Great Ideas in Computing

University of Toronto CSC196
Fall 2020

Week 1: September 13-September 17 (2021)

Course Organization

Course Instructor: Allan Borodin

- Email: bor@cs.toronto.edu
- Instructor course email: 196instr@cs.toronto.edu

Teaching Assistant : Gabriela (Gabi) Morgenshtern and Rachel Phinnemore
TA Course Email : 196ta@cs.toronto.edu
Note: These course emails may take a couple of days to set up

I strongly encourage questions and discussions in class. In addition, we are also using piazza for questions and discussions.
Piazza link piazza.com/utoronto.ca/fall2021/csc196

Using piazza, you can also answer questions posed by others. You can pose or answer questions anonymously or using your name. The benefit of in class and piazza (over email) is that the entire class benefits from the discussion. Please sign up.

Course organization continued

The class will normally meet M,W,F at 9AM in MY 350 but as I will soon mention, we will also meet (at the same) tme remotely for the firstv two weeks and then we will see. Of course, if COVID continues to spread in Toronto we will have to adjust our plans.

Usually I will be presenting on Mondays and Wednesdays and the tutorials will be on Fridays. You should not have a conflict with any of these three hour classes.

Course web site: http://www.cs.toronto.edu/~196f21/
My slides will be on the web page. I will also post various documents on the web page.
There are also links to the previous version of CSC196 (fall 2020) and versions of SCI199 on the web page.

I will mainly be using Quercus for Announcements but might also have some announcements on the web page.

COVID considerations

The University is continually updating information about good practices and what is required for students and instructors. I have some comments and links on the course web page.

- By October 14, everone ((with some well defined exceptions) on campus must have had their final shot of a COVID vaccine.
- Everyone should be wearing masks inside buildings including in classrooms. There is a possible allowance if the masks are considered to be a significant impediment to the class (e.g., if we can't be understood when talking). But we would have to be granted such an exception.
- I want (really require) your participation so please speak as loudly as necessary to be understood. **Make sure that I am being clear when I speak. I won't be offended.**
- Weather permitting we might have office hours (or maybe even the occasional class) outside (not requiring masks) if we can find a good place to talk without a lot of noise. On the other hand this room is really nice and made for a seminar.

COVID considerations continued

I recognize that everyone has their own sense of comfort as to how much to interact. We will be using a hybrid class for the first two weeks.

If you do not feel comfortable coming to class in person after the first two weeks please speak to me.

While we can ask for an exemption regarding face make wearing, I have not asked for that and hope we can conduct the class in person with face masks until further notice.

For a seminar course, interaction is very important but safety is clearly important.

Preliminaries

Getting to know each other

- A little about myself
- Your plans at the University?

What is this course about?

- FAS Calendar about First Year Foundational Seminars
  https://www.artsci.utoronto.ca/future/academic-opportunities/first-year-opportunities/first-year-foundations-seminars
  One might say that all of these first year courses are an "Introduction to Critical Thinking".
- Brief theme of our CSC196 course: Great Ideas in Computing.
  What constitutes a "great idea"?

Preliminaries

Getting to know each other

- A little about myself
- Your plans at the University?

What is this course about?

- FAS Calendar about First Year Foundational Seminars
  https://www.artsci.utoronto.ca/future/academic-opportunities/first-year-opportunities/first-year-foundations-seminars
  One might say that all of these first year courses are an "Introduction to Critical Thinking".
- Brief theme of our CSC196 course: Great Ideas in Computing.
  What constitutes a "great idea"?

  Impact, surprise, elegance. Of course, it is easier to agree on great ideas in retrospect rather than as ideas are being introduced. And of course ideas rarely occur in a "vacuum"; usually there are similar ideas known and often the timing of when an idea becomes viable is very critical. It is also the case that credit for an idea is not always completely fair to all those involved.

More thoughts on what constitutes a great idea

In class, students made the following suggestions:

- "It is a breakthrough"; meaning it accomplishes something that was not possible before.
- "It allows for new possibilities"
- "It provides an *optimal* solultion to a problem'
- "It is the first thing I would want to describe to someone not in CS"'

Great ideas, good decisions, good plans

- "At the time, it was not considered a good idea" Can we call an idea a great idea if it never gets adopted?

