Review of[1]

Ideas that Created the Future:
Classic Papers of Computer Science
Edited by Harry Lewis
Publisher: MIT Press
517 pages, Year: 2021, $60.00 paperback, $42.00 Kindle

Reviewer: William Gasarch `gasarch@umd.edu`

1 Disclosure

Harry Lewis, the editor of this book, was my PhD adviser.

2 Introduction

What are the most important papers in computer science? What are the most important 46 papers in computer science? Yikes! Far more than 46 seem to qualify. Picking out the top 46 papers in computer science is the task that befallen Harry Lewis (henceforth Harry). I suspect it was both a burden (gee, which ones to omit?) and a joy (Wow, these papers are insightful!). He used two (probably more) criteria that helped him cut it down (1) he prefers short readable papers to long unreadable ones (don't we all!), and (2) no paper past 1980 (arbitrary but firm cutoff). There were also absurd financial constraints based on how much the rights to republish a paper costs. In some cases a paper that is available online for free cost too much to put into a book. See my comment on Turing's 1936 paper.

This book is a collection of the 46 most important papers in computer science, in the opinion of Harry, relative to some constraints. The complete list is at the end of the review. While I am sure many readers will think *why isn't X in the list*, or *why is X in the list*, or *I never heard of X*, I suspect that 2/3 of the people reading this review will agree with 2/3 of the papers on the list. I would urge people to read the book AND THEN have an intelligent debate about which papers should or should not be in it.

What does this book add that you could not get by just finding the papers online and reading them?

1. Harry carefully picked out which papers are important, short, and readable.

2. Each paper has an introduction (written by Harry) which tells you about the author(s), about the paper, and why the paper is important.

3. The papers are not there in their entirety. Some parts have been edited out which makes them more readable. In essence, he edited out the boring parts.

4. There is something about having a book in front of you, or even a kindle-version, that makes you want to read it, rather than going to the web and reading them one at a time. I used to think this was my inner-Luddite talking, but young people tell me that they also are more inclined to read solid books than ephemeral screens.

[1] ©2021 William Gasarch

5. By reading these papers you can pick up certain threads: some people thought computers could eventually think or do all kinds of miraculous things, others were more modest. Some were interested in scientists using computers, others with business people. Later there was issues about ordinary people using them, an issue that just had not come up earlier. There are other threads to pick up on.

6. One of the proofreaders for the review is dyslexic who uses text-to-speech software. After she gave me (much appreciated) corrections and suggestions I inquired if she thinks she would like the book. She responded: *A book of past papers in it is great for someone like me since older papers are usually not in a format for text-to-speech software. But books published recently are in such a format. I had accepted that I was just never going to be able to read papers like these, but since they are in this book, I can!*

Some of the papers are from theory, so the reader of this review (in SIGACT News) probably knows what's in them, but probably *has not read them.* I was surprised to realize how many classic papers I had not actually read. Its great to see what the original authors thought of their work. Some of the papers were quite prescient. Others were ... less so.

3 Comments on a Few of the Papers

It would be madness to comment on all 46 papers in the space of a review. So I comment on a selection, hoping to hit several types of papers. I won't comment on Harry's comments for every article; however, suffice it to say that he sets up what you are about to read very well.

3.1 Prior Analytics, by Aristotle

Computer Science goes back to Aristotle. Really! Aristotle was concerned with being able to make inferences based solely on the *form* of a set of sentences rather than their *content*, what we would call deductions. The excerpt we get is 3 pages which is plenty. It is surprisingly readable, but wordy given today's understanding. Harry's introduction is particularly helpful here. For example, Harry points out that the paper gives rise to the modern notion of a set.

Boole learned logic out of Aristotle's text. (An excerpt from Boole's book, *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*, is in this book). Boole took Aristotle's ideas further and Shannon applied them to real computers (Shannon paper, *A symbolic analysis of switching circuits* is in this book). The interplay of logic and computers is an example of a thread one can follow from this book that would be hard to follow if one looked for papers on the web.

3.2 The True Method by Gottfried Wilhelm Leibniz

Leibniz might be the first computer scientist (questions of *who was the first X* are always complicated). He build a nested-loop calculator that could multiply and divide. One competitor for the honor is Pascal who built an adding machine. Both feats are impressive.

