

Great Ideas in Computing

University of Toronto CSC196
Fall 2022

Week 6: October 17 - October 21 (2022)

Week 6 slides

Announcements:

- This week we have Professor Fan Long giving a guest presentation on Wednesday peaking about block chain technology. One or two faculty members will be joining us as they are interested in Professor Long's presentation.
- We have our first quiz on Friday. The quiz starts at 9:10 and ends promptly at 10.
- Next week, we have another guest presenter, Professor Nisarg Shah speaking about fair division of resources.
- I have posted some questions for Assignment 2. As announced last week, given all the guest presenters it has become harder to create an assignment using our class discussions and slides. Therefore the assignment is now due on Tuesday, November 1 at 11PM.

Today's agenda

- Since I rushed through some material regarding undecidability, I will begin today with a review of the transformations and diagonalization.
- A brief discussion of the encoding of a Turing machine computation.
- New topic: search engines.

Transformations: A very special type of reduction

We will use a very simple type of reduction which I will call a *transformation* and which we can denote by \leq_{τ} . We say that $A \leq_{\tau} B$ if there is a computable function f such that $w \in A$ iff $f(w) \in B$.

It should be easy to see that $A \leq_{\tau} B$ is a special case of $\leq_{\mathcal{T}}$. **Is this obvious?**

When we do some complexity theory, we will further refine this concept by requiring that the transformation function f is computable in polynomial time (by a Turing machine).

Showing the desired transformation $\mathcal{L}_3 \leq_T \mathcal{L}_2$.

Here is a description of the transformation of $\langle M \rangle$ to $f(\langle M \rangle)$ such that $\langle M_3 \rangle \in \mathcal{L}_3$ iff $f(\langle M_3 \rangle) \in \mathcal{L}_2$.

Given an encoding of a Turing machine TM M_3 , we are going to construct a TM M_2 . That is, $\langle M_2 \rangle = f(\langle M_3 \rangle)$. M_2 operates on an input string w as follows:

If $w \neq 010$, M_2 halts. If $w = 010$, then M_2 simulates M_3 on input w .

Claim: M_2 halts on the input string $w = 010$ iff M_3 halts on all inputs w . That is, $\langle M_3 \rangle \in \mathcal{L}_3$ iff $f(\langle M_3 \rangle) = \langle M_2 \rangle \in \mathcal{L}_2$

Given that the halting problem is undecidable, many other problems can be proved to be undecidable using reductions. Most of these problems do not mention Turing machines but undecidability comes from the problem “being able to encode the computation of a Turing machine M ”.

Expanding on slide 9 in the Week 5 slides

A halting computation is a composition of Turing machine *configurations* C_1, C_2, \dots, C_t such that C_{i+1} is the configuration of the Turing machine that follows from executing one step of the Turing machine when it is in configuration C_i and C_t is in a halting state.

In the transformation we described, we used the fact that a Turing machine can simulate the computation of another Turing machine. This is what Turing called a *universal Turing machine* (UTM). In modern terms, a UTM is an interpreter.

A universal Turing machine (UTM) \mathcal{U} is a T.M. such that

$$\mathcal{U}(\langle \mathcal{M} \rangle, w) = \mathcal{M}(w)$$

That is, \mathcal{U} simulate exactly what \mathcal{M} does on input w . Turing showed how to design a UTM.

The Entscheidungsproblem

In his seminal paper “On Computable Numbers With an Application to the Entscheidungsproblem (i.e. decision problem), Turing uses his model and the undecidability of the halting problem, to prove the undecidability of the “Entscheidungsproblem” posed by Hilbert in 1928. (Church provided an independent proof within his formalism.)

Sometimes this is informally stated as “can mathematics be decided ?”

The Entscheidungsproblem question refers to the decidability of predicate logic which Church and Turing independently resolved in 1936-1937. It would take a little while to formally define this “Entscheidungsproblem” but here is an example of the kind of question that one wants to answer:

Given a formula such as $\forall x \exists y : y < x$

can we determine if such a formula is always true no matter what what ordered domain x, y and $<$ refer to?

For example, $x < y$ and $y < z$ implies $x \neq z$ is always true.

But $x < y$ implies $\exists z : x < z < y$ is not true of all ordered domains (e.g., consider the integers) but is true of the rationals.

