

Quantum computing



Quantum computing



Let's rewind back to **1981**...



IBM 5150 PC

16 kB memory

\$1,565 (\$5,000 today)



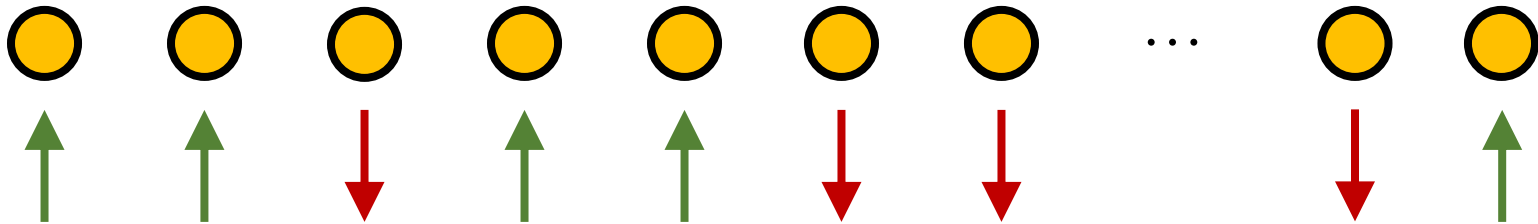
And physicists had a problem:

They couldn't simulate **quantum systems**



Suppose you have n electrons

Each electron can have a **spin**:  or 



So that's 2^n possible spin configurations

Quantum Simulation Problem

Given: a starting configuration of the n electrons

Goal: what will the system look like after time T ?

Best algorithm for this takes time 2^n !



1981 talk



Richard Feynman

"Nature isn't classical, dammit, and if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem because it doesn't look so easy."

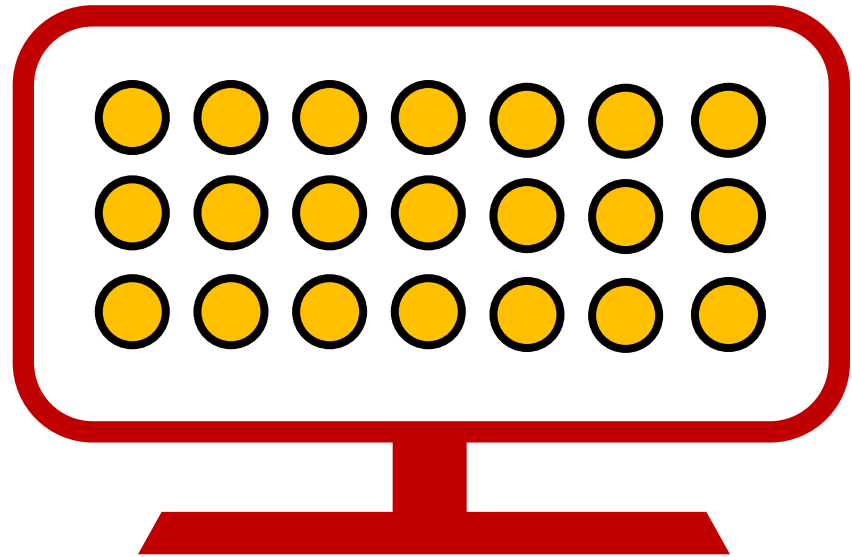
"Can you do it with a new kind of computer – a quantum computer? Now it turns out, as far as I can tell, that you can simulate this with a quantum system, with quantum computer elements. It's not a Turing machine, but a machine of a different kind."

1981 talk

1. Build a computer out of electrons
2. Make it programmable
3. Let the electrons simulate themselves



Richard Feynman



After this idea was introduced, people wondered:
how **powerful** are quantum computers?

Sure, they can simulate quantum physics quickly.
But **what else**?

5 algorithms:

1. Deutsch-Jozsa algorithm
2. Bernstein-Vazirani algorithm
3. Simon's algorithm
4. Shor's algorithm
5. Grover's algorithm

I will tell you about **these**

Deutsch-Jozsa problem (1992)



David Deutsch

Input: a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, either:

1. $f(x) = 0$ for all x
2. $f(x) = 1$ for all x
3. $f(x) = 0$ for half of x , 1 for other half

Goal: which case are we in?



Richard Jozsa

Deterministic algorithms

Have to check $2^{n-1} + 1$ values of $f(x)$.

