ossu / computer-science Public

<> Code ⊙ Issues 8 Pull requests 1 Actions Projects Wiki

master ▾ 4 branches 5 tags Go to file Add file ▾ Code ▾

... 9a33b0d 9 days ago 955 commits

.git...	Update issue...	2 years ago
cou...	Fix typo	27 days ago
extr...	Add DSA Te...	2 months ago
.giti...	Update .gitig...	2 years ago
CH...	Make note m...	2 years ago
CO...	Update link t...	2 years ago
CU...	Clarify that C...	2 years ago
FA...	Add link to ...	6 months ago
HEL...	Use Discord ...	2 years ago
LIC...	Update Lice...	3 years ago
PR...	Update PRO...	10 days ago
REA...	Rename intr...	27 days ago

About

🎓 Path to a free self-taught education in Computer Science!

#computer-science #curriculum #courses #awesome-list

Readme

MIT license

122k stars

5.3k watching

16.3k forks

Releases

5 tags

Packages

No packages published

Contributors 124

+ 113 contributors

OSSU
Open Source Society University

Open Source Society

University

Path to a free self-taught education in Computer Science!

awesome

OSSU computer-science

Contents

Summary

The OSSU curriculum is a **complete education in computer science** using online materials. It's not merely for career training or professional development. It's for those who want a proper, *well-rounded* grounding in concepts fundamental to all computing disciplines, and for those who have the discipline, will, and (most importantly!) good habits to obtain this education largely on their own, but with support from a worldwide community of fellow learners.

It is designed according to the degree requirements of undergraduate computer science majors, minus general education (non-CS) requirements, as it is assumed most of the people following this curriculum are already educated outside the field of CS. The courses themselves are among the very best in the world, often coming from Harvard, Princeton, MIT, etc., but specifically chosen to meet the following criteria.

Courses must:

- Be open for enrollment
- Run regularly (ideally in self-paced format, otherwise running multiple times per year)
- Be of generally high quality in teaching materials and pedagogical principles
- Match the curricular standards of the CS 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science

When no course meets the above criteria, the coursework is supplemented with a book. When there are courses or books that don't fit into the curriculum but are otherwise of high quality, they belong in extras/courses or extras/readings.

Organization. The curriculum is designed as follows:

- *Intro CS*: for students to try out CS and see if it's right for them
- *Core CS*: corresponds roughly to the first three years of a computer science curriculum, taking classes that all majors would be required to take
- *Advanced CS*: corresponds roughly to the final year of a computer science curriculum, taking electives according to the student's interests
- *Final Project*: a project for students to validate, consolidate, and display their knowledge, to be evaluated by their peers worldwide

Duration. It is possible to finish within about 2 years if you plan carefully and devote roughly 20 hours/week to your studies. Learners can use this spread to estimate their end date. Make a copy and input your start date and expected hours per week in the `Timeline` sheet. As you work through courses you can enter your actual course completion dates in the `Curriculum Data` sheet and get updated completion estimates.

Cost. All or nearly all course material is available for free. However, some courses may charge money for assignments/tests/projects to be graded. Note that both Coursera and edX offer financial aid.

Decide how much or how little to spend based on your own time and budget; just remember that you can't purchase success!

Process. Students can work through the curriculum alone or in groups, in order or out of order.

- We recommend doing all courses in Core CS, only skipping a course when you are certain that you've already learned the material previously.
- For simplicity, we recommend working through courses (especially Core CS) in order from top to bottom, as they have already been topologically sorted by their prerequisites.
- Courses in Advanced CS are electives. Choose one subject (e.g. Advanced programming) you want to become an expert in and take all the courses under that heading. You can also create your own custom subject, but we recommend getting validation from the community on the subject

you choose.

Content policy. If you plan on showing off some of your coursework publicly, you must share only files that you are allowed to. *Do NOT disrespect the code of conduct* that you signed in the beginning of each course!

How to contribute

Getting help (Details about our FAQ and chatroom)

Community

- We have a discord server! 2408 online This should be your first stop to talk with other OSSU students. Why don't you introduce yourself right now? Join the OSSU Discord
- You can also interact through GitHub issues. If there is a problem with a course, or a change needs to be made to the curriculum, this is the place to start the conversation. Read more here.
- Subscribe to our newsletter.
- Add **Open Source Society University** to your Linkedin profile!
- Note: There is an unmaintained and deprecated firebase app that you might find when searching OSSU.