Formalization of a Turing machine

- Formally, a Turing machine algorithm is described by the following function $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
 Q is a *finite* set of *states*. Γ is a finite set of symbols (e.g., $\Gamma = \{\#, 0, 1, a, b, \dots\}$ and perhaps $\Sigma = \{0, 1\}$)
- Note: Each δ function is the definition of a single Turing machine; that is, each δ function is the statement of an algorithm.
- We can assume there is a halting state q_{halt} such that the machine halts if it enters state q_{halt} . There is also an initial state q_0 .
- We view a Turing machine P as computing a function $f_P : \Sigma^* \rightarrow \Sigma^*$ where $\Sigma \subseteq \Gamma$ where $y = f(x)$ is the string that remains if (and when) the machine halts. There can be other conventions as to interpreting the resulting output y .
- Note that the model is precisely defined as is the concept of a computation step. A *configuration* of a TM is specified by the contents of the tape, the state, and the position of the tape head. A computation of a TM is a sequence of configurations, starting with an initial configuration.
- For decision problems, we can have YES and NO halting states.

A more general Turing machine model

The Turing machine model has been extended to allow separate read (for the input) and write (for the output when computing a function) tapes and any finite number of work tapes. Here is a figure of a multi-tape TM (but without separate input and output tapes).

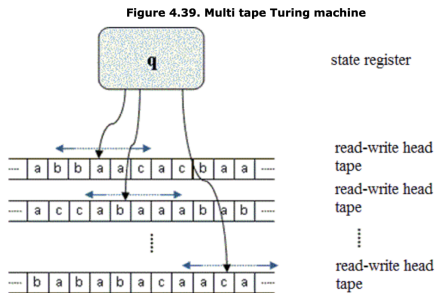


Figure: Figure from the Bela Gyires Informatics Curriculum Repository

Diagonalization

You may have learned in high school why using the diagonalization method it can be proved that the set of real numbers say in $(0, 1)$ is *uncountable* while the set of rationals in $(0, 1)$ is *countable*.

The idea is that since the rationals are countable we can list them and let's say that $r_i =$ the i^{th} rational number in $(0, 1)$ (in some agreed upon listing of these rationals). Suppose as a binary fraction $r_i = .r_i(1)r_i(2)r_i(3) \dots r_i(m_i)$ for some m_i and $r_i(j) = 0$ for all $j > m_i$.

We can create a non-rational number $x = .s_1 s_2 s_3 \dots$ where $s_i = 1 - r_i(i)$.

Diagonalization and the halting problem

We will just sketch why the halting problem is undecidable.

Using diagonalization, we can show that the following halting problem is undecidable. Namely, the set of Turing machines is a countable set and let \mathcal{M}_i denote the i^{th} TM. Consider the following function
 $f(i) = \text{YES}$ if $\mathcal{M}_i(i) = \text{NO}$ or $\mathcal{M}_i(i)$ does not halt, and $f(i) = \text{NO}$ otherwise.

If the halting problem were decidable, then using a UTM and the claimed decidability of the halting problem, f would be a computable function but that would be a contradiction since f is defined to be different than every TM \mathcal{M}_j .

What Turing showed is that we can encode whether or not a Turing machine M accepts input w (i.e. the halting problem) by a statement in predicate logic.

New topic: search engines

I think of search engines as a great idea in the sense of being a "killer application" and also leading to interesting computational issues that have energized the field of computing.

In doing so I am mostly talking about search engines as they are mainly used today in terms of searching web documents. That is, search engines that take queries (usually in the form of key words or phrases) and produce a *ranked list* of documents.

I am mostly going to talk about search engines independent of the importance (and necessity) of having large pools of fast machines, high speed communication and massive storage.

Search engines intro continued

That is, I am mostly going to talk about search engines in terms of their functionality and the basic computational ideas that make them work (so) well. This is another example (like deep neural nets) of a great idea where greatness depended on new technology. Today's quality search engines simply could not exist say using the technology of the 1960s and 70s.

It is also an example where its greatness may also be an inhibitor for thinking about how to “significantly” move beyond the *current norm of key word based search*.

Of course, in some sense we have moved beyond just key word search in that one can now input an image and ask the search engine to find examples like that image. And in addition to ML being used for image recognition, ML is now playing a more algorithmic role in the quality the key word search.