The paper by Leibniz is quite readable but a bit long for its content (by modern standards). I was struck that he thought computing devices could settle all questions, both in Physics (where experiments are expensive) and Metaphysics (where experiments are impossible). He was an optimist. He was right in that computers can often do simulations and hence solve problems in Physics. He was wrong in that the problems of metaphysics are still unsolved and probably always will be.

This paper can be viewed as a research proposal. The following line in it would likely not be in an NSF grant proposal today:

It is one of my ambitions to finish this project if God grants me the time.

3.3 On Computable Numbers, With an Application to the Entscheidungsproblem, by Alan Turing

This was the most expensive paper to get permission to reprint here, even though its free on line in the following places:

`https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf`

`https://academic.oup.com/plms/article/s2-42/1/230/1491926?login=true`

`http://www.turingarchive.org/browse.php/b/12`

`https://www.wolframscience.com/prizes/tm23/images/Turing.pdf`

As you may have guessed, this is the paper where Turing machines (not called that in the paper) are defined and HALT is shown to be undecidable. Interesting historically since he has to also convince the reader that Turing-computable is computable (which we, with 20-20 hindsight, find obvious) and that computable is Turing-computable (which we call Church's thesis or the Church-Turing thesis).

3.4 A Logical Calculus of the Ideas Immanent in Nervous Activity, by Warren McCulloh and Walter Pitt

The goal of this paper was to model how the brain works. That sounds hard. And indeed it is. This paper both fails and succeeds.

1. The model of the brain they come up with is not accurate for the real brain.
2. They lay the groundwork for (1) finite automata, which Kleene and others picked up on, and (2) Neural Nets, which Rosenblatt picked up on (Rosenblatt's paper *The Perceptron* is in the book).

This paper is also the first one that talks of having computers mimic what people do by trying to do it the way people do it. In later years some people in AI wanted to have computers just do X that people do well, while others wanted computers to do X the way people do it. This difference is an interesting thread to follow.

3.5 As We May Think, by Vannevar Bush

This article appeared in *The Atlantic*, a popular magazine. That goes to show that you can say scientifically important things in popular media if (1) you're smart enough, and (2) the science isn't to advanced.

When I read this article my first reaction was:

Wow! This article, written in 1945, predicts Google, Kindle, Browsing the web, Data Structures, Data Base, and P vs NP.

Upon reading it again I realized that I was projecting the future into the past. But not much. Truly a visionary article.

3.6 Some Moral and Technical Consequences of Automation, by Norbert Wiener

This paper warns of the the dangers of computers. The dangers fall into two categories.

1. Computers can be trained for some scenarios but then another scenario happens and the machines reaction could be dangerous. Wiener was probably thinking of automated weapons systems and accidental nuclear war. Today the same problem might plague self-driving cars.
2. In the story *The Sorcerer's Apprentice*, within the movie *Fantasia*, Mickey Mouse asks a broom to get him some water. The broom keeps doing this and Mickey almost drowns. Computers may take us to literally and that could be dangerous. I wonder if computer-trading on Wall Street may have this problem, or computers doing bidding at auctions.

Has anything like *The Sorcerer's Apprentice* really occurred? As it happens, the second edition of *Blown to Bits* by Hal Abelson, Ken Ledeen, Harry Lewis (Same Harry Lewis!), and Wendy Seltzer, which is reviewed in this column by William Gasarch (Same William Gasarch!) has the following story which I paraphrase: On Amazon a used book *The Making of a Fly*, list price \$70.00, was listed at around \$23,000,000. It does get 4.1 stars (out of 5) so it is probably a very good book. But this was not the reason why it was so expensive. There were two sellers: Bordeebook and Profnath. They had different programs to price their books:

- Bordeebook was programmed to raise the price when others did. In particular, it charged approximately 1.23 times what Profnath was charging.
- Profnath was programmed to undercut its competitors by just a little. In particular, it charged approximately 0.998 times what Bordeebook charged.

I leave it to the reader to work out how this causes prices to go to infinity.

This story is more amusing than dangerous. But it is still a cautionary tale and a proof-of-concept.

3.7 Cramming More Components onto Integrated Circuits, by Gordon Moore

One sign that computer science is a new field is that some of the writers of classic papers are still alive. As of April 2021, when I am writing this review, Gordon Moore is alive at the age of 92. In fact, for the last 20 (or so) papers, most of the authors are still alive.