Great ideas, good decisions, good plans

- "At the time, it was not considered a good idea" Can we call an idea a great idea if it never gets adopted?
- How should we characterize a good decision or a good plan?

Great ideas, good decisions, good plans

- "At the time, it was not considered a good idea" Can we call an idea a great idea if it never gets adopted?
- How should we characterize a good decision or a good plan?
  A point of view: one probably should not judge the quality of a decision by outcomes. The quality of a good decision may simply be whether or not, given all the information available at the time the decision was made, the decision was a good or the best decision one could make.
- Decisions can be irrevocable or can be modified. Plans usually do evolve over time.

Great ideas, good decisions, good plans

- "At the time, it was not considered a good idea" Can we call an idea a great idea if it never gets adopted?
- How should we characterize a good decision or a good plan?
  A point of view: one probably should not judge the quality of a decision by outcomes. The quality of a good decision may simply be whether or not, given all the information available at the time the decision was made, the decision was a good or the best decision one could make.
- Decisions can be irrevocable or can be modified. Plans usually do evolve over time.
- It is important to understand that in all ideas, decisions, plans, there is always some degree of chance as to how the "world" will unfold.
- In research and development (and more generally in life) how long should we stick with our intuition or when do we see that our beliefs and plans are inconsistent with facts or "common wisdom"?

Great ideas may have negative consequences

There can be unintended, undesireable consequences for ideas we may come to accept as great ideas? Is it then still a great idea?

- If a technology becomes a standard, it can also become a barrier to innovation. For example, we will soon encounter our first great idea "The von Neumann" model. Some have argued that it has impeded progress on alternative computational architectures.
- Social networks allow for the rapid spread of information and mis-information. It is claimed that social media deliberately fails to act on hate speech while social media companies argue they are actively removing hate speech above and beyond reported cases.
- Moreover, some claim that these companies do not want to eliminate hate speech and that they often target users which in turn reinforces divisions in society. (See the links to article by Starbird et al and a text file with some links to the issue of hate speech and bias on social media platforms.)
- For controversial and ethical issues, we may sometimes use a debate format where students argue for a point of view and not their view.

Predictions and the nature of research

Niels Bohr, Mark Twain, Yogi Berra: "Predicitng the future is hard because it hasn'rt happened yet" and "it's tough to make predictions, especially about the future" Who knows who said it first.

- Predictions about computing and what and can and can't be done within some predicted time frame are often wrong. I posted a link to the 1955 Dartmouth summer project on AI (where the term seems to have first appeared). Turing's 1950 article provided what is now called the Turing Test as to "whether or not it is possible for machinery to show intelligent behaviour". The Dartmouth project suggests that the indicated challenges for AI could be done over the summer.
- I posted a link to an article by John Backus about the view that source level languages could never be nearly as efficient as machine code.
- There is also an article giving what one individual calls the "7 Worst Tech Predictions of All Time." But note that the quote attributed to T.J. Watson may never have happened.

Grading scheme, syllabus and possible topics

- The grading scheme will be based on 4 assignments (15% each), two quizzes (10% each), and class participation (20%). Students are expected to attend all classes regularly and participate actively. There will be no final exam.
- Assignments will be submitted on Markus
  https://markus.teach.cs.toronto.edu/csc196-2021-09/
  I will post tentative dates soon.
- It is difficult but not impossible to fail this course. Mainly do NOT plagarize. If you have any questons about plagarism, ask me.
- The syllabus (listed on the course web page) contains other organizational information.
- There is also an ambitious list of possible topics on the web page.
  What have we missed?
  When I ask a question in red, thats a strong invitation for YOU to join the conversation but don't wait for an invitation to speak up.

Relevant dates

- A0 September 20. This is part of the participation grade
- A1 October 8
- Q1 October 22
- A2 October 29
- *Note:* November 8 is the last day to drop a Fall (F) course.
- Reading week is November 8-12.
- A3 November 19
- Q2 November 26
- A4 December 3
- Fall classes end December 8.