But still from a functional point of view, while the quality of search has greatly improved, we are still basically doing what we did since say the late 1990s.

A little search engine history

Search engines are part of the topic of "information retrieval" once the domain of library science. Computerized information retrieval has been an application idea since the start of modern computing.

On the web page there is a link to a prophetic July, 1945 Atlantic article "As We May Think" by Vannevar Bush where he envisions something quite close in many respects to the modern web and hyperlinked documents.

The article begins with the following: "Consider a future device . . . in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory."

That is, some kind of semi-automated information retrieval has been thought about for over 75 years.

Some quotes from the Vannevar Bush article

There are a lot of anachronisms (in terms of what the underlying technology will be, gender roles) in this article but more important there are many insightful ideas about the future of accessing information. Here are some quotes from that article.

“Much needs to occur, however, between the collection of data and observations, the extraction of parallel material from the existing record, and the final insertion of new material into the general body of the common record. For mature thought there is no mechanical substitute. But creative thought and essentially repetitive thought are very different things. For the latter there are, and may be, powerful mechanical aids.”

Note: Bush is then clearly drawing his line here between human intelligence (and say creating new knowledge) vs retrieving existing knowledge.

More quotes from Bush's article in the Atlantic

“Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. ... The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain.”

“Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it. In minor ways he may even improve; e.g., for his records have relative permanency. The first idea, however, to be drawn from the analogy concerns selection. **Selection by association, rather than indexing**, may yet be mechanized.”

“Wholly new forms of encyclopedias will appear, ready made with a mesh of associative trails running through them, ready to be dropped into the **memex** and there amplified.” [Think now of hyperlinks.](#)

“Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and, to coin one at random, **memex** will do.”

The debate as to the nature of information retrieval

In the 1960's and 70', there was a "debate" (albeit not widely discussed outside of those interested in information retrieval) between those who felt that information retrieval (IR) (i.e. finding documents to satisfy an "information need") was a subfield of AI (and more specifically natural language understanding) verses those who thought it could be best realized by more well established combinatorial, algebraic and statistical ideas. **Bush seems to have already settled his views well before this debate is taking place**

That is, one constituency felt that we needed to be able to "understand" what a document was saying (and what people were requesting) so as to find relevant documents.

The other constituency felt that the claims of many AI researchers were not at all feasible and that again a more statistical/algebraic/combinatorial approach (devoid of any real "intelligence") would produce better results.

The debate continued

I had a course (1967) in IR from Gerald Salton, who (according to Wikipedia) was "perhaps the leading computer scientist working in the field of information retrieval during his time". His group at Cornell developed the SMART Information Retrieval System".

I am not a great historian but I believe the vector space model (which we will discuss) was his idea. Salton was a proponent of the statistical/algebraic/combinatorial approach. I think that he always felt that AI was over-hyped.

So who the debate?

The debate continued

I had a course (1967) in IR from Gerald Salton, who (according to Wikipedia) was "perhaps the leading computer scientist working in the field of information retrieval during his time". His group at Cornell developed the SMART Information Retrieval System".

I am not a great historian but I believe the vector space model (which we will discuss) was his idea. Salton was a proponent of the statistical/algebraic/combinatorial approach. I think that he always felt that AI was over-hyped.

So who the debate?

As of today, it is clear that the approach of the constituency represented by Salton has turned out to be the basis for the way we currently do search in the internet.

The debate continued

I had a course (1967) in IR from Gerald Salton, who (according to Wikipedia) was "perhaps the leading computer scientist working in the field of information retrieval during his time". His group at Cornell developed the SMART Information Retrieval System".

I am not a great historian but I believe the vector space model (which we will discuss) was his idea. Salton was a proponent of the statistical/algebraic/combinatorial approach. I think that he always felt that AI was over-hyped.

So who the debate?

As of today, it is clear that the approach of the constituency represented by Salton has turned out to be the basis for the way we currently do search in the internet. However, search engines today do incorporate ML into their retrieval algorithms.

Key word search

At a very very general level, we can think of current search as the following process:

- 1 A user converts an “information need” into a query (i.e. a set of key words)
- 2 The search engine is then an algorithm for the mapping:
 $\text{query} \times \{\text{collection of documents}\} \rightarrow \langle \text{ranked list of “relevant documents”} \rangle$.
- 3 Upon receiving highly ranked documents, the user may choose to refine the query.
- 4 This process continues until the user is either satisfied or gives up.
How often do you have to refine your queries? How often do you abandon a search?

