevant?

02

Data Collection

How should we collect relevant papers?

03

Data Evaluation

Which collected papers should be included?

04

Data Analysis

What do the included papers tell us?

Methods Overview

01

Problem Formulation

Which topics and papers are relevant?

02

Data Collection

How should we collect relevant papers?

03

Data Evaluation

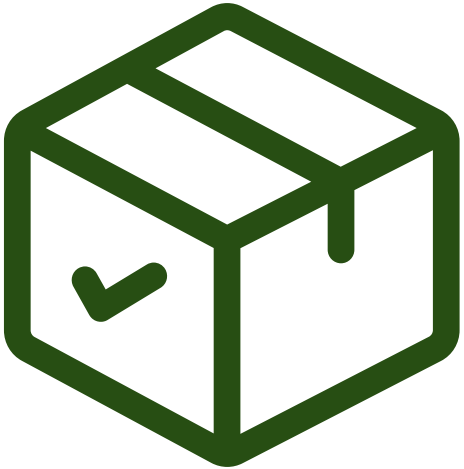
Which collected papers should be included?

04

Data Analysis

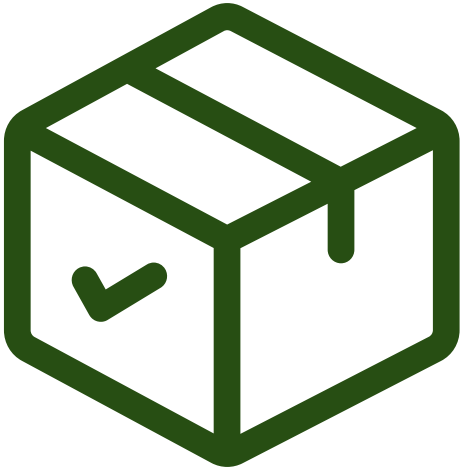
What do the included papers tell us?

What counts as an algorithms topic?



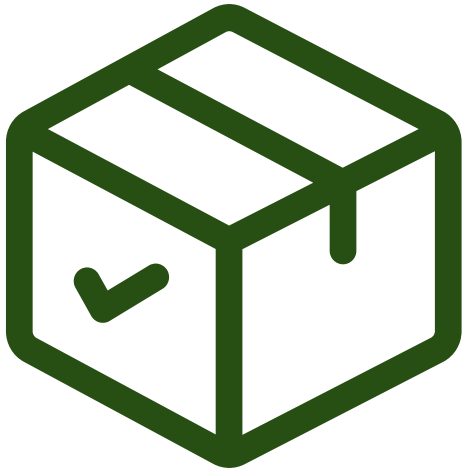
What counts as an algorithms topic?

Dynamic Programming



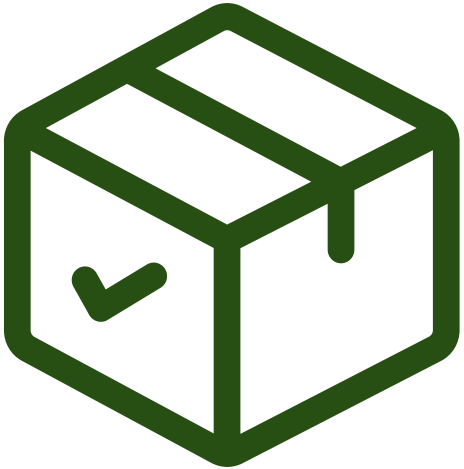
What counts as an algorithms topic?

Dynamic Programming



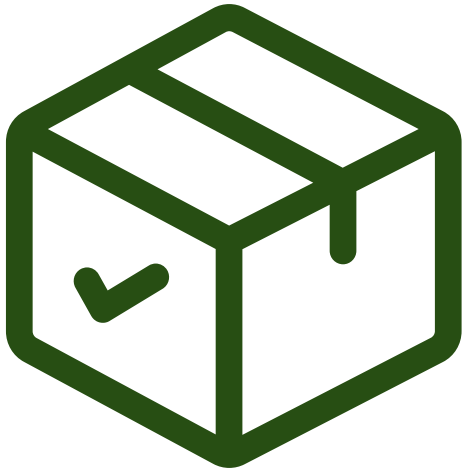
What counts as an algorithms topic?

