TE 594: Quantum Computing & Information Processing

Prof. Jerry John Kponyo
&
Dr. Justice Owusu Agyemang

Course Outline

- Quantum Information: The mathematical formalism of qubits, operations, measurements. Superposition and entanglement. Teleportation etc.
- Quantum Algorithms: Quantum gates and circuits. Deutsch's algorithm, the Deutsch-Jozsa algorithm, Grover's algorithm, Shor's factoring algorithm etc.
- Cryptography: Bit-commitment, Quantum key distribution.
- Nonlocality: CHSH game, etc.
- Special Topics: Entropy and compression, quantum communication complexity, etc.

Prerequisites

- Introduction to linear algebra.
- Basic knowledge of probability theory.
- Python programming.

Recommended Books

- Terry Rudolph, "Q is for Quantum", 2017.
- Thomas G. Wong, "Introduction to Classical and Quantum Computing", 1st edition, 2022.
- Hassi Norlen, "Quantum Computing in Practice with Qiskit and IBM Quantum Experience - Practical recipes for quantum computer coding at the gate and algorithm level with Python", Packt, 2020.

Resource Link: https://github.com/jayluxferro/TE-594-QCIP

Course Grading

Assignments	20%
Semester Project	40%
End of Semester Exams	40%
Total	100%

Classical Information and Computation

Classical Information and Computation

Concept	Classical
Fundamental Unit	Bit
Gates	Logic Gates
Gates Reversible	Sometimes
Universal Gate Set (Example)	{NAND}
Programming Language (Example)	Verilog
Algebra	Boolean
Error Correcting Code (Example)	Repetition Code
Complexity Class	P
Strong Church-Turing Thesis	Supports

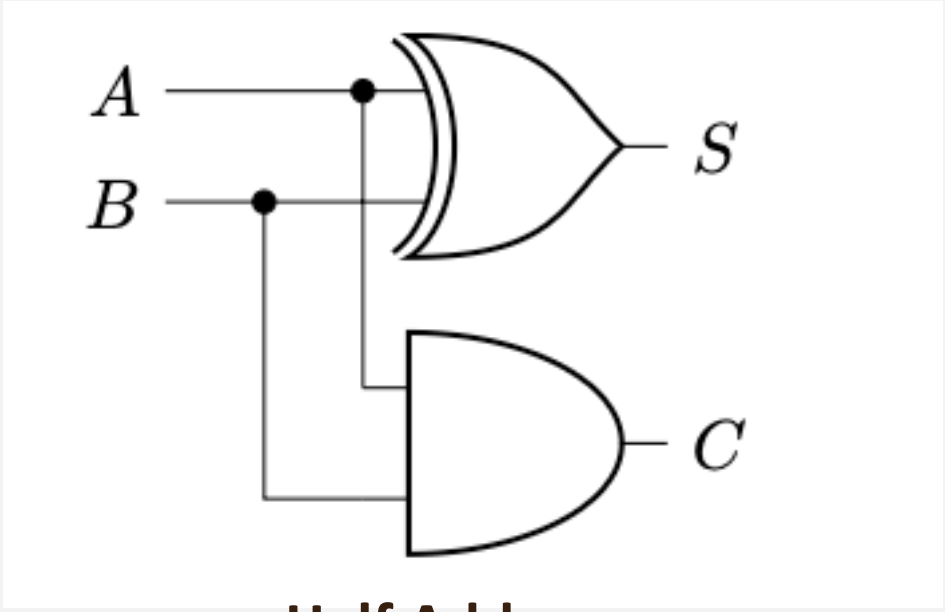
Classical Information and Computation

Recap of basic concepts:

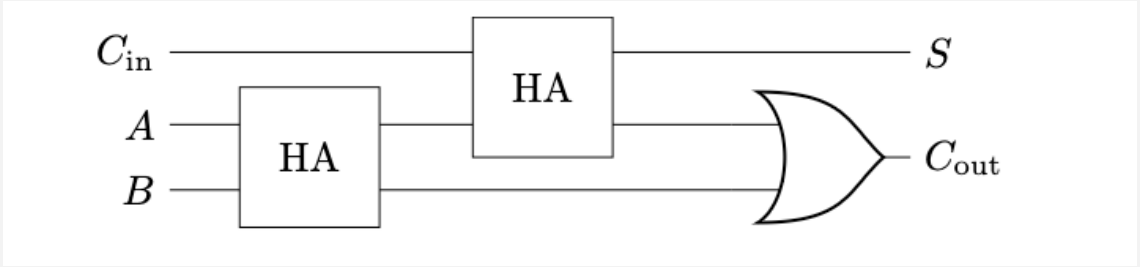
1. Bits
2. Information Encoding
3. ASCII
4. Logic Gates
5. Adders & Verilog
6. Circuit Complexity
7. Reversible Logic Gates
8. Error Correction and Detection

Classical Information and Computation

- Programming of logic gates using Verilog: https://tutorialspoint.com/compile_verilog_online.php