As we discuss the ideas behind key word search in search engines, it should be noted that there are many specific ideas and engine specific details that go into making a search engine successful (in terms of the quality, speed, and coverage) and these ideas and details are kept reasonably confidential.

Why?

Why the secrecy?

There are two main reasons for not disclosing specific search engine ideas and details:

- Not surprisingly, these ideas are trade secrets that give a company an edge
- Perhaps less obvious, knowing exactly how a company does its searches allows one to easily spam documents so as to raise their ranking (and hence lower the quality of the ranking).

So please be advised that what I am discussing is just the high level ideas and not the specifics say being utilized by Google, Yahoo or Microsoft.

It clearly took significant progress in technology (i.e., the speed and memory capabilities of large numbers of distributed machines) to make key word search as successful as it is today. Equally important, many significant algorithmic ideas plus extensive and ongoing experience with user requests has been necessary for search engine success.

However, collecting information from user interactions is, of course, an important privacy issue.

The challenge of real time information retrieval

In addition to algorithmic ideas used to improve search quality (i.e., precision, recall), commercial search engines are dealing with enormous collections of sites/documents and must return responses in what appears to be "real time".

Estimates of the size of the web vary. One site (WorldWideWebSize.com) provides daily reports on the size of the web: That site reported "The Indexed Web contains at least 5.42 billion pages (Sunday, 04 October, 2020)" but as of October 14, 2021, it reports "The Indexed Web contains at least 4.81 billion pages". **Did the size really decline?**

Precision in a set of documents (for an information need) is defined as the fraction of documents that are relevant. In a ranked list we can say that precision means that the higher the rank of the document, the more likely it is to be relevant.

Recall in a set of documents is defined as the fraction of all relevant documents in the set. In a ranked list of retrieved documents (where there can be many thousands of relevant documents), we want the most relevant documents to occur earliest in the ranked list.

Do we want diversity in the documents retrieved?

We may (or may not) want the highest ranked documents to reflect some desired diversity.

For example, what if I provide the query “What did Donald Trump accomplish as US president”? Do I want just what is reported as his positive accomplishments? Or do I want just the negative aspects of his presidency? Or do I want a diversity of opinions? Do we want denials of the Holocaust to be presented in the name of diversity and “balance” as some in the Texas legislature demand? What is a “legitimate” opinion vs a conspiracy theory devoid of facts?

Similarly, if I ask whether the stock market has recently been rising? Do I want some overall assessment, or do I want reports on different sectors of the market?

Even for a more classical and now perhaps a more mundane example, when I ask for recent information about “jaguars”, do I mean the car, the animal or the NFL football team? I probably only want one of these. When I make my request clear, a search engine should avoid ambiguous meanings.

Should a search engine use my previous history of requests to better

The basic bag of words model

Suppose $\mathcal{C} = \{D_i\}$ is a collection of web documents (URLs).

- We can treat each document as a *bag of words*. Let's just say 200 words per document as some very rough average.
- Each query can also be considered as a very small bag of words. Most queries are two or three words. One estimate is an average of 2.2 words per query.
- The most naive approach. Find all the documents that contain all the words in the query. As a naive first approach call these the "relevant documents".
- The most naive way to find all these (potentially) relevant documents would look at each document and check if all the query words occur.
- Even if all the documents were stored locally (which is not possible), **what would be a rough estimate for the time to find all the relevant documents?**

A quick calculation

You can do a quick calculation: compute $|\mathcal{C}| \cdot \frac{\text{number words}}{\text{document}} \cdot \frac{\text{number words}}{\text{query}}$ and then divide by $\frac{\text{number comparisons}}{\text{second}}$ to estimate the time for naively looking for documents that contain all the query terms.

Let's say that we have approximately 5 billion URLs, 200 words per document, 2 words per query which naively would result in $2 \cdot (10)^{12}$ comparisons. And let's say $(10)^7$ comparisons per second. Then a query would take $2 \cdot (10)^5 = 200,000$ seconds.

OK I might have some miscalculations but clearly this is *not* "real time" and not even feasible.