Quantum algorithm

Solves it in time $O(n)$!

(Why is this a little unsatisfying?)

Bernstein-Vazirani problem (1992)



Ethan Bernstein

Input: a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ s.t.

$$f(x) = x_2 + x_5 + \cdots + x_{n-1} \pmod{2}$$

(some subset of the input)

Goal: which bits are in the sum?

Classical algorithm:

0. Look at $f(0, 0, 0, \dots 0)$

1. Look at $f(1, 0, 0, \dots 0)$

2. Look at $f(0, 1, 0, \dots 0)$

...

$n + 1$
looks at f



Umesh Vazirani

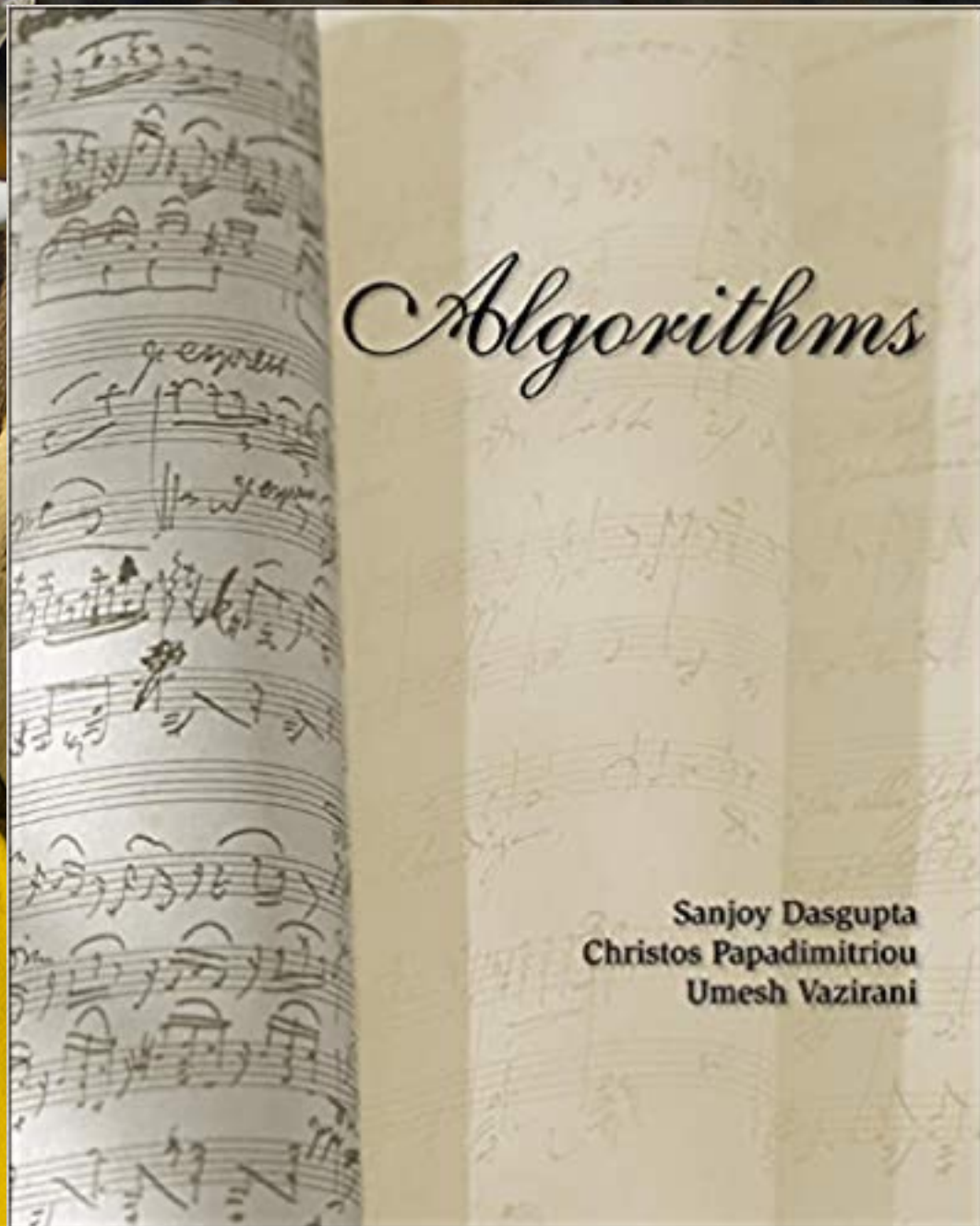
Quantum algorithm:

Only need to “quantum look” at f once!