While Loops



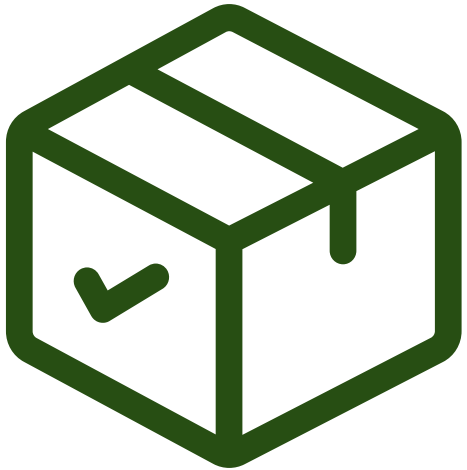
What counts as an algorithms topic?

While Loops



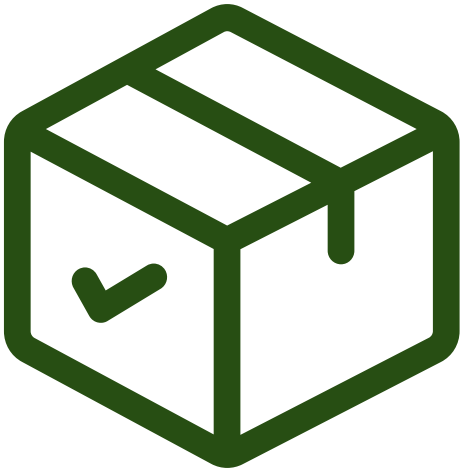
What counts as an algorithms topic?

Insertion Sort



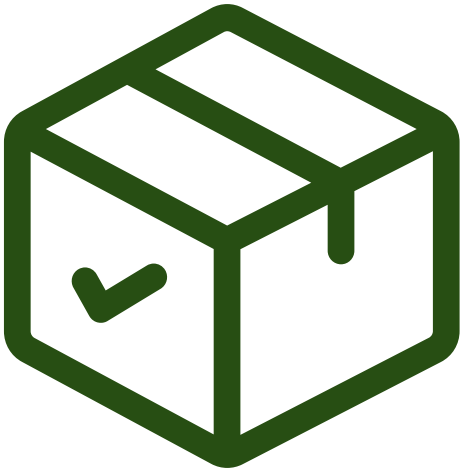
What counts as an algorithms topic?

Insertion Sort



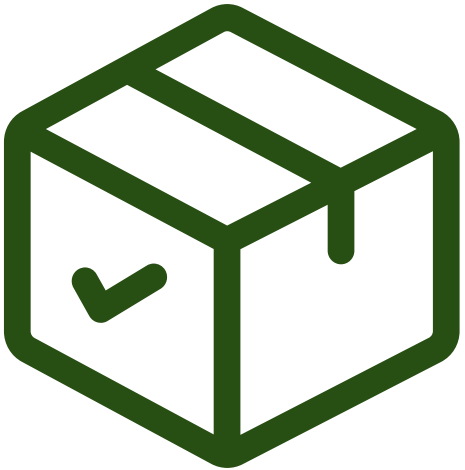
What counts as an algorithms topic?

P vs. NP



What counts as an algorithms topic?

A Data Structures and Algorithms Course



What counts as an algorithms topic?

Criterion 1: It is an algorithm design paradigm.

What counts as an algorithms topic?

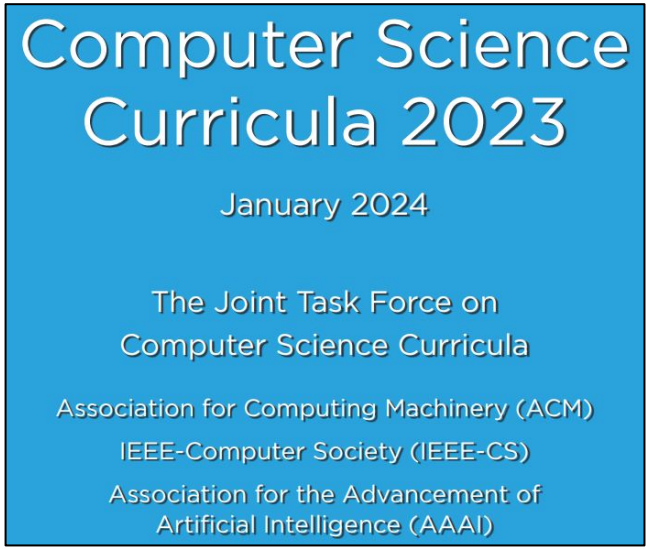
Criterion 1: It is an algorithm design paradigm.