This article appeared in *Electronics*, a popular electrical engineering magazine. As with Bush's article, this shows that you can say scientifically important things in popular media if (1) you're smart enough, and (2) the science isn't to advanced.

The paper states, almost in passing, that the number of components on a chip will double every two year. He later revised that down to once-a-year and then to once-every-1.5 years. He also predicted it would stop after about 40 years, which was about right. (The article appeared in 1965.)

Moore never stated that speed doubles every 1.5 years; however, I suspect he was happy with this law being named after him. That version of Moore's law held for about 40 years also.

The paper is a good read. Moore tells us why he thinks component density will increase. His arguments are intelligent and he is correct.

3.8 Solution of a Problem in Concurrent Program Control, by Edsger Dijkstra

In this paper Dijkstra gives an elegant proof that a concurrent program is correct.

There are several other papers in the volume on proving programs correct:

1. *The Structure of the "THE"–Mulitprogramming System* by Edsger Dijkstra. This paper introduces semaphores, which are a method for reasoning about atomic actions in a program. They also apply the methods to an actual system.

2. *Go To Considered Harmful* by Edsger Dijkstra. This paper argues why we should get rid of the Goto statement so that it would be easier to reason about programs.

3. *An Axiomatic Basis for Computer Programming* by Tony Hoare. This paper lays out a framework for proving programs correct; however, it also points out obstacles to the endeavor.

4. *Social Processes and Proofs of Theorems and Programs* by Richard DeMillo, Richard Lipton, and Alan Perlis. This paper argues that the field of proving programs correct is doomed to failure. This may have been a self-full-filling prophecy since this paper led to funding being cut.

All of these papers were influential. It is interesting to see them all in the same volume.

3.9 Managing the Development of Large Software Systems, by Winston Royce

Having a team of programmers write a large piece of software is hard! This paper provides, to quote Harry's introduction to it: *useful wisdom that has passed the test of time and remains good advice today.* The paper is also quite readable, particularly since it has lots of nice diagrams that build on each other.

Another excellent paper (actually a book excerpt) on this topic that appears in this book is *The Mythical Man-Month* by Fred Brooks. This paper makes the obvious-in-hindsight point that putting more people on a project that is falling behind will only make it fall further behind.

3.10 The Complexity of Theorem-Proving Procedures, by Stephen Cook

This is the paper that proves SAT is NP-complete. (actually, this paper deals with TAUT). Karp's paper *Reducibility Among Combinatorial Problems* (which is also in this book) showed 21 problems are NP-complete. Cook's paper and Karp's paper together launched modern complexity theory.

3.11 A Statistical Interpretation of Term Specificity and its Application to Retrieval, by Karen Sparck Jones

This is an early paper on search. I quote from the introduction by Harry:

The more often a term occurs in a document, the more relevant it is to the document's content. For example, a paper that uses the term `zebra` *repeatedly is probably at least somewhat about zebras.*

But of course, the same paper will use the term `the` *repeatedly, so mere frequency within a document is an unreliable indicator of a word's significance.*

The introduction then goes on to say that you must compare the frequency of a word in a document with its frequency in *other documents*. This paper shows how to really do this.

3.12 A Protocol for Packet Network Intercommunication, by Vinton Cerf and Robert Kahn

No, Al Gore did not invent the internet; however, there is a good argument that Cerf and Kahn did. In this paper, written in 1974, they designed standards and protocols for what we now call the internet. It was very important that the protocols be standardized to make the world wide web truly world wide.

3.13 New Directions in Cryptography, by Whitfield Diffie and Martin Hellman

This paper started a revolution in cryptography in two ways: (1) they showed that there was a way Alice and Bob can establish a shared secret key without having a secret channel (by making some reasonable hardness assumptions), (2) making cryptography into a rigorous field of study.

Its interesting to read it now to see that *they knew* it would start a revolution. This paper lead to *A Method For Obtaining Digital Signatures and Public-Key Cryptosystems*, by Rivest-Shamir-Adleman (which is also in the book). Both the Diffie-Helman paper and the RSA-paper are quite readable. This is not surprising since they were early and use what are now well known concepts.

4 Who Should Read This Book?

You should read it. You should also give it to your CS-inclined niece or great niece for a graduation present. Note that the price, $60.00 softcover, $42.00 Kindle, for a book that is over 500 pages is, as the kids say, jawsome (jaw dropping awesome).

