Part 1

Let's imagine you are on a quest to find the most complex object in the universe. It is astounding that you will eventually discover it in the form of 1.5 kilograms of soft, wet tissue just above your own eyes. Our brain, the home of our mind, is, by far, the most complex object we know of. It can solve problems, create art, build things, fall in love, figure out how the universe works, and pose profound questions. As a system, it not only looks outward but also inward and attempts to understand how our mind functions, how we experience feelings, and, above all, becomes aware of its own existence. Starting from antiquity, we have tried to uncover the mysteries of thought and reasoning, and this remains the most profound and complex intellectual challenge.

As we ponder the mystery of our minds, we have consistently compared it to the latest and most complex technologies of our time. At one point, we envisioned it as a clockwork mechanism of immense intricacy. Next, we called it the engine of thought during the Industrial Revolution. In the early twentieth century, we likened it to a telephone network. Ultimately, beginning in the mid-twentieth century, we compared it to a computer. While this latest analogy may eventually be replaced by another technology we have yet to discover, there is a deeper reason why it might endure. A computer is not just a specific object we construct, but also a conceptual framework for processing information. Therefore, as long as we perceive the mind as an information-processing mechanism, a computer will continue to serve as a fitting analogy and a mechanism for simulating the mind.

We always fostered the desire to build machines that can emulate our minds, and one of the finest examples of a mind at work

