Impossibilities in Computing

Lecture 1

Unit 2

ML and Society (Spring 2024)

Atri Rudra

Pass Phrase for today: **Sorelle Friedler**

Sorelle Friedler

Shibulal Family Associate Professor of Computer Science

Sorelle Friedler is the Shibulal Family Associate Professor of Computer Science at Haverford College. She served as the Assistant Director for Data and Democracy in the White House Office of Science and Technology Policy under the Biden-Harris Administration where her work included the AI Bill of Rights. Her research focuses on the fairness and interpretability of machine learning algorithms, with applications from criminal justice to materials discovery.

Sorelle is a Co-Founder and former Executive Committee Member of the ACM Conference on Fairness, Accountability, and Transparency (FAccT) as well as a former Program Committee Co-Chair of FAccT and FAT/ML. She has received grants for her work on fairness in machine learning, fairness and social networks, using interpretable machine learning techniques to inform scientific hypotheses, Responsible CS Education, and policy and discriminatory machine learning. Key papers include work on disparate impact in machine learning and on accelerating materials discovery with interpretable machine learning.

Before Haverford, Sorelle was a software engineer at Alphabet (formerly Google), where she worked in the X lab and in search infrastructure. She holds a Ph.D. in Computer Science from the University of Maryland, College Park, and a B.A. from Swarthmore College.

Checking In

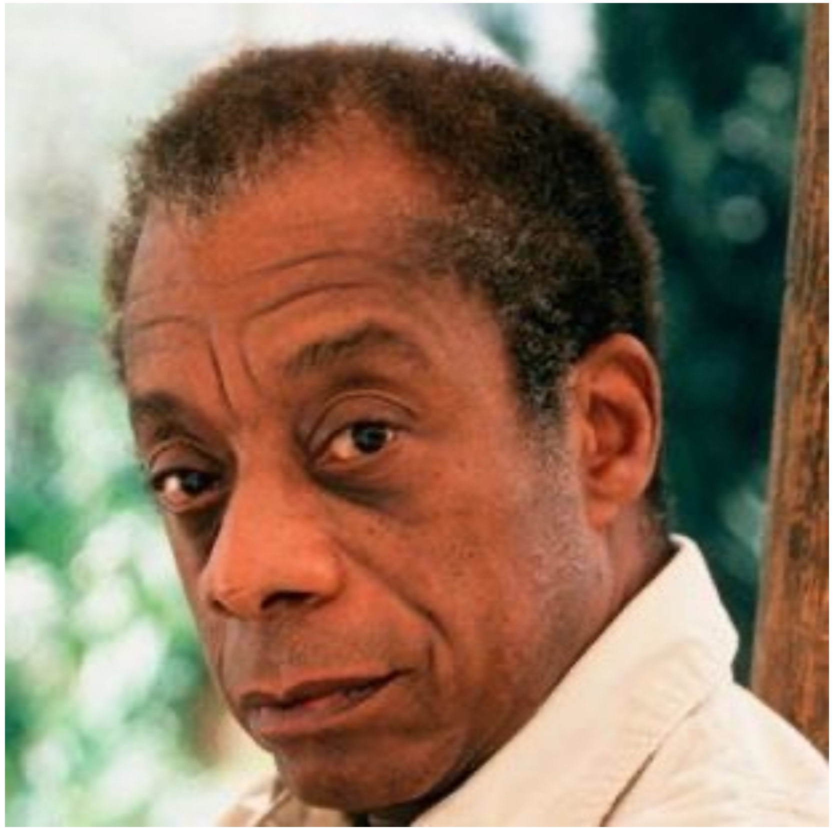
How was the unit 1 group submission?

Back to the James Baldwin quote….

Discuss!

How do you interpret the above statement?

How do you interpret the above statement for computational problems?



" The impossible is the least that one can demand "

Class Responses…

Keep testing the boundaries of computing folks: how far we can push what our algos

No progress can be made without exploring the uknown

My first interpretation

Only by knowing the impossible, we know the limits of what is possible

A 1936 paper

Does anyone know the significance of this paper?

The paper ended Hilbert's plan to automatize all of mathematics

Pic from Wikipedia

This paper started CSE!!

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

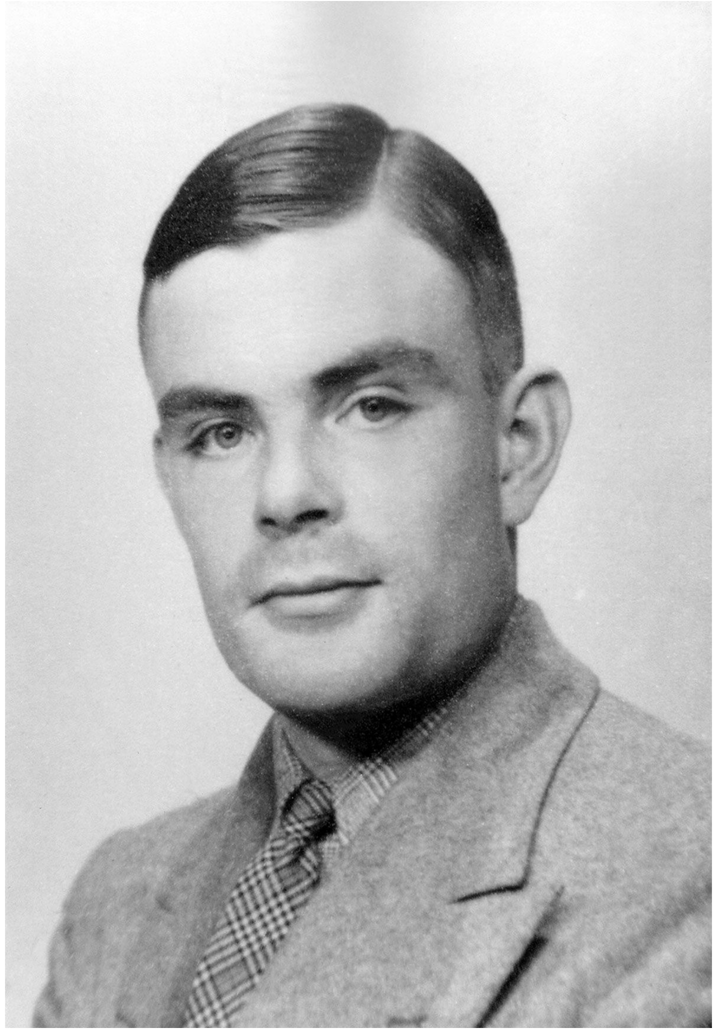
In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

Who was Alan Turing?

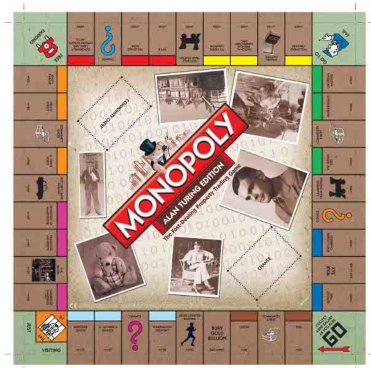
Pick ALL choices that are TRUE

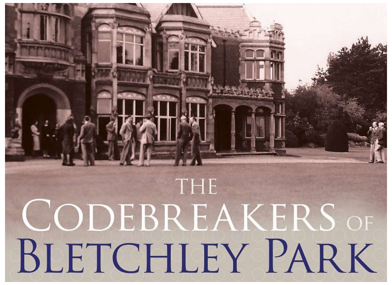


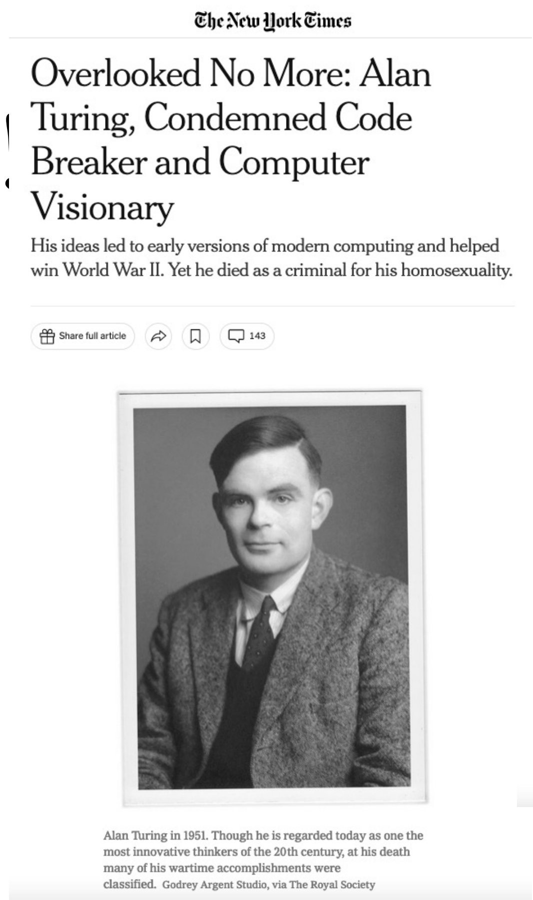
https://www.britannica.com/biography/Alan-Turing

1. Benedict Cumberbatch played Alan Turing in the 2014 movie The Imitation Game
2. Turing was an avid Monopoly player
3. Turing was 5th in the British marathon trials for the 1948 Olympics
4. Turing led the effort to break Nazi code ("Enigma") at Bletchley Park
5. Turing was a gay man who given a choice between "chemical castration" and imprisonment for homosexual acts
6. Turing wrote what is the considered the first major paper on AI

Correct answer: ALL of them!







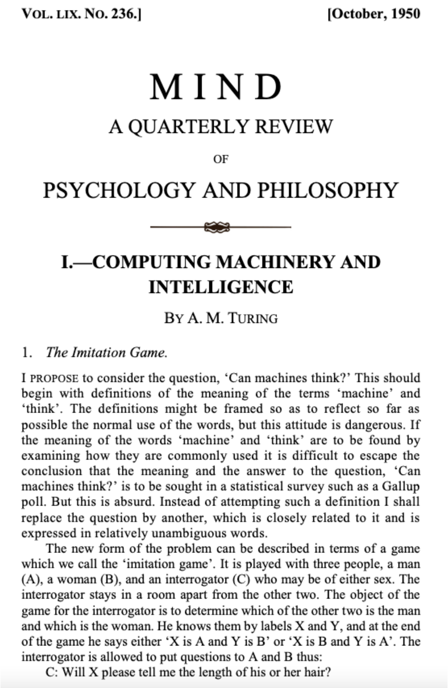
The New York Times

Overlooked No More: Alan Turing, Condemned Code Breaker and Computer Visionary

His ideas led to early versions of modern computing and helped win World War II. Yet he died as a criminal for his homosexuality.

Share full article 143

Alan Turing in 1951. Though he is regarded today as one the most innovative thinkers of the 20th century, at his death many of his wartime accomplishments were classified. Godrey Argent Studio, via The Royal Society



VOL. LIX. NO. 236.] [October, 1950

MIND

A QUARTERLY REVIEW OF PSYCHOLOGY AND PHILOSOPHY

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 a game which we call the 'imitation game'. It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either 'X is A and Y is B' or 'X is B and Y is A'. The interrogator is allowed to put questions to A and B thus:

C: Will X please tell me the length of his or her hair?