Half Adder



Full Adder

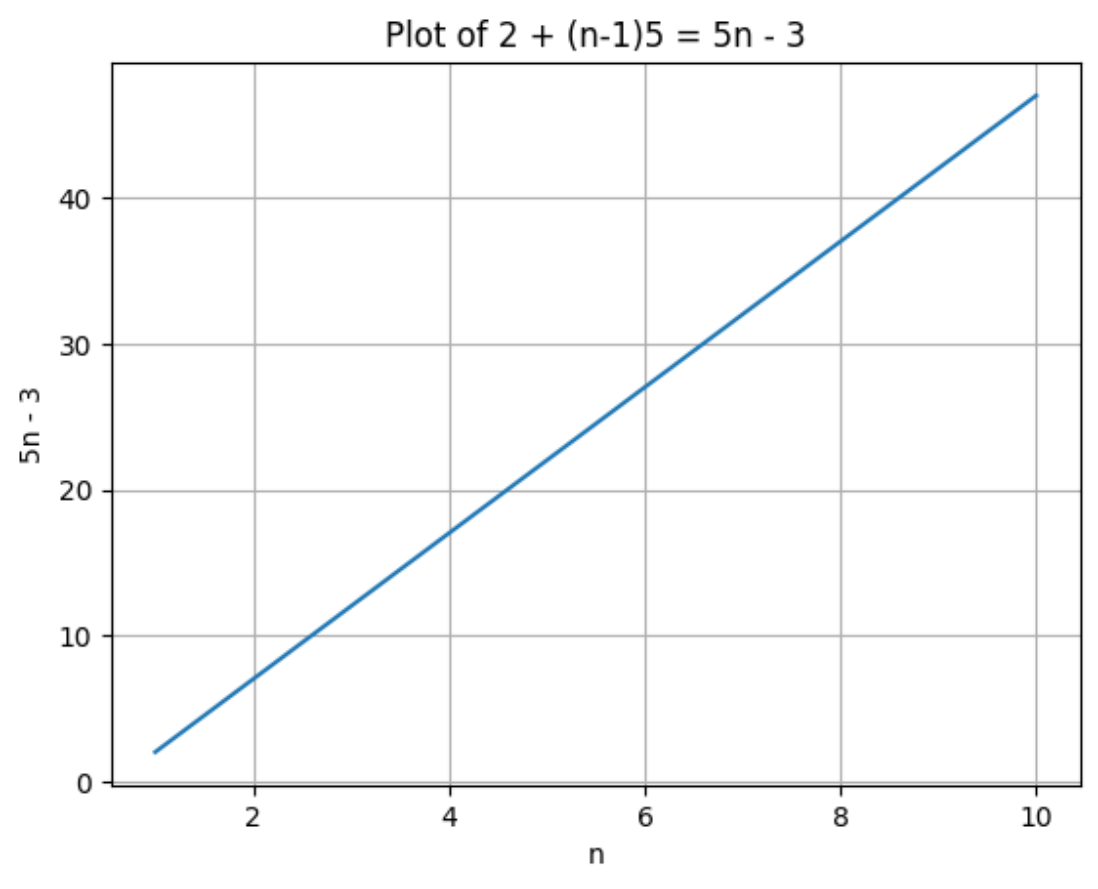
Computational Complexity

Asymptotic Notation

Ripple-Carry Adder (Adding two 4-bit numbers):

$$2 + (n - 1)5 = 5n - 3$$

- Big-O national is used to give an upper bound of the asymptotic behavior.
- Examples
  - $5n - 3 = O(nlog(n))$
  - $5n - 3 = O(2^n)$
  - $5n - 3 = O(n)$



Computational Complexity

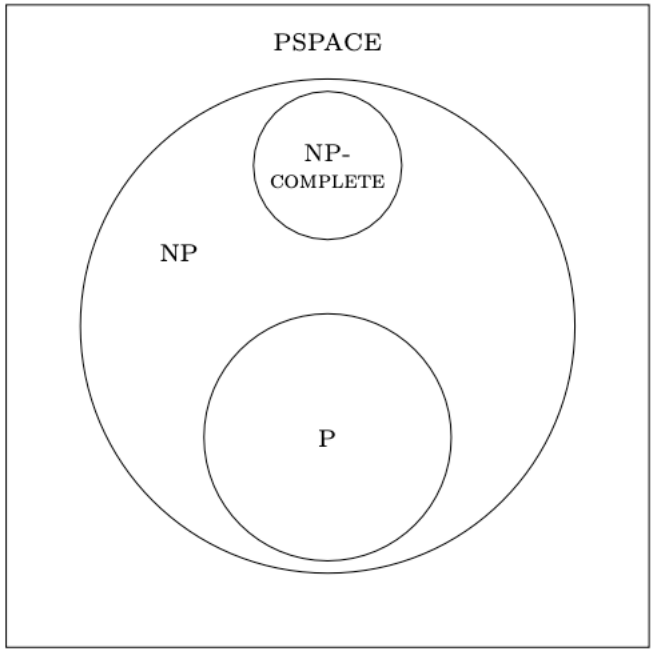
Asymptotic Notation

Notation	Description	Definition
$f(n) = O(g(n))$	f scales $\leq g$	$\exists\, c, n_0 \ni f(n) \leq cg(n)\ \forall\, n > n_0$
$f(n) = o(g(n))$	f scales $< g$	$\exists\, c, n_0 \ni f(n) < cg(n)\ \forall\, n > n_0$
$f(n) = \Omega(g(n))$	f scales $\geq g$	$\exists\, c, n_0 \ni f(n) \geq cg(n)\ \forall\, n > n_0$
$f(n) = \omega(g(n))$	f scales $> g$	$\exists\, c, n_0 \ni f(n) > cg(n)\ \forall\, n > n_0$
$f(n) = \Theta(g(n))$	f scales $= g$	$\exists\, c_1, c_2, n_0 \ni c_1 g(n) \leq f(n) \leq c_2 g(n)\ \forall\, n > n_0$