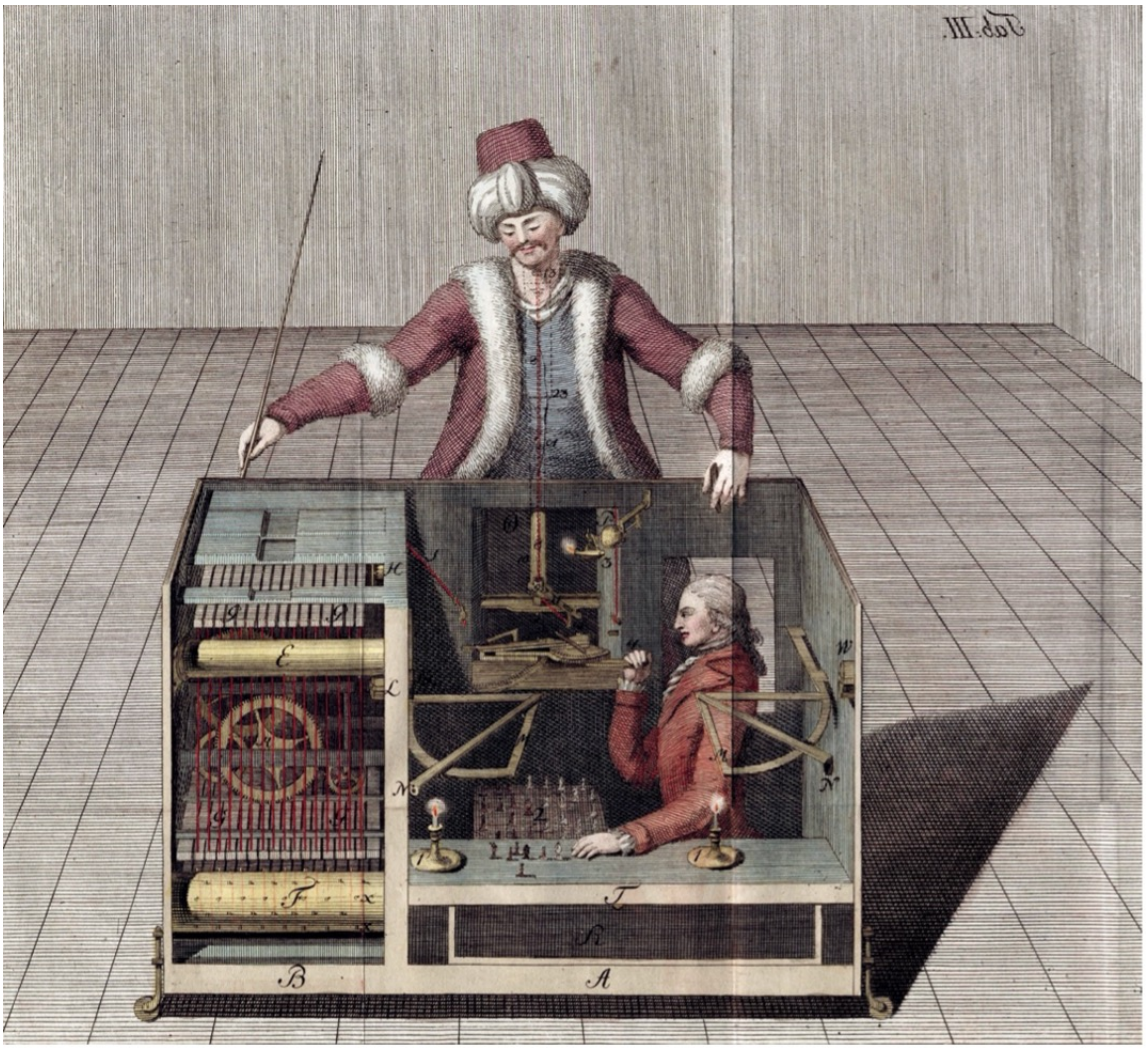


Figure 1: Humboldt University Library, Public Domain.

was our ability to play the game of chess. In 1770, as we discovered the power of clockworks, Wolfgang von Kempelen (1734-1804) designed and demonstrated a machine that could play a strong game of chess. Known as The Mechanical Turk (Fig. 1), it was an automaton that made mechanical moves. The contraption was shown in exhibitions, often defeating its opponents, including Napoleon Bonaparte and Benjamin Franklin. This continued for the next 84 years until a fire at the Philadelphia Museum destroyed the machine. There were speculations about how it worked, but the correct explanation was revealed after it was destroyed. The machine hid a human player inside the dense array of gears, with a clever arrangement that gave the visitors an illusion that they were seeing through all the machinery. [0]

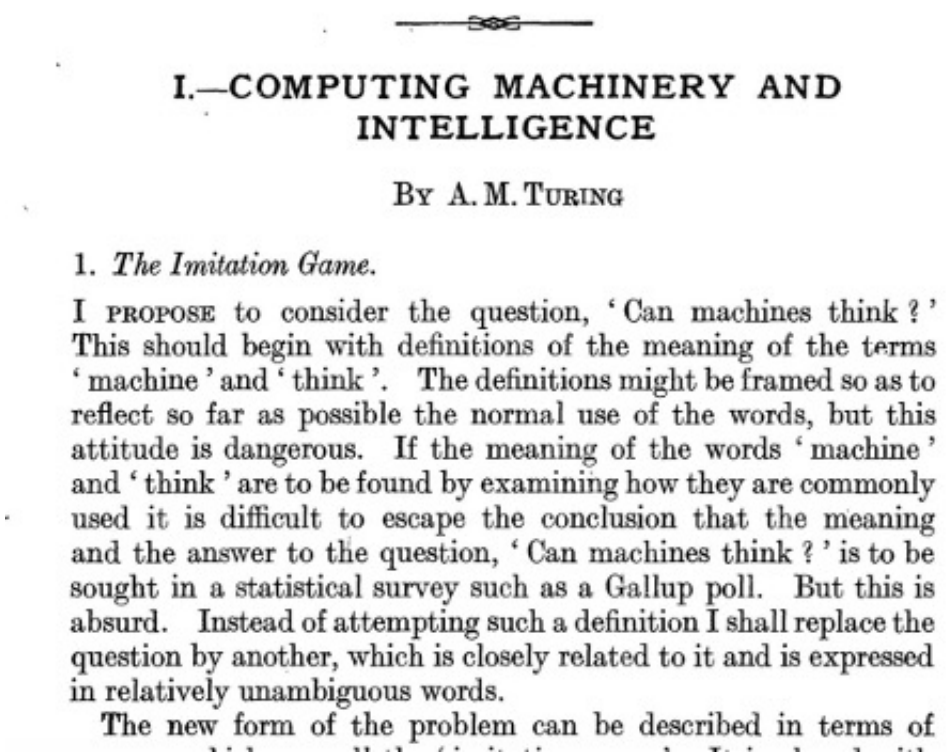
This machine was a fake, but our desire to build machines that could do things that, if done by humans, would be considered to require intelligence expanded. However, we could never define "intelligence" in a satisfactory manner. It had been a sliding scale ever since. Many people considered performing simple arithmetic to require intelligence. The earliest mechanical calculators and, eventually, the first electronic computers in the mid-twentieth century could handle mathematical operations accurately and much faster than humans could. [1] We eventually decided to demote the ability to perform arithmetic operations from the realm of intelligence. Instead, they moved into the realm of algorithms. They are tasks that can be done by a series of well-defined logical steps, just like we teach kids how to add, subtract, multiply, and divide using simple logical steps. There is an old saying — "Artificial intelligence is whatever machines can't do," and consequently, the definition keeps shifting.