Some possible topics

- What is a great idea?
- What responsibilties do computer professionals have for the impact (and possible misuse) of the technology?
- What is a computer? The von Neumann architecture. Digital vs analogue. What were the alternatives? What else is possible (parallel, quantum)?
- The genius of Alan Turing; A mathematical definition of computable function. Interpreters. Non computable functions.
- How did computers and computing become a commodity? The amazing advances in software and algorithms came along with advances in hardware (cost, speed, memory size, physical size, power) and communication (cost, capacity and speed) . Demand for "killer applications" such as word processors, email, search engines, navigation systems, games).
- The internet; packet routing. TCP/IP.

More possible topics

- Fortran, the first commericial source level language and compiler. John Backus vs the prevailing view that compiled code would be too slow compared to machine code.
- The semantic web.
- A local great idea: NP completeness. What is and what is not *efficiently* computable.
- Complexity based cryptography; public key cryptography; digital signatures. Captchas.
- Another local great idea: deep neural networks and the success of machine learning (ML). Computers vs human thought and learning.
- HCI (human computer interaction). The mouse. Menus, click, paste and drag. Visualization.

And a few more possible topics

- Information theory: the genius of Claude Shannon. Error correcting codes; compression.
- Social networks and the spread of information (and mis-information, conspiracies, etc). Targetting information to different communities. (What is a social network community?)
- Open Source. Wikipedia. Blogs
- Relational data bases.

What great ideas have we missed?

In class we had three additional suggestions; namely (algorithmic) game theory and mechanism design, video conferencing, virtual reality. The latter two suggestions fall into the scope of HCI. I will see if Fanny Chevalier might touch on these topics. One of the remote students suggested "big data privacy".

In Assignment A0, I asked you to rank your top 3 choices for a topic to be discussed. In addtion to any of the topics mentioned, feel free to add any topic not mentioned as part of your top three.

End of Monday, September 13 class

We ended Monday with a list of possible topics and added three more suggestions by students. After class, I received one more topic suggestion. After quickly listing our guest presentations, we are ready to begin our first topic.

Slides and guest speakers

I will have preliminary slides available before the week begins and then repost sometime at the end of the week (correcting typos, adding comments from the discussions, etc.)

NOTE: My slides will only be an outline of our discussions. **And we will often "wander" (i.e., take tangents) as we discuss ideas.**

I have lined up some guest speakers to lead discussions on some recent "great ideas". Here is the tentative schedule for these guest presentations:

- David Duvenaud (ML and Deep Learning) Wednesday September 29
- Nathan Wiebe (Quantum computation) Wednesday Oct 13
- Fanny Chevalier (HCI and Visualization) Wednesday November 3
- Kyros Kutulakos (Vision) End of November

Our first great idea

Our first topic/great idea for discussion: the von Neumann architecture

- By the mid 40s the first computers were being built. (I am not going to talk say about Babbage and the recent implementation of Babbages 1850 machine.)
- The earliest computers had fixed programs and were not general purpose machines.
- The stored program computer. The conceptual idea is associated with von Neumann who first described the model in a paper "First Draft of a Report on the EDVAC" dated June 30, 1945.
- It is clear that the stored program idea is present in Turing's 1936 paper which we will discuss later. But the Turing model was not meant to be a model of an actual computer.

The von Neumann architecture and digitilization

- The basic organization consists of 4 units: memory, I/O, ALU, control. (Some would say that a "bus" to cary signals between units is a fifth component but most say 4 units.) The memory is organized into a list or array of "words", each with its own address.
- Each word w_i (with say address i) is some fixed length string of bits bits ; i.e., $w = b_{n-1}b_{n-2}, \ldots, b_0$ which as an integer represents $\sum_{j=0}^{n-1} b_j 2^j$. In order to represent negative numbers, we can reserve the high order bit as a sign bit. Thus $b_{n-1}b_{n-2}\ldots b_0$ respresents $(-1)^{b_{n-1}} \sum_{j=0}^{n-2} b_j 2^j$. That is, $b_{n-1} = 1$ specifies a negative number. (The choice of 2 is not essential mathematically.). It is also possible to think of a word as a string of "bytes" in which each byte can also addressable but we will ignore that. Why not unary, decimal, or some other representation?