You can safely ignore it. Read more in the FAQ.

Curriculum

Curriculum version: `8.0.0` (see CHANGELOG)

Prerequisites

- Core CS assumes the student has already taken high school math, including algebra, geometry, and pre-calculus.

- Advanced CS assumes the student has already taken the entirety of Core CS and is knowledgeable enough now to decide which electives to take.
- Note that Advanced systems assumes the student has taken a basic physics course (e.g. AP Physics in high school).

Intro CS

Introduction to Programming

If you've never written a for-loop, or don't know what a string is in programming, start here. This course is self-paced, allowing you to adjust the number of hours you spend per week to meet your needs.

Topics covered: `simple programs` `simple data structures`

Courses	Duration	Eff
Python for Everybody	10 weeks	1 hours

Introduction to Computer Science

This course will introduce you to the world of computer science. Students who have been introduced to programming, either from the courses above or through study elsewhere, should take this course for a flavor of the material to come. If you finish the course wanting more, Computer Science is likely for you!

Topics covered: `computation` `imperative programming` `basic data structures and algorithms` `and more`

Courses	Duration	
Introduction to Computer Science and Programming using Python (alt)	9 weeks	ho

Core CS

All coursework under Core CS is **required**, unless otherwise indicated.

Core programming

Topics covered: `functional programming` `design for testing` `program requirements` `common design patterns` `unit testing` `object-oriented design` `static typing` `dynamic typing` `ML-family languages (via Standard ML)` `Lisp-family languages (via Racket)` `Ruby` `and more`

The How to Code courses are based on the textbook How to Design Programs. The First Edition is available for free online and includes problem sets and solutions. Students are encouraged to do these assignments.

Courses	Duration	
How to Code - Simple Data	7 weeks	ho
How to Code - Complex Data	6 weeks	ho
Programming Languages, Part A	5 weeks	ho
Programming Languages, Part B	3 weeks	ho
Programming Languages, Part C	3 weeks	ho

Object-Oriented Design	4 weeks	ho
Design Patterns	4 weeks	ho
Software Architecture	4 weeks	ho

Core math

Discrete math (Math for CS) is a prerequisite and closely related to the study of algorithms and data structures. Calculus both prepares students for discrete math and helps students develop mathematical maturity.

Topics covered: `discrete mathematics` `mathematical proofs` `basic statistics` `O-notation` `discrete probability` `and more`

Courses	Duration	
Calculus 1A: Differentiation (alt)	13 weeks	hc
Calculus 1B: Integration	13 weeks	hc
Calculus 1C: Coordinate	6 weeks	

Systems & Infinite Series		hc
Mathematics for Computer Science (alt)	13 weeks	hc

CS Tools

Understanding theory is important, but you will also be expected to create programs. There are a number of tools that are widely used to make that process easier. Learn them now to ease your future work writing programs.

Topics covered: `terminals and shell scripting` `vim` `command line environments` `version control` `and more`

README.md

The Missing Semester of Your CS Education	2 weeks	12 hours/

Core systems

Topics covered: `procedural programming` `manual memory management` `boolean algebra` `gate logic` `memory` `computer architecture` `assembly` `machine language` `virtual machines` `high-level languages` `compilers` `operating systems` `network protocols` `and more`

Courses	Duration	E
Build a Modern Computer from First Principles: From Nand to Tetris (alt)	6 weeks	7 hou
Build a Modern Computer from First Principles: Nand to Tetris Part II	6 weeks	1 hou
Operating Systems: Three Easy Pieces	10-12 weeks	6 hou
Computer Networking: a Top-Down Approach	8 weeks	4 hou

Core theory

Topics covered: `divide and conquer` `sorting and searching` `randomized algorithms` `graph search` `shortest paths` `data structures` `greedy algorithms` `minimum spanning trees` `dynamic programming` `NP-completeness` `and more`