It is very interesting to read these papers and see the roots of computer science. It's interesting to (1) follow some threads, (2) see where people were correct in their predictions, (3) see where people were wrong in their predictions (Turing thought that ESP was real and might distinguish humans from machines in the paper *Computing Machinery and Intelligence*). And it's interesting to read papers that you thought you knew about and find out they didn't quite say what you thought.

5 The 46 Papers

1. Prior Analytics ($\sim$ 350 BCE) Aristotle
2. The True Method (1677) Gottfried Wilhelm Leibniz
3. Sketch of the Analytical Engine (1843) L. F. Menabrea, with Notes by the Translator, Ada Augusta, Countess of Lovelace
4. An Investigation of the Laws of Thought on Which Are Founded the Mathematical Theories of Logic and Probabilities (1854) George Boole

5. Mathematical Problems (1900) David Hilbert
6. On Computable Numbers, with an Application to the Entscheidungsproblem (1936) Alan Mathison Turing
7. A Proposed Automatic Calculating Machine (1937) Howard Hathaway Aiken
8. A Symbolic Analysis of Relay and Switching Circuits (1938) Claude Shannon
9. A Logical Calculus of the Ideas Immanent in Nervous Activity (1943) Warren McCulloch and Walter Pitts
10. First Draft of a Report on the EDVAC (1945) John von Neumann
11. As We May Think (1945) Vannevar Bush
12. A Mathematical Theory of Communication (1948) Claude Shannon
13. Error Detecting and Error Correcting Codes (1950) R. W. Hamming
14. Computing Machinery and Intelligence (1950) Alan Mathison Turing
15. The Best Way to Design an Automatic Calculating Machine (1951) Maurice Wilkes
16. The Education of a Computer (1952) Grace Murray Hopper
17. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem (1956) Joseph B. Kruskal, Jr.
18. The Perceptron: A Probabilistic Model for Information Storage and Organization (1958) Frank Rosenblatt
19. Some Moral and Technical Consequences of Automation (1960)1 Norbert Wiener
20. Man–Computer Symbiosis (1960) J. C. R. Licklider
21. Recursive Functions of Symbolic Expressions and Their Computation by Machine (1960) John McCarthy
22. Augmenting Human Intellect: A Conceptual Framework (1962) Douglas C. Engelbart
23. An Experimental Time-Sharing System (1962) Fernando Corbato, Marjorie Merwin Daggett, and Robert C. Daley
24. Sketchpad (1963) Ivan E. Sutherland
25. Cramming More Components onto Integrated Circuits (1965) Gordon Moore
26. Solution of a Problem in Concurrent Program Control (1965) Edsger Dijkstra
27. ELIZA—A Computer Program for the Study of Natural Language Communication between Man and Machine (1966) Joseph Weizenbaum

28. The Structure of the “THE”-Multiprogramming System (1968) Edsger Dijkstra
29. Go To Statement Considered Harmful (1968) Edsger Dijkstra
30. Gaussian Elimination is Not Optimal (1969) Volker Strassen
31. An Axiomatic Basis for Computer Programming (1969) C. A. R. Hoare
32. A Relational Model of Large Shared Data Banks (1970) Edgar F. Codd
33. Managing the Development of Large Software Systems (1970) Winston W. Royce
34. The Complexity of Theorem-Proving Procedures (1971) Stephen A. Cook
35. A Statistical Interpretation of Term Specificity and Its Application in Retrieval (1972) Karen Späck Jones
36. Reducibility among Combinatorial Problems (1972) Richard Karp
37. The Unix Time-Sharing System (1974) Dennis Ritchie and Kenneth Thompson
38. A Protocol for Packet Network Intercommunication (1974) Vinton Cerf and Robert Kahn
39. Programming with Abstract Data Types (1974) Barbara Liskov and Stephen Zilles
40. The Mythical Man-Month (1975) Frederick C. Brooks
41. Ethernet: Distributed Packet Switching for Local Computer Networks (1976) Robert Metcalfe and David R. Boggs
42. New Directions in Cryptography (1976) Whitfield Diffie and Martin Hellman
43. Big Omicron and Big Omega and Big Theta (1976) Donald E. Knuth
44. Social Processes and Proofs of Theorems and Programs (1977) Richard DeMillo, Richard Lipton, and Alan Perlis
45. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems (1978) Ronald Rivest, Adi Shamir, and Len Adleman
46. How to Share a Secret (1979) Adi Shamir