1. Benedict Cumberbatch played Alan Turing in the 2014 movie The Imitation Game https://www.imdb.com/title/tt2084970/
2. Turing was an avid Monopoly player https://www.theguardian.com/technology/2012/sep/10/alan-turing-monopoly-board-google
3. Turing was 5th in the British marathon trials for the 1948 Olympics https://kottke.org/18/04/alan-turing-was-an-excellent-runner
4. Turing led the effort to break Nazi code ("Enigma") at Bletchley Park https://www.nationalww2museum.org/war/articles/alan-turing-betchley-park
5. Turing was a gay man who given a choice between "chemical castration" and imprisonment for homosexual acts https://www.nytimes.com/2019/06/05/obituaries/alan-turing-overlooked.html
6. Turing write what is the considered the first major paper on AI https://academic.oup.com/mind/article/LIX/236/433/986238

Turing didn't invent programs we know today

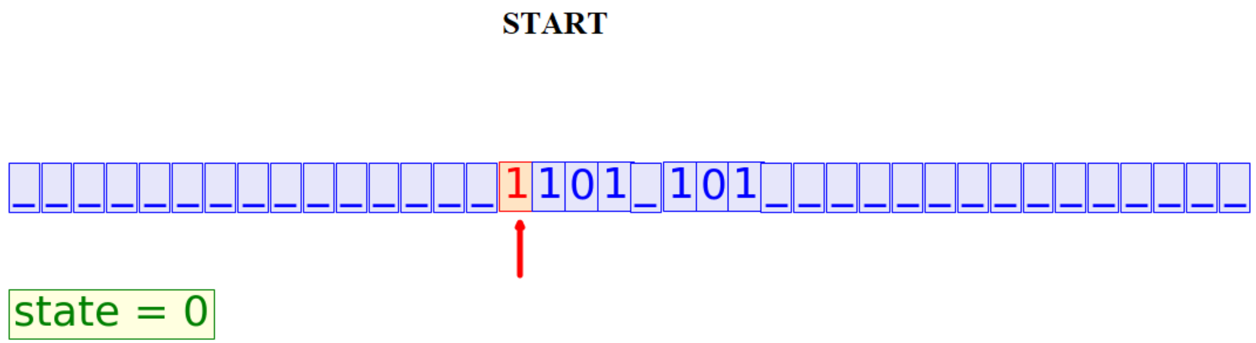
He invented what we today call Turing machines

These are programs that directly manipulate memory

Python code to add a and b

```
def add (a,b): # This means function name is add an                s: a and b
    c = a+b # assign c to be the sum of a and b
    return c # The function then returns the sum
```

Sandipan Dey (UMBC)



Turing machine to add two numbers

The Turing Machine

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

$\delta(p, X) = (q, Y, L)$.

The program for binary addition

move right to end of first block

$\delta(0, 0) = (0, 0, R)$
$\delta(0, 1) = (0, 1, R)$
$\delta(0, \sqcup) = (1, \sqcup, R)$

move right to end of second block

$\delta(1, 0) = (1, 0, R)$
$\delta(1, 1) = (1, 1, R)$
$\delta(1, \sqcup) = (2, \sqcup, L)$

subtract one in binary

$\delta(2, 0) = (2, 1, L)$
$\delta(2, 1) = (3, 0, L)$
$\delta(2, \sqcup) = (5, \sqcup, R)$

move left to end of first block

$\delta(3, 0) = (3, 0, L)$
$\delta(3, 1) = (3, 1, L)$
$\delta(3, \sqcup) = (4, \sqcup, L)$

add one in binary

$\delta(4, 0) = (0, 1, R)$
$\delta(4, 1) = (4, 0, L)$
$\delta(4, \sqcup) = (0, 1, R)$

clean up

$\delta(5, 1) = (5, , R)$
$\delta(5, \sqcup) = (\mathbf{halt}, \sqcup, *)$

https://stackoverflow.com/questions/59045832/turing-machine-for-addition-and-comparison-of-binary-numbers

Let's assume TM and programs are same

Why abstract out computation as Turing machines?

The paper ended Hilbert's plan to automatize all of mathematics

Pic from Wikipedia

Allows us to ask "meta" questions on computation

I.e. what can we compute ?

My second interpretation

If something is impossible, it might make something *else* possible

Any example?

Cryptography!

But, wait…

What do you think when you hear impossible?

Discuss!

In the context of computational problems, what does an impossible problem (that is defined *mathematically*) mean to you?

Class Responses…

Problems take a long time to solve (using existing techniques)

Problems with no finite answer

Problems that cannot be defined precisely

Problems that do not have a solution

Problems that make other things possible

Format of my (five!) interpretations

Start off with the interpretation with an example

Talk about the “work around” folks have figured out

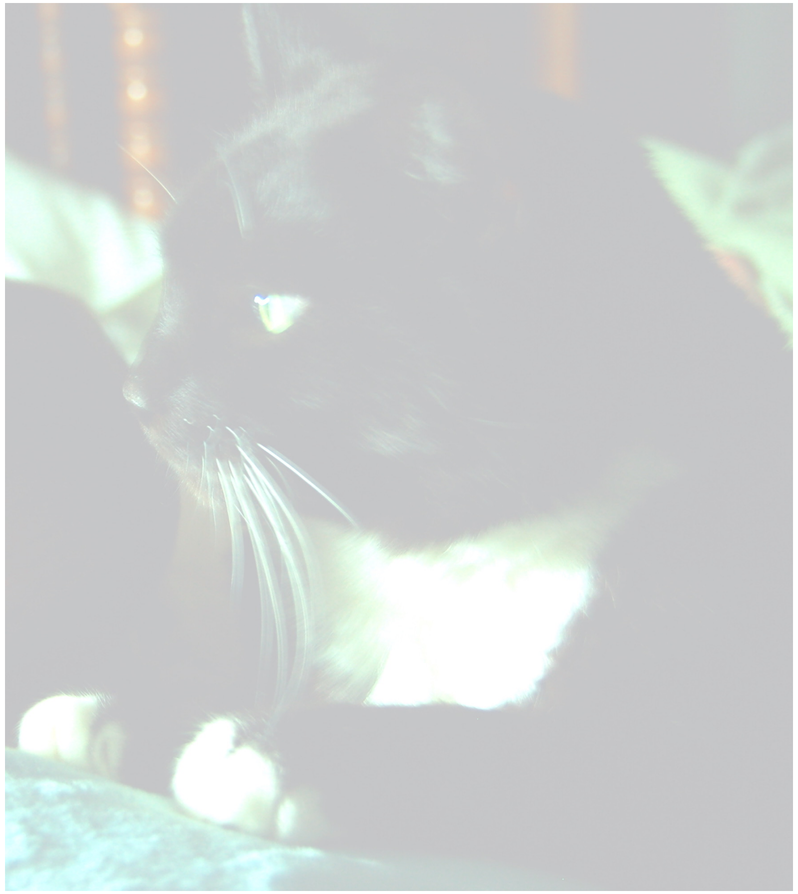
First interpretation

Essentially not possible to come up with a precise mathematical description of a problem

At least not in the sense of being able to write the math formulation down

Any example?

Cat vs. Dogs



Warren and Billy

How do you “define” a dog vs cat image?

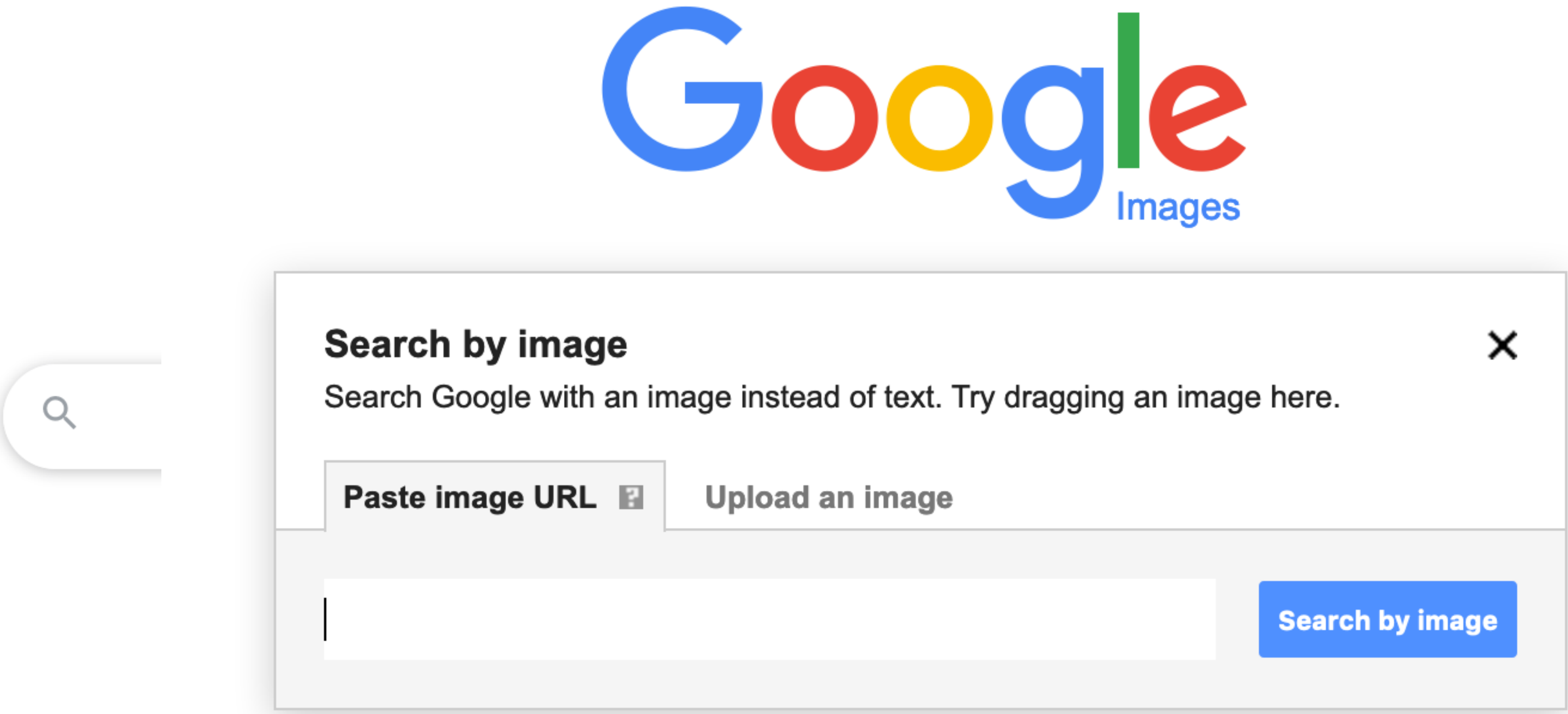
First interpretation

Essentially not possible to come up with a precise mathematical description of a problem

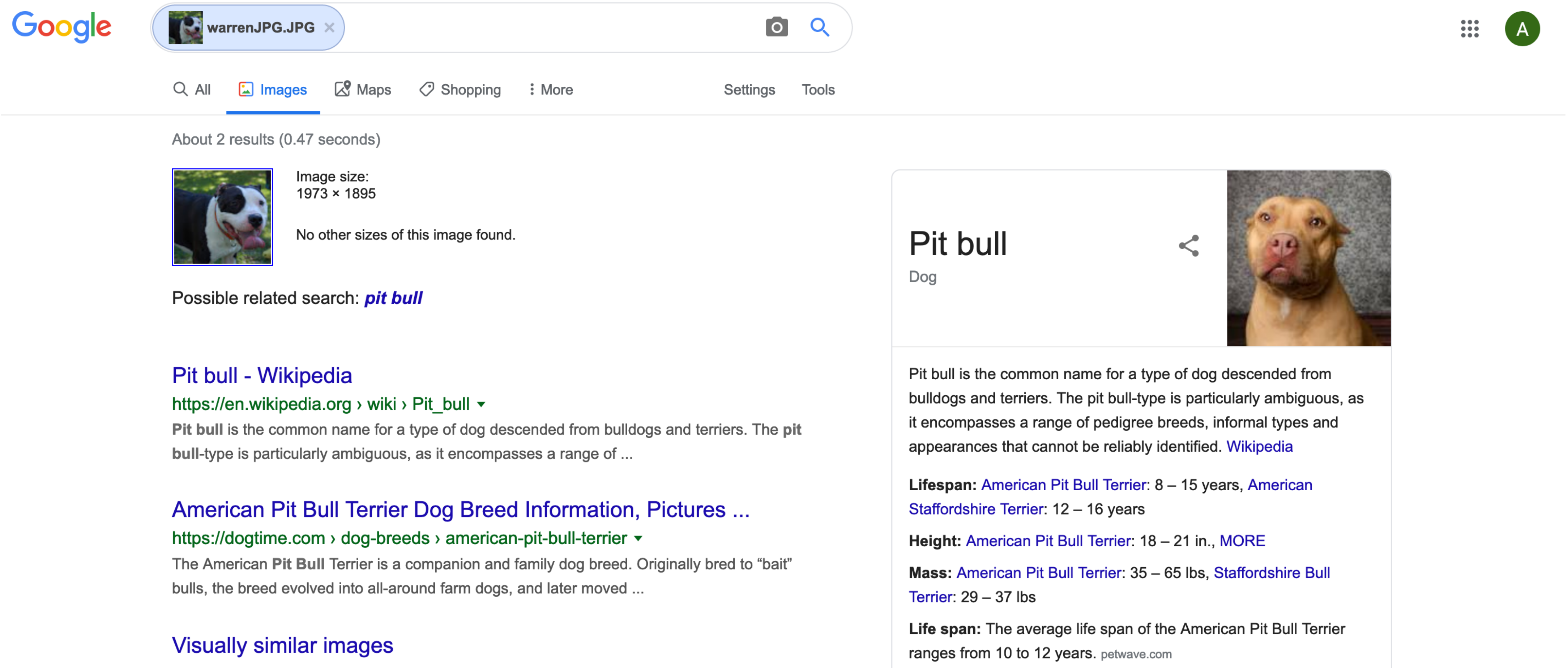
At least not in the sense of being able to write the math formulation down

Try to learn the problem from data itself!