Just when computers were coming into being, a few scientists were convinced that more intelligent tasks could be reduced to manipulating information that a computer can execute. Two branches of science started to develop simultaneously.

On the one hand, psychologists, neuroscientists, and behavioral scientists continued their quest to understand the animal brain. This research mainly concerned subconscious activities such as predicting and manipulating animal behavior, a deeper understanding of our senses, and a rudimentary exploration of some of our more complex activities.

However, Alan Turing's 1950 paper "Computing Machinery and Intelligence," (Fig. 2) in which he asked, "Can machines think?" started the modern quest for a deeper understanding of the thinking process and attempts to replicate it in machines. He introduced the idea of the Turing Test as a measure of machine intelligence. The idea was to connect two rooms by a typing and

display device. In one room, we have the subject of the experiment, which can be a human or a machine, and in the other, a panel of human judges. The judges can ask any question, and the subject has to answer. If, after a given period, the judges conclude that the subject in the other room is a human, even though a machine responded to the questions, then this machine would pass the Turing Test.



I.—COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

1. *The Imitation Game.*

I propose to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

The new form of the problem can be described in terms of

Figure 2: Groundbreaking paper by Alan Turing.

In 1956, the landmark Dartmouth Conference brought together a group of visionary scientists who collectively believed that every facet of human intelligence, including learning, could, in principle, be accurately described so that a machine could simulate it. There was tremendous optimism that the construction of intelligent machines was just around the corner. However, this optimism has been challenged repeatedly throughout history, and it is often referred to as AI winters.

As scientists explored whether computers could handle some of our intelligent tasks, it became apparent that our understanding

of the processes involved was rudimentary. Also, the computers available were bulky and slow and had a tiny fraction of the capabilities of today's computers. The biggest, most powerful machines in the 1950s were far weaker than the computer that runs our wristwatch today.

Therefore, the scientists had to find toy problems. They could only tackle relatively simple intelligent tasks. They looked for simple, structured problems such as puzzles, games, and linguistic tasks. The impetus behind making the Mechanical Turk was that we considered playing chess to require high intelligence. However, this time, the scientists did not try to cheat but wanted to build a computer program that could actually play this complex game.

In this two-part article, we will follow this single example, playing the game of chess, and see how it reflects the development of Artificial Intelligence over the next 75 years. We will explore the challenge of making machines play strategic games like chess, how they evolved, and how machines became unbeatable in any strategic game, even by the best human players.