Computational Complexity

Complexity Classes

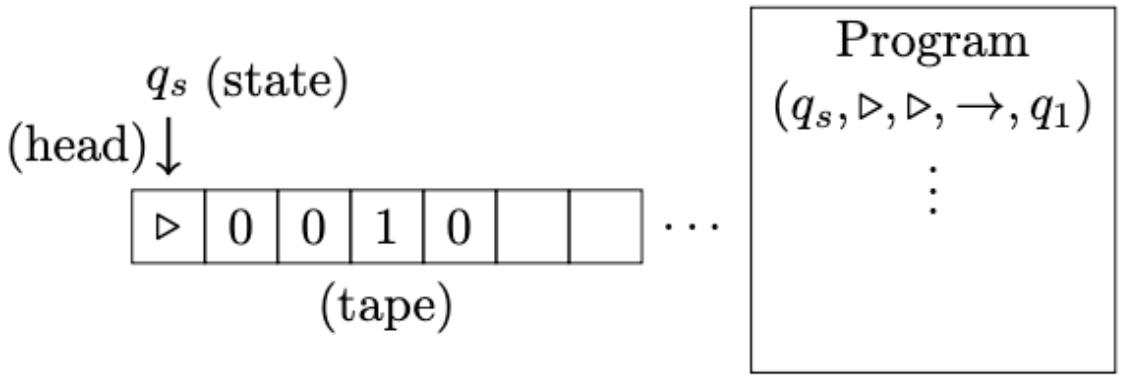
- P: Polynomial-Time
- NP: (the N stands for a non-deterministic Turing machine)
- PSPACE: Contains all the problems that can be solved by a computer using a polynomial amount of memory, without any limits on time.



Computational Complexity

Turing Machines

A Turing machine consists of four parts:



Computational Complexity

Church-Turing Thesis

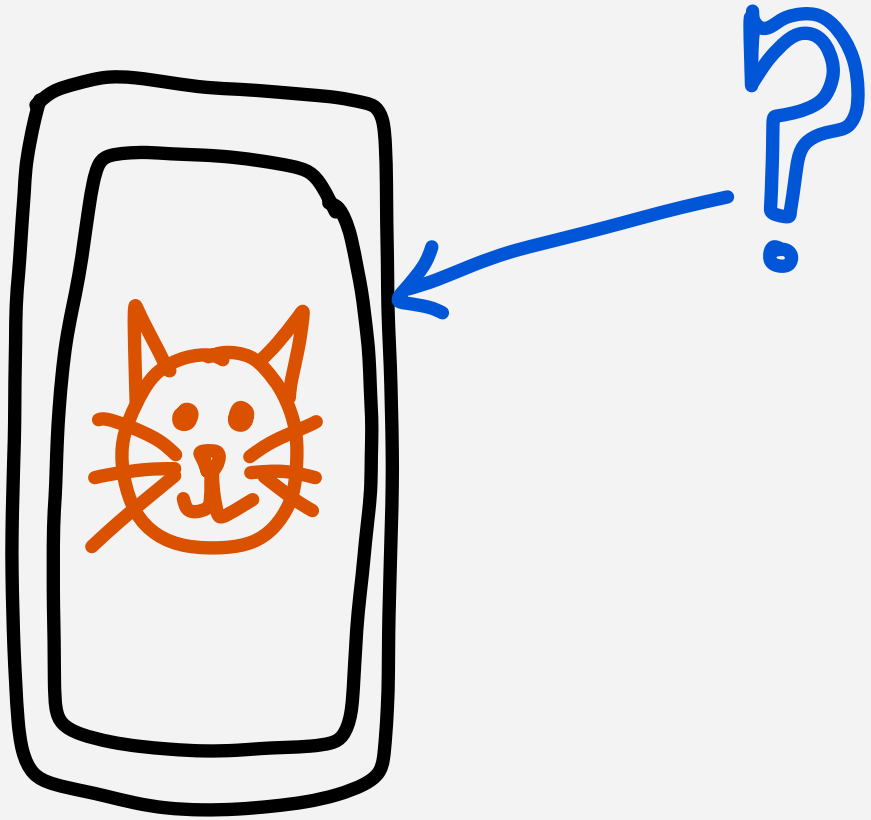
Thesis: ***Anything that can be computed by an algorithm (a step-by-step procedure) can be computed by a Turing machine.***