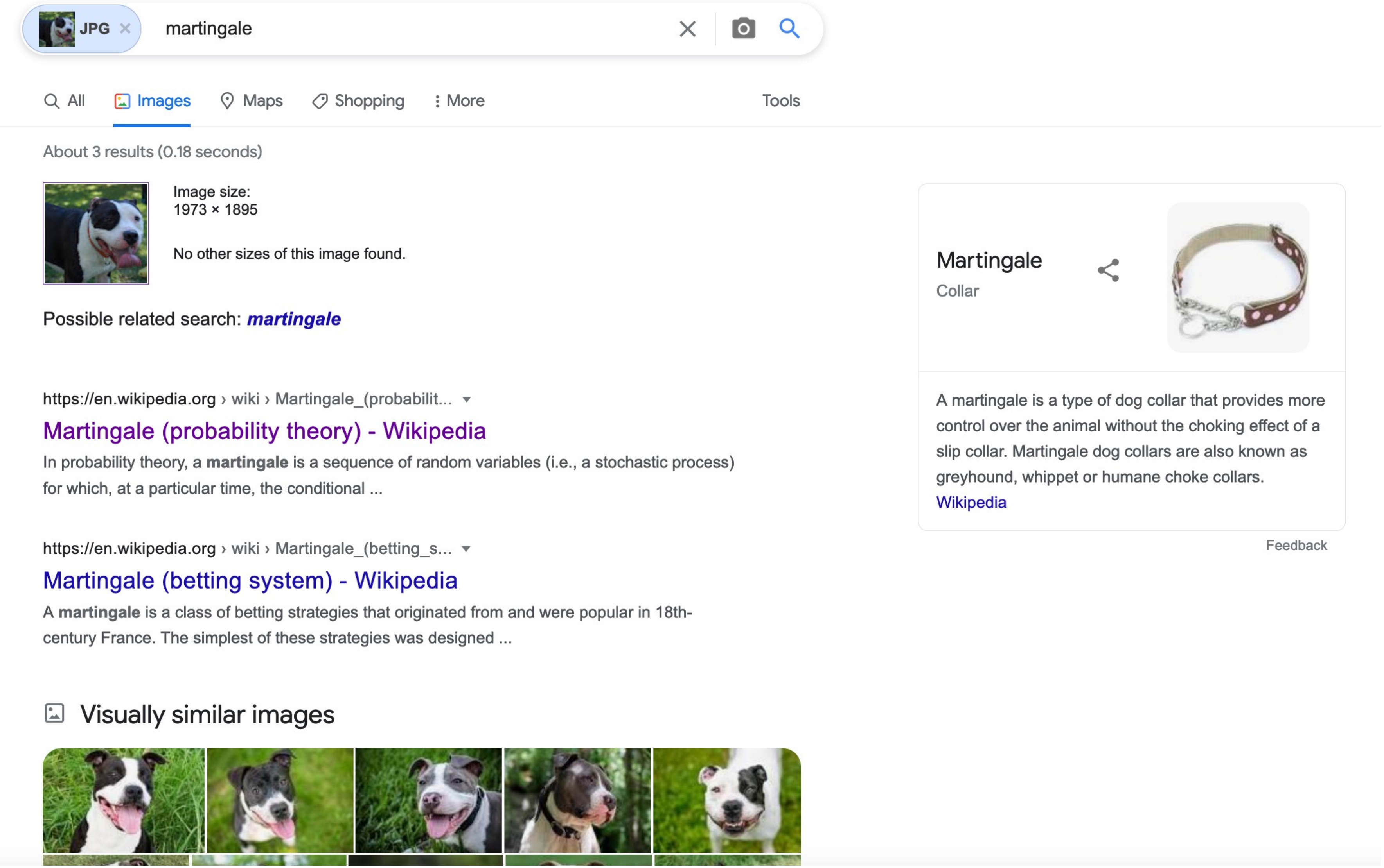
Google Images has “solved” this problem



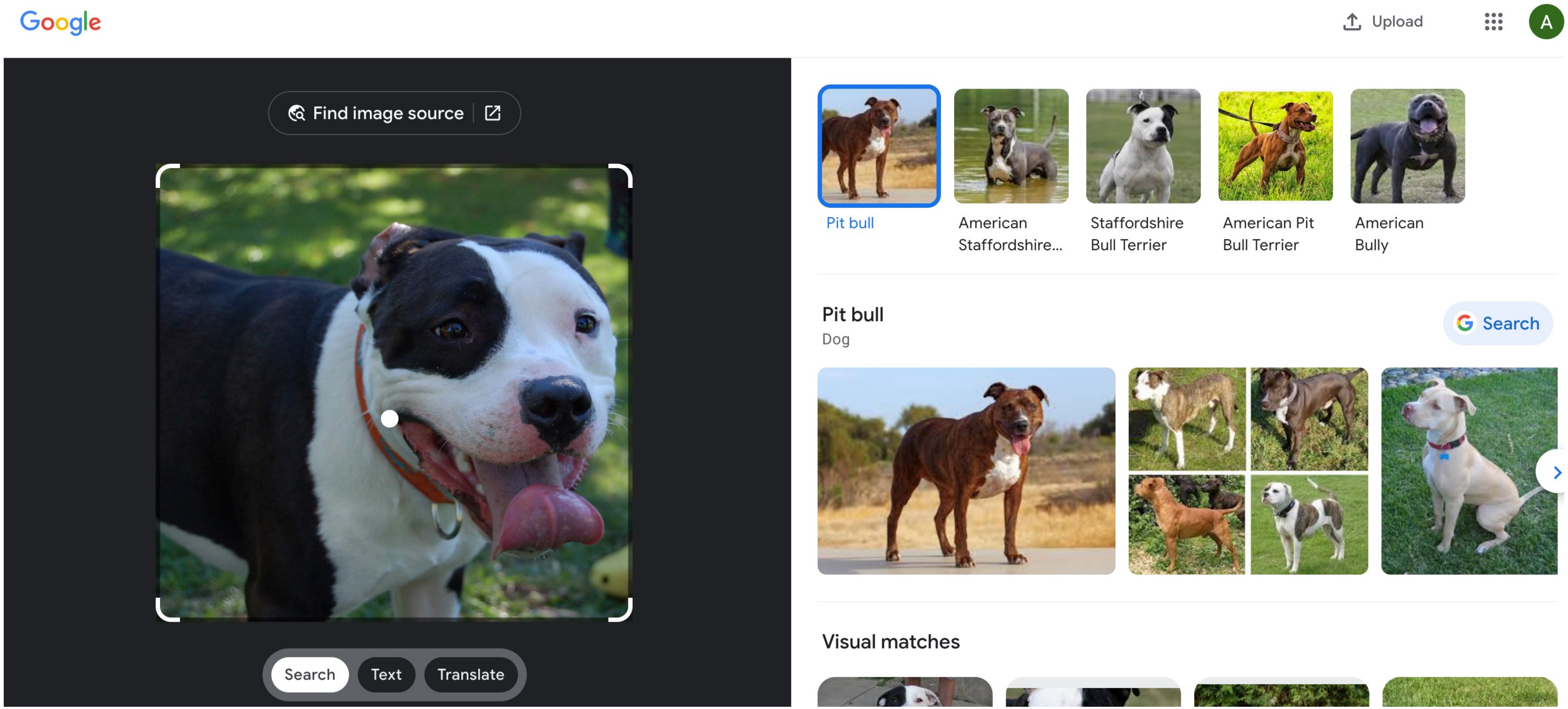
My result for Warren (Spring 20)



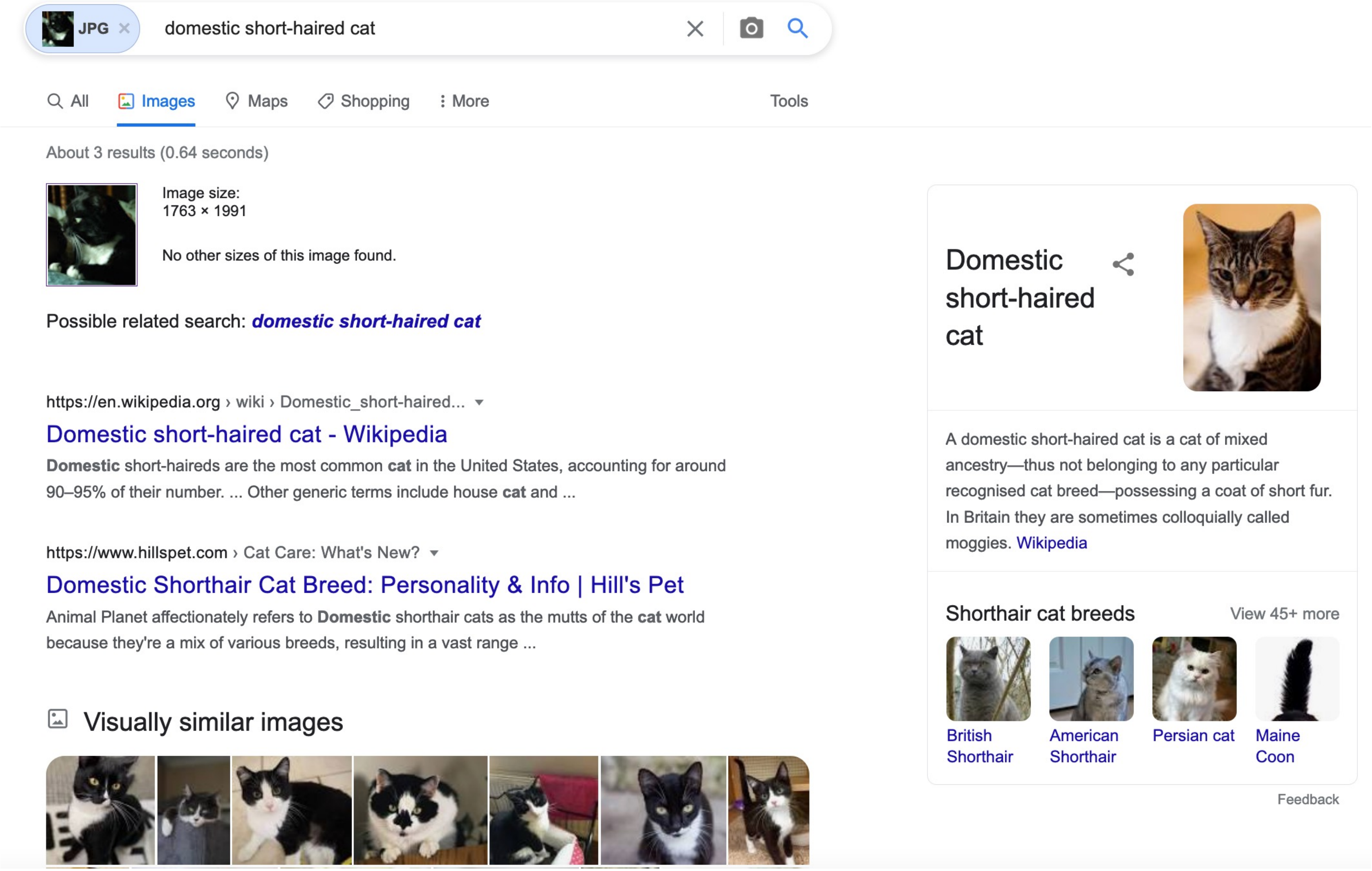
My result for Warren (Spring 22)