Making search feasible

One simple idea but but very useful idea is the following. When a search engine *crawls* the web to find documents, it indexes documents so that for each *term* (i.e., word and frequent 2 word and 3 word phrases it maintains a sorted list of documents that contain that term. We usually ignore common articles such as “the”, “an, etc.

A term may also represent a number of strongly related terms. For example, a match for “cook” might be satisfied by “cooking”.

Making search feasible

One simple idea but but very useful idea is the following. When a search engine *crawls* the web to find documents, it indexes documents so that for each *term* (i.e., word and frequent 2 word and 3 word phrases it maintains a sorted list of documents that contain that term. We usually ignore common articles such as “the”, “an, etc.

A term may also represent a number of strongly related terms. For example, a match for “cook” might be satisfied by “cooking”.

You can get spelling suggestions, or maybe get a partial match, and sometimes be told that no documents match your query or there are no good matches but still get some suggested matches.

What can happen often is that there be too many documents matching the query terms. So as we already suggested we really need a ranked list of documents in which the “most relevant” documents are ranked highest.

The vector space model and ranking documents

Instead of simply matching for query terms, we want to account for the fact that the occurrence of certain terms are more important for relevance.

Gerald Salton's idea was that a document (and a query) are represented by a vector of weighted counts of words/terms. Here are some ways to weight the occurrences of terms in a document.

- 1 Count the number of occurrences of a query term in a document, and better yet normalize this count by the relative frequency of terms in "the corpus of documents". This normalized count is called *tf-idf* standing for term frequency-inverse document frequency. Terms that occur infrequently throughout the corpus but appear frequently in a document should be weighted more. Wikipedia quotes a 2015 study that states "83 % of text based recommender systems in digital libraries use tf-idf".
- 2 Terms that appear in the title of the document or the title of a section heading should be given higher weights.
- 3 Terms that appear in the *anchor text* are important.

The vector space model continued

The above ideas for weighting terms are independent of the user queries. In contrast, we could also give higher weights to terms that relate to an individual's interests (say as learned by previous searches).

There can be many other ways to weight terms say by using machine learning techniques.

Now once we adopt this vector space representation, we can measure the similarity of a document and a query by say the cosine of these vectors.

An additional idea (in addition to the term similarity of the document and the query) is to exploit the “popularity” of a document. Popularity of a document in Google was done using *page rank* which is basically a random walk on the graph defined by the hyperlinks. This leads to a stationary distribution (i.e., an equilibrium) on the vertices (i.e., the relevant documents).

Some further comments on the history of search engines

Page rank was touted as an essential idea in the early days of Google search but not clear how much of a role it now plays.

At about the same time as page rank, Jon Kleinberg introduced another graph based popularity method called *hubs and authorities* which was used in IBM's search engine (which they never commercialized).

With regard to *td-idf* (now accepted as an important idea), I saw the following comment in a web post (Language Log)
<https://languagelog.idc.upenn.edu/nll/?p=27770>

“one of Marvin Minsky's students once told me that Minsky warned him ‘If you're counting higher than one, you're doing it wrong’. Still, Salton's students (like Mike Lesk and Donna Harman) kept the flame alive.”

Marvin Minsky is recognized as one of the pioneers of artificial intelligence.

Why is search so profitable?

Companies such as IBM and (initially) Microsoft did not try to commercialize search, not recognizing the profitability of search. Indeed, should one charge for information or should the business model be based on advertising? Or it possible that search would not be profitable?

We now know that search has turned out to be extremely profitable for companies based on advertising. The main way that Google and other comapnies sell advertising for search has spawned major research in algorithm design and auction theory. We will say more about autions, game theory and mechanism design.

We can view the process of assigning queries to advertisers (say wanting to display an *ad* as an *online bipartite graph matching problem*).

When a query arrives it needs to be assigned to one (or more, depending on how many advertising slots will be displayed) ads.

The “adwords” assignment problem

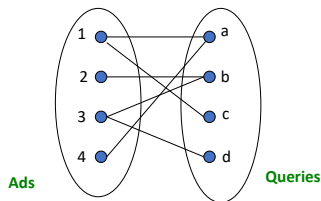


Figure: Figure taken from USC lecture notes by Rafael Ferreira da Silva

Each advertiser may have a budget (say for a given day) and indicates for given queries (or keywords) what it is willing to pay for that query but never exceeding its budget for all the queries assigned to that advertiser.