Issues with Classical Turing Machines

1. Speed and efficiency.
2. Complexity

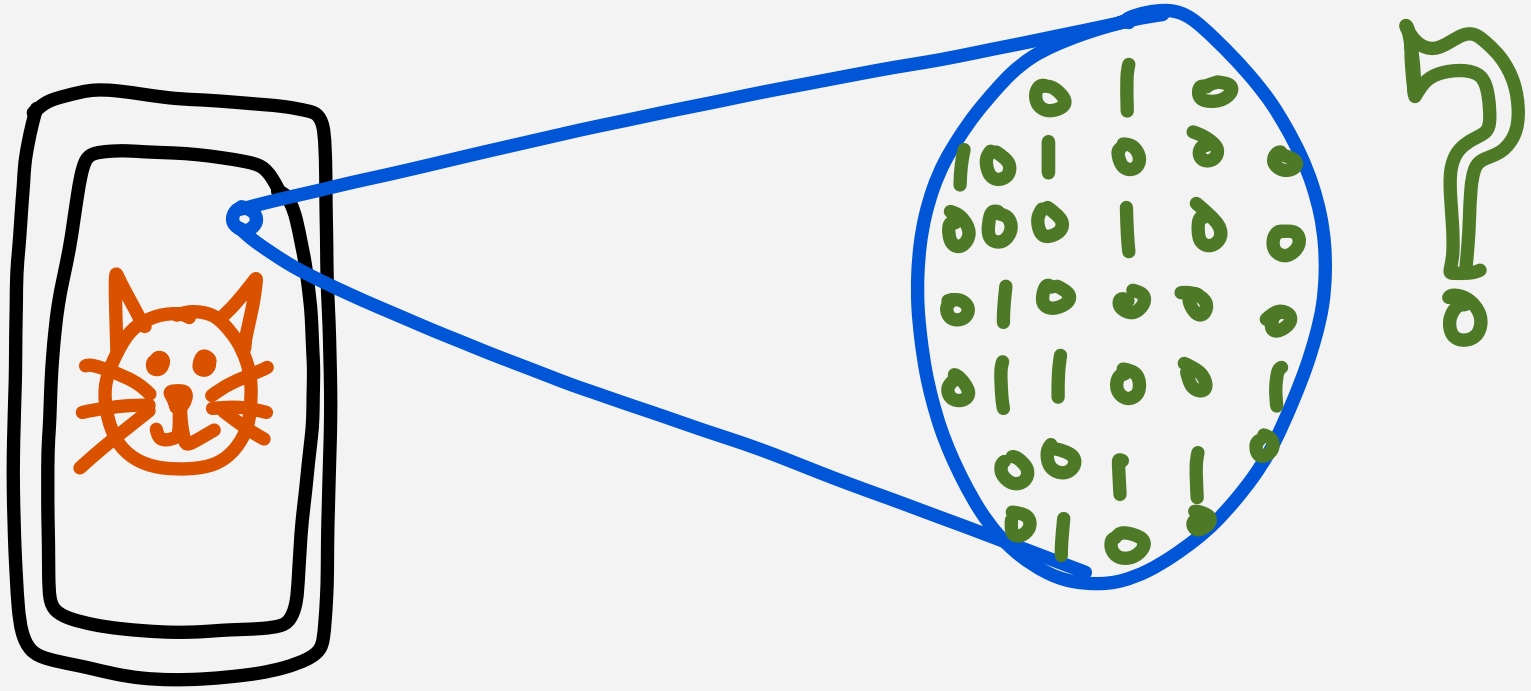
Is computation physical?

What's inside your phone?



Is computation physical?

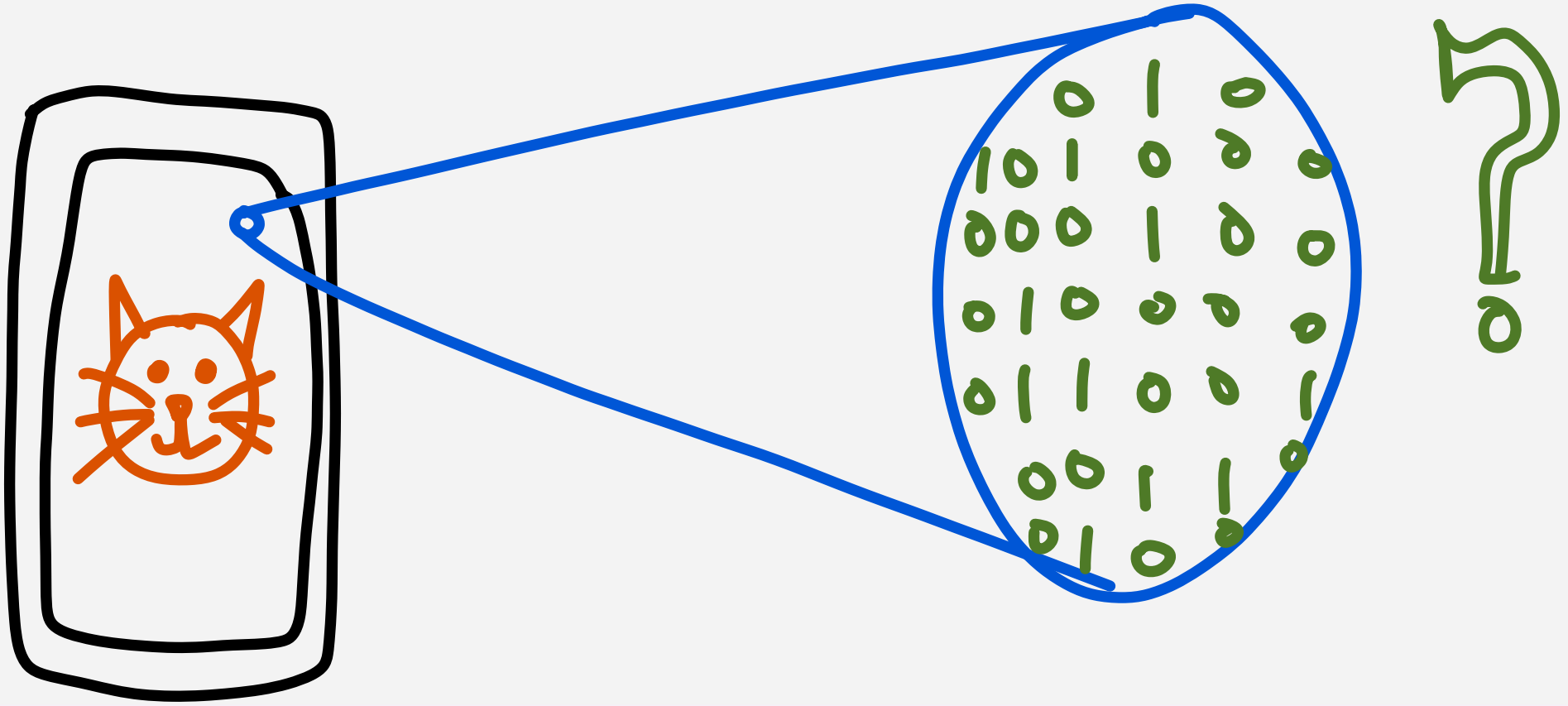
What's inside your phone?