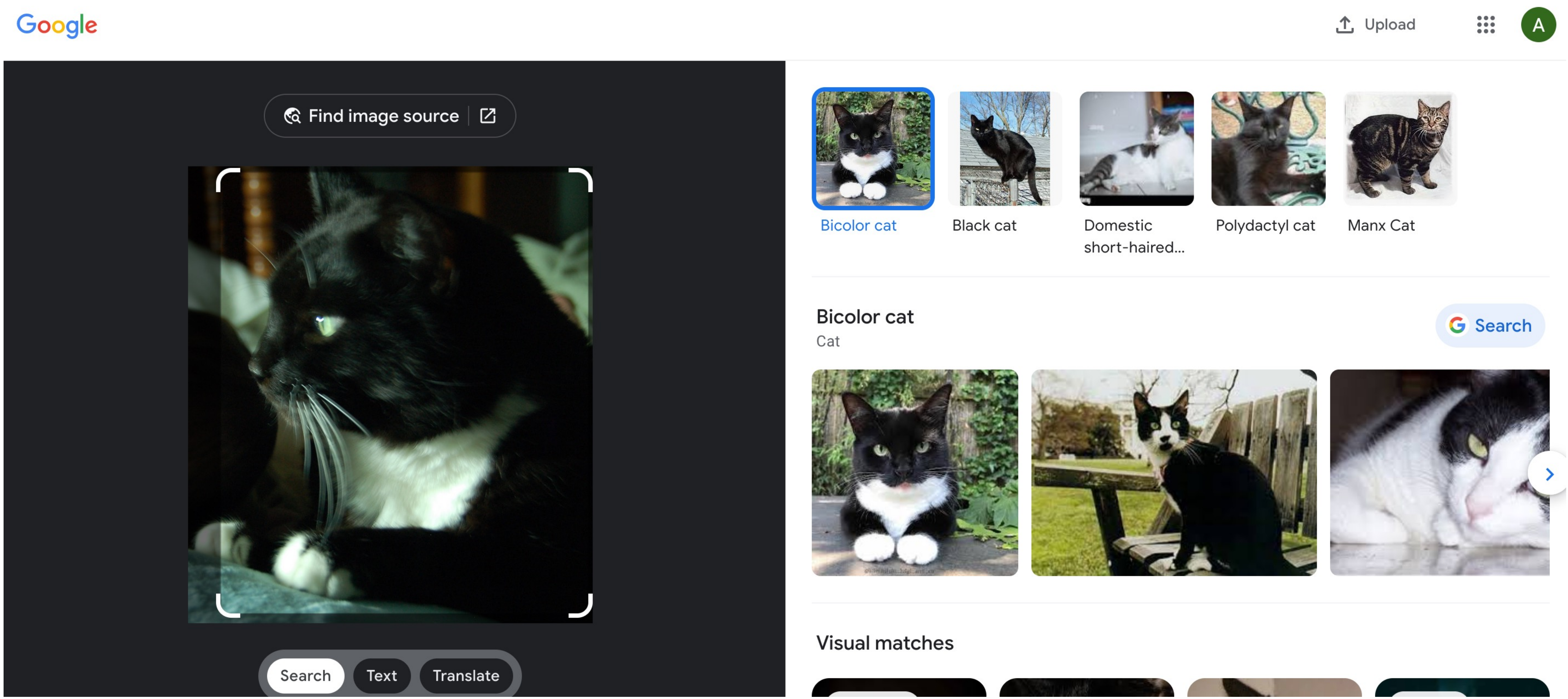
My result for Warren (Spring 23)



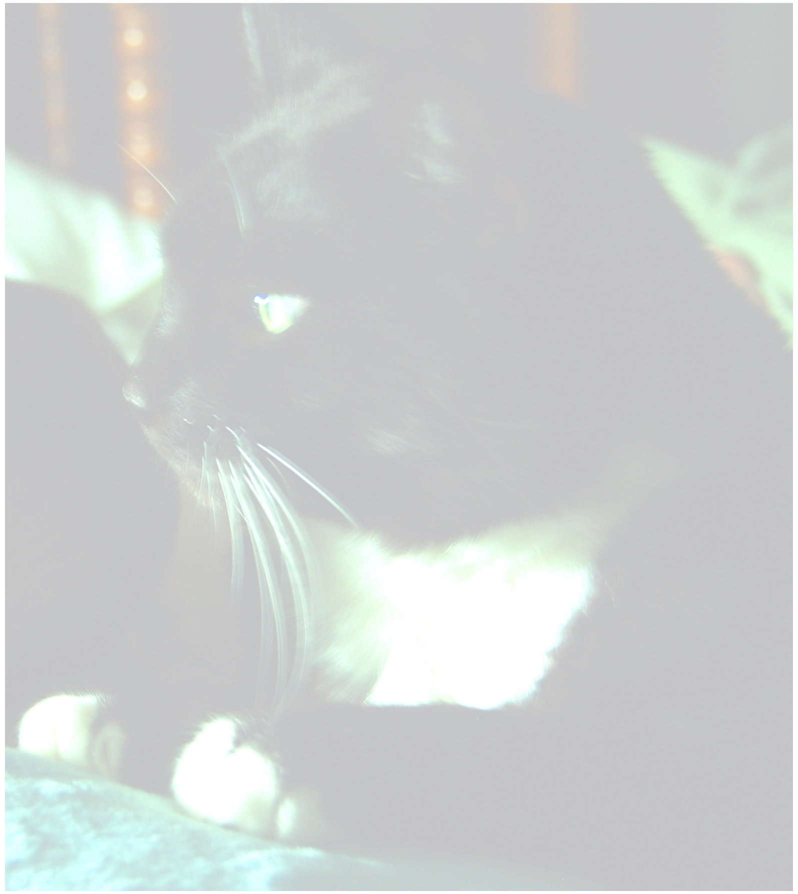
My result for Billy (Spring 20+22)



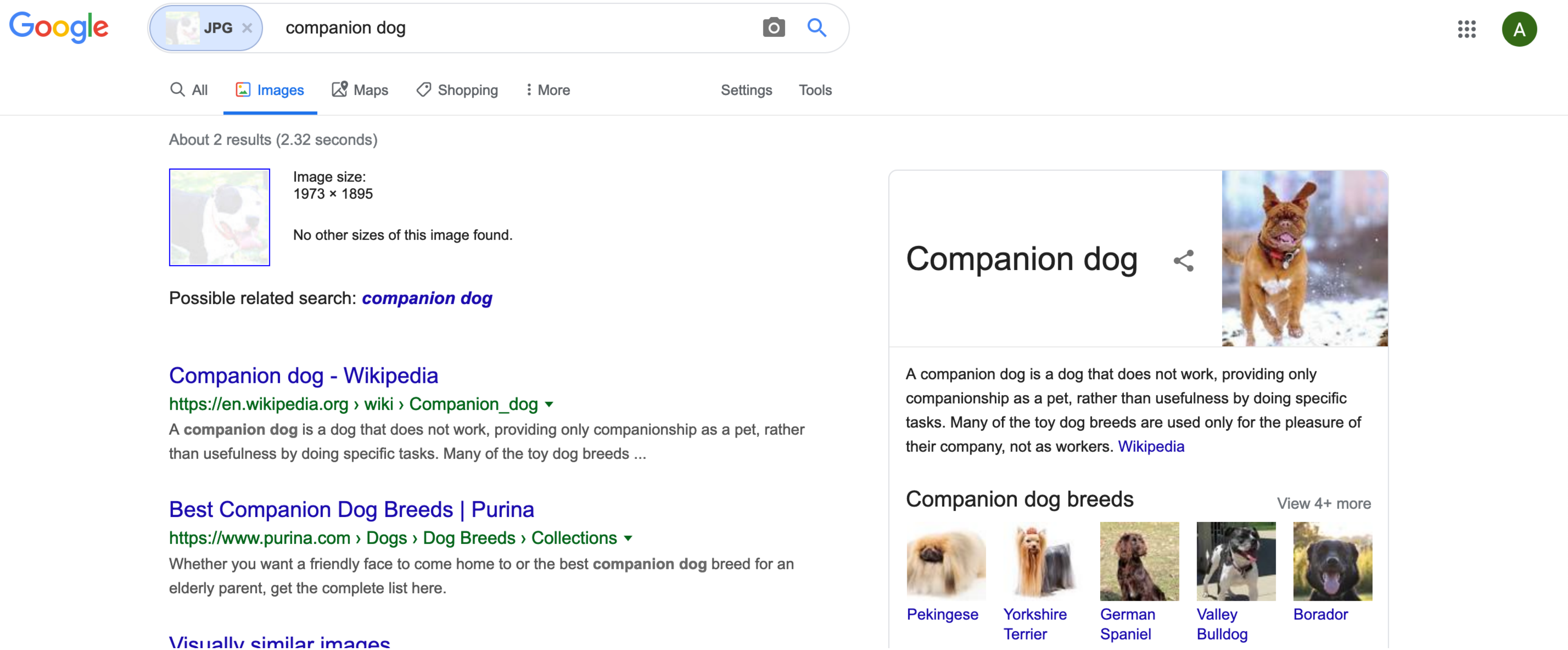
My result for Billy (Spring 23)



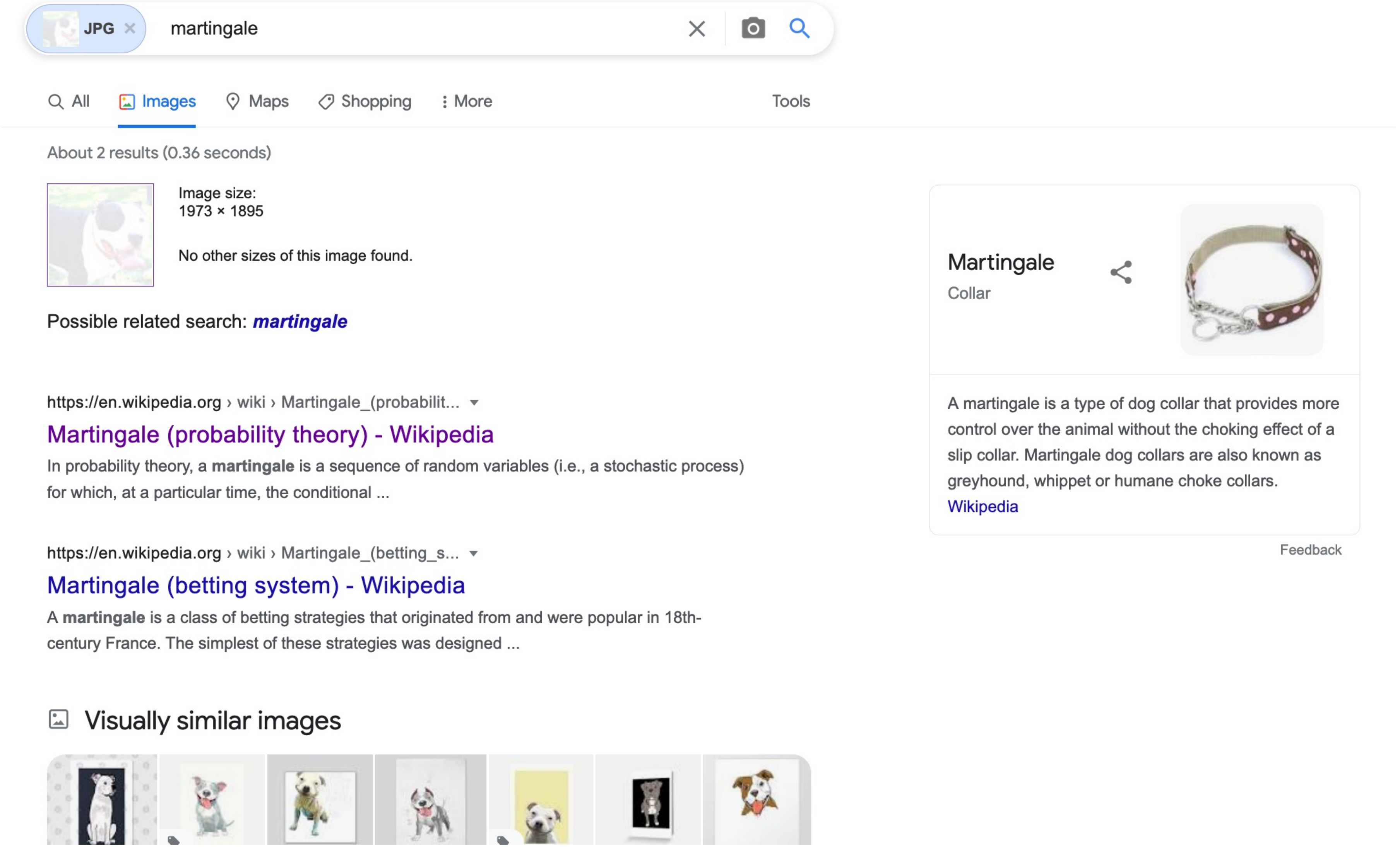
So cats vs dogs problem solved?