The von Neumann architecture and digitilization

- The basic organization consists of 4 units: memory, I/O, ALU, control. (Some would say that a "bus" to cary signals between units is a fifth component but most say 4 units.) The memory is organized into a list or array of "words", each with its own address.
- Each word w_i (with say address i) is some fixed length string of bits bits ; i.e., $w = b_{n-1}b_{n-2}, \ldots, b_0$ which as an integer represents $\sum_{j=0}^{n-1} b_j 2^j$. In order to represent negative numbers, we can reserve the high order bit as a sign bit. Thus $b_{n-1}b_{n-2}\ldots b_0$ respresents $(-1)^{b_{n-1}} \sum_{j=0}^{n-2} b_j 2^j$. That is, $b_{n-1} = 1$ specifies a negative number. (The choice of 2 is not essential mathematically.). It is also possible to think of a word as a string of "bytes" in which each byte can also addressable but we will ignore that. Why not unary, decimal, or some other representation?
- But before we go any further we need to step back and talk about *digitalization and encoding*. That is, what are we storing in these words?

The von Neumann architecture and digitilization

- The basic organization consists of 4 units: memory, I/O, ALU, control. (Some would say that a "bus" to cary signals between units is a fifth component but most say 4 units.) The memory is organized into a list or array of "words", each with its own address.
- Each word w_i (with say address i) is some fixed length string of bits bits ; i.e., $w = b_{n-1}b_{n-2}, \ldots, b_0$ which as an integer represents $\sum_{j=0}^{n-1} b_j 2^j$. In order to represent negative numbers, we can reserve the high order bit as a sign bit. Thus $b_{n-1}b_{n-2}\ldots b_0$ respresents $(-1)^{b_{n-1}} \sum_{j=0}^{n-2} b_j 2^j$. That is, $b_{n-1} = 1$ specifies a negative number. (The choice of 2 is not essential mathematically.). It is also possible to think of a word as a string of "bytes" in which each byte can also addressable but we will ignore that. Why not unary, decimal, or some other representation?
- But before we go any further we need to step back and talk about *digitalization and encoding*. That is, what are we storing in these words? Data and instructions (as part of a program)

Digital computing vs the continuous real world

- The concept of digitalization is one of the most profound concepts that underlies science today.

Digital computing vs the continuous real world

- The concept of digitalization is one of the most profound concepts that underlies science today.
- In the "real world", space and time are continuous A typical computing application might wish to study the trajectory or movement of an object (person, ball, missle, etc). An object is moving through continuous 3 dimensional space in continuous time according to some (hopefully) known laws of motion.
- In contrast, digital computers use discrete finitely represented data and instructions.

Digital computing vs the continuous real world

- The concept of digitalization is one of the most profound concepts that underlies science today.
- In the "real world", space and time are continuous A typical computing application might wish to study the trajectory or movement of an object (person, ball, missle, etc). An object is moving through continuous 3 dimensional space in continuous time according to some (hopefully) known laws of motion.
- In contrast, digital computers use discrete finitely represented data and instructions.

  However, a real value x can be approximated by a rational number. Every integer has a finite representation and every rational can be represented by a pair of integers and hence and hence has a finite reperesentation.
- Furthermore, we will see how we can approximate any number (within some range) using *floating point representation*.

Digital computing vs the real world continued

- Furthermore there are some stochastic aspects (such as wind and air polution in the trajectory example) in real world processes.
- In contrast, digital computers are determinsitic; that is, each "state" of the computer is precisely determined from the previous state.

Digital computing vs the real world continued

- Furthermore there are some stochastic aspects (such as wind and air polution in the trajectory example) in real world processes.
- In contrast, digital computers are determinsitic; that is, each "state" of the computer is precisely determined from the previous state.

  However, we can simulate stochastic events by drawing random variables from some (pseudo) random distribution.
- Stochastic processes can then be simulated by a discrete deterministic machine. Of course, the conclusions to be made from simulations relative to a specific outcome in the real world will depend on how well we have modelled the system and how unpredictable is the system.
- The early debate: relative benefits of digital computation vs that of analogue computation. (In the past, thermostats and other control devices were essentially simple analogue computers.) Why did digitlilization win the debate?

End of Wednesday, September 15 class

In have a few more slides relating to the von Neuman model and floating point representation which In will leave in this weeks slides and repeat in the slides for next week.