The search engine adjusts this advertiser *bid* for a query based on how well it thinks the ad matches the query and then decides whether or not to assign an advertising slot to an advertiser and the price paid by the advertiser (depending on the slot) for each click by search users for the ad.

The semantic web

We will end our discussion of search engines about where we began when I said, like other great ideas, sometimes these great ideas become so entrenched that it is hard to make further progress.

Is this the case with key word search? What kinds of “information needs” are beyond today’s search engines? See 2008 “Ontologies and the Semantic Web” article by Ian Horrocks and also his 2005 Lecture by the same title.

The vague goal of the semantic web is “to allow the vast range of web-accessible information and services to be more effectively exploited by both humans and automated tools.”

A more specific goal is to *integrate* information that occurs in the web but not in one document.

Some specific examples of information that might not exist in any one document

One example Horrocks gave is to retrieve a “list of all the heads of state of EU countries”. Of course, once such an example is given, it is likely (as in this example) that one can successfully find the required information in a single query. (Why was this a difficult search in 2008 and an easy search today? It was the fourth document in my search on October 17, 2021.

“The classic example of a semantic web application is an automated travel agent that, given various constraints and preferences, would offer the user suitable travel or vacation suggestions”. This example still seems beyond something we can easily do with current search engines.

I decided to create the following query “list of all computer scientists whose last name is Cook”. In my first search, most of the retrieved documents are not useful but the first of the retrieved documents is for Stephen Cook and the second document is a very incomplete list of computer scientists.

Screenshot of my query for computer scientists with last name Cook

The screenshot shows a Google search results page for the query "list of computer scientists whose last name is Cook". The browser is Chrome, and the search results are displayed in a list format. The first result is from Encyclopedia.com, followed by a result from TheBestSchools.org, a result from Future Students | York University, a result from lamturing.acm.org, and a result from books.google.ca. A "People also search for" box is visible, listing related search terms like "leonid levin", "gordon cook", and "stephen cook obituary".

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Inbox (15,346) - abborndi x New Tab x G list of computer scientist: x G list of all the heads of sta: x +

google.com/search?q=list+of+computer+scientists+whose+last+name+is+Cook&rlz=1C8CHFA_enCA904CA... ☆ Update

Apps Getting Started Latest Headlines Imported From Fir... CSC200_Lecture4... Application List, D... Reading List

Google list of computer scientists whose last name is Cook X

https://www.encyclopedia.com/science/stephen-arth... **Stephen Arthur Cook | Encyclopedia.com**
He earned his M.S. from Harvard in 1962, and his Ph.D. in 1966. He then took a position as assistant professor of mathematics and computer science at the ...
Missing: name | Must include: name

https://thebestschools.org/magazine/most-influential... **The Most Influential Computer Scientists - TheBestSchools.org**
Sep. 7, 2021 — Who are the scientists shaping and framing the computer-driven world ... to put names and faces to the esoteric acronyms and the machinery.

People also search for

leonid levin	famous computer scientists and their inventions
gordon cook	10 computer inventors and their inventions
stephen cook obituary	20 computer inventors and their inventions

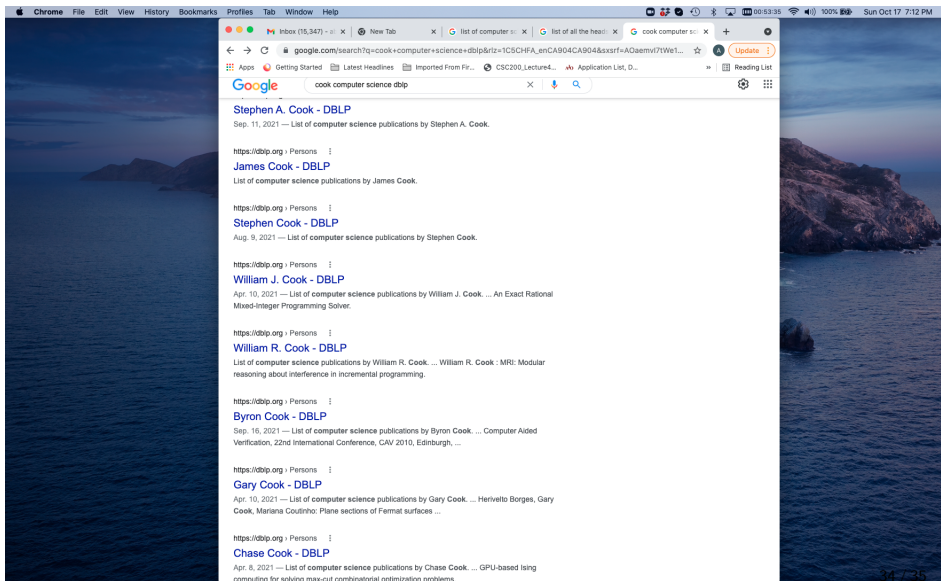
https://futurestudents.yorku.ca/program/computer-s... **Computer Science | Future Students | York University**
This program is intensive in Mathematics and Computer Science courses. ... am a high-school student I have completed at least one year of full-time study at ...

