

DeepMind's AlphaDev finds faster sorting routines now in the C++ library

Source copy kept by wheresthe.ai for the story "DeepMind's AlphaDev finds faster sorting routines now in the C++ library" (<https://wheresthe.ai/d/gb-deepmind-alphadev-sorting-2023-06/>).

Source: <https://deepmind.google/blog/alphadev-discovers-faster-sorting-algorithms/>

Text saved 18 Oct 2026 (the page as published).

SHA-256 of the saved text: e201f7d000e2b60a902efbee0120fe9a3bb7bd94f78c2d32addab5a5458d6e1e

June 7, 2023 Science AlphaDev discovers faster sorting algorithms Daniel J. Mankowitz and Andrea Michi

Share

New algorithms will transform the foundations of computing Digital society is driving increasing demand for computation, and energy use. For the last five decades, we relied on improvements in hardware to keep pace. But as microchips approach their physical limits, it's critical to improve the code that runs on them to make computing more powerful and sustainable. This is especially important for the algorithms that make up the code running trillions of times a day. In our paper published today in *Nature*, we introduce AlphaDev, an artificial intelligence (AI) system that uses reinforcement learning to discover enhanced computer science algorithms – surpassing those honed by scientists and engineers over decades. AlphaDev uncovered a faster algorithm for sorting, a method for ordering data. Billions of people use these algorithms everyday without realising it. They underpin everything from ranking online search results and social posts to how data is processed on computers and phones. Generating better algorithms using AI will transform how we program computers and impact all aspects of our increasingly digital society. By open sourcing our new sorting algorithms in the main C++ library, millions of developers and companies around the world now use it on AI applications across industries from cloud computing and online shopping to supply chain management. This is the first change to this part of the sorting library in over a decade and the first time an algorithm designed through reinforcement learning has been added to this library. We see this as an important stepping stone for using AI to optimise the world's code, one algorithm at a time. What is sorting? Sorting is a method of organising a number of items in a particular order. Examples include alphabetising three letters, arranging five numbers from biggest to smallest, or ordering a database of millions of records. This method has evolved throughout history. One of the earliest examples dates back to the second and third century when scholars alphabetised thousands of books by hand on the shelves of the Great Library of Alexandria. Following the industrial revolution, came the invention of machines that could help with sorting – tabulation machines stored information on punch cards which were used to collect the 1890 census results in the United States. And with the rise of commercial computers in the 1950s, we saw the development of the earliest computer science algorithms for sorting. Today, there are many different sorting techniques and algorithms which are used in codebases around the world to organise massive amounts of data online.

Illustration of what a sorting algorithm does. A series of unsorted numbers is input into the algorithm and sorted numbers are output.

Contemporary algorithms took computer scientists and programmers decades of research to develop. They're so efficient that making further improvements is a major challenge, akin to trying to find a new way to save electricity or a more efficient mathematical approach. These algorithms are also a cornerstone of computer science, taught in introductory computer science classes at universities. Searching for new algorithms AlphaDev uncovered faster algorithms by starting from scratch rather than refining existing algorithms, and began looking where most humans don't: the computer's assembly instructions. Assembly instructions are used to create binary code for computers to put into action. While developers write in coding languages like C++, known as high-level languages, this must be translated into 'low-level' assembly instructions for computers to understand. We believe many improvements exist at this lower level that may be difficult to discover in a higher-level coding language. Computer storage and operations are more flexible at this level, which means there are significantly more potential improvements that could have a larger impact on speed and energy usage.

Code is typically written in a high level programming language such as C++. This is then translated to low-level CPU instructions, called assembly instructions, using a compiler. An assembler then converts the assembly instructions to executable machine code that the computer can run.

Figure A: An example C++ algorithm that sorts up to two elements. Figure B: The corresponding assembly representation of the code.

Finding the best algorithms with a game AlphaDev is based on AlphaZero, our reinforcement learning model that defeated world champions in games like Go, chess and shogi. With AlphaDev, we show how this model can transfer from games to scientific challenges, and from simulations to real-world applications. To train AlphaDev to uncover new algorithms, we transformed sorting into a single player 'assembly game'. At each turn, AlphaDev observes the algorithm it has generated and the information contained in the central processing unit (CPU). Then it plays a move by choosing an instruction to add to the algorithm. The assembly game is incredibly hard because AlphaDev has to efficiently search through an enormous number of possible combinations of instructions to find an algorithm that can sort, and is faster than the current best one. The number of possible combinations of instructions is similar to the number of particles in the universe or the number of possible combinations of moves in games of chess (10¹² games) and Go (10⁷ games). And a single, wrong move can invalidate the entire algorithm.

