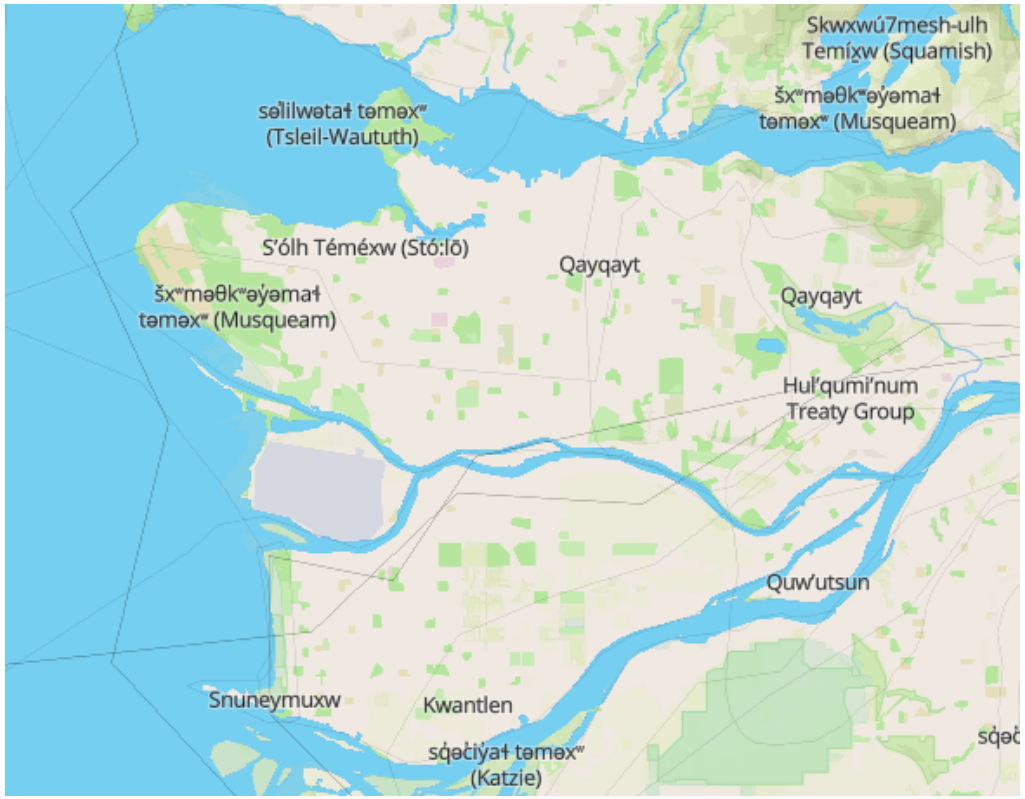
CPEN 400Q Lecture 01
Overview and intro to gate model quantum computing

Monday 6 January 2025

Land acknowledgement

The land on which we gather today is the traditional, ancestral, and unceded territory of the Musqueam People.



Explore more:

- `https://www.musqueam.bc.ca/`
- `https://native-land.ca/`

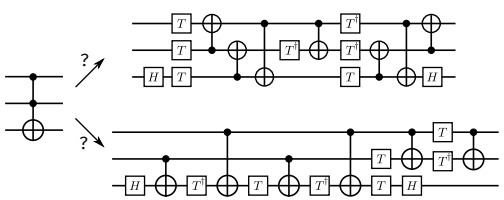
Olivia Di Matteo – olivia@ece.ubc.ca

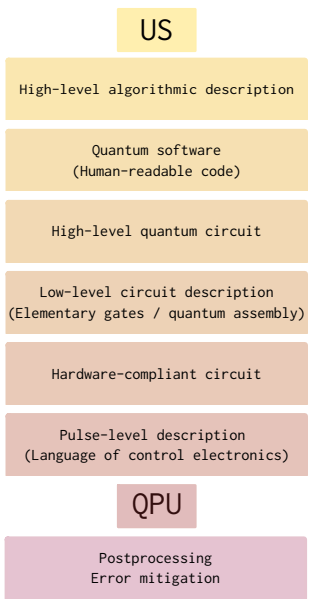
I have a physics background my group works on open-source quantum software and algorithms.