Criterion 2: It is a paradigm in the curricular guidelines.

AL-Strategies: Algorithmic Strategies

CS Core:

1. Paradigms
   a. Brute-Force (e.g., linear search, selection sort, traveling salesperson, knapsack)
   b. Decrease-and-Conquer
      i. By a Constant (e.g., insertion sort, topological sort),
      ii. By a Constant Factor (e.g., binary search),
      iii. By a Variable Size (e.g., Euclid's)
   c. Divide-and-Conquer (e.g., binary search, quicksort, mergesort, Strassen's)
   d. Greedy (e.g., Dijkstra's, Kruskal's, Knapsack)
   e. Transform-and-Conquer
      i. Instance simplification (e.g., find duplicates via list presort)
      ii. Representation change (e.g., heapsort)
      iii. Problem reduction (e.g., least-common-multiple, linear programming)
      iv. Dynamic programming (e.g., Floyd's, Warshall, Bellman-Ford)
   f. Space vs time tradeoffs (e.g., hashing)
2. Handling exponential growth (e.g., heuristic A*, branch-and-bound, backtracking)
3. Iteration vs recursion (e.g., factorial, tree search)



What counts as an algorithms topic?

Criterion 1: It is an algorithm design paradigm.

Criterion 2: It is a paradigm in the curricular guidelines.

Criterion 3: It is most commonly first taught in algorithms.

What counts as an algorithms topic?

Criterion 1: It is an algorithm design paradigm.

Criterion 2: It is a paradigm in the curricular guidelines.

Criterion 3: It is most commonly first taught in algorithms.

What is an Algorithms Course? Survey Results of Introductory Undergraduate Algorithms Courses in the U.S. IN-PERSON

Michael Luu University of California, Irvine, Matthew Ferland University of Southern California, Varun Nagaraj Rao Princeton University, Arushi Arora University of California, Irvine, Randy Huynh University of California Irvine, Frederick Reiber Boston University, Jennifer Wong-Ma University of California, Irvine, Michael Shindler University of California, Irvine

What counts as an algorithms topic?

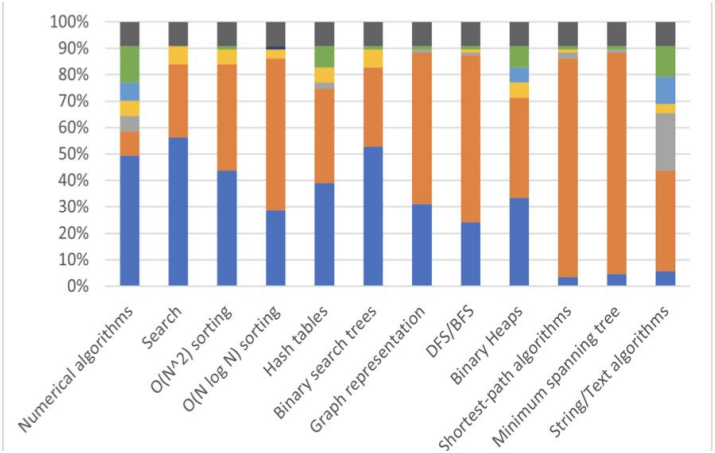
Criterion 1: It is an algorithm design paradigm.

Criterion 2: It is a paradigm in the curricular guidelines.

Criterion 3: It is most commonly first taught in algorithms.

What is an Algorithms Course? Survey Results of Introductory Undergraduate Algorithms Courses in the U.S. IN-PERSON

Michael Luu University of California, Irvine, Matthew Ferland University of Southern California, Varun Nagaraj Rao Princeton University, Arushi Arora University of California, Irvine, Randy Huynh University of California Irvine, Frederick Reiber Boston University, Jennifer Wong-Ma University of California, Irvine, Michael Shindler University of California, Irvine



Final List of Algorithms Topics

Branch-and-bound	Brute-force algorithms	DFS/BFS	Divide and conquer
Dynamic Programming	Greedy algorithms	Heuristics	Minimum spanning trees
Pattern matching and string/text algs	Recursive backtracking	Reduction	Representations of graphs
Shortest-path algorithms	O(n log n) Sorting Algorithms		

Problem Formulation: Research Questions

- **RQ1:** What algorithms topics are currently represented in the literature, and what kinds of studies are being conducted on these topics?
- **RQ2:** What kinds of interventions have been developed to teach algorithm design?
- **RQ3:** What knowledge currently exists in the literature about undergraduate algorithm design courses? What are the major takeaways?