Most intelligent tasks we perform in our lives, we learn those skills through examples rather than being told step-by-step how to do them. For example, no one told us how to recognize the alphabet, but showed us many examples, and our minds figured out some subconscious rules to distinguish an "A" from all other letters. Scientists quickly realized that if machines must do complex tasks, they cannot be taught algorithmically, with step-by-step instructions, as we ourselves may not know these logical steps, but rather by showing many examples of the correct behavior. Although that was the holy grail of AI, Machine Learning was a hard task. Therefore, most early experiments in Artificial Intelligence relied on human-created algorithms rather than self-learning. In the second part of this article we will explore how we

eventually made machines that can learn how to play complex board games without any human-created step-by-step instructions. In this installment, we will explore how the algorithmic approach worked and what it achieved.

In the early days of AI, a group of scientists tried to tackle the problem of programming machines to play strategic games, and the king of such games was chess. In chess, as in tic-tac-toe, chance plays no role other than which player makes the first move. Beyond the first move, everything is purely based on logical thinking. At every stage of the game, each player must consider all possible legal moves available to them. The player then has to think through how the opponent might react to each of their potential moves. Each of the opponent's moves must be responded with the best countermove. This back-and-forth calculation can go on for many moves until the game ends. In a simple game like tic-tac-toe, it is possible to think through till the end of the game and decide on the next optimal move. If both players do that, every tic-tac-toe game will end in a draw. In real life, we seldom think that deeply, and at some point, one player makes a fatal mistake that leads to the other person winning the game.

Game	**Board Size**	**State Space Complexity**	**Game Tree Complexity**	**Av. Game Length**	**Branching Factor**
Tic-Tac-Toe	9	10^3	10^5	9	4
Chess	64	10^{44}	10^{123}	70	35
Go	361	10^{170}	10^{505}	211	250

To make a good move, a player must think of all possible moves available to them and then think of every way the opponent can respond. A good player must think beyond the first pair of move-countermoves and imagine all possible moves they can make after

each of the move-countermove pairs. This can go on and on. Let's say that in the middle game position, player-A has a choice of 35 possible moves, which is the average number of legal moves available in this game. Let's say player-B also has 35 possible legal responses. Therefore, player-A has to think of 35 * 35, or 1,225 possible board positions, after just a single move-countermove pair. If player-A wants to think through three pairs of move-countermove before deciding, then the number of board positions that must be considered becomes 1.8 billion. Yet, a good chess player must think beyond just three move-countermove pairs. This raises two important questions. How do human players manage this task, and how can computers do the same?

This problem can be generalized to a problem of finding an optimal solution in a very large search space. We can imagine the game play as a tree, where each node is a particular board position and the branches coming out of that node are the possible legal moves (Fig-3). Each branch ends in a new board position, represented by another node in the tree. Based on this tree, one can estimate the total number of distinct games that can be played. This number is often referred to as Game-Tree Complexity.

Claude Shannon estimated the number of possible chess games to be around 10120, or the mind-blowing number of 1 followed by 120 zeroes. To compare, the number of possible tic-tac-toe games is just 105. On the other hand, the number of possible Go games, an ancient Asian game, is 10505. Looking at these numbers, it becomes evident that an exhaustive search of the game space of any non-trivial game is beyond the capacity of humans or any machine we know of.

Therefore, to tackle this problem, we take three shortcuts.

First, neither human players, nor chess-playing programs look beyond a small number of move-countermove pairs. We don't ex-