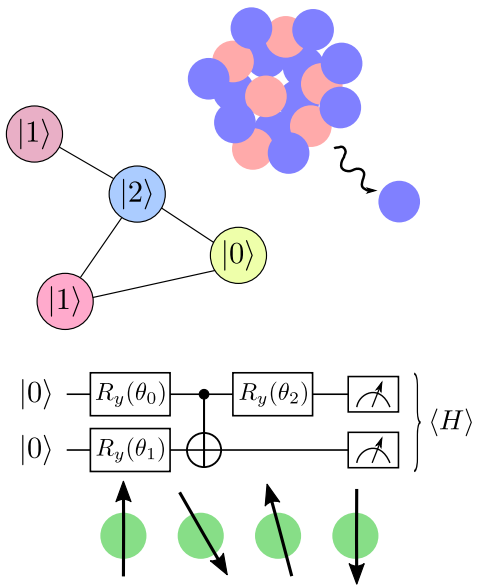
>QSAR QUANTUM SOFTWARE & ALGORITHMS RESEARCH

PENNYLANE

```
@qml.qnode(dev)
@qml.compile()
def circuit(params):
    qml.RX(params[0], wires=0)
    qml.RY(params[1], wires=0)
    ...
    return qml.expval(H)
```







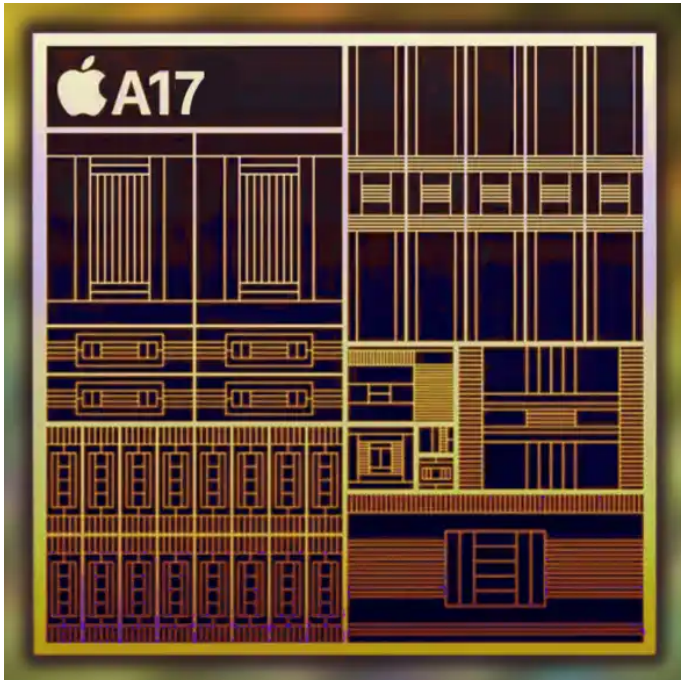
Intros

Go to `https://www.menti.com` and type in the code.

How do we make computers more powerful?

How do we make computers more powerful?