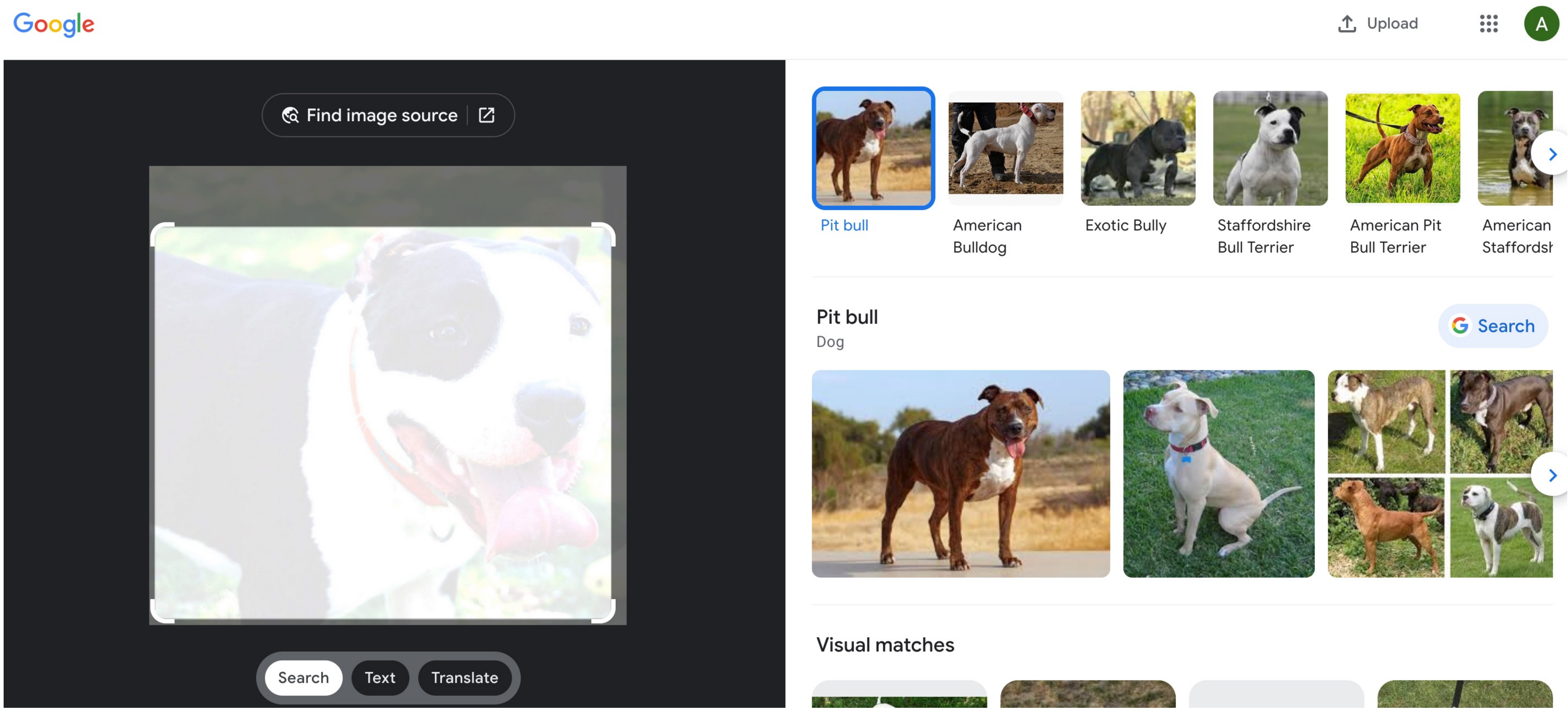
My result for modified Warren (Spring 20)



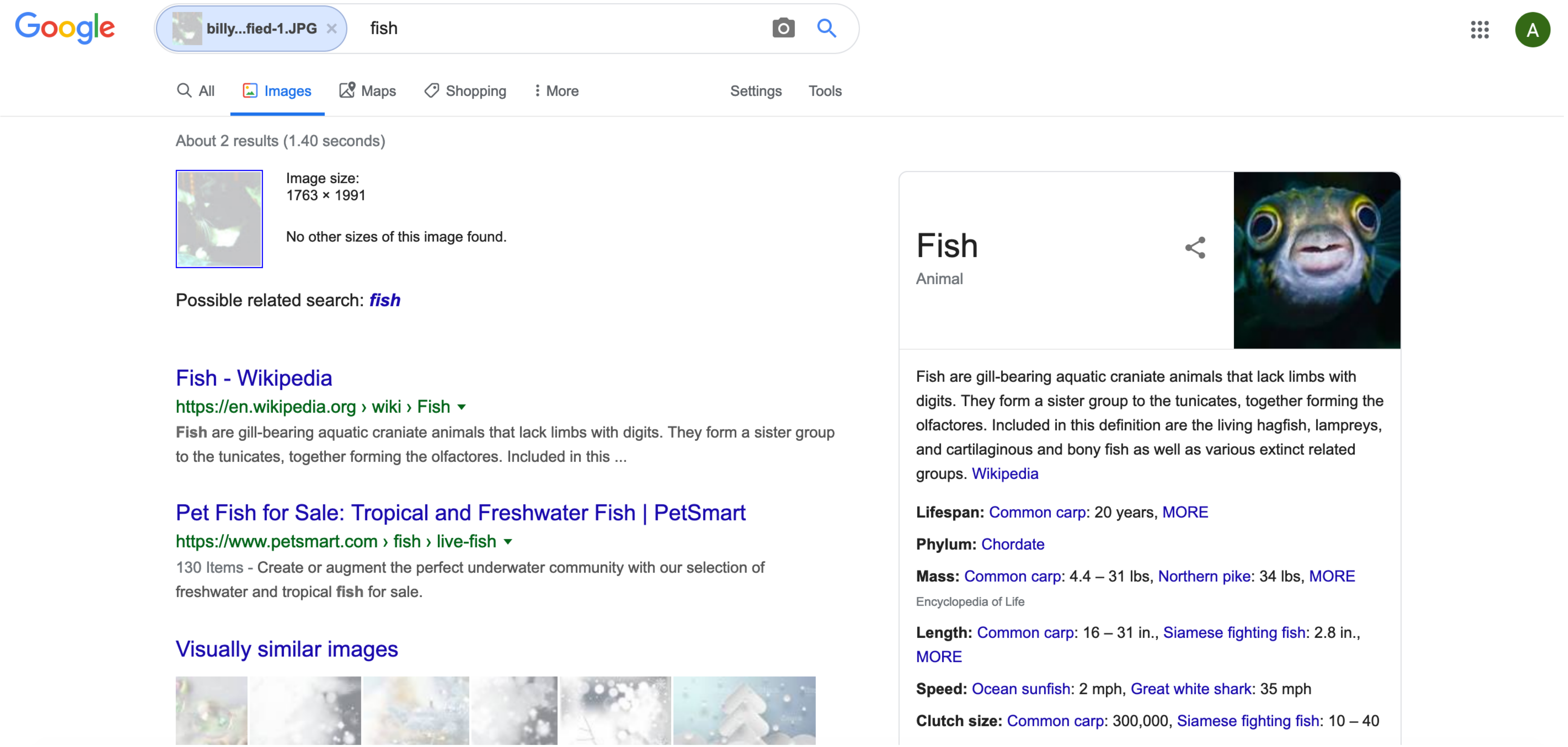
My result for modified Warren (Spring 22)



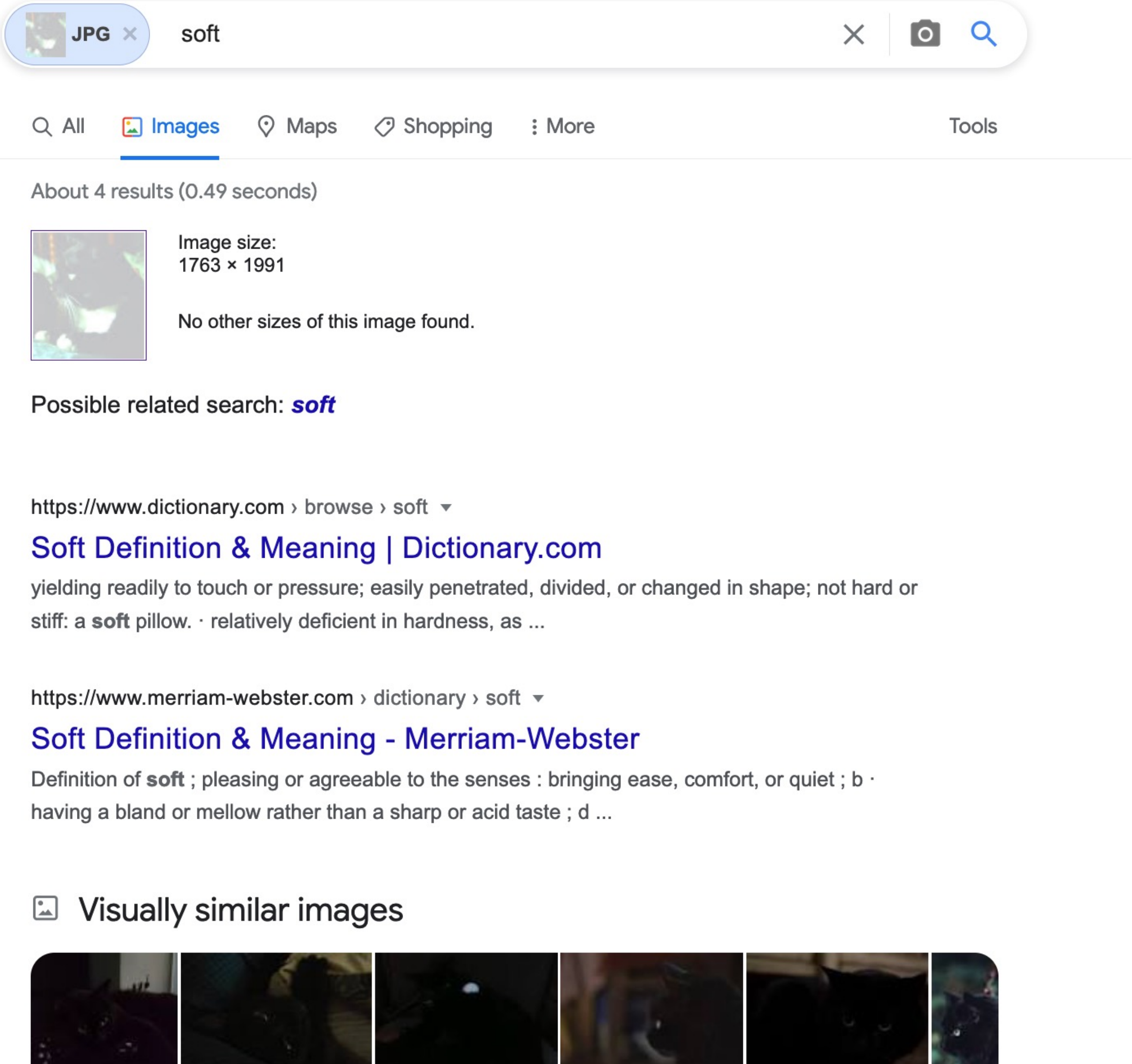
My result for modified Warren (Spring 23)