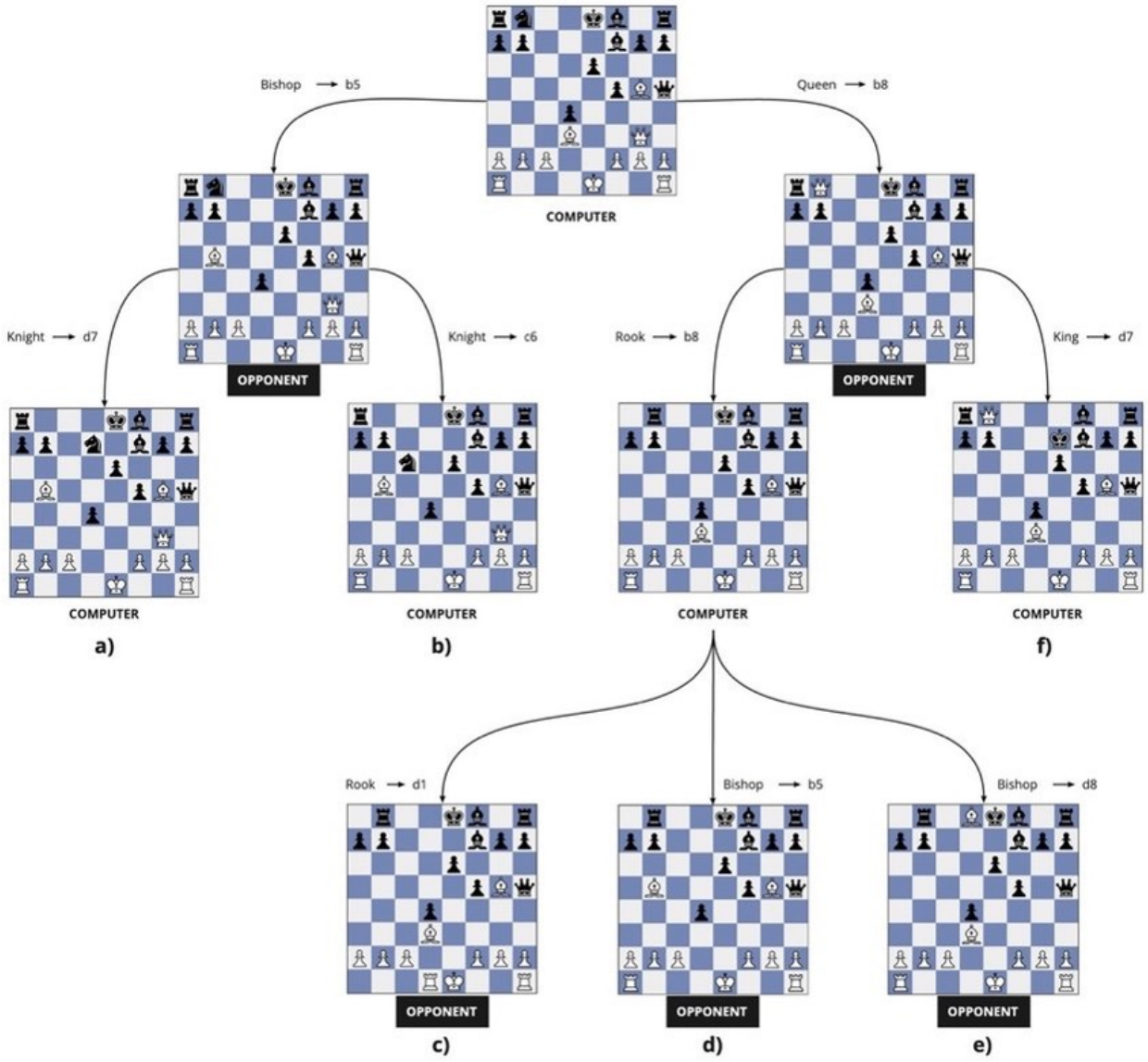


Figure 3: A small subset of a decision tree for chess [2] [Accessed 2 Mar, 2025].

actly know how many steps chess champions can consider, but it is possibly below ten. That is, we look as far as we can comfortably see, judge the final board positions, and try to determine which player will be ahead in the game if we can reach that board position. Based on this analysis, we pick the most advantageous board position and make the initial move that leads to that position, even if the opponent makes all the best moves.