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each *fixed length word* is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each *fixed length word* is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each *fixed length word* is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy
- Algorithms consist of individual instructions that say what "basic operations" to perform on data and also to indicate what instruction to do next.
- Now here is a great idea (relating to digitalization): Instructions can also be represented by strings of symbols (indeed by strings of bits)! So instructions can also be stored in the memory, say for example one instruction per word!

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each *fixed length word* is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy
- Algorithms consist of individual instructions that say what "basic operations" to perform on data and also to indicate what instruction to do next.
- Now here is a great idea (relating to digitalization): Instructions can also be represented by strings of symbols (indeed by strings of bits)! So instructions can also be stored in the memory, say for example one instruction per word!
  Why is this such a great idea?

Memory and the von Neumann architecture

The von Neumann architecture is a *random access stored program model*.

- The von Neuman model assumes a random access memory in which each *fixed length word* is "addressable" and can be accessed at some unit (of time) cost.
  Is this how your lap top memory is organized?
  The"von Neumann bottleneck" addressed by caching and the memory hierarchy
- Algorithms consist of individual instructions that say what "basic operations" to perform on data and also to indicate what instruction to do next.
- Now here is a great idea (relating to digitalization): Instructions can also be represented by strings of symbols (indeed by strings of bits)! So instructions can also be stored in the memory, say for example one instruction per word!
  Why is this such a great idea?
  Why is the von Neumann model such a great idea?
  Are we stuck in a "von Neumann tarpit?"

Dataflow architecture

Direct from Wikipedia:

Dataflow architecture is a computer architecture that directly contrasts the traditional von Neumann architecture or control flow architecture. Dataflow architectures do not have a program counter (in concept): the executability and execution of instructions is solely determined based on the availability of input arguments to the instructions,[1] so that the order of instruction execution is unpredictable, i.e. behavior is nondeterministic. Although no commercially successful general-purpose computer hardware has used a dataflow architecture, it has been successfully implemented in specialized hardware such as in digital signal processing, network routing, graphics processing, telemetry, and more recently in data warehousing.[citation needed] It is also very relevant in many software architectures today including database engine designs and parallel computing frameworks.[citation needed]

More on data flow architecture

From J. Paul Morrison's Flow-Based Programming text

The von Neumann machine is perfectly adapted to the kind of mathematical or algorithmic needs for which it was developed: tide tables, ballistics calculations, etc., but business applications are rather different in nature. . . .
Business programming works with data and concentrates on how this data is transformed, combined, and separated. . . . Broadly speaking, whereas the conventional approaches to programming (referred to as "control flow") start with process and view data as secondary, business applications are usually designed starting with data and viewing processes as secondary—processes are just the way data is created, manipulated, and destroyed. We often call this approach "data flow." (21)

Multicore and Parallel Computation

- As you probably already know, computers today are often multicore machines meaning that some "small" constant number of processes can be running simultaneously.
- When people refer to large scale parallelism hey have in mind that the number of processes running in parallel can depend (at least conceptually) on the computation.
- The von Neumann architecture is an abstract model for *sequential computation*.
- In contrast to the well accepted von Neumann model for sequential computation, the situation for *parallel computation* is more nuanced.
  - There is the issue of a constant number of parallel processes vs a nnumber of processes that depends on the size of the data and/or the computation as it evolves.
  - Do the processes run syncronously (i.e. according to some golbal clock) or asynchronously?
  - Do the processes communicate mainly through a shared memory or via some communication bus?
  - How do we maintain consistency of the information being shared?

The benefits of a well agreed upon abstract model of computation

One of the reasons to consider the von Neumann model a great idea is that by being a well agreed upon model, coordination amongst different people is minimized. That is,

- A computer architect doesn't need to know which programming languages will be run on their specific architecture. (The von Neumann model doesn't specify the instruction set, the memory management, how interupts are handled, etc.)
- A compiler writer for a programming language $\mathcal{L}$ doesn't have to know what algorithms will be implemented using the language $\mathcal{L}$.
- Without complete knowledge of the architecture, and the compiler, the algorithm designer can make a rough approximation for the memory and time requirements of their algorithm.

The progress in parallel computation had been relatively slow but there is now some common approaches (e.g., MapReduce for large scale parallel computation).