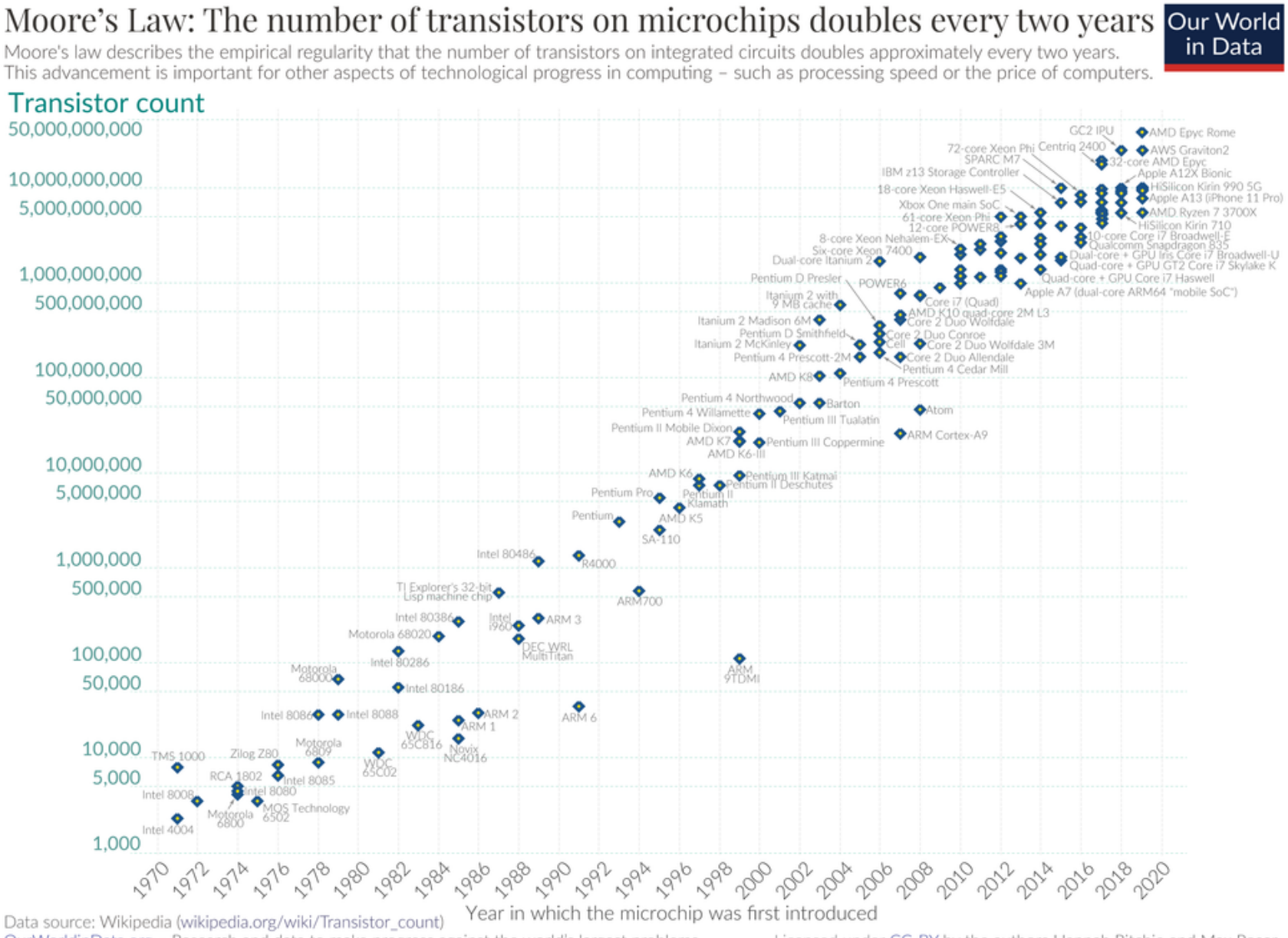
1. Make smaller transistors
2. Put more transistors onto a single chip



Apple A17: 19 billion transistors (iPhone 15 Pro / Pro Max, 2023)

Image: https://www.macworld.com/article/1532925/a17-bionic-preview-benchmarks-cpu-gpu-ram.html

How do we make computers more powerful?



How do we make computers more powerful?

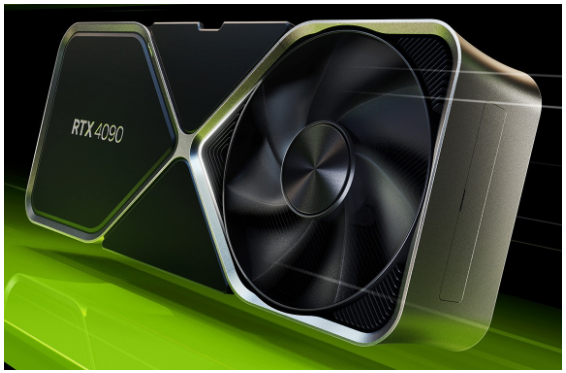
3. Use multiple processors in parallel

El Capitan (LLNL): ∼11 million cores (CPU+GPU), ∼1.742 exaFLOPS

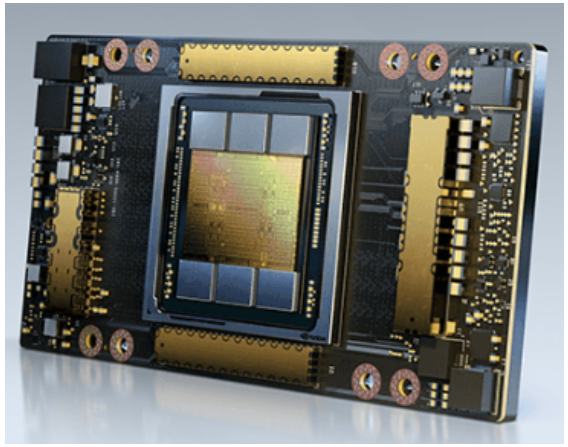
Image: https://asc.llnl.gov/exascale/el-capitan

How do we make computers more powerful?