UC Berkeley
prof



Umesh Vazirani



Shor's algorithm (1994)

Last class:

To factor an n -digit number, the best **classical** alg runs in time $O(e^{1.9 \cdot n^{1/3} \cdot \log(n)^{2/3}})$



Peter Shor

Shor's algorithm factors in $O(n^2)$ time
on a **quantum computer**

(also solves discrete log)

With a quantum computer,
we can **break** the RSA cryptosystem!

See: <https://www.youtube.com/watch?v=6qD9XEITpCE>

Grover's algorithm (1996)

Lecture 21:

Circuit-SAT is **NP**-complete
Best classical alg runs in $\mathbf{O}(2^n)$ time



Lov Grover

Grover's algorithm solves **Circuit-SAT**
in $\mathbf{O}(\sqrt{2^n}) = \mathbf{O}(1.414^n)$ time
on a **quantum computer**

(actually solves many other problems
with a **square-root-speedup**)

Quantum speedups

Three types:

1. "Shor-type speedups"

Pro: **Exponentially** faster than classical!

Con: Only for certain problems (e.g. factoring)

2. "Grover-type speedups"

Pro: Works for many problems!

Con: Only **polynomially** faster than classical

3. "Physics simulation speedups"

Pro: **Exponentially** faster than classical!

Con: Only for certain problems (e.g. physics)

Quantum speedups

Q: What is behind these speed-ups?

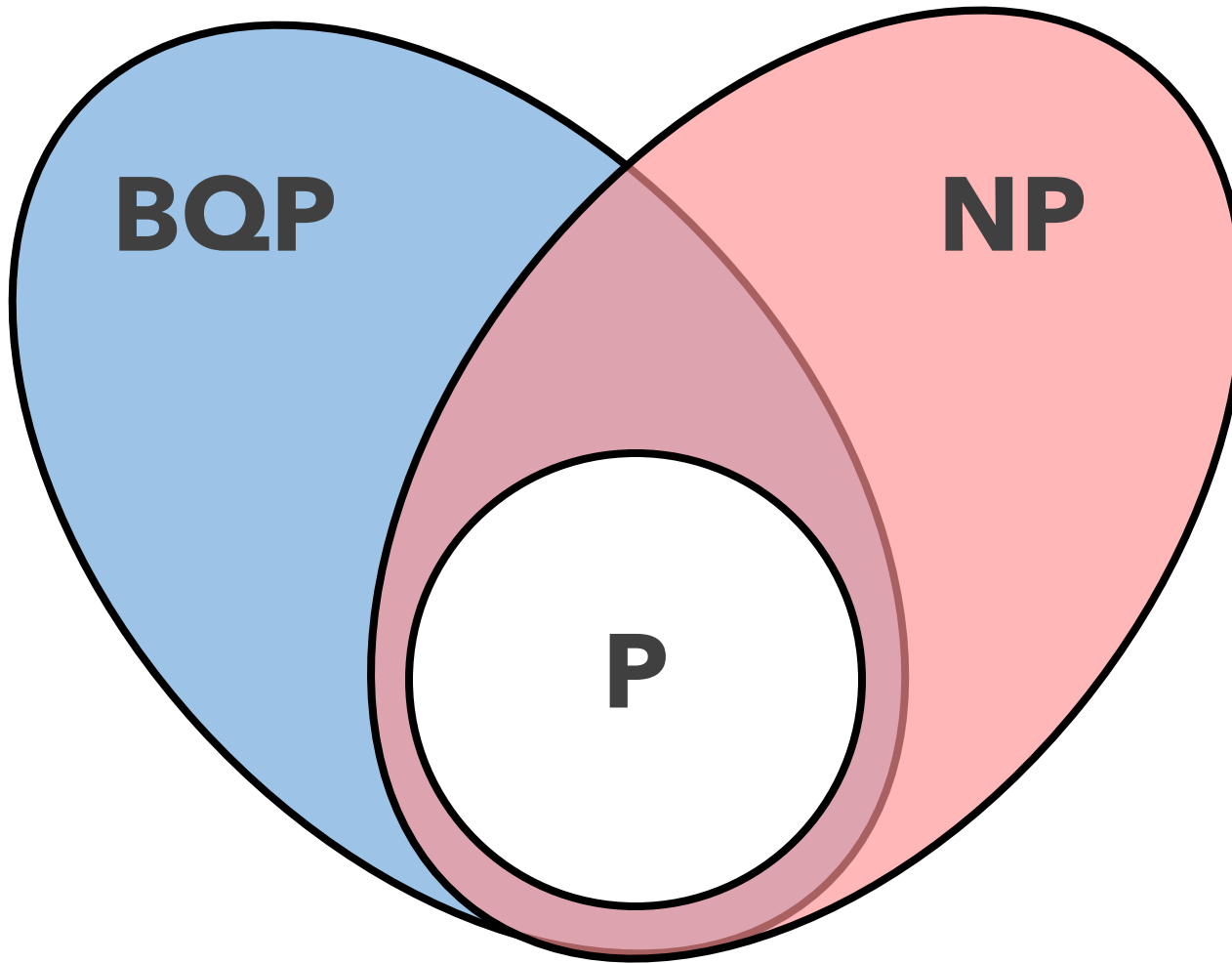
Why do these
outperform classical
algorithms?

