Benjamin Bremer

Jeremy Brown

10/03/2025

CSCI-472 - Historical and Current Computer Science

Edsger Wybe Dijkstra

I. Introduction

My first job interview came about halfway through my second year of college. Despite upholding a cordial facade, I was sweating bullets. Would my preliminary skills hold up to the scrutiny of someone with decades of experience? Then came the question: write a Python function to print a given amount of Fibonacci Numbers. My body relaxed. What a simple task!

```python
def print_fib(n):
	prev_fibs = [0, 1]
	for i in range(2, n + 2):
		prev_fibs.append(prev_fibs[i - 2] + prev_fibs[i - 1])
	print(prev_fibs)
```

Many, if not all of us, would take a function like this for granted, just as I did in my interview. It's just a for loop, right? Wrong.

Prior to 1968, writing a function to do exactly what print_fib() does would require a mess of Go To statements that would be horribly difficult to understand and debug, even for such a simple problem. In contrast, the function above could be understood by any CS101 student today. One of the major defining rules of modern programming that makes this function so easy to understand is known as Structured Programming and was the contribution of Edsger Dijkstra, a Dutch mathematician, and one of the first so-called "computer scientists" (Encyclopædia Britannica, Inc., 2025).

While arguably the most important contribution Dijkstra made to modern computing, Structured Programming wasn't his only idea. In 1956, 3 years before even attaining his PhD, Dijkstra solved the shortest path problem. The solution, aptly named "Dijkstra's Algorithm", solves for the path between any two nodes in a graph such that the combined weight of all edges in the path is minimized (DSA Dijkstra's Algorithm, n.d.). This algorithm would also be familiar to any CS101 student today.

II. Formative Years

Born Edsger Wybe Dijkstra on May 11, 1930 in Rotterdam, Netherlands, the first few decades of his life were very similar to many of the early influential computing minds. Dijkstra's father was a well-regarded chemist who recommended that he study mathematics and theoretical physics at the University of Leiden in Leiden, Netherlands. Dijkstra then began a career at his alma mater, and through his supervisor met the director of the Computation Department of the Mathematical Centre in Amsterdam in March of 1952, where he began a new job, effectively becoming the first "programmer" in the Netherlands. While this job title is recognized today, it was not back then. When filing for his marriage license with his wife in 1957, he tried to tell the government that he was a programmer, to which they responded that the profession didn't exist in The Netherlands (Dijkstra, 1972).

In 1959, Dijkstra received his PhD from the University of Amsterdam for a thesis in which he thoroughly described the assembly language that had been developed for The Netherlands' first commercial computer (Dijkstra, 1959). He continued working at the Mathematical Centre in Amsterdam until 1962, when he moved first to Eindhoven, Netherlands, and then to Nuenen, Netherlands, to become a professor within the Mathematics Department at

the Eindhoven University of Technology (Technische Hogeschool Eindhoven, or THE in Dutch) (Apt, 2002). Dijkstra never felt like he quite fit in at THE, as they didn't have a separate computer science department. He tried his best to build a collaborative group of computer scientists, but this method was an unusual approach for the department. His most notable accomplishment at THE was the development of the THE operating system, aptly named after the school (Silverschatz et al., 1988).

Thus, Dijkstra's early years were, in essence, not that much different from many of the early great computing minds. He came from a family well-off enough to put him through college at a time when only about 3% of 18-year-olds in The Netherlands enrolled as first-year university students (Canton and Jong, 2005). In addition, being raised by his highly academic father likely ingrained in Dijkstra the skills needed to succeed in a STEM undergraduate degree and, subsequently, in greater academia. Furthermore, Dijkstra exhibits more similarity to the fathers of computing in that he largely fell into the discipline by accident before it even really existed. Before finally becoming a member of the Burroughs Corporation as a research fellow in August 1973, and later accepting a position as Centennial Chair of the Department of Computer Science at the University of Texas in Austin in 1984, Dijkstra was, by title, just a mathematician (Encyclopædia Britannica, Inc., 2025). It is worth noting, however, that all of his most significant contributions to modern computer science came before he ever held a formal position in the field. Like so many of the other visionaries that came before him, Dijkstra was ahead of his time–a deep critical thinker that had no qualms about exploring the unexplored head first.