4. Make special-purpose computers



GPU: NVIDIA RTX 4090



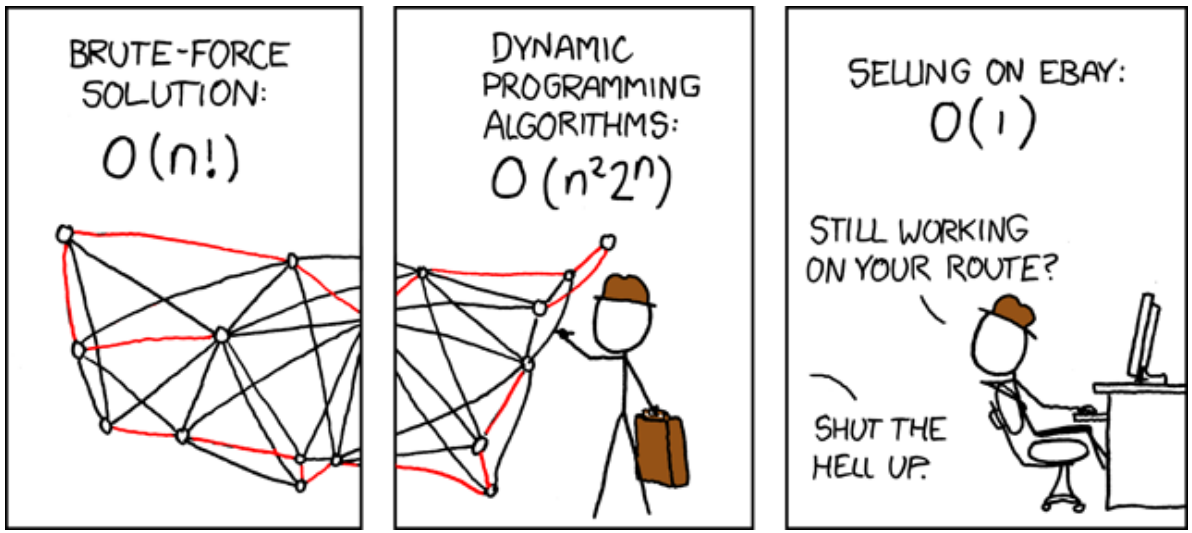
TPU: NVIDIA RTX A100

TPU: Google Trillium

Images:
https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4090/
https://www.nvidia.com/en-us/data-center/a100/
https://techcrunch.com/2024/05/14/googles-next-gen-tpus-promise-a-4-7x-performance-boost/

What's the point?

Some problems will remain intractable.



Sometimes that's good:

- our cryptographic infrastructure relies on some mathematical problems being *computationally hard*

Image: https://imgs.xkcd.com/comics/travelling_salesman_problem.png

What's the point?

But usually it just prevents us from doing interesting things:

- solving complex optimization problems
- simulation of molecules and quantum systems
- searching large spaces
- machine learning with large amounts of data