Methods Overview

01

Problem Formulation

Which topics and papers are relevant?

02

Data Collection

How should we collect relevant papers?

03

Data Evaluation

Which collected papers should be included?

04

Data Analysis

What do the included papers tell us?

Data Collection: Search Corpus

- Searched the ACM Digital Library
  - Only venues in collaboration with SIGCSE
    - Includes SIGCSE TS, ICER, ITiCSE, TOCE, Koli, etc.
- Also searched *Computer Science Education* through Taylor and Francis

- Restricted to papers/articles
- *Any time* before July 2024

Data Collection: Search Strategy

Topic	Search Query
Divide and Conquer	“Divide and Conquer”
Greedy Algorithms	“Greedy”
Reduction: transform-and-conquer	"algorithm reduction" OR "reduction algorithm"
Shortest-path algorithms	"shortest path algorithm" OR "dijkstra’s algorithm" OR "floyd’s algorithm"

Result: 946 papers collected

Methods Overview

01

Problem Formulation

Which topics and papers are relevant?

02

Data Collection

How should we collect relevant papers?

03

Data Evaluation

Which collected papers should be included?

04

Data Analysis

What do the included papers tell us?

Data Evaluation

Three major criteria:

- Presents student data
- Students are college-level
- Topic relevance
  - Study was conducted in an “Algorithms” course, or
  - Results pertain to one of our topics

4 coders reached inter-rater reliability (Fleiss’ Kappa $\kappa > 0.87$)

Result: 102/946 papers included

Methods Overview

01

Problem Formulation

Which topics and papers are relevant?

02

Data Collection

How should we collect relevant papers?

03

Data Evaluation

Which collected papers should be included?

04

Data Analysis

What do the included papers tell us?

Data Analysis: Codebook Development

- Began with deductive codebook split into four categories:

Topic	Which specific course topic is studied, if any?
Research Methods	How is the data being collected and evaluated?
Evaluation Metric	What metric(s) are being evaluated?
Intervention Target	What course feature is being changed?

- Coded 10 papers independently, discussed and iterated
- Four coders reached inter-rater reliability ($\kappa > 0.80$)

Methodological Limitations

- Our query may have missed some papers
  - Logistically, could not search “algorithms”
  - Validated our search on a few years of SIGCSE
- Some venues were not included

Outcomes Overview

Understand paper landscape

Consolidate major takeaways

Identify understudied topics

Facilitate in-depth exploration

Outcomes Overview

Understand paper landscape

Consolidate major takeaways

Identify understudied topics

Facilitate in-depth exploration

Landscape: Overall

Table 3. Tags by Topic. Note that some papers cover multiple topics, and contribute to each corresponding row in the table.

		Evaluation							Metric			Intervention						
Topic	Total	Quantitative	Survey	Stat. Tests	Control Trial	Qualitative	Interview	Thm. Coding	Performance	Affect	Persistence	None	Tools	Course Policy	Content	Discussion	Student Work	Exams
Branch-and-bound	1	1	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0
Brute-force Algorithms	2	2	1	2	2	0	0	0	2	1	1	0	2	0	0	1	1	0
DFS/BFS	7	6	5	3	3	3	1	1	4	5	0	0	3	0	2	1	3	0
Divide-and-Conquer	4	4	1	3	1	2	1	0	4	1	0	2	1	0	0	1	0	1
Dynamic Programming	12	11	6	6	3	6	3	2	10	6	1	4	4	0	1	1	4	1
Greedy Algorithms	8	8	4	0	0	6	2	5	6	4	0	2	3	0	3	2	2	0
Heuristics	3	3	0	0	0	2	0	2	3	0	0	1	1	0	0	0	2	0
Minimum Spanning Trees	6	6	5	2	2	2	0	0	4	4	1	0	4	0	0	2	4	1
Pattern Matching and String/-Text Algorithms	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
Recursive Backtracking	3	3	2	1	1	2	0	1	3	2	0	1	2	0	1	1	0	0
Reductions	7	5	3	1	0	5	1	2	7	2	0	4	0	1	2	1	2	0
Representations of Graphs	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shortest-path Algorithms	7	7	5	3	3	1	0	0	6	5	1	0	7	0	1	2	2	0
$O(n \log n)$ Sorting Algorithms	10	9	7	6	2	4	1	0	6	9	0	2	3	0	2	2	3	0
Total (any topic)	51	46	30	18	11	25	7	9	40	30	2	12	21	1	10	11	15	2
Not topic-specific	51	46	29	25	7	22	6	7	34	33	2	12	16	11	11	7	16	2
Total	102	92	59	43	18	47	13	16	74	63	4	24	37	12	21	18	31	4