Is computation physical?

What's inside your phone?

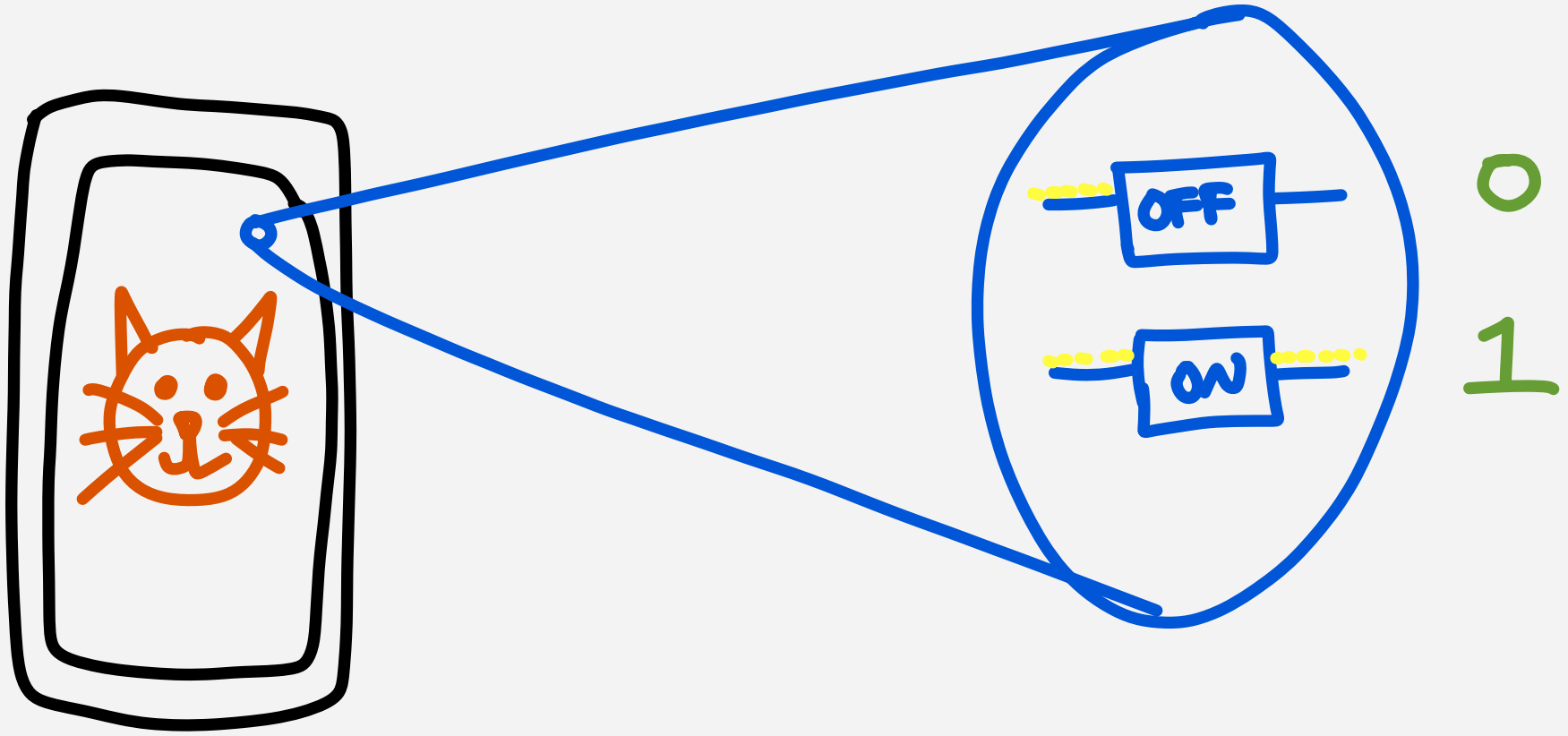
Not exactly…



Is computation physical?

What's inside your phone?

