2019 MCS HS Workshop

Curriculum Draft for Grade 10, 11, and 12 Computer Science

Overview and Notes

Teacher Feedback

Some care is required when developing a curriculum because Grade 10 CS is *not a requirement* for Grade 11 CS. Thereby Grade 10 needs to be a *self-contained* class that can service students of a wide variety of skills.

It is not uncommon that students are taking the CS courses for completing degree requirements and *not* out of pure interest.

It is hard (if not impossible) to address plagiarism or non-completion. This ill-equips students for transitioning into University where plagiarism is taken seriously and is strictly punished.

Some of Walter Isaacson "The Innovators'' should be required reading.

There are curriculum requirements, like Environmental Impact considerations, that need to be shoehorned into the teaching.

Students are losing their connection with computer hardware. Phones have abstracted away the file system.

Students no longer understand the file and directory abstraction. They are being "spoon-fed" by us to take care of these issues -- Google Classroom, frameworks to verify their submission correctness (e.g. that the student can't submit a file unless named "ABC.XYZ").

CS education guru, Mark Guzdial; Engineering Professor, University of Michigan. Mark will be attending the International Computing Education Research Conference (ICER 2019) occuring on August 12-14, 2019 in Toronto. ***This is open to high-school teachers and they are encouraged to attend.***

Student Feedback

Students reported conflicting opinions about the "Game-ification" of the curriculum. On the one hand it was motivating to have a final product (i.e. a game) to show to friends; on the other hand this approach tends to trivialize the subject matter.

Working in groups was a new, difficult, but ultimately useful experience.

Having a final product to show off at the end of the year is a good motivator.

Python was the preferred starting language. Switching languages to Java the following year was regarded as beneficial as it emphasized the portability of algorithms. C is definitely too hard because of memory management.

Students feel that syntax mistakes *should be penalized* at their level because otherwise they develop bad habits.

Writing out algorithms *on paper* should not be phased out --- it is a helpful aspect of the design phase.

Grade 10

Note: This course is designed to be self contained.

Digital Literacy

Markup

Introduce markup languages like HTML with CSS, LaTeK, and XML. Explain the differences between these and WYSIWYG ("what you see is what you get"). Use this as a starting point to demonstrate computers need to be instructed in a rigid manner.

Culminating task: Personal website.

Markdown

Structured plaintext language. Excellent for quickly creating notes and documents that compile to something nice looking. Supported by Wordpress.

Random school-board requirements can be shoehorned here. Have students write reflections, journal entries or reports using the markdown.

Hardware

A great way to engage practically minded students with a “hands on” approach. Re-establishes a connection with instructions and the machinery that is being instructed. A great place to have competitions (i.e. fastest robot through maze). Emphasize the memory model and that memory is *physically changed* in space.

Use of Micro:bit, Arduino, Raspberry Pi is advisable here. All have good APIs and enjoy wide community support.

Introduction to Programming

Use some of the programming games (e.g. LOGO) to introduce the students to the notion of issuing *ordered* and *unambiguous* instructions to the computer.

Grade 11

Note: This course should not and does not require Grade 10 as a prerequisite.

Introduction to Programming

Use a game to (re)introduce the notion of communicating to the computer with *non-ambiguous*, *ordered* instructions, that *terminate* (i.e. the notion of an algorithm).

Python

Introduce Python and an IDE. Start with structured printing and using the IDE as a calculator. Consider Dr Kevin Browne's "Simon Says Programming" where the teacher programs and has the class follow along on their own machines.

Introduce: Boolean Logic, If and Case statements, Loops, Variables. X := X + 1

Code Refactoring

Talk about Code Equivalence (when two pieces of code do the same thing). Demonstrate how a simple piece of code can do something complicated and that simpler code is desirable because it is easier to maintain and debug.

Have students simplify code with redundant if-statements and non-triggered conditions.

Basic Data Structures

Introduce the notion of a Data Structure by way of arrays and strings. Have the students implement:

1. All of the vector operations they have been taught (pointwise addition and dot product).
2. Various simple cryptography (Caesar Ciphers, encoding and decoding).

Debugging

Introduce code tracing and generic debugging techniques. Problem solving code is key here.

Formatting and Style

Naming: files, functions, variables, etc.. Following "good" standards/conventions (e.g. PEP-8, Code Conventions, etc…).

Grade 12

Consider switching the language to Java.

Advanced Data Structures

Introduce Matrices and then have the students build their own Linear Algebra library. Indicate to them that they will be using this library for subsequent projects so it is advisable for them to document and organize.

Linear Algebra Package

This will require the students to decide on a *data representation* for Matrices and Vectors. The commands required are

1. Dot product,
2. Matrix Product,
3. Transpose, and
4. Inverse (may be too difficult).

Here is a good place to mention *complexity* as the students would only need to analyze embedded loops.

Culminating task: Do Linear Transformations by using matrices to move points around the plane. Introduce the notion of a .jpg as a matrix of RGB values and then have the students translate jpg pictures.