Landscape: Topic Coverage

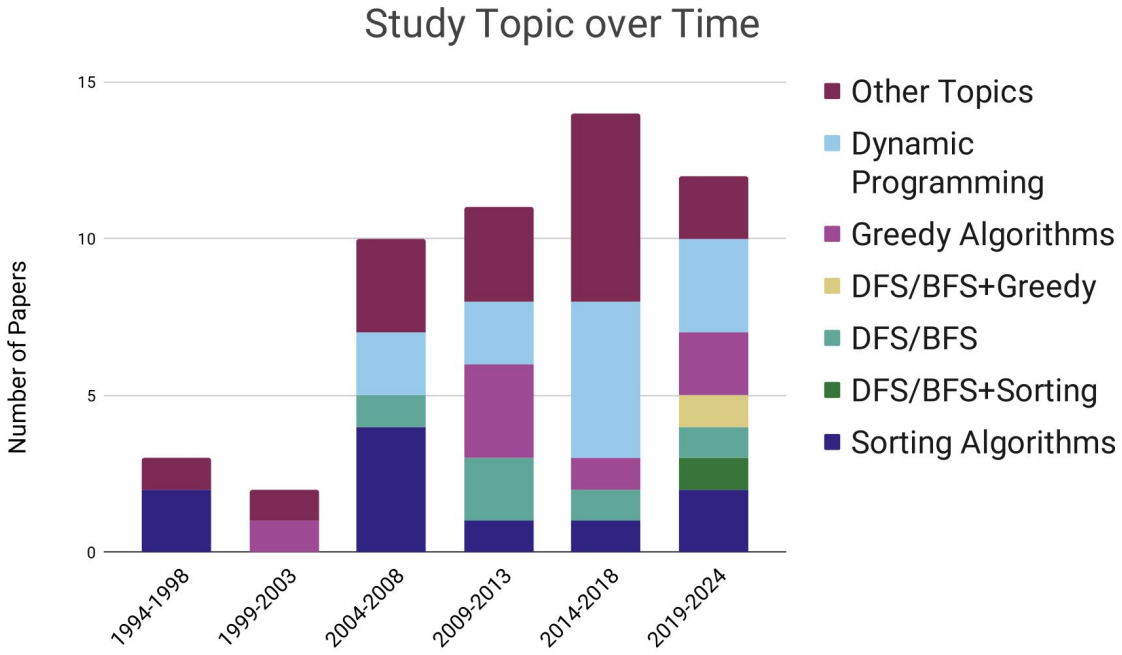
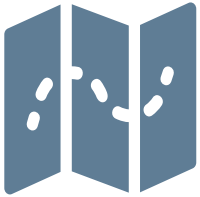


Fig. 1. Topic of topic-specific papers, by year

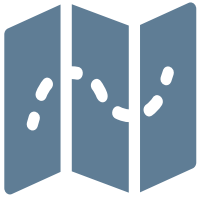
Landscape: Dynamic Programming



- **Broad coverage** with variety of approaches and targets
  - Tools scaffolding different steps of DP problems
  - Course projects using “real-world” applications
  - Investigations of roadblocks in solving DP problems

Topic	Total	Evaluation							Metric			Intervention						
		Quantitative	Survey	Stat. Tests	Control Trial	Qualitative	Interview	Thm. Coding	Performance	Affect	Persistence	None	Tools	Course Policy	Content	Discussion	Student Work	Exams
Dynamic Programming	10	10	6	5	3	5	2	1	8	5	1	3	4	0	1	1	3	0