Courses	Duration	
Divide and Conquer, Sorting and Searching, and Randomized Algorithms	4 weeks	ho
Graph Search, Shortest Paths, and Data Structures	4 weeks	ho
Greedy Algorithms, Minimum Spanning Trees, and Dynamic Programming	4 weeks	ho
Shortest Paths Revisited,		

NP-Complete Problems and What To Do About Them	4 weeks	ho

Core security

Topics covered
Confidentiality, Integrity, Availability Secure Design Defensive Programming Threats and Attacks Network Security Cryptography and more

Courses	Duration	
Cybersecurity Fundamentals	8 weeks	hc
Principles of Secure Coding	4 weeks	hc
Identifying Security Vulnerabilities	4 weeks	hc

Choose **one** of the following:

Courses	Duration
Identifying Security Vulnerabilities in C/C++Programming	4 weeks
Exploiting and Securing Vulnerabilities in	4 weeks

Java Applications	

Core applications

Topics covered: `Agile methodology` `REST` `software specifications` `refactoring` `relational databases` `transaction processing` `data modeling` `neural networks` `supervised learning` `unsupervised learning` `OpenGL` `ray tracing` and more

Courses	Duration
Databases: Modeling and Theory	2 weeks
Databases: Relational Databases and SQL	2 weeks
Databases: Semistructured Data	2 weeks
Machine Learning	11 weeks
Computer Graphics	6 weeks
Software Engineering: Introduction	6 weeks

Core ethics

Topics covered: `Social Context` `Analytical Tools` `Professional Ethics` `Intellectual Property` `Privacy and Civil Liberties` `and more`

Courses	Duration	
Ethics, Technology and Engineering	9 weeks	hc
Introduction to Intellectual Property	4 weeks	hc
Data Privacy Fundamentals	3 weeks	hc

Advanced CS

After completing **every required course** in Core CS, students should choose a subset of courses from Advanced CS based on interest. Not every course from a subcategory needs to be taken. But students should take *every* course that is relevant to the field they intend to go into.

Advanced programming

Topics covered: debugging theory and practice goal-oriented programming parallel computing object-oriented analysis and design UML large-scale software architecture and design and more

Courses	Duration	
Parallel Programming	4 weeks	ho
Compilers	9 weeks	ho
Introduction to Haskell	14 weeks	
Learn Prolog Now! (alt)*	12 weeks	
Software Debugging	8 weeks	ho
Software Testing	4 weeks	ho

(*) book by Blackburn, Bos, Striegnitz (compiled from source, redistributed under CC license)

Advanced systems

Topics covered: digital signaling combinational logic CMOS technologies sequential logic finite state machines processor instruction sets caches pipelining virtualization parallel processing virtual memory synchronization primitives system call interface and more

Courses	Duration	I
Computation Structures 1: Digital Circuits alt1 alt2	10 weeks	hou
Computation Structures 2: Computer Architecture	10 weeks	hou
Computation Structures 3: Computer Organization	10 weeks	hou

Advanced theory

Topics covered: `formal languages` `Turing machines` `computability` `event-driven concurrency` `automata` `distributed shared memory` `consensus algorithms` `state machine replication` `computational geometry theory` `propositional logic` `relational logic` `Herbrand logic` `game trees` `and more`

Courses	Duration	
Theory of Computation (Lectures)	8 weeks	h
Computational Geometry	16 weeks	h
Game Theory	8 weeks	h

Advanced Information Security

Courses	Duration	
Web Security Fundamentals	5 weeks	l
Security Governance & Compliance	3 weeks	l
Digital Forensics	3 weeks	l

Concepts		
Secure Software Development: Requirements, Design, and Reuse	7 weeks	h
Secure Software Development: Implementation	7 weeks	h
Secure Software Development: Verification and More Specialized Topics	7 weeks	h

Advanced math

Courses	Duration	E
Essence of Linear Algebra	-	
Linear Algebra	14 weeks	hou
Introduction to Numerical Methods	14 weeks	hou
Introduction	10	

to Logic	weeks	hou
Probability	24 weeks	hou

Final project

OSS University is project-focused. The assignments and exams for each course are to prepare you to use your knowledge to solve real-world problems.