- 
1. Deutsch-Jozsa algorithm
 2. Bernstein-Vazirani algorithm
 3. Simon's algorithm
 4. Shor's algorithm
 5. Grover's algorithm

A: Quantum computers can do
ridiculously fast **Fourier transforms**.

(Recall **lecture 4**: fast Fourier transform
helps multiplying polynomials!)

BQP = the set of all problems efficiently solvable
on a **quantum computer**



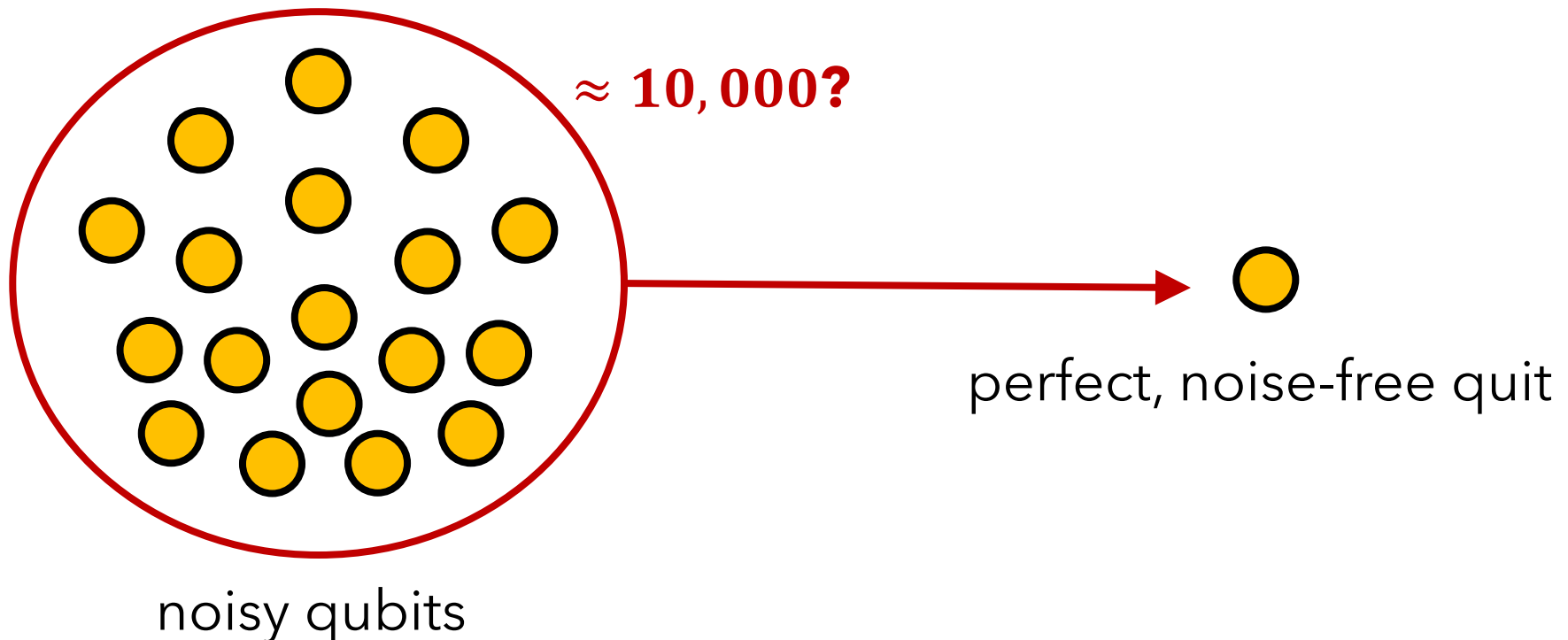
Quantum computers **not believed**
to solve **NP**-complete problems efficiently!