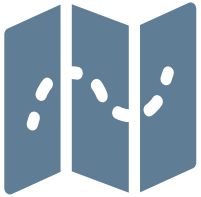
Landscape: Greedy Algorithms



- **Less breadth in approach**
  - Six papers about interactive learning activities
  - One about misconceptions
  - One about collective argumentation solving greedy
- No statistical tests

		Evaluation							Metric			Intervention						
Topic	Total	Quantitative	Survey	Stat. Tests	Control Trial	Qualitative	Interview	Thm. Coding	Performance	Affect	Persistence	None	Tools	Course Policy	Content	Discussion	Student Work	Exams
Greedy Algorithms	8	8	4	0	0	6	2	5	6	4	0	2	3	0	3	2	2	0

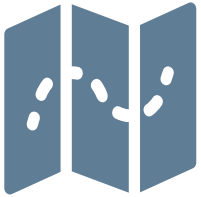
Landscape: Less-covered topics



- All four topics below taught in more than 80% of algorithms classes
- Very little work, especially qualitative
- Work suffers from lack of breadth as well

		Evaluation							Metric			Intervention						
Topic	Total	Quantitative	Survey	Stat. Tests	Control Trial	Qualitative	Interview	Thm. Coding	Performance	Affect	Persistence	None	Tools	Course Policy	Content	Discussion	Student Work	Exams
Divide-and-Conquer	3	3	1	2	1	2	1	0	3	0	0	2	1	0	0	1	0	0
Minimum Spanning Trees	5	5	4	1	1	2	0	0	3	3	1	0	3	0	0	1	4	1
Representations of Graphs	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shortest-path Algorithms	6	6	4	2	2	1	0	0	5	4	1	0	6	0	1	1	1	0

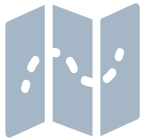
Landscape: Methodology



- Similar methodology spread (except course policy)
- Heavy focus on quantitative performance data
- Nearly 40% develop a new tool
- Only 4 papers about persistence, 3 about exams

		Evaluation							Metric			Intervention						
Topic	Total	Quantitative	Survey	Stat. Tests	Control Trial	Qualitative	Interview	Thm. Coding	Performance	Affect	Persistence	None	Tools	Course Policy	Content	Discussion	Student Work	Exams
Total (any topic)	46	43	28	15	10	23	6	8	35	27	2	9	20	1	10	9	15	1
Not topic-specific	48	45	27	23	7	22	6	7	34	30	2	11	16	11	9	7	16	2
Total	94	88	54	38	17	45	12	15	69	57	4	20	36	12	19	16	31	3

Outcomes Overview



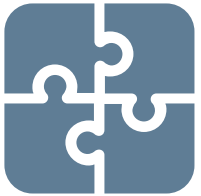
Understand paper landscape

Consolidate major takeaways

Identify understudied topics

Facilitate in-depth exploration

Major Takeaways: Problem-Solving Skills

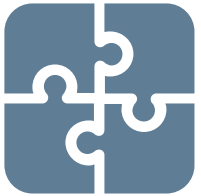


Abstraction (and related skills) are useful but underutilized when solving problems.

- Abstraction ability correlates with performance in only Algorithms courses [Bennedssen and Caspersen (2008)]
- Students utilize abstraction at different, distinguishable levels [Ginat and Blau (2017); Izu et al. (2017)]

- Related skills also learned but underutilized:
  - Reduction [Armoni (2009)]
  - Recursion [Ginat (2004)]
  - Counterexamples [Ginat (2003)]

Major Takeaways: Problem Selection



Intentional problem selection can be leveraged to motivate students.

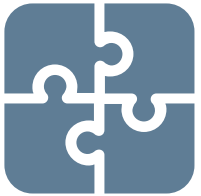
- Algorithms lends itself to real-world applicable problems, giving students motivation and problem familiarity
  - Integrating maps/GPS into shortest-path problems [Holdsworth and Lui (2009); Teresco et al. (2018)]
  - Bipartite matching for making project groups [Mehta et al. (2012)]
- Comparing multiple solutions is also enlightening and motivating
  - Can compare implementation, output, runtime
  - e.g. BFS vs. DFS, Kruskal's vs. Prim's [Erkan and Gochev (2008, 2018)]

Major Takeaways: Algorithm Visualization

Students appreciate visualizations, but their effectiveness is unclear.