III. Character

Despite the similarities his early life shared with other early computer scientists, Dijkstra was anything but average. And, given that the average computer scientist is far from normal, it's safe to say that Dijkstra was a bit odd. His teaching style was rather unorthodox and old-school. For example, he preferred writing on a blackboard as opposed to using overhead projectors (Misra et al., 2003). He would assign challenging homework assignments, and expected students to think critically and contribute ideas to the class (Misra et al., 2003). The final exams would be conducted orally, with each exam typically lasting for several hours (Misra et al., 2003). His idiosyncratic approach to teaching did not indicate a lack of care for his students; however, it indicated the exact opposite. Dijkstra viewed teaching as a serious endeavor, not just a required byproduct of being a professor (Misra et al., 2003).

Lecturing wasn't the only thing Dijkstra did differently from his colleagues. It was only at the pestering encouragement of his UT Austin colleagues that he bought a Mac, which he used exclusively for email and browsing the internet. He didn't use word processors, since he believed that one should be able to write a letter or, yes, even an academic article, with no rough drafts, rewriting, or significant editing. He preferred instead to use either his typewriter or fountain pen. Dijkstra's unique ability to work with his pen and typewriter demonstrates his critical thinking ability. He would often know exactly what he wanted to write before he even started, and even mentioned that while he was a physics student, he would often solve homework problems in his head while walking and simply write them down later (Misra et al., 2003).

Dijkstra's disdain for technology bled into his personal life. He never owned a television, video player, or mobile telephone, and never went to the movies. He would rather spend his time playing the piano, going to concerts, and listening to classical music (Apt, 2002).

The rest of Dijkstra's life could be described as odd by a modern American, but would seem very normal to any Dutch person. His home was modest and unassuming. His work style was characterized by elegance and efficiency. He authored more than 1300 papers over the course of his career, many of which had no co-authors and were written by hand. His devoted work ethic was combined with an intense imagination and quick wit. Among his papers were parables, fairy tales, and warnings. In a series of papers, labeled "EWDs" after his initials, Dijkstra described himself as the chairman of a fictitious company that monetized mathematical proofs in a similar way to how modern firms monetize software (Dijkstra, 1975).

IV. Contributions to Computer Science

A. Structured Programming

While many who consider themselves "programmers" may not be able to describe exactly what structured programming is, it is a paradigm fundamental to all modern coding languages. Structured programming largely concerns itself with the if-then-else, switch, and iterative control flow elements. Prior programming methods utilized Go To calls in code, in which the programmer would tell the program which line of code to execute next (Dijkstra, March 1968). The first instances of Structured Programming appeared in the ALGOL 58 and ALGOL 60 languages, the second of which Dijkstra used during his academic career (Kruseman, 2003). The paradigm was popularized largely by the publication of Dijkstra's letter titled "Go To Statement Considered Harmful" in 1968 (Dijkstra, March 1968).

The theoretical foundation for Structured Programming, the Structured Program Theorem, states that sequencing, selection, and iteration are sufficient to express any computable function (Böhm and Jacopini, 1966). This theorem accurately describes the operation of a CPU's

instruction cycle and a Turing Machine, even if the code itself is not part of a program adhering to such guidelines. This discovery is accredited to other academics who published a 1966 paper, which Dijkstra later cited in his own paper (Dijkstra, March 1968). While the theorem that gave the foundation for Structured Programming was not attributed to Dijkstra, he was one of the most influential characters in defining how to write structured programs.

Prior to the idea of Structured Programming, which utilized sequentially executed programs with a single entry and exit point containing if-elif-else statements, for loops, and while loops, the Go To statement was utilized to perform jumps to various pieces of code in the program. For example, a “for loop” might look like the following:

```
None
loop_ctr = 0
if loop_ctr >= 10:
       Go To(8)
print(loop_ctr)
loop_ctr += 1
Go To(2)

print("Done with loop")
```

As you can probably imagine, as programs grew increasingly complex, it grew increasingly harder to write, understand, debug, and maintain programs in this manner. Dijkstra cites this very reason in his *Notes on Structured Programming (EWD249)* (Dijkstra, 1969). Under the Structured Programming paradigm, the 7-line for loop above could be written in just 2, with no need for a line of code telling the program to jump to another line of code. Thus, Dijkstra pioneered the idea that programs could be written without ever needing a Go To statement, leading to more understandable and maintainable code (Structured Programming Details, n.d.).