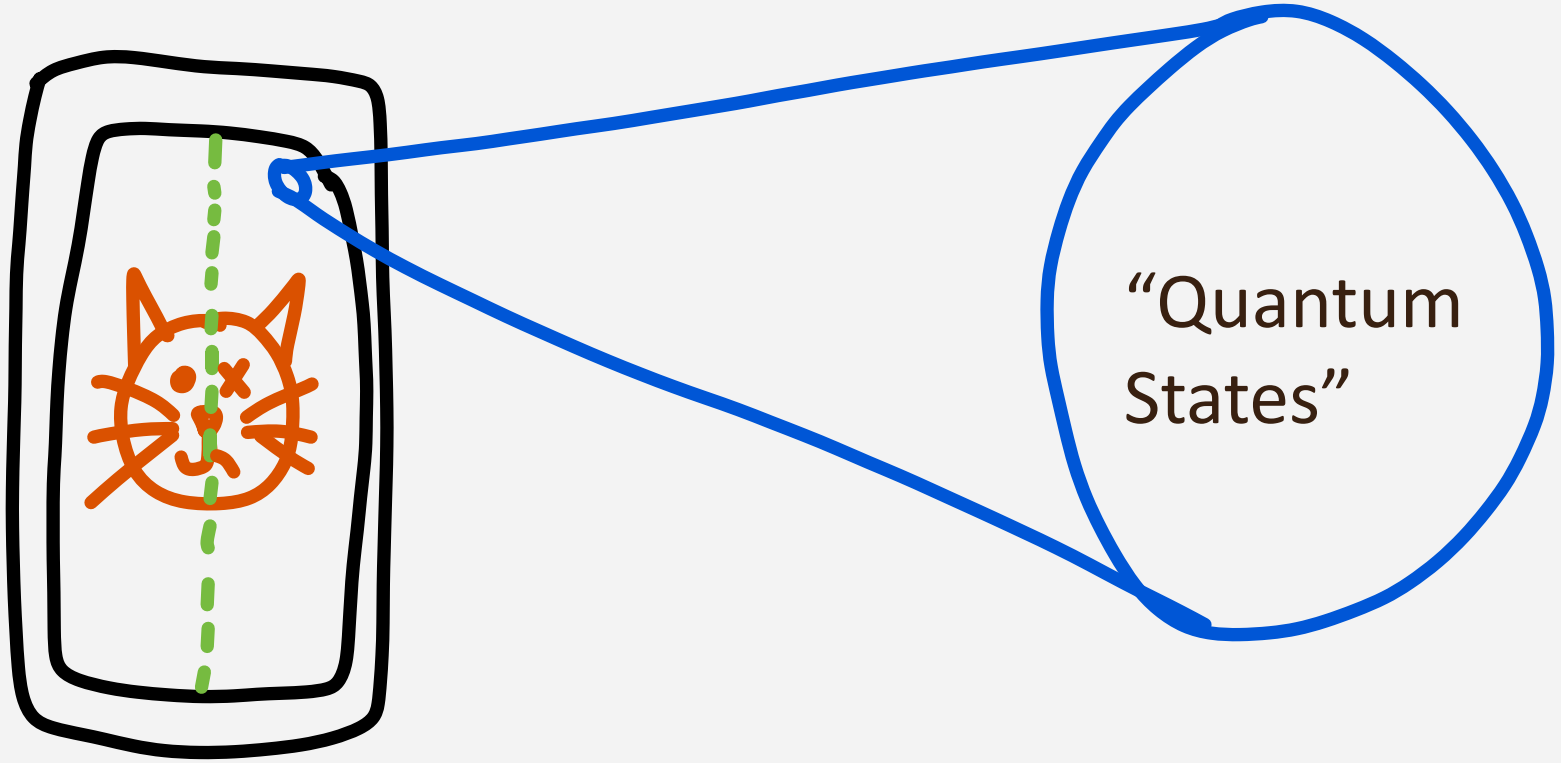
Transistors



These are different "states"

Is computation physical?

What's in a quantum computer?



Why quantum computing?