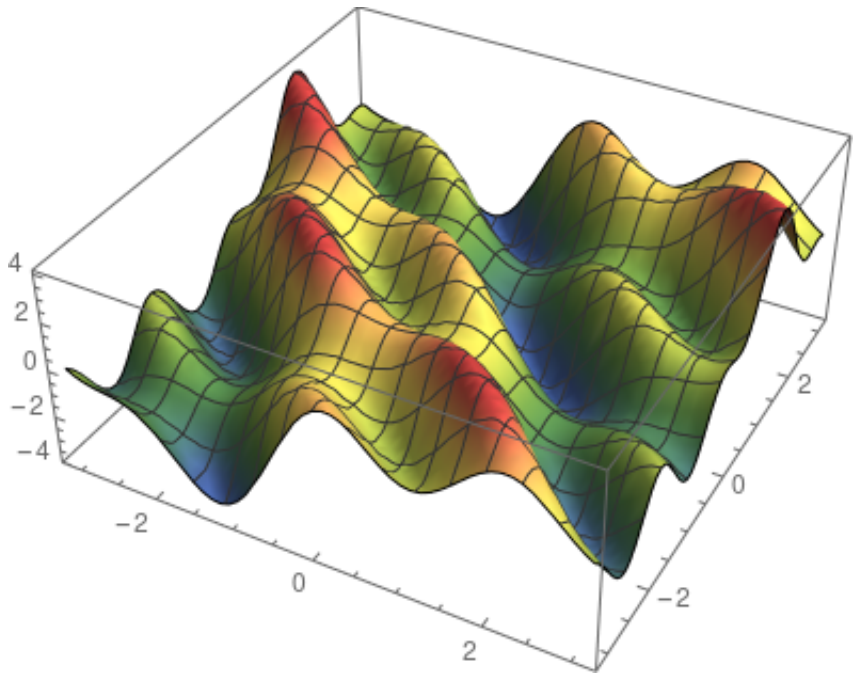
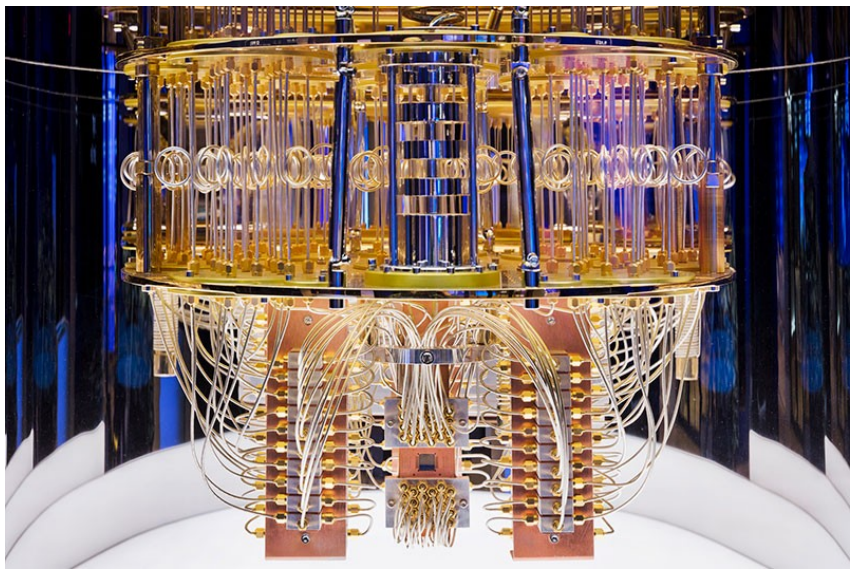


Image:

https://towardsdatascience.com/the-mathematics-of-optimization-for-deep-learning-11af2b1fda30

How do we make computers more powerful?

4+. Make special-purpose computers *that work in a fundamentally different way*.



"Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems."

– Nielsen & Chuang

We know quantum algorithms that have an *exponential speedup* over the best-known classical algorithms (and some that don't, but are still good).