In 1972, Dijkstra was bestowed with the Turing Award for his contributions to Structured Programming. His ideas helped to form computer science into a rigorous academic discipline, creating standards that modern programming languages still follow today (ACM, n.d.). Additionally, the Structured Programming framework directly addressed the so-called "software crisis" by introducing the idea that care should be taken to make sure code is readable, understandable, and maintainable (ACM, n.d.). The influence Dijkstra's work on Structured Programming had on the computer science field is still being felt today, and will be felt for many decades to come.

B. Solution to the Shortest Path Problem - Dijkstra's Algorithm

Graph theory is a fundamental building block of computer science. Many problems can be represented with graphs, from geospatial problems to networking problems. When dealing with a graph problem, one solution of interest is how to traverse between any two nodes in the graph in the shortest possible distance. Could you imagine a world in which Google Maps could only find the shortest route to your destination by brute force? Before any of these modern applications of graph theory existed, Dijkstra proposed his solution to the shortest path problem.

For being such a fundamental concept in computer science, Dijkstra's Algorithm had fairly humble beginnings. Dijkstra himself said in an interview that he imagined the problem as if he was traveling from city to city (Frana and Misa, 2010). Dijkstra claims that he and his fiancee got tired while shopping in Amsterdam, and, when he sat down, he imagined the algorithm in his head in only about 20 minutes (Frana and Misa, 2010). Dijkstra claimed that, since he developed the algorithm with no pen and paper, he was "almost forced to avoid all avoidable complexities" (Frana and Misa, 2010). Even though Dijkstra thought of his solution in

1956, it was not published until 1959, maybe due to the lack of dedicated computer science journals at that time that may be interested in publishing such a discovery, or maybe because something influential enough to put in a paper surely couldn't be thought up in just 20 minutes. Alas, Dijkstra's solution was certainly the perfect testament to his deep mind. It is also worth noting that, when Dijkstra thought of the solution, he hadn't even earned his PhD yet.

As Dijkstra was working at the Mathematical Centre in Amsterdam when the algorithm was born, he immediately applied it to a graph of 64 Dutch cities in order to demonstrate the new ARMAC computer's capabilities (Frana and Misa, 2010). The original algorithm was found to run in $O(V^2)$ time, and was later improved by others to run in $O(E + Vlog(V))$, which is the fastest known asymptotic time complexity solution to the shortest path problem (Schrijver, 2012). The algorithm is also often considered to be the most straightforward algorithm for solving the shortest path problem (DSA Dijkstra's Algorithm, n.d.).

Today, Dijkstra's Algorithm has applications in navigation, network routing, transportation, robotics, video games, communication, social networking, DNA sequencing, resource allocation, and more (Bhattacharyya, 2023). Despite being developed in his head while drinking coffee on a street in Amsterdam, Dijkstra's Algorithm laid the base for his fame.

C. Other Notable Achievements

As if contributing two foundational ideas of modern-day computer science wasn't enough, Dijkstra made some other notable contributions to the computer science field over the course of his career. The first of which was the THE operating system. The THE OS introduced the first form of paged virtual memory (Dijkstra, 1968). It achieved this using an ALGOL compiler, which was one of the first representations of Structured Programming (before anyone

even knew what Structured Programming really was) (Dijkstra, 1968). Additionally, semaphores were utilized for the first time in the THE OS to help support multitasking - a programming construct that is still used today, and has been further morphed into the also-common mutex (Dijkstra, undated, 1962 or 1963). While these achievements led into and utilized Dijkstra's Algorithm and Structured Programming, they are still notable achievements in their own right.

V. Conclusion

Edsger Dijkstra's accomplishments should feel familiar to nearly any computer scientist today. What many don't take the time to think about is the man behind the inventions. Dijkstra was a deep thinker: able to solve the shortest path problem in his head in 20 minutes over a cup of coffee. He was a simple man: never owning many things considered to be modern luxuries at the time, even attributing much of his ability to reason to being free from avoidable complexity. He wasn't afraid to try new things: he was The Netherlands' first programmer before the profession even technically existed. He gathered a group of computer scientists at THE to collaborate on research - a method that deviated from the university's norm. Despite feeling like he didn't fit in at THE, it was there that Dijkstra developed the THE operating system, one of the first steps that led to his major breakthrough: Structured Programming.