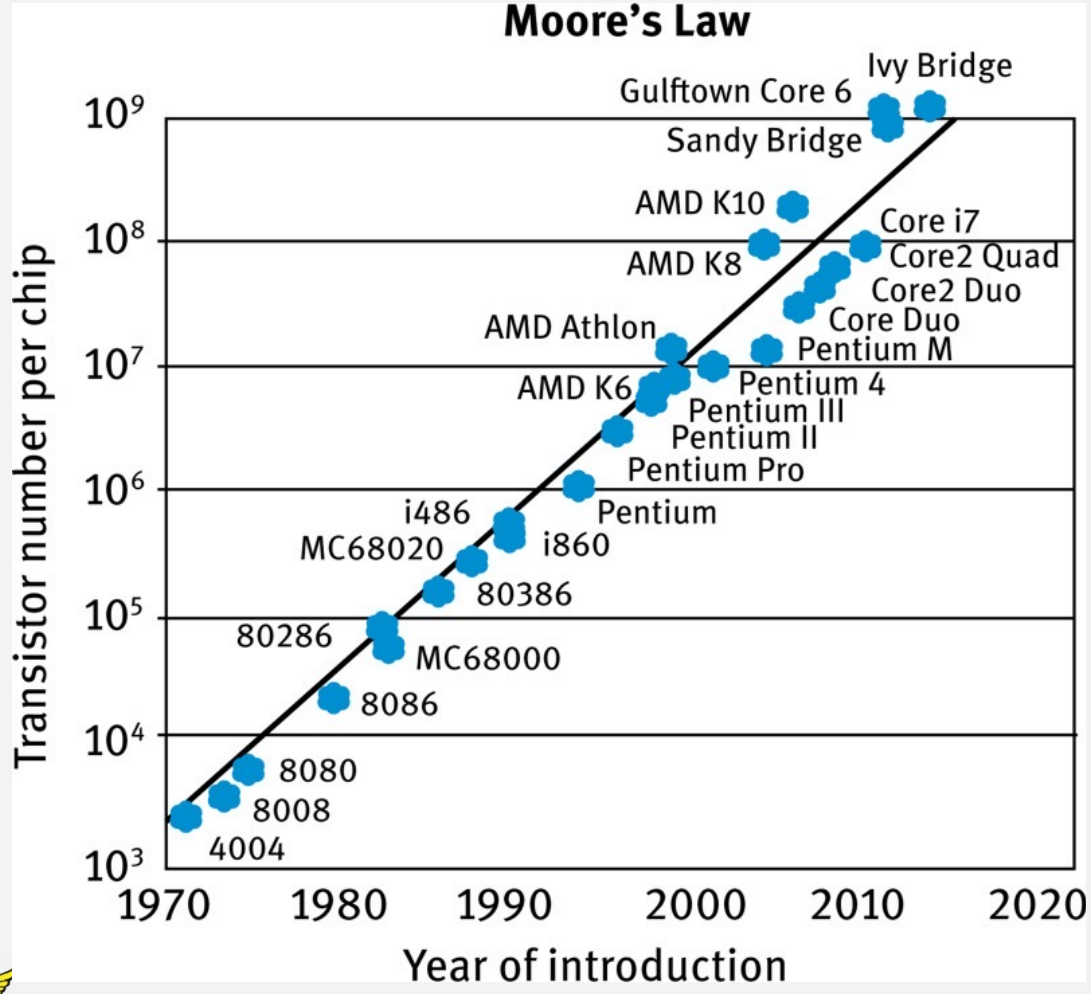
Well, there are a few good answers

We argued that physics plays a role

Quantum Physics = study of tiny things
(e.g., electrons, photons, etc.)

But, are there tiny things inside your phone?

Why quantum computing?



Issue #1: How small can transistors get?

Issue #2: Do quantum effects start playing a role?

Bug, or a feature?

This trend raises 2 issues…

The math behind the physics

Physics → Math → Computer Science

Transistors being OFF or ON.

Playing with transistors

Bits being 0 or 1.

$0 \oplus 1 = 1$

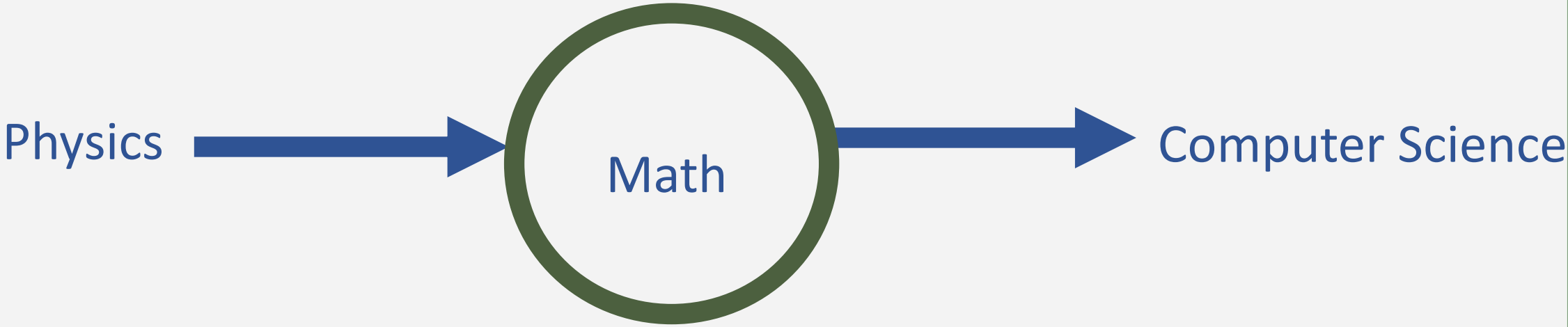
$1 \oplus 1 = 0$ …

Adding is easy.

Factoring is hard?

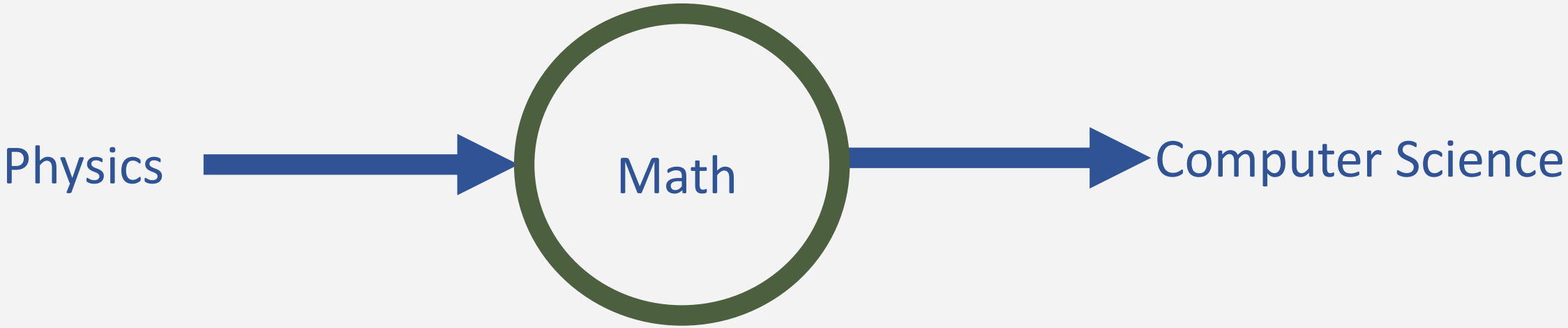
Robots!