- Effectiveness results are mixed:
  - Only one paper finds statistically significant learning improvements [Farghally et al. (2017)]
    - Two find improvement with no statistical tests
  - One paper finds no statistical improvement [Jarc et al. (2000)]
  - One finds improvement in efficiency but not performance [Velázquez-Iturbide and Pérez-Carrasco (2016)]

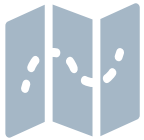
Major Takeaways: Coding Problems



Coding assignments in algorithms courses have shown to benefit students.

- Around 25% of algorithms courses contain no programming [Luu et al. (2023)]
- Research has shown that coding assignments increase:
  - Student engagement [Gárcia-Mateos and Fernández-Alemán (2009)]
  - Algorithm design skills [Izu and Alexander (2018)]
  - Runtime analysis skills [Coore and Fokum (2019)]
- Online competitions with public leaderboards may only motivate strong students [Coore and Fokum (2019)]

Outcomes Overview



Understand paper landscape

Consolidate major takeaways

Identify understudied topics

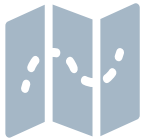
Facilitate in-depth exploration

Understudied Topic: Student Experiences

Little is known about student experiences in (standard) algorithms courses.

- Studies exist about affect regarding interventions, but not baseline experiences
- [Danielsiek et al. (2017)] develops an instrument for measuring student self-efficacy in algorithms
  - They use it to measure the perceived benefit of collaborative labs [Toma and Vahrenhold (2018)]
- This work demonstrates the potential for emotional responses to guide intervention design

Outcomes Overview



Understand paper landscape

Consolidate major takeaways

Identify understudied topics

Facilitate in-depth exploration

Conclusion

Contributions are generally sparse, especially per topic.

Well-established strategies also work here, but students struggle with algorithms-specific skills.

Studies about student experience are particularly lacking.

The literature review can be used to orient future research to advance topic discourse.

Presentation Roadmap

1. *Teaching Algorithm Design: A Literature Review* with Jonathan Liu, Erica Goodwin, Hongxuan Chen, Grace Williams, Yael Gertner, Diana Franklin, TOCE 2025 (slides from Jon)
2. *How Do Students Select and Algorithm Design Technique?* with Dip Kiran Pradhan Newar and Peter Fowles, ACE 2026 (slides from Dip)
3. What's Next?

Related Work

- Zehra, Shamama, et al. "Student misconceptions of dynamic programming." Proceedings of the 49th ACM technical symposium on Computer Science Education. 2018.
  - Interviewed students solving Dynamic Programming Problems
  - Students didn’t realize they were supposed to use Dynamic Programming

Related Work

- Shindler, Michael, et al. "Student misconceptions of dynamic programming: a replication study." Computer Science Education 32.3 (2022): 288-312.
  - Interviewed students solving Dynamic Programming Problems
  - Students didn’t realize they were supposed to use Dynamic Programming

Related Work

- Chen, Hongxuan, et al. "Novice Difficulties in Graph Layering for Algorithm Design." Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2. 2025.
  - Interviewed students solving Dynamic Programming Problems
  - Students didn’t realize they were supposed to use Graph Algorithms

Introduction

“An important part of computing is the ability to select algorithms appropriate to particular purposes and to apply them, recognizing the possibility that no suitable algorithm may exist.” - ACM Curricular guidelines

Algorithm design technique needed for exams and job interviews.

Introduction

If its so important, why isn’t it being taught anywhere?

(I talked to lots of algorithms instructors, read lots of books, etc.)

Introduction

- Research Question: What problem-solving techniques do level students exhibit when selecting an appropriate algorithm design technique for an advanced algorithm design problem?
- Focus on techniques taught in an upper-level algorithms course:
  - Dynamic Programming
  - Graph Modelling
  - Divide & Conquer

Algorithm Design Selection Process

- Understanding Examples:
  - Examine some example inputs for the problem
  - examine some subproblems of your example problem
  - look at more examples if you don't understand
- Data Structure Selection: Should the input be represented as a graph, or can it be left as the input array?
- Selecting a Design Technique:
  - Divide-and-conquer: combine disjoint sub-problem solutions to get the whole problem solution?
  - Dynamic Programming: Do certain sub-problems have sub sub-problems which are identical?
  - If graph: Does one of the standard graph algorithms (minimum spanning tree, breadth- or depth-first search, pathfinding) solve the problem?