Figure A: The assembly game. The player, AlphaDev, receives the state of the system *st* as input and plays a move at by selecting an assembly instruction to add to the algorithm that has been generated thus far. Figure B: The reward computation. After each move, the generated algorithm is fed test input sequences - for *sort3*, this corresponds to all combinations of sequences of three elements. The algorithm then generates an output, which is compared to the expected output of sorted sequences for the case of sorting. The agent is rewarded based on the algorithm's correctness and latency.

As the algorithm is built, one instruction at a time, AlphaDev checks that it's correct by comparing the algorithm's output with the expected results. For sorting algorithms, this means unordered numbers go in and correctly sorted numbers come out. We reward AlphaDev for both sorting the numbers correctly and for how quickly and efficiently it does so. AlphaDev wins the game by discovering a correct, faster program. Discovering faster sorting algorithms AlphaDev uncovered new sorting algorithms that led to improvements in the LLVM libc++ sorting library that were up to 70% faster for shorter sequences and about 1.7% faster for sequences exceeding 250,000 elements. We focused on improving sorting algorithms for shorter sequences of three to five elements. These algorithms are among the most widely used because they are often called many times as a part of larger sorting functions. Improving these algorithms can lead to an overall speedup for sorting any number of items. To make the new sorting algorithm more usable for people, we reverse-engineered the algorithms and translated them into C++, one of the most popular coding languages that developers use. These algorithms are now available in the LLVM libc++ standard sorting library, used by millions of developers and companies around the world. Finding novel approaches AlphaDev not only found faster algorithms, but also uncovered novel approaches. Its sorting algorithms contain new sequences of instructions that save a single instruction each time they're applied. This can have a huge impact as these algorithms are used trillions of times a day. We call these 'AlphaDev swap and copy moves'. This novel approach is reminiscent of AlphaGo's 'move 37' – a counterintuitive play that stunned onlookers and led to the defeat of a legendary Go player. With the swap and copy move, AlphaDev skips over a step to connect items in a way that looks like a mistake but is actually a shortcut. This shows AlphaDev's ability to uncover original solutions and challenges the way we think about how to improve computer science algorithms.

Left: The original implementation with $\min(A, B, C)$. Right: AlphaDev Swap Move - AlphaDev discovers that you only need $\min(A, B)$.

Left: The original implementation with $\max(B, \min(A, C, D))$ used in a larger sorting algorithm for sorting eight elements. Right: AlphaDev discovered that only $\max(B, \min(A, C))$ is needed when using its copy move.

From sorting to hashing in data structures After discovering faster sorting algorithms, we tested whether AlphaDev could generalise and improve a different computer science algorithm: hashing. Hashing is a fundamental algorithm in computing used to retrieve, store, and compress data. Like a librarian who uses a classification system to locate a certain book, hashing algorithms help users know what they're looking for and exactly where to find it. These algorithms take data for a specific key (e.g. user name "Jane Doe") and hashes it – a process where raw data is turned into a unique string of characters (e.g. 1234ghfty). This hash is used by the computer to retrieve the data related to the key quickly rather than searching all of the data. We applied AlphaDev to one of the most commonly used algorithms for hashing in data structures to try and discover a faster algorithm. And when we applied it to the 9-16 bytes range of the hashing function, the algorithm that AlphaDev discovered was 30% faster. This year, AlphaDev's new hashing algorithm was released into the open-source Abseil library, available to millions of developers around the world, and we estimate that it's now being used trillions of times a day.

Optimising the world's code, one algorithm at a time By optimising and launching improved sorting and hashing algorithms used by developers all around the world, AlphaDev has demonstrated its ability to generalise and discover new algorithms with real-world impact. We see AlphaDev as a step towards developing general-purpose AI tools that could help optimise the entire computing ecosystem and solve other problems that will benefit society. While optimising in the space of low-level assembly instructions is very powerful, there are limitations as the algorithm grows, and we are currently exploring AlphaDev's ability to optimise algorithms directly in high-level languages such as C++ which would be more useful for developers. AlphaDev's discoveries, such as the swap and copy moves, not only show that it can improve algorithms but also find new solutions. We hope these discoveries inspire researchers and developers alike to create techniques and approaches that can further optimise fundamental algorithms to create a more powerful and sustainable computing ecosystem.

Acknowledgements Juanita Bawagan, Arielle Bier, Gabriella Pearl, Duncan Smith, Katie McAtackney, Kathryn Seager, Max Barnett, Ross West, Dominic Barlow, Hollie Dobson, Domhnall Malone for their help with text and figures. This work was done by a team with contributions from Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Koppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals and David Silver. Mikita Sazanovich and Danila Kutenin for their contributions to the hashing algorithm.

Related posts

Discovering novel algorithms with AlphaTensor October 2022 Science [Learn more](#)

Competitive programming with AlphaCode December 2022 Research [Learn more](#)