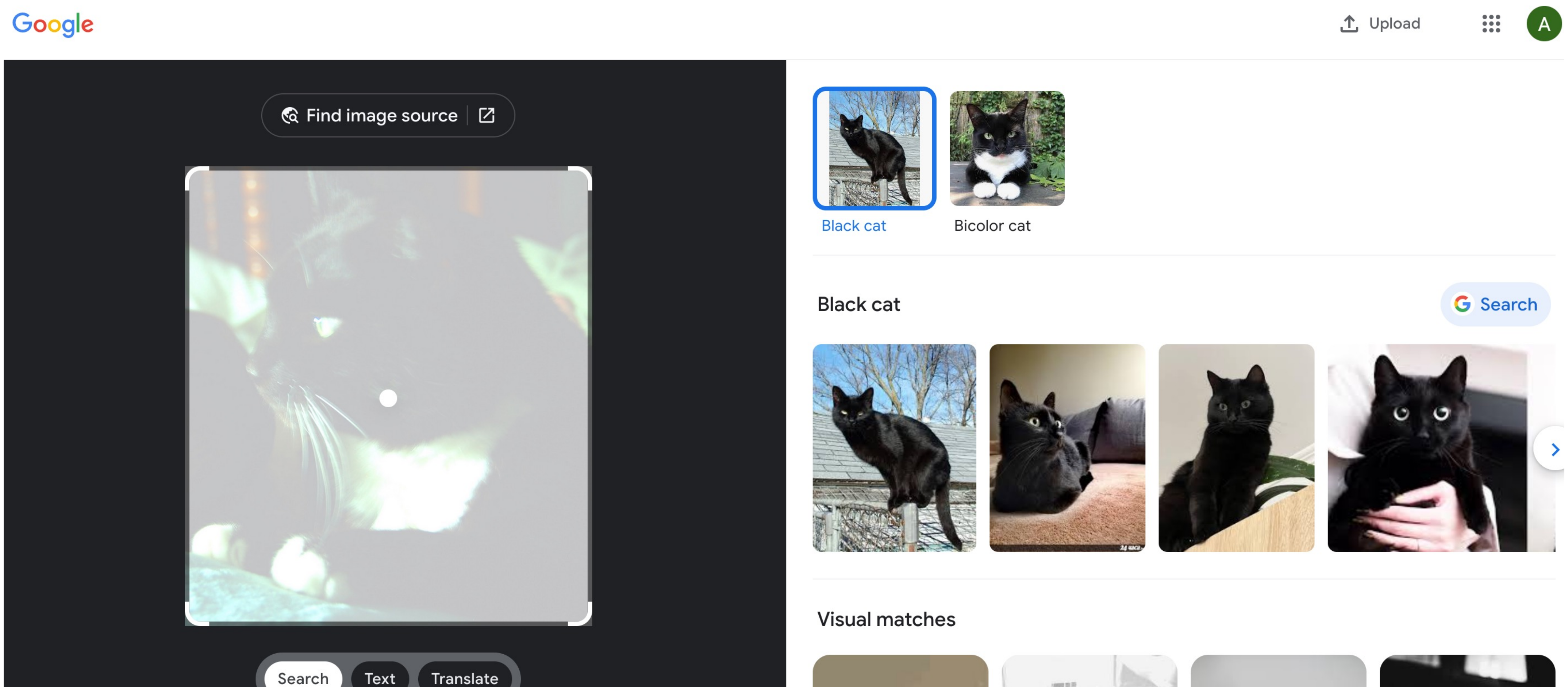
My result for modified Billy (Spring 20)



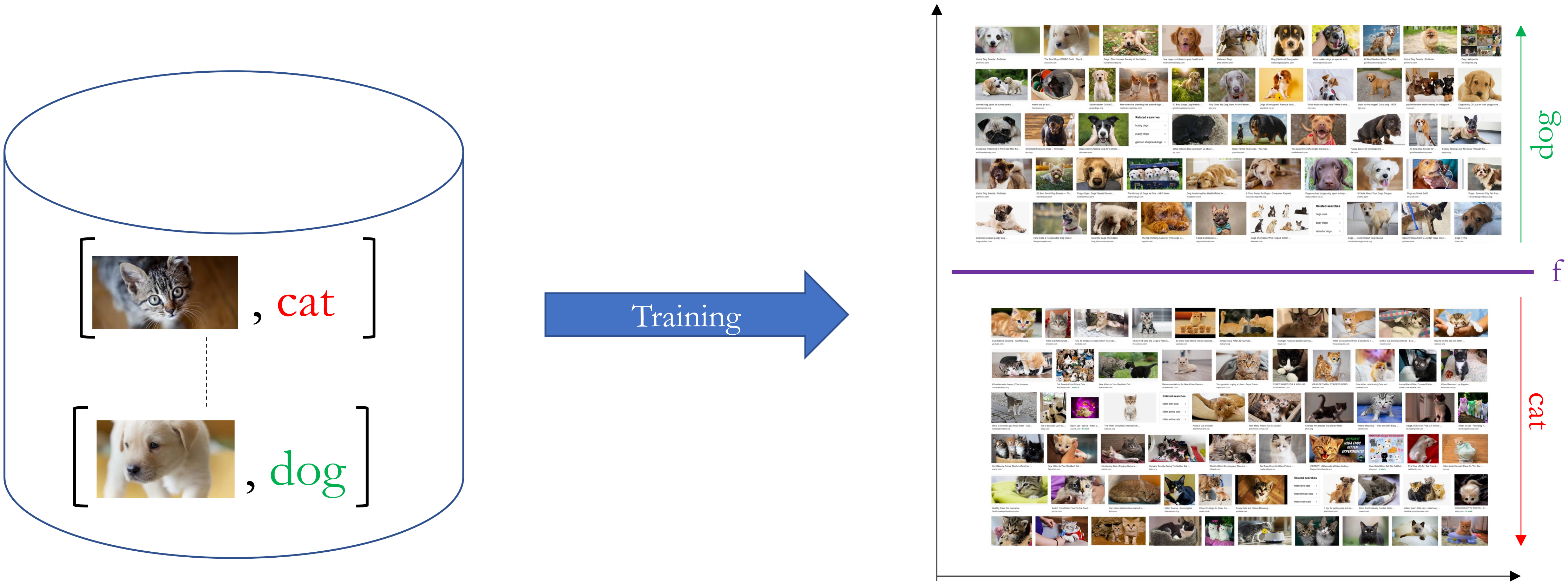
My result for modified Billy (Spring 22)



My result for modified Billy (Spring 23)

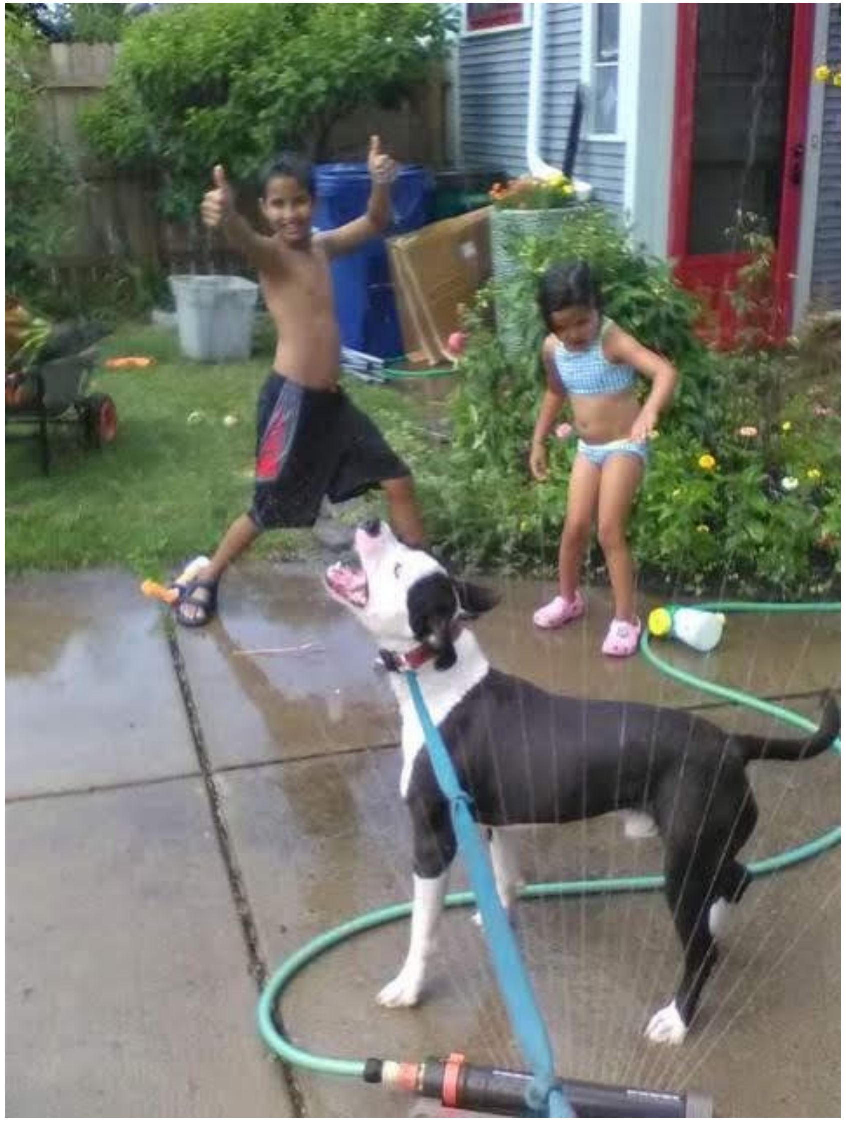


How does Google Images work?



When a new image comes in

Let’s do a quick break



Impossibilities in Computing

What do you think when you hear impossible?

In the context of computational problems, what does an impossible problem (that is defined *mathematically*) mean to you?

Second interpretation

It is possible to precisely define the problem but there does not exist *any* solution

Any example?

Arrow's Impossibility Theorem

Arrow's impossibility theorem

文A 22 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Arrow's impossibility theorem, the **general possibility theorem** or **Arrow's paradox** is an impossibility theorem in social choice theory that states that when voters have three or more distinct alternatives (options), no ranked voting electoral system can convert the **ranked preferences** of individuals into a community-wide (complete and transitive) ranking while also meeting the specified set of criteria: *unrestricted domain*, *non-dictatorship*, *Pareto efficiency*, and *independence of irrelevant alternatives*. The theorem is often cited in discussions of voting theory as it is further interpreted by the Gibbard–Satterthwaite theorem. The theorem is named after economist and Nobel laureate Kenneth Arrow, who demonstrated the theorem in his doctoral thesis and popularized it in his 1951 book *Social Choice and Individual Values*. The original paper was titled "A Difficulty in the Concept of Social Welfare".[1]

In short, the theorem states that no rank-order electoral system can be designed that always satisfies these three "fairness" criteria:

- If every voter prefers alternative X over alternative Y, then the group prefers X over Y.
- If every voter's preference between X and Y remains unchanged, then the group's preference between X and Y will also remain unchanged (even if voters' preferences between other pairs like X and Z, Y and Z, or Z and W change).
- There is no "dictator": no single voter possesses the power to always determine the group's preference.

Cardinal voting electoral systems are not covered by the theorem, as they convey more information than rank orders.[2][3] Gibbard's theorem and the Duggan–Schwartz theorem show that strategic voting remains a problem. The axiomatic approach Arrow adopted can treat all conceivable rules (that are based on preferences) within one unified framework. In that sense, the approach is qualitatively different from the earlier one in voting theory, in which rules were investigated one by one. One can therefore say that the contemporary paradigm of social choice theory started from this theorem.[4]

The practical consequences of the theorem are debatable. Arrow has said: "Most systems are not going to work badly all of the time. All I proved is that all can work badly at times."[5] When asked what he would change about US elections, he said, "The first thing that I'd certainly do is go to a system where people ranked all the candidates."[6] Arrow's impossibility theorem does not apply to multi-winner voting such as proportional representation.