So, let us not just remember the concepts of Structural Programming or Dijkstra's Algorithm. Let us remember who Edsger Dijkstra was. After he returned to his home in Nuenen, Netherlands and succumbed to an arduous battle with cancer on August 6, 2002, the University of Texas at Austin, where Dijkstra ended up serving for 15 years, released an obituary praising him as a

> “prodigious writer…notorious for his wit, eloquence, and way with words, such as in his remark ‘The question of whether computers can think is like the question of whether submarines can swim’, his advice to a promising researcher…’Do only what you can do”, and his remark in his Turing Award lecture ‘In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.’” (UT Computer Sciences Department’s Obituary - Edsger Wybe Dijkstra, 2012).

Dijkstra was not a square peg forcing himself into a round hole; instead, every aspect of who he was, indeed, even his skills as a pianist and his love for exploring national parks in his Volkswagen bus called the “Touring Machine” prepared him to make the groundbreaking discoveries he is still remembered for today (UT Computer Sciences Department’s Obituary - Edsger Wybe Dijkstra, 2012).

References

ACM. (n.d.). Edsger Wybe Dijkstra. Edsger W. Dijkstra – A.M. Turing Award Laureate. https://amturing.acm.org/award_winners/dijkstra_1053701.cfm

Apt, K. R. (2002). Edsger Wybe Dijkstra (1930–2002): A portrait of a genius. Formal Aspects of Computing, 14(2), 92–98. https://doi.org/10.1007/s001650200029

Bhattacharyya, S. (2023, September 11). How is Dijkstra's algorithm used in the real world? Analytics Steps. https://www.analyticssteps.com/blogs/how-dijkstras-algorithm-used-real-world

Böhm, C., & Jacopini, G. (1966). Flow diagrams, Turing machines and languages with only two formation rules. Communications of the ACM, 9(5), 366–371. https://doi.org/10.1145/355592.365646

Canton, E., & de Jong, F. (2005). The demand for higher education in the Netherlands, 1950–1999. Economics of Education Review, 24(6), 651–663. https://doi.org/10.1016/j.econedurev.2004.09.006

Dijkstra, E. (1959, October 28). Communication with an automatic computer.

Dijkstra, E. W. (1968, March). Letters to the editor: Go to statement considered harmful. Communications of the ACM. https://dl.acm.org/doi/10.1145/362929.362947

Dijkstra, E. W. (1968). The Structure of the THE-Multiprogramming System (EWD-196).

Dijkstra, E. W. (1969). Notes on Structured Programming (EWD249).

Dijkstra, E. W. (1972). The Humble Programmer (EWD-340).

Dijkstra, E. W. (1975). Mathematics, Inc., a Private Letter from Its Chairman (EWD-539).

Dijkstra, E. W. (undated, 1962 or 1963). Over de Sequentialiteit van Procesbeschrijvingen (EWD-35).

DSA Dijkstra's Algorithm. W3Schools Online Web Tutorials. (n.d.). https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php

Encyclopædia Britannica, Inc. (2025, August 2). Edsger Dijkstra. Encyclopædia Britannica. https://www.britannica.com/biography/Edsger-Dijkstra

Frana, P. L., & Misa, T. J. (2010). An interview with Edsger W. Dijkstra. Communications of the ACM, 53(8), 41–47. https://doi.org/10.1145/1787234.1787249

Kruseman Aretz, F. E. J. (2003, August 25). The Dijkstra-Zonneveld ALGOL 60 compiler for the Electrologica X1 (PDF). Software Engineering. History of Computer Science. Amsterdam: Centrum Wiskunde & Informatica. ISSN 1386-3711. Archived from the original (PDF) on 2004-01-17.

Misra, J., Richards, H., Boyer, R. S., Gries, D., McIlroy, M. D., & Hoare, T. (2003, August 25). In Memoriam – Edsger Wybe Dijkstra (1930–2002). The University of Texas at Austin Computer Science. https://www.cs.utexas.edu/~EWD/MemRes(USLtr).pdf

Schrijver, A. (2012). On the history of the shortest path problem. Documenta Mathematica Series, 155–167. https://doi.org/10.4171/dms/6/19

Structured Programming Details. DePaul Computer Science. (n.d.). http://facweb.cs.depaul.edu/sjost/it211/documents/structured.htm

UT Computer Sciences Department's Obituary – Edsger Wybe Dijkstra. Edsger W. Dijkstra Archive. (2012, March 6). https://www.cs.utexas.edu/~EWD/CSobit.html