Building quantum computers

Building quantum computers is really, really tough!

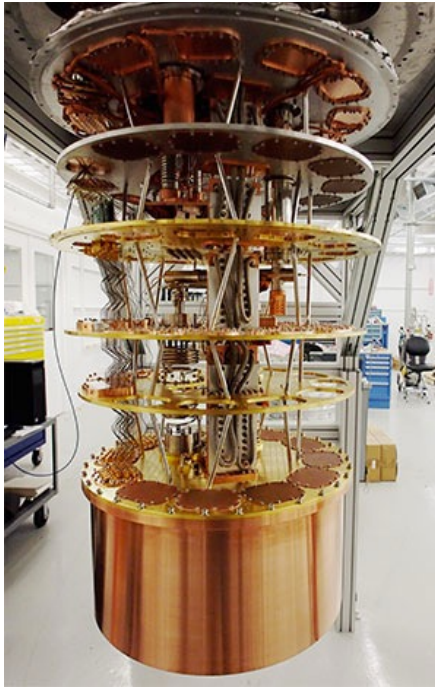
Key challenge: **Noise!** Even the smallest amount of noise completely **ruins** a quantum computation.

Solution: Error correction



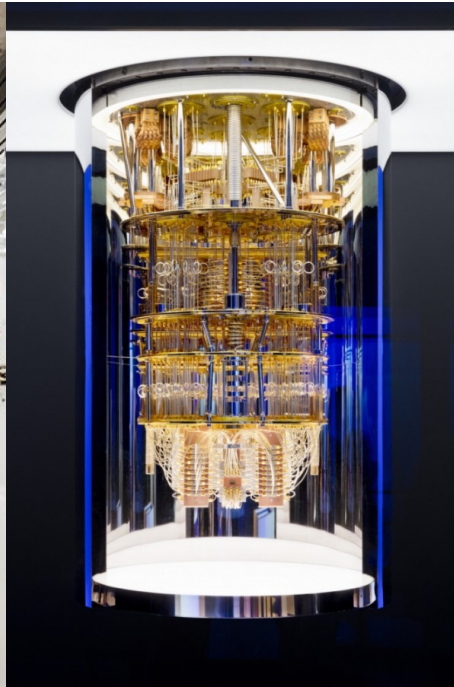
2011: Factored $N = 21 = 3$ using two qubits

Today:



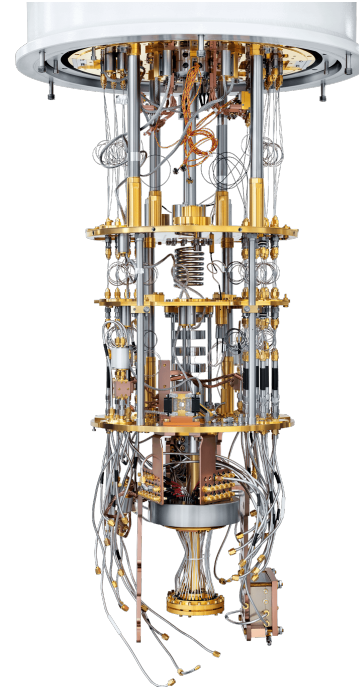
Google

51 qubits



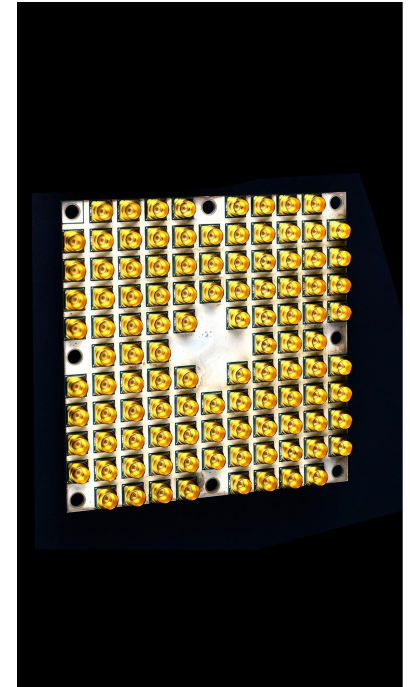
IBM

65 qubits



rigetti

31 qubits



intel

49 qubits

(Noisy qubits. Need 10,000 for 1 clean qubit!)

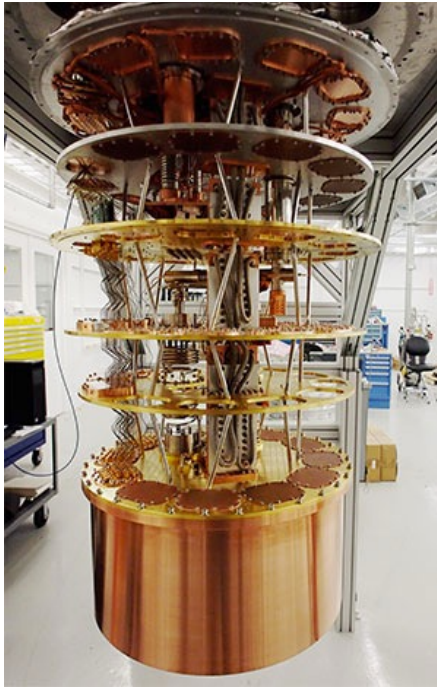
2011: Factored $N = 21 = 3$ using two qubits

Today:

Hello quantum world! Google publishes landmark quantum supremacy claim

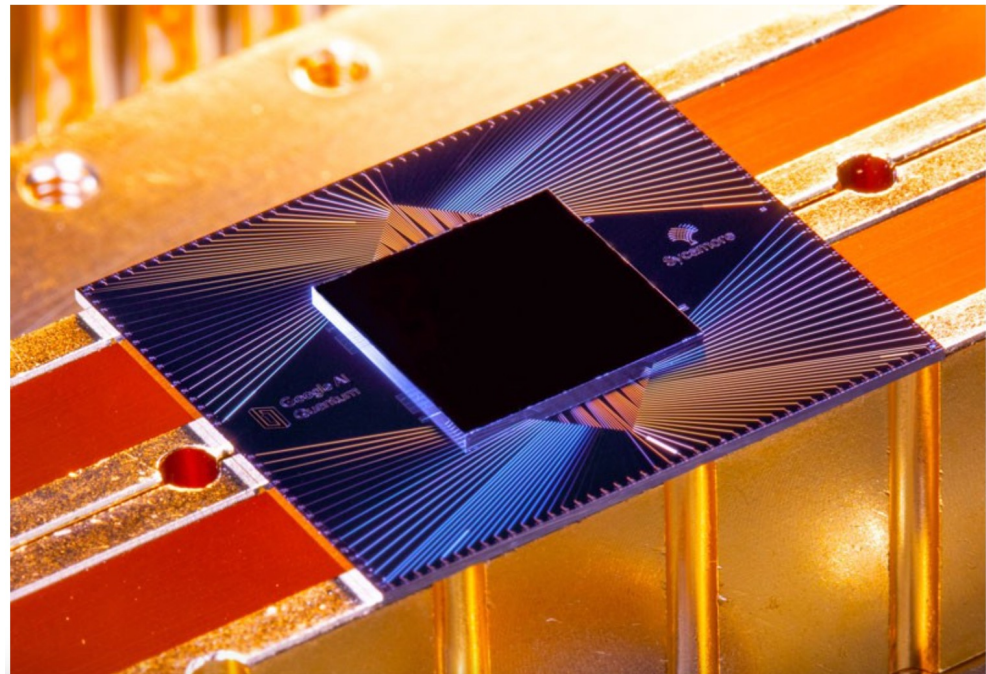
The company says that its quantum computer is the first to perform a calculation that would be practically impossible for a classical machine.

[Elizabeth Gibney](#)



Google

51 qubits



Now let's see
a quantum algorithm