Let’s do another example: fairness definition

Have you heard of COMPAS?

COMPAS (software)

From Wikipedia, the free encyclopedia

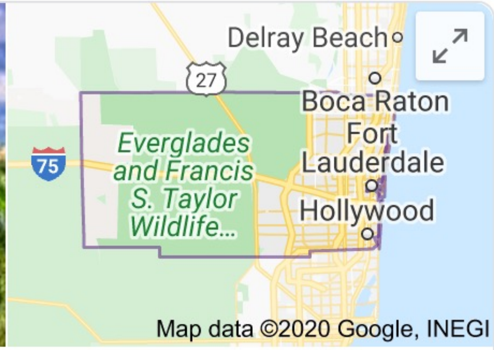
COMPAS, an acronym for Correctional Offender Management Profiling for Alternative Sanctions, is a case manag
Equivant) used by U.S. courts to assess the likelihood of a defendant becoming a recidivist.[1][2]

COMPAS has been used by the U.S. states of New York, Wisconsin, California, Florida's Broward County, and oth

Contents [hide]

1 Risk Assessment
2 Critiques and legal rulings
3 Accuracy
4 Further reading
5 See also
6 References

Risk Assessment [edit]



Broward County

County in Florida

Broward County is a county in southeastern Florida, US. According to a 2018 census report, the county had a population of 1,951,260, making it the second-most populous county in the state of Florida and the 17th-most populous county in the United States. The county seat is Fort Lauderdale. Wikipedia

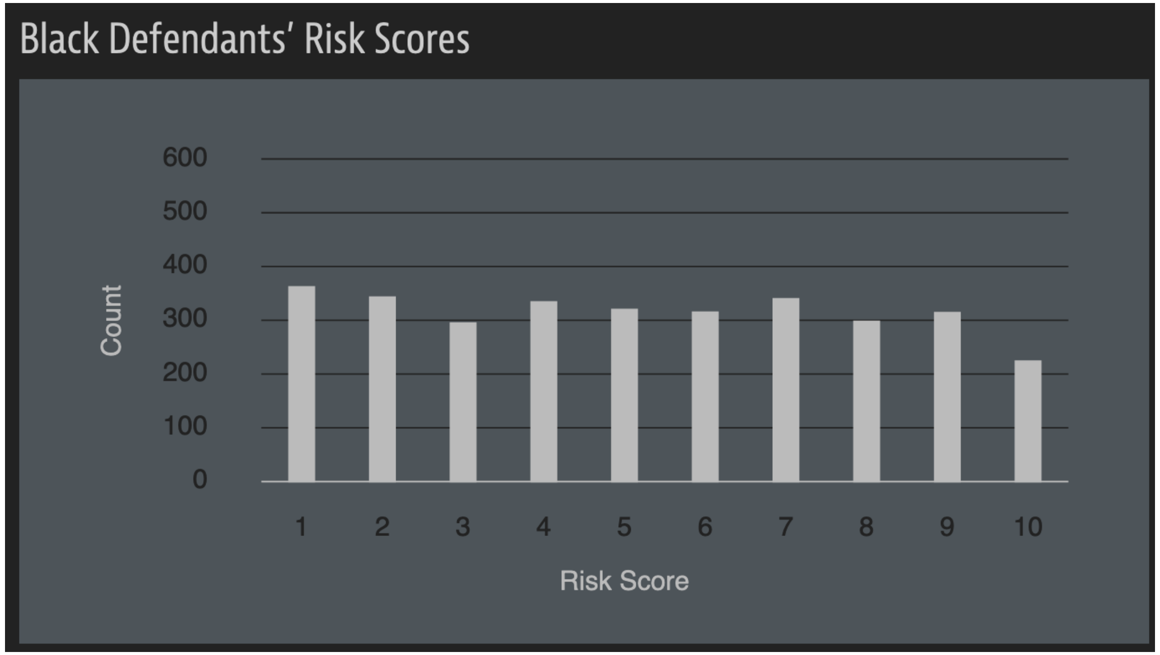
Incorporated cities: 24

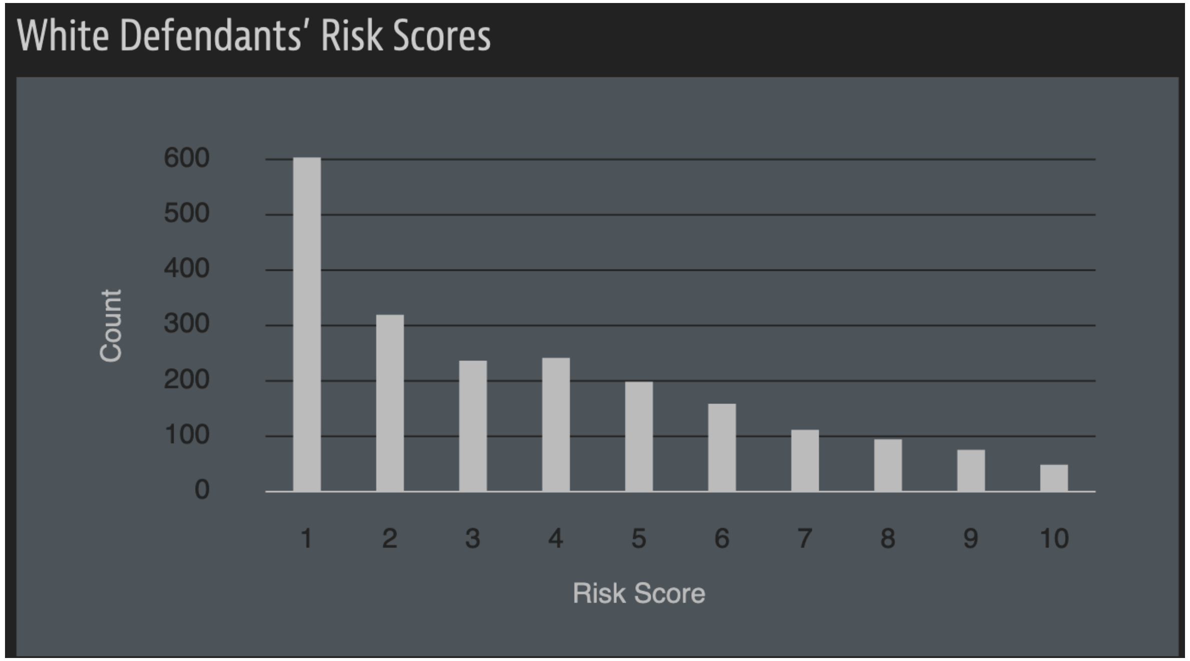
Population: 1.936 million (2017)

Mayor: Mark D. Bogen


ProPublica
Donate
Machine Bias
There's software used across the country to predict future criminals. And it's biased against blacks.
by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica
May 23, 2016

A sample of their result





False Positives, False Negatives, and False Analyses: A Rejoinder to "Machine Bias: There's Software Used Across the Country to Predict Future Criminals. And It's Biased Against Blacks."

Anthony W. Flores
California State University, Bakersfield
Kristin Bechtel
Crime and Justice Institute at CRJ
Christopher T. Lowenkamp
Administrative Office of the United States Courts
Probation and Pretrial Services Office

Both are correct….

How can that be????

Second interpretation

It is possible to precisely define the problem but there does not exist *any* solution

Solve an approximate version of the impossible problem

DOI:10.1145/3587930

Standards for fair decision making could help us develop algorithms that comport with our consensus views; however, algorithmic fairness has its limits.

BY MANISH RAGHAVAN

What Should We Do when Our Ideas of Fairness Conflict?

force us to re-examine the broader contexts within which algorithms are deployed. Here, we survey these responses and discuss their implications for the use of algorithms in decision making.

We are constantly faced with decisions in our daily lives. Some appear fairly inconsequential: an ad shown before the next video you watch or the sequence of posts on your social media feed. Others can change our lives—for example, whether we get a certain job or are approved for a loan. Algorithms play a growing role in these types of decisions. In response, a nascent field has formed, bridging disciplines such as computer science, economics, sociology, and legal studies in an effort to understand the impact of algorithmic decision making on society.[34]

One key area within this field considers fair decision making. When algorithms are used to make or assist with consequential decisions, how do we ensure that they do so fairly? This question is particularly salient when it comes to machine learning and other data-driven tools, where we might expect algorithms trained on data produced by humans to inherit the same biased and discriminatory behavior that humans exhibit. Researchers and practitioners have begun developing tools to address concerns over these behaviors, often using phrases like "algorithmic fairness" or "fairness in machine learning" to describe their efforts.

Impossibilities in Computing

What do you think when you hear impossible?

In the context of computational problems, what does an impossible problem (that is defined *mathematically*) mean to you?

Third interpretation

It is possible to precisely define the problem that has a solution but COMPUTING the solution is impossible/very hard

Any example?

Case 3.1

It is possible to precisely define the problem that has a solution but COMPUTING the solution is impossible (period)

Meta Q: Halting Problem

Input: A program P

Output: Yes if P terminates on all possible inputs
No otherwise

Let A be a program that solves the Halting problem on all inputs

```
def add (a,b):
    c = a+b
    return c
```

→ **A**

Yes, if on *every*

```
def add (a,b):
    c = a+b
    return c
```

returns *some* c input (a , b)

No, if *any* c on *some*

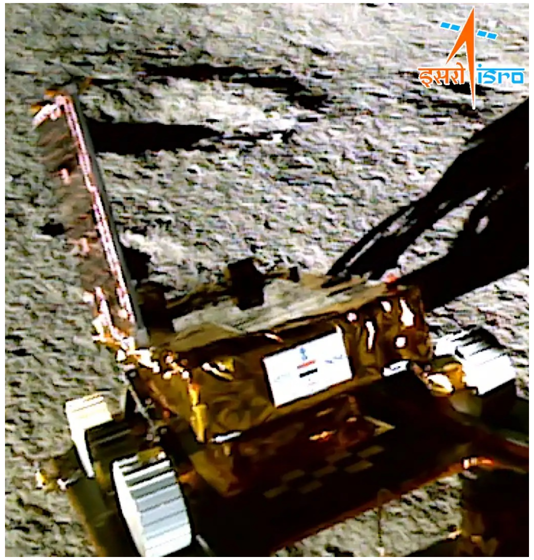
```
def add (a,b):
    c = a+b
    return c
```

doesn't return input (a , b)

What if you had this "magic box" A?

Input: A program P

Output: Yes if P terminates on all possible inputs
No otherwise



https://www.isro.gov.in/chandrayaan3_gallery.html

A solves the Halting problem on all inputs

Application 1: Take your programming question solution and feed it to A!

Chandrayaan 3

India Moon Landing | A Successful Mission | Videos and Photos | India's Path to Breakthroughs | Moon Landings and Crashes

'India Is on the Moon': Lander's Success Moves Nation to Next Space Chapter

The Chandrayaan-3 mission makes India the first country to reach the lunar south polar region in one piece and adds to the achievements of the country's homegrown space program.

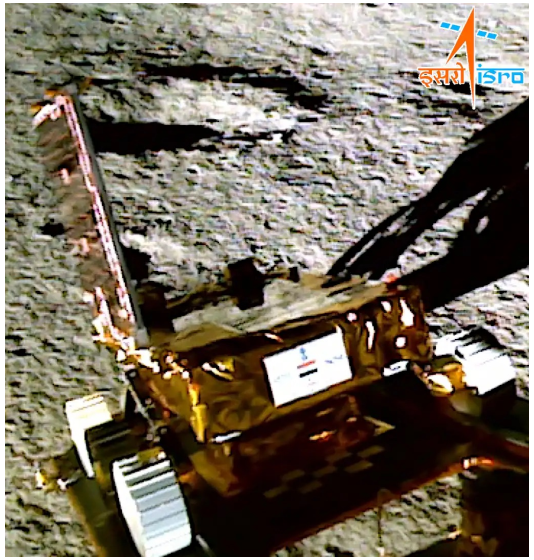
Share full article | 608

Schoolchildren watching a live feed of the Chandrayaan-3 mission to the moon celebrated its success in Guwahati, India, on Wednesday. Anupam Nath/Associated Press

What if you had this "magic box" A?

