Modern Course Design

Kalpathi Subramanian, Erik Saule krs@uncc.edu, esaule@uncc.edu

The University of North Carolina at Charlotte

BRIDGES Summer Workshop 2022

Table of Contents

- Modern Course Design
 - High level view
 - Principles of Alignment
 - Curriculum Guidelines
- 2 A Quick Plug for CS Materials
- 3 Activities

High Level Picture

Structured in Module

Content is grouped themes

Dependencies between modules

- Catch up modules
- Mandatory Modules
- Optional Modules

Goals

- Topics
- Course Learning Outcomes
- Program Learning Outcomes
- Competencies
- Pedagogical Strategies

A Variety of Content

- Lectures
- Videos
- In-class activities
- Assignments
- Exam
- Projects

Accreditation

- SACS
- ABET
- Quality Matters
- insert your accreditation body here

What is Alignment?

Properties of how content flow in • Program

- Course
- Module
- Materials

That could apply to

- Topics
- Outcomes
- Competencies

That could be in term of

- What they cover
- What they assume students know

Aligning Modules with Course Objectives

Courses usually have objectives that come from program descriptions and assessments. How do we ensure that the content of the class actually serve these higher objective? We want to align the objective modules with the objective of the course.

Two main properties to check:

- Are all the course objectives covered appropriately by a module objective?
- Are there module objectives that serve no course objective?

Alignment within Module

Typical module structure

- Exposition to new concept (lecture, textbook)
- Clarification of concept (discussion, hands-on activity)
- Reinforcement of concept (problem, programming assignment)

Properties you want

- The clarification should not introduce new concepts
- The reinforcement should strengthen the exposition and clarification topics
- The materials should cover the topics the module is meant to cover
- The materials should not wander too far from the module objectives

Assessment

Exam should never introduced new concepts

Plan for ITCS 6114: Algorithms and Data Structures

Overreaching Learning Outcomes

OLO1. Articulate that design, complexity, and correctness of algorithms and data structures matter in the real world

OLO2. Design correct and low complexity algorithms and data structures by employing standard techniques

OLO3. Analyze and implement given algorithms and data structures

OLO4. Recognize faulty algorithmic logic

Detailed Learning Outcomes

On Complexity

 ${\tt DLOC1.}\ Interpret\ complexity\ notation\ and\ their\ implications\ on\ the\ performance/resource\ consumption\ of\ algorithms\ ({\tt OLO1},{\tt OLO2})$

 $\hbox{DLOC2. Articulate the real-world implication of the design of algorithms and data structures in term of performance (OLO1) \\$

DLOC3. Derive the complexity of algorithms using various techniques (for instance, master theorem, amortized analysis, and average case analysis) (OLO1, OLO2, OLO3, OLO4)

DLOC4. Prove the NP-Completeness of classic problems (OLO2, OLO4)

DLOC5. Leverage the P != NP conjecture to recognize dubious algorithmic claims (OLO4)

On Correctness

DLOCo1, Recognize and prove the invariant of data structures and algorithms (OLO2, OLO4)

On Data Structures

DLOD1. Design, analyze, and implement tree-based indexes (OLO2, OLO3)

DLOD2. Design, analyze, and implement hash-based indexes (OLO2, OLO3)

DLOD3. Design, analyze, and implement classic algorithms on graphs (OLO2, OLO3)

On Algorithmic Techniques

DLOA1. Create, analyze, and implement divide and conquer algorithms (OLO2, OLO3)

DLOA2. Create, analyze, and implement greedy algorithms (OLO2, OLO3)

DLOA3. Create, analyze, and implement dynamic programming algorithms (OLO2, OLO3)

Week 1. Sep 8.

Lecture

- 1 Introduction
- 2. Complexity notations [DLOC1].

Activity:

- 1. Proving simple complexity notation properties [DLOC1].
- 2. Interpreting complexity notation in term of practical cost or feasibility [DLOC1, DLOC2].

Week 2. Sep 15.

Lecture:

- 1. Analyzing simple algorithms [DLOC3].
- 2. Invariant and correctness [DI OCo1]
- Simple recursive complexity formulas [DLOC3].

Activity:

- Given simple algorithms (binary search, insertion sort, simple nearest neighboor), prove their correctness and complexity (DLOCa). DLOCa).
- 2. Implement and benchmark insertion sort and simple nearest neighboor [DLOC1, DLOC2].

Week 3. Sep 22.

Lecture:

- 1. Divide and Conquer [DLOA1].
- Merge sort [DLOA1, DLOCo1].
 Master Theorem [DLOC3].

Activity:

- 1. Solve some other problem using D&C [DLOA1].
- Implement and benchmark Merge Sort [DLOC2].

Week 4. Sep 29.

Lecture:

- 1. Tree-based indexing [DLOD1].
- Invariant of data structure [DLOCo1].
- 3. Using BST for associative array [DLOD1, DLOC2].

Activity:

- 1. Run BST manually on toy example [DLOD1].
- Design, analyze, and implement a tree base index for nearest neighbor query [DLOD1, DLOC2].

Week 5. Oct 6.

Lecture-

Curriculum Guidelines

What are they?

Usually they are recommendation of what should/could be taught across a program. Expressed in term of topics, learning outcome, and competencies. Not in term of courses.