https://amturing.acm.org/cook_n991950 **Stephen A Cook - A.M. Turing Award Laureate**
Cook entered the University of Michigan in 1957, majoring in science engineering. He was introduced to computer programming in a freshman course taught by ...
Missing: name | Must include: name

https://books.google.ca/books **Coding as a Playground: Programming and Computational ...**
Marina Umaschi Bers · 2020 · Education
Basic research must inform the debate about the role of computer science in the ... and looking up a name in an alphabetical list (linear; starting at the ...

https://suonline.asu.edu/.../Online graduate programs **Online master of computer science (MCS)**

Another search to find other computer scientists with last name Cook



The screenshot shows a Google search results page for the query "cook computer science dblp". The search results list several computer scientists with the last name Cook, each with a link to their DBLP profile and a brief description of their work.

Search results for "cook computer science dblp":

- Stephen A. Cook - DBLP**
Sep. 11, 2021 — List of computer science publications by Stephen A. Cook.
- James Cook - DBLP**
List of computer science publications by James Cook.
- Stephen Cook - DBLP**
Aug. 9, 2021 — List of computer science publications by Stephen Cook.
- William J. Cook - DBLP**
Apr. 10, 2021 — List of computer science publications by William J. Cook. ... An Exact Rational Mixed-Integer Programming Solver.
- William R. Cook - DBLP**
List of computer science publications by William R. Cook. ... William R. Cook : MRI: Modular reasoning about interference in incremental programming.
- Byron Cook - DBLP**
Sep. 16, 2021 — List of computer science publications by Byron Cook. ... Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, ...
- Gary Cook - DBLP**
Apr. 10, 2021 — List of computer science publications by Gary Cook. ... Herivelto Borges, Gary Cook, Mariana Coutinho: Plane sections of Fermat surfaces ...
- Chase Cook - DBLP**
Apr. 8, 2021 — List of computer science publications by Chase Cook. ... GPU-based Ising computing for solving max-cut combinatorial optimization problems.

October 14, 2022 search to find a computer scientist named Cook not living in Canada.

The screenshot shows a Chrome browser window with the following elements:

- Address Bar:** `google.com/search?q=computer+scientists+with+last+name+cook+but+not+living+in+canada&rlz=1C5CHF_enCA904CA904&sxsr=ALICzsZnGq6P...`
- Search Bar:** `computer scientists with last name cook but not living in canada`
- Search Results:**
 - Showing results for **computer scientist** with last name cook but not living in canada
 - Search instead for **computer scientists with last name cook but not living in canada**
 - Results for **Stephen Cook - Wikipedia**:
 - URL: `https://en.wikipedia.org/wiki/Stephen_Cook`
 - Text: Stephen Arthur Cook, OC, OOnt (born December 14, 1939) is an American-Canadian computer scientist and mathematician who has made major contributions to the ...
 - Results for **Gordon Cook - Wikipedia**:
 - URL: `https://en.wikipedia.org/wiki/Gordon_Cook`
 - Text: Gordon Cook (born December 3, 1978, in Toronto) is a two-time Canadian Olympic sailor. ... He is the son of computer scientist Stephen Cook.
 - Results for **Stephen Cook, 1982 ACM Turing Award Recipient - YouTube**:
 - URL: `https://www.youtube.com/watch`
 - Text: Describes his early life, education, introduction to electronics,
- Taskbar:** Shows various application icons and the system clock at 3:35 PM on 10/14/2022.