Input: A program P

Output: Yes if P terminates on all possible inputs
No otherwise



https://www.isro.gov.in/chandrayaan3_gallery.html

A solves the Halting problem on all inputs

Application 1: Take your programming question solution and feed it to A!

Application 2: Use A to make sure that code in rovers in Chandrayaan 3 will never hang!

My favorite CSE 199 example

Solve Collatz conjecture

Statement of the problem

Consider the following operation on an arbitrary positive integer:

- If the number is even, divide it by two.
- If the number is odd, triple it and add one.

In modular arithmetic notation, define the function f as follows:

$$f(n) = \begin{cases} n/2 & \text{if } n \equiv 0 \pmod{2}, \\ 3n+1 & \text{if } n \equiv 1 \pmod{2}. \end{cases}$$

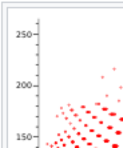
Now form a sequence by performing this operation repeatedly, beginning with any positive integer, and taking the result at each step as the input at the next.

In notation:

$$a_i = \begin{cases} n & \text{for } i = 0, \\ f(a_{i-1}) & \text{for } i > 0 \end{cases}$$

(that is: a_i is the value of f applied to n recursively i times; $a_i = f^i(n)$).

The Collatz conjecture is: *This process will eventually reach the number 1, regardless of which positive integer is chosen initially.*



https://en.wikipedia.org/wiki/Collatz_conjecture

Wouldn't it be nice to have this magic box A?

No such A can exist!

The paper ended Hilbert's plan to automatize all of mathematics

Pic from Wikipedia

This paper started CSE!!

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

Case 3.1

It is possible to precisely define the problem that has a solution but COMPUTING the solution is impossible (period)

Solve the problem for "real world" cases

Model checking

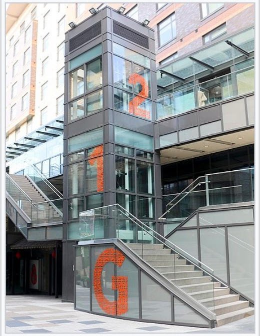
文A 13 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

This article is about checking of models in computer science. For the checking of models in statistics, see statistical model validation.

In computer science, **model checking** or **property checking** is a method for checking whether a finite-state model of a system meets a given specification (also known as correctness). This is typically associated with hardware or software systems, where the specification contains liveness requirements (such as avoidance of livelock) as well as safety requirements (such as avoidance of states representing a system crash).



Elevator control software can be model-checked to verify both safety properties, like *"The cabin never moves with its door open"*,[1] and liveness properties, like *"Whenever the n^{th} floor's call button is pressed, the cabin will eventually stop at the n^{th} floor and open the door"*.

In order to solve such a problem algorithmically, both the model of the system and its specification are formulated in some precise mathematical language. To this end, the problem is formulated as a task in logic, namely to check whether a structure satisfies a given logical formula. This general concept applies to many kinds of logic and many kinds of structures. A simple model-checking problem consists of verifying whether a formula in the propositional logic is satisfied by a given structure.

Overview [edit]

Property checking is used for verification when two descriptions are not equivalent. During refinement, the specification is complemented with details that are unnecessary in the higher-level specification. There is no need to verify the newly introduced properties against the original specification since this is not possible. Therefore, the strict bi-directional equivalence check is relaxed to a one-way property check. The implementation or design is regarded as a model of the system, whereas the specifications are properties that the model must satisfy.[2]

An important class of model-checking methods has been developed for checking models of hardware and software designs where the specification is given by a temporal logic formula. Pioneering work in temporal logic specification was done by Amir Pnueli, who received the 1996 Turing award for "seminal work introducing temporal logic into computing science".[3] Model checking began with the pioneering work of E. M. Clarke, E. A. Emerson,[4][5][6] by J. P. Queille, and J. Sifakis.[7] Clarke, Emerson, and Sifakis shared the 2007 Turing Award for their seminal work founding and developing the field of model checking.[8][9]

Case 3.2.1

It is possible to precisely define the problem that has a solution but COMPUTING the solution efficiently with **current** technology is very hard

Any example?

Quantum Computing

Quantum computing

32 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

A **quantum computer** is a computer that takes advantage of quantum mechanical phenomena.

IBM Q System One, a quantum computer with 20 superconducting qubits[1]

At small scales, physical matter exhibits properties of both particles and waves, and quantum computing leverages this behavior, specifically quantum superposition and entanglement, using specialized hardware that supports the preparation and manipulation of quantum states.

Classical physics cannot explain the operation of these quantum devices, and a scalable quantum computer could perform some calculations exponentially faster (with respect to input size scaling)[2] than any modern "classical" computer. In particular, a large-scale quantum computer could break widely used encryption schemes and aid physicists in performing physical simulations; however, the current state of the art is largely experimental and impractical, with several obstacles to useful applications. Moreover, scalable quantum computers do not hold promise for many practical tasks, and for many important tasks quantum speedups are proven impossible.

The basic unit of information in quantum computing is the qubit, similar to the bit in traditional digital electronics. Unlike a classical bit, a qubit can exist in a superposition of its two "basis" states. When measuring a qubit, the result is a probabilistic output of a classical bit, therefore making quantum computers nondeterministic in general. If a quantum computer manipulates the qubit in a particular way, wave interference effects can amplify the desired measurement results. The design of quantum algorithms involves creating procedures that allow a quantum computer to perform calculations efficiently and quickly.

Physically engineering high-quality qubits has proven challenging. If a physical qubit is not sufficiently isolated from its environment, it suffers from quantum decoherence, introducing noise into calculations. Paradoxically, perfectly isolating qubits is also undesirable because quantum computations typically need to initialize qubits, perform controlled qubit interactions, and measure the resulting quantum states. Each of those operations introduces errors and suffers from noise, and such inaccuracies accumulate.

Why do folks care about quantum computing?

Remember RSA/factoring?

Quantum Computer can "easily" factor

Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored. These two problems are generally considered hard on a classical computer and have been used as the basis of several proposed cryptosystems. (We thus give the first examples of quantum cryptanalysis.)

1 Introduction

Since the discovery of quantum mechanics, people have found the behavior of the laws of probability in quantum mechanics counterintuitive. Because of this behavior, quantum mechanical phenomena behave quite differently than the phenomena of classical physics that we are used

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e., memory). The field of analysis of algorithms considers the asymptotic demands that algorithms make for these resources as a function of the problem size. Theoretical computer scientists generally classify algorithms as efficient when the number of steps of the algorithms grows as a polynomial in the size of the input. The class of problems which can be solved by efficient algorithms is known as P. This classification has several nice properties. For one thing, it does a reasonable job of reflecting the performance of algorithms in practice (although an algorithm whose running time is the tenth power of the input size, say, is not truly efficient). For another, this classification is nice theoretically, as different reasonable machine models

So RSA is dead?

Records for efforts by quantum computers [edit]

The largest number reliably factored[clarification needed] by Shor's algorithm is 21 which was factored in 2012.[23] 15 had previously been factored by several labs.

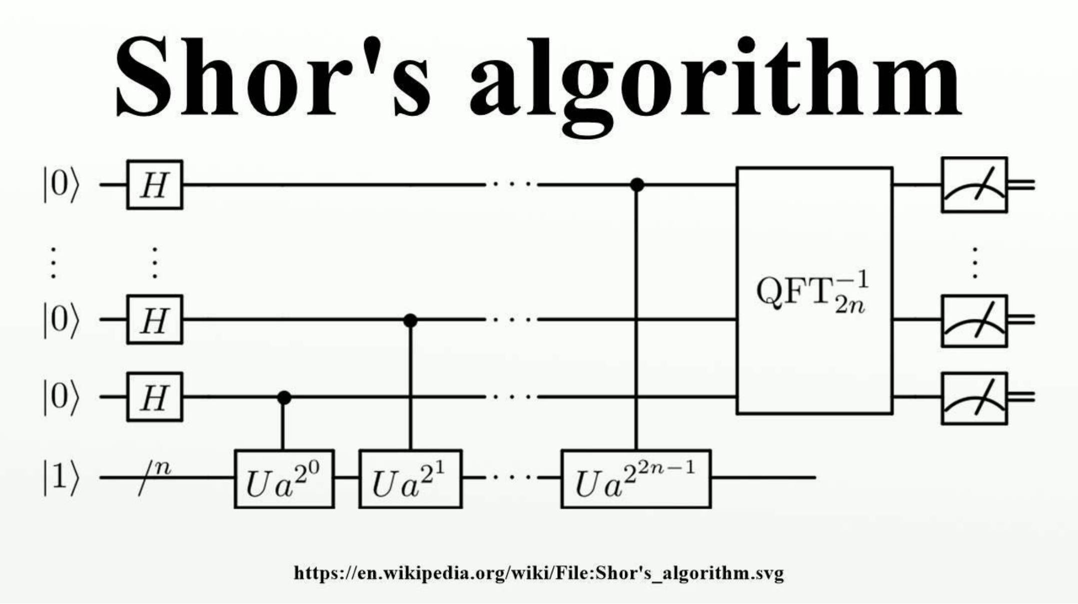
In April 2012, the factorization of 143 = 13 × 11 by a room-temperature (300 K) NMR adiabatic quantum computer was reported by a group led by Xinhua Peng.[24] In November 2014 it was discovered that the 2012 experiment had in fact also factored much larger numbers without knowing it.[clarification needed][25][26] In April 2016 the 18-bit number 200,099 was factored using quantum annealing on a D-Wave 2X quantum processor.[27] Shortly after, 291 311 was factored using NMR at higher than room temperature.[28] In late 2019, Zapata computing claimed to have factored 1,099,551,473,989,[29] and in 2021 released a paper describing this computation.[30]. In 2024, Samer Rahmeh applied adiabatic quantum computation (AQC) to successfully factor prime numbers up to 201 digits (666 bit) which have been computed on the Dynex Neuromorphic Computing Cloud[31].

As such, claims of factoring with quantum computers have however been criticized for depending heavily on classical computation to reduce the number of qubits required.[32] [33] For example, the factorization of 1,099,551,473,989 relied on classical pre-processing to reduce the problem to a three-qubit quantum circuit.[30] Furthermore, the three numbers factored in this paper (200,099, 291,311, and 1,099,551,473,989) can easily be factored using Fermat's factorization method, requiring only 3, 1, and 1 iterations of the loop respectively.

Case 3.2.1

It is possible to precisely define the problem that has a solution but COMPUTING the solution efficiently with **current** technology is very hard

Solve the problem mathematically



« NAND now for something completely different

Quantum Computing Since Democritus Lecture 10: Quantum Computing »

Shor, I'll do it

I've been talking a lot recently about how quantum algorithms *don't* work. But last week JR Minkel, an editor at *Scientific American*, asked me to write a brief essay about how quantum algorithms *do* work, which he could then link to from *SciAm*'s website."OK!" I replied, momentarily forgetting about the $10^{10^{5000}}$ quantum algorithm tutorials that are already on the web. So, here's the task I've set for myself: to explain Shor's algorithm without using a single ket sign, or for that matter any math beyond arithmetic.

Alright, so let's say you want to break the RSA cryptosystem, in order to rob some banks, read your ex's email, whatever. We all know that breaking RSA reduces to finding the prime factors of a large integer N. Unfortunately, we also know that "trying all possible divisors in parallel," and then instantly picking the right one, isn't going to work. Hundreds of popular magazine articles notwithstanding, trying everything in parallel just isn't the sort of thing that a quantum computer can do. Sure, in some sense you can "try all possible divisors" — but if you then measure the outcome, you'll get a *random* divisor, which almost certainly won't be the one you

Case 3.2.2

It is possible to precisely define the problem that has a solution but implementing the solution efficiently in **current** world is hard

Any example?