Usually make recommendation on how much one should learn in a particular topic, sometimes specified in number of hours.

How can we use them?

Give us a reference of what we should/could be teaching.

Am I covering all that? Should I? Why not?

Give us a common language to communicate between instructors.

General Guidelines: ACM/IEEE CS 2013

Structured in

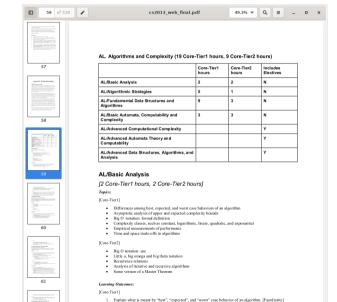
- Knowledge Area
- Knowledge Unit

Topics and Learning Outcomes are classified as

- Tier-1
- Tier-2
- Elective

Other general guidelines:

- Data Science
- Computer Engineering
- Upcoming revised CS



Specific Guidelines: NSF/IEEE-TCPP PDC 2012

Structured in domains:

- Programming
- Algorithm
- Architecture

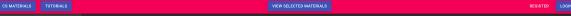
More descriptive.

Bloom levels.

Other specific guidelines: graphics, security

Table of Contents

- Modern Course Design
 - High level view
 - Principles of Alignment
 - Curriculum Guidelines
- A Quick Plug for CS Materials
- Activities



Analyzing Fig. Material Views A Fig. Select Materials *It Select Collections A. Radial View ACM-CSC 2013 A. Radial View PDC 2012 II. Harmonization View

Comparison

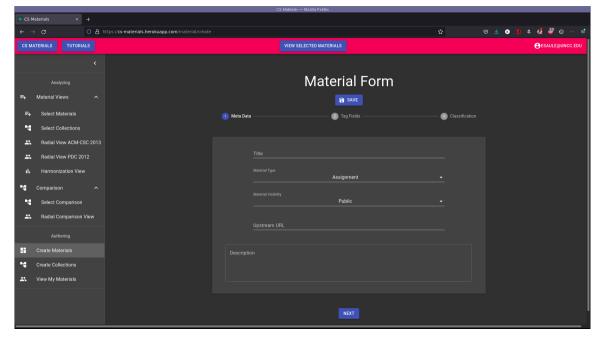
Select Comparison

Radial Comparison View

CS Materials

Create, Analyze and Search for computer science materials that are classified against the ACM and PDC quidelines.





CS Materials						
	→ c		O 8	https://cs-materials.herokuapp.com/material/create	슙	♥ よ ⑤ S よ ⑥ # ○ → □
		IALS TUTOR	IALS	VIEW SELECTED MATERIALS		Aesaulemuncc.et
	← A	ACM CSC 201		Q Search SAVE		
		Root::ACM/IEEE	Curricul	um Guidelines for Undergraduate Degree Programs in Computer Science		VIEW SELECTED TAGS
] Knowledge Are	a::Algo	ithms and Complexity		
Ε.		☐ Knowledge l	Jnit::Bas	ic Analysis		
		☐ Knowledge I	Jnit::Adv	vanced Data Structures Algorithms and Analysis		
		☐ Knowledge I	Jnit::Alg	orithmic Strategies		
		☐ Knowledge I	Jnit::Fur	adamental Data Structures and Algorithms		
		☐ Learning (Outcom	e::Implement basic numerical algorithms.		
		☐ Learning (Outcom	e::Implement simple search algorithms and explain the differences in their time complexities.		
		Learning (Outcom	e::Be able to implement common quadratic and O(N log N) sorting algorithms.		
- t		☐ Learning (outcom	e::Describe the implementation of hash tables, including collision avoidance and resolution.		
		Learning (outcome	e:Discuss the runtime and memory efficiency of principal algorithms for sorting, searching, and hashing.		
		Learning (Outcome	e:Discuss factors other than computational efficiency that influence the choice of algorithms, such as programming	time, maintainability, and the use of application-specific p	atterns in the input data.
		Learning (Outcom	e::Explain how tree balance affects the efficiency of various binary search tree operations.		
		Learning (Outcom	e::Solve problems using fundamental graph algorithms, including depth-first and breadth-first search.		
Earning Outcome::Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithms.					or that selection, and to implement the algorithm in a parti	cular context.
•t		Learning (Outcome	e::Describe the heap property and the use of heaps as an implementation of priority queues.		
_		Learning (outcom	e::Solve problems using graph algorithms, including single-source and all-pairs shortest paths, and at least one minim	num spanning tree algorithm.	
		Learning (Jutcom	e:Trace and/or implement a string-matching algorithm.		
		☐ Topic::Gra	phs and	graph algorithms (Tier 2)		
		☐ Topic::Hea	aps			

CS Materials — Mozilla Firefox

Why?

Study of Coverage

- We can easily understand what one course is covering.
- We can understand across multiple offfering of the same course what that particular course is about.
- We can identify different "flavors" of that course.

Search and Recommendation

- A Nifty Issue
- Curriculum Guidelines as Features
- Given a set of materials, find more that use these topics/outcomes

Table of Contents

- Modern Course Design
 - High level view
 - Principles of Alignment
 - Curriculum Guidelines
- 2 A Quick Plug for CS Materials
- Activities

Activities

Upload your materials in Google Drive

- Structure them in modules
- BRIDGES team will look at them overnight