After you've gotten through all of Core CS and the parts of Advanced CS relevant to you, you should think about a problem that you can solve using the knowledge you've acquired. Not only does real project work look great on a resume, but the project will also validate and consolidate your knowledge. You can create something entirely new, or you can find an existing project that needs help via websites like CodeTriage or First Timers Only.

Students who would like more guidance in creating a project may choose to use a series of project oriented courses. Here is a sample of options (many more are available, at this point you should be capable of identifying a series that is interesting and relevant to you):

Courses	Duration
Fullstack Open	12 weeks
Modern Robotics (Specialization)	26 weeks
Data Mining (Specialization)	30 weeks
Big Data (Specialization)	30 weeks
Internet of Things (Specialization)	30 weeks
Cloud Computing (Specialization)	30 weeks
Data Science (Specialization)	43 weeks
Functional Programming in Scala (Specialization)	29 weeks
Game Design and Development with Unity 2020 (Specialization)	6 months

Evaluation

Upon completing your final project:

- Submit your project's information to PROJECTS via a pull request.

- Put the OSSU-CS badge in the README of your repository!
  OSSU computer-science

  - Markdown: `[![Open Source Society University - Computer Science](https://img.shields.io/badge/OSSU-computer--science-blue.svg)](https://github.com/ossu/computer-science)`
  - HTML: `<a href="https://github.com/ossu/computer-science"><img alt="Open Source Society University - Computer Science" src="https://img.shields.io/badge/OSSU-computer--science-blue.svg"></a>`

- Use our community channels to announce it to your fellow students.

Solicit feedback from your OSSU peers. You will not be "graded" in the traditional sense — everyone has their own measurements for what they consider a success. The purpose of the evaluation is to act as your first announcement to the world that you are a computer scientist and to get experience listening to feedback — both positive and negative.

The final project evaluation has a second purpose: to evaluate whether OSSU, through its community and curriculum, is successful in its mission to guide independent learners in obtaining a world-class computer science education.

Cooperative work

You can create this project alone or with other students! **We love cooperative work**! Use our channels to communicate with other fellows to combine and create new projects!

Which programming languages should I use?

My friend, here is the best part of liberty! You can use **any** language that you want to complete the final project.

The important thing is to **internalize** the core concepts and to be able to use them with whatever tool (programming language) that you wish.

Congratulations

After completing the requirements of the curriculum above, you will have completed the equivalent of a full bachelor's degree in Computer Science. Congratulations!

What is next for you? The possibilities are boundless and overlapping:

- Look for a job as a developer!
- Check out the readings for classic books you can read that will sharpen your skills and expand your knowledge.
- Join a local developer meetup (e.g. via meetup.com).
- Pay attention to emerging technologies in the world of software development:
  - Explore the **actor model** through Elixir, a new functional programming language for the web based on the battle-tested Erlang Virtual Machine!
  - Explore **borrowing and lifetimes** through Rust, a systems language which achieves memory- and

thread-safety without a garbage collector!
    - Explore **dependent type systems** through Idris, a new Haskell-inspired language with unprecedented support for type-driven development.

Code of conduct

OSSU's code of conduct.

How to show your progress

1. Create an account in Trello.
2. Copy this board to your personal account. See how to copy a board here.

Now that you have a copy of our official board, you just need to pass the cards to the `Doing` column or `Done` column as you progress in your study.

We also have **labels** to help you have more control through the process. The meaning of each of these labels is:

- `Main Curriculum` : cards with that label represent courses that are listed in our curriculum.

- `Extra Resources` : cards with that label represent courses that were added by the student.
- `Doing` : cards with that label represent courses the student is currently doing.
- `Done` : cards with that label represent courses finished by the student. Those cards should also have the link for at least one project/article built with the knowledge acquired in such a course.
- `Section` : cards with that label represent the section that we have in our curriculum. Those cards with the `Section` label are only to help the organization of the Done column. You should put the *Course's cards* below its respective *Section's card*.

The intention of this board is to provide our students a way to track their progress, and also the ability to show their progress through a public page for friends, family, employers, etc. You can change the status of your board to be *public* or *private*.

Team

- **Eric Douglas**: founder of OSSU

- Josh Hanson: lead technical maintainer
- Waciuma Wanjohi: lead academic maintainer
- Contributors