Methodology

- Interviewed 9 + 8 students who had taken CS 5050: Advanced Algorithms
- Four questions:
  - Q1: Dynamic Programming
  - Q2: Graph Modeling
  - Q3: Divide and Conquer
  - Q4: DP or Graph Modeling
- Used inductive categorization qualitative analysis.

Results

Step	Code	Code Name	#Students	#Questions	#Successes
Step 1: Understanding Examples	C1	Experimenting with given example to understand the problem better	8/8	29/32 (91%)	17/29 (59%)
	C2	Generates new examples to understand the problem better	4/8	4/32 (36%)	2/4 (50%)
	C3	Considering how to construct the solution to the problem using a sub-problem solution	5/8	6/16 (36%)	4/6 (67%)

Step	Code	Code Name	#Students	#Questions	#Successes
Step 2: Data Structure Selection	C4	Realizes graph data structure could be used	8/8	15/16 (94%)	14/15 (93%)

Step	Code	Code Name	#Students	#Questions	#Successes
Step 3: Selecting a Design Technique	C5	Connecting to similar problems	5/8	8/32 (25%)	6/8 (75%)
	C6	Thinking about some existing technique	8/8	25/32 (78%)	19/25 (76%)
	C7	Realizes that an attempted approach is incorrect	5/8	5/32 (16%)	3/5 (60%)

Step	Code	Code Name	#Students	#Questions	#Successes
Step 4: Others	C8	Immediately understands solution	5/8	6/32 (19%)	6/6 (100%)
	C9	Considering time complexity during the solution	6/8	12/32 (38%)	7/12 (58%)
	C10	Realizes brute force approach is expensive	6/8	11/32 (34%)	3/11 (27%)
	C11	Creating algorithm that only works for some inputs	3/8	3/32 (9%)	1/3 (33%)
	C12	Using a greedy approach when not appropriate	4/8	4/32 (13%)	3/4 (75%)

Finding and Conclusion

- There are a few behaviors students don’t seem to do enough!
  - Connecting to similar problem
  - Identify subproblems

Finding and Conclusion

- Students continue to fail even after realizing their approach is wrong (rather than recovering and trying a different method)
- Students work with examples but still generate solution that only works for some inputs.

Future Work

- Developing Algorithmic Paradigm Assessment, to scale up assessment of this skill
- Testing the steps developed using a randomized controlled trial

Presentation Roadmap

1. *Teaching Algorithm Design: A Literature Review* with Jonathan Liu, Erica Goodwin, Hongxuan Chen, Grace Williams, Yael Gertner, Diana Franklin, TOCE 2025 (slides from Jon)
2. *How Do Students Select and Algorithm Design Technique?* with Dip Kiran Pradhan Newar and Peter Fowles, ACE 2026 (slides from Dip)
3. What's Next?

What’s Next For Algorithms Education?

- Hopefully, more controlled trials
- Hopefully, more evidence-backed interventions that can be scaled across Universities

TheoryABCs Project

- $1.36M Collaborative grant (#2434362, #2434363, #2434364)
- PIs Diana Franklin, Jeff Erickson, Geoffrey Herman, Seth Poulsen

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF COMPUTER SCIENCE

Computer Science
The Grainger College of Engineering
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

UtahState University

TheoryABCs Project Subgroups

Personalized Problems:
Leads: Erica Goodwin (UChicago), Katherine Brought (UIUC)

Teaching Dynamic Programming
Lead: Jon Liu (UChicago)

Teaching Graph Layering
Lead: Hongxuan Chen (UIUC)

Assessing Algorithms Knowledge
Lead: Dip Kiran Pradhan Newar (USU)

Other People working in Algorithms Education (non-exhaustive)

- Jan Vahrenhold
- Matthew Ferland
- Ryan Dougherty
- Mike Shindler
- Tim Randolph
- Robbie Weber
- <Your Name Here>

Thank You!

Any Questions?