The math behind the physics



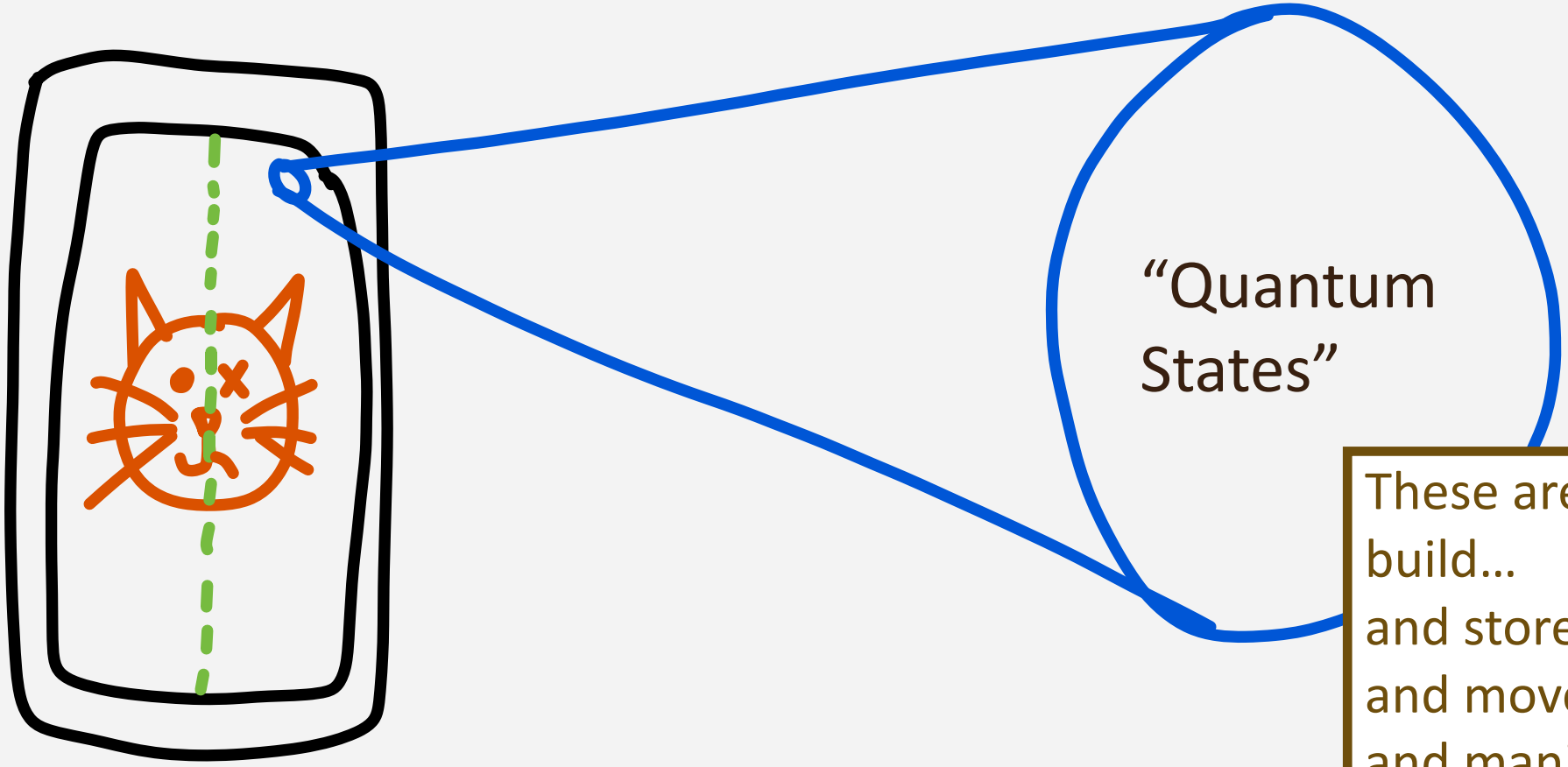
We'll concentrate on this for a while.

The math behind the physics



Just like how we can study bits without transistors, we can study the math of quantum today without worrying how we might build quantum computers tomorrow!

Why don't we have quantum computers yet?



These are hard to build…
and store…
and move…
and manipulate…

Why Quantum?

1. **Parallelism**: Quantum computer use quantum bits (qubits) that can represent both 0 and 1 simultaneously due to superposition. This allows quantum computers to process a vast number of possibilities at once.
2. **Entanglement**: Qubits can be entangled, meaning the state of one qubit can depend on the state of another, even across long distance. This property enables computational possibilities.
3. **Quantum Algorithms**: Algorithms like Shor's algorithm for factoring large numbers or Grover's algorithm for searching unsorted database can solve problems exponentially faster than classical algorithms.

Interactive Demo Websites

1. https://lab.quantumflytrap.com/lab/nonorthogonal-state-discrimination?mode=waves
2. https://quantum-computing.ibm.com/