Intro Data Science and Visualization

Implement the numerical algorithms from mathematics: Integer Division, Newton's Method, 2D gradient descent. The Linear Algebra package will be required to do this.

Have the students output pictures of their optimizations so they can see an animation of a line “best fitting” some scatter plot. Move to 3D (by just giving the equations for the grad descent) if group is strong.

Introduce *libraries* and use one (like) for plotting data in various ways.

Read .CSV files and pull information out that Excel would not be able to handle. This is basically introduction to databases.

Culminating task: Use open data to discover something interesting.

Resources

Software Practice and Tools

Name	Description	Link
LeetCode	Collection of problems taken from industry coding interviews. Problems range from very easy to difficult. Solutions can be in a variety of languages and are compiled in browser.	Leetcode.com Selected Solutions
HackerRank		hackerrank.com
Code	Encourages schools to include more computer science classes in the curriculum by offering free coding lessons and other initiatives.	code.org
Coursera	Coursera works with universities to offer online courses, specializations, and degrees in a variety of subjects. Digital LinkedIn certificated can be obtained for program completions.	coursera.org
MIT Open Courseware	MIT is trying to publish all of their undergraduate and graduate courses online for free.	ocw.mit.edu/index.htm
Code Academy	Codecademy is an online platform that offers free coding classes in 12 different programming languages	codecademy.com
Khan Academy	Education through short lessons in the form of videos.	khanacademy.org
Wordpress	Most associated with blogging, this is a free and open-source content management system based on PHP & MySQL. It includes a plugin architecture and a template system.	wordpress.com
Tinkercad	3D Modelling and Design Software	tinkercad.com
Autodesk Inventor		autodesk.com/products/inventor/overview
Vectary		vectary.com
Integrated Development Environment (IDE)	A software application that provides comprehensive facilities to programmers/developers for software development.	**Python**
		jetbrains.com/pycharm jupyter.org
		Java
		eclipse.org jetbrains.com/idea

		netbeans.apache.org
Overleaf	Online LaTeX editor allowing collaboration.	overleaf.com

Hardware Resources

Name	Description	Link
Raspberry Pi	Low-cost computer capable enabling people of all ages to explore computing, and to learn programming in Scratch and Python.	raspberrypi.org
Arduino	Single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control both physically and digitally.	arduino.cc
LEGO Mindstorm	A software platform produced by LEGO for robot construction and development.	lego.com/en-us/mindstorms
VEX Robotics Design System	Tools for educators to assist in robotics and other software related endeavours.	www.vexrobotics.com

Plagiarism Software

Name	Description	Link
Measure of Software Similarity (MOSS)	An automated system for detecting similarities in software programs; primarily used for detecting plagiarism.	theory.stanford.edu/~aiken/moss

Open Data

Name	Description	Link
Province of Ontario	Data provided by the Province of Ontario.	ontario.ca/search/data-catalogue
Government of Canada	Data provided by the Government of Canada.	open.canada.ca/en/open-data
Twitter Data	Twitter's summary account revolving around data-driven insights.	twitter.com/twitterdata
Google Trends	A site that analyzes the popularity of top	trends.google.com/tren

	search queries	ds

Programming Games

Name	Description	Link
Code Combat	Live coding with a game component teaching JavaScript and Python.	codecombat.com
Code Wars	On the platform, software developers train on programming challenges known as kata.These discrete programming exercises train a variety of skills in a variety of programming languages, and are completed within an online IDE.	codewars.com
Python: PyGame	Free and Open Source python programming language library for making multimedia applications	pygame.org
Python: Zelle's Graphics	A simple object-oriented graphics library.	mcsp.wartburg.edu/zelle/python/
Python: Turtle Graphics	Vector graphics using a relative cursor (the "turtle") upon a Cartesian plane.	docs.python.org/3.8/library/turtle.html

Hard "Easy" Problems

These are good questions to give to students who are overconfident in their abilities or convinced they have learned everything.

The Backpack Problem

Without dynamic programming the naive solution has an exponential number of cases.

Assume you have a backpack of capacity (integer) weight C. You can choose among N treasures and the k^{th} treasure has value $V[k]$ and weight $W[k]$. Assume you can choose the same treasure multiple times.

What is the maximum value that can be carried?

Longest Increasing Subsequence

Also hopeless without dynamic programming.

The integer sequence

3, 1, 8, -1, 2, 4, 2

has 2^7 *subsequences*. One of them is

1, 2, 4

and has length three.

Given an integer sequence $x[0],\ x[1],\ \ldots,\ x[n]$, **find the longest increasing subsequence.**

Note: 1, 2, 2, 4 is **not** an increasing subsequence (it is only non-decreasing).

Rod Cutting Problem with Cost

Given a rod of length L and list of prices of rod of length i, where $1 \le i \le n$, **find the optimal way to cut the rod into smaller pieces in order to maximize the profit**.

Sudoku

Given a 9×9 matrix with some set (but not all) $\{0..9\}$ integer values, **solve the sudoku**.