Image: https://www.nature.com/articles/d41586-021-03476-5

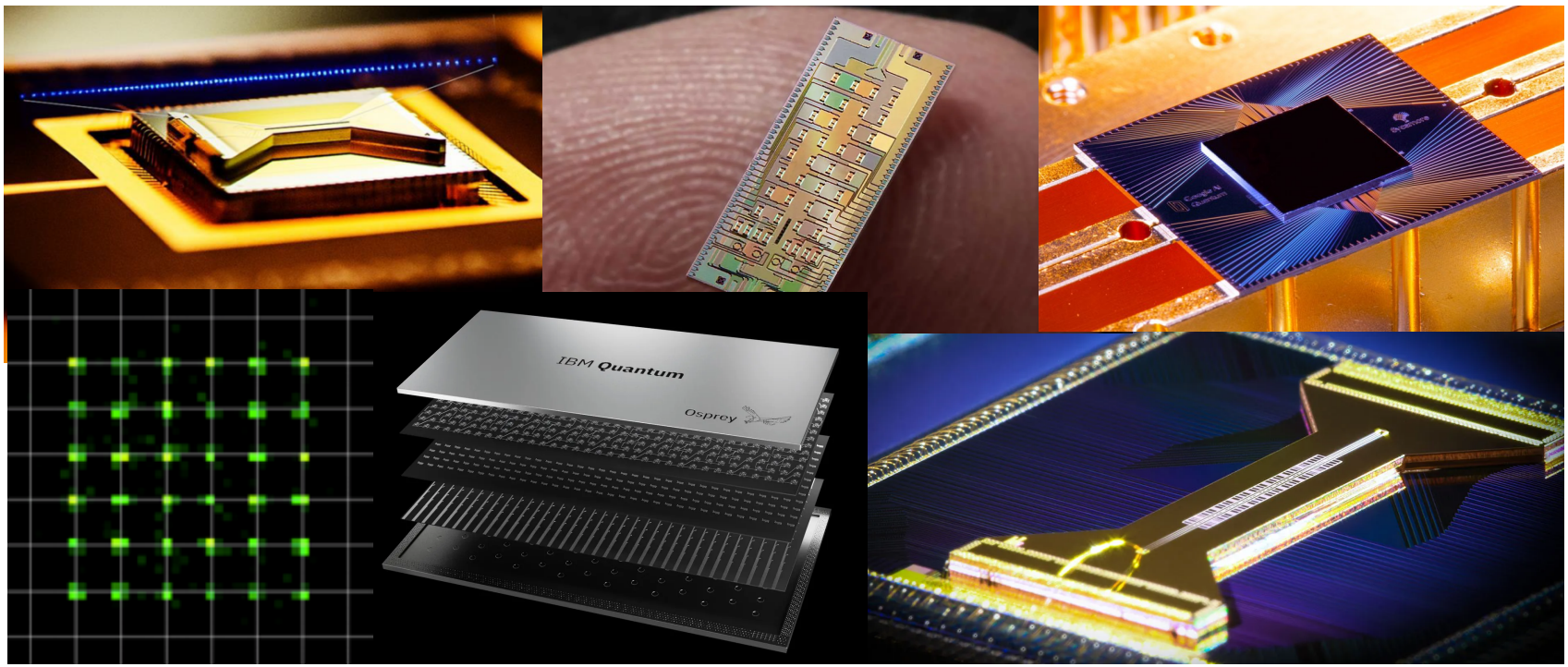
Quantum computing

Lots of potential applications:

- Simulating and calculating properties of physical systems
- Searching large spaces
- Discrete mathematics, (breaking) cryptography
- Optimization
- Machine learning
- Things we haven't thought of yet!

We are already capable of doing small, proof-of-concept implementations of some of these algorithms...

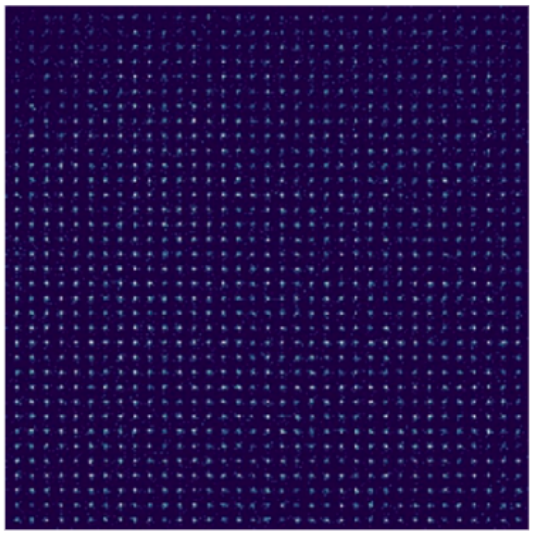
Quantum computers



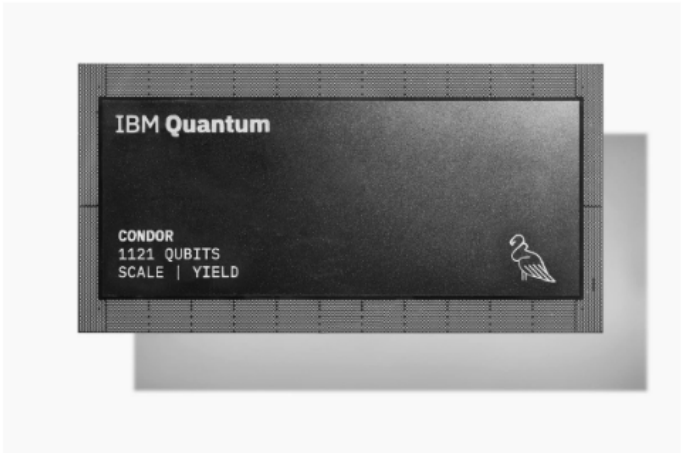
Commercial quantum processors (clockwise from top left): IonQ (trapped-ion), Xanadu (photonic), Google (superconducting), Quantinuum (trapped-ion), IBM (superconducting), Pasqal (neutral atoms)

Quantum computers