Two, we don't explore all possible moves at every stage of the analysis but only consider those that seem to make the most sense. In a typical chess game, the average number of moves available

is around 35. If we consider all these moves then the number of board positions we have to consider will grow exponentially, quickly becoming impossibly large. Instead, out of all possible moves, we pick just a small fraction of them as the most promising and drill down further. This selection is done based on some rules of thumb assessment. For humans, this comes from experience, and we develop an intuitive feel of what is useful and what can be discarded. For computers, the programmer defines the rules of thumb. However, both for humans and machines, this premature pruning of the tree could be very risky. We may prune out a branch, but the best solution could be lurking in that part of the tree. For weaker human players, that is where we could go wrong, while better players make fewer mistakes of pruning out the wrong branch.

Third, as we reach the end of our search, based on whatever limit of move-countermove pairs we can afford, we need to evaluate the final board position and decide which of these looks the best. Once again, this is often a difficult decision to make. Let's say, in one board position, the player has far fewer pieces than the opponent, then the decision is easier. However, if the two board positions have no such obvious difference, then the choice becomes harder. Once again, a human player uses their experience-based intuition and a computer program uses rules of thumb or heuristics to make that decision.

All initial chess-playing programs used the same algorithm to search the game tree. However, with time, the rules of thumb they used improved, and so did the raw power of the machines they ran on. The very first full chess-playing computer program was developed by IBM in 1957. It took another twenty years when a computer program developed by Bell Labs reached Master-level rating in 1978. In another ten years, IBM's Deep Thought defeated a Grandmaster. In 1996 IBM's Deep Blue managed to

defeat the world champion Gary Kasparov in one game, and an improved version of the same machine defeated Kasparov in a six-game match, thus becoming the first chess champion. That was the end of the human domination of the game.

Despite this achievement, it was clear that the machines were beating humans primarily by brute force. The tremendous processing speed of computers allowed them to go wider and deeper in exploring the game tree, evaluating billions of positions. For example, IBM's Deep Blue could process 200 million board positions per second. In a typical game, with 3 minutes per move tournament time control, it could, on average, evaluate 36 billion positions per move. A human player typically analyzes a handful of moves deeply, intuitively rejecting all other possibilities. There is almost no comparison between humans and machines regarding the precision of the search.

Stockfish, an open-source chess engine, evolved during the 2000s. Compared to Deep Blue, it was much better at judging board positions and could, therefore, focus its search more precisely. While Deep Blue ran on huge machines with custom hardware, today's Stockfish, running on a laptop, can beat Deep Blue and most other human players. Though much less dependent on pure brute force, it still relies on massive searches and utilizes an extensive database of human-played opening sequences.

In the next part of the article, we will explore how the next generation of chess-playing computers learn to play the game. Instead of relying on an array of human-created rules of thumb, these machines can learn those rules simply by watching other players or even learn on their own by playing many games with themselves. These learning machines not only outperform all of their algorithmic counterparts but can also tackle more difficult board games like the Asian game Go, which were beyond the reach of the algorithmic approach described above.

Another important distinction between the algorithms described in this article and the "learning" systems we will discuss in Part-2 is the generality of the latter. A chess-playing program, as described above, can do just one thing: play a game of chess. However, as we will see later, the new generation of self-learning systems can learn other tricks. The same program that plays chess very well can be taught an entirely different board game simply through experience and training.

(To be continued.)

References

[0] Wikipedia Contributors, *Mechanical Turk — Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Mechanical_Turk&oldid=1285076875, [Accessed: 24-April-2025], 2025.

[1] CHM Author, *Timeline of Computer History*, https://www.computerhistory.org/timeline/computers/, [Accessed: 2025-4-24].

[2] N. C. Thompson, S. Ge, and G. F. Manso, *The Importance of (Exponentially More) Computing Power*, 2022. arXiv: 2206.14007 [cs.AR]. [Online]. Available: https://arxiv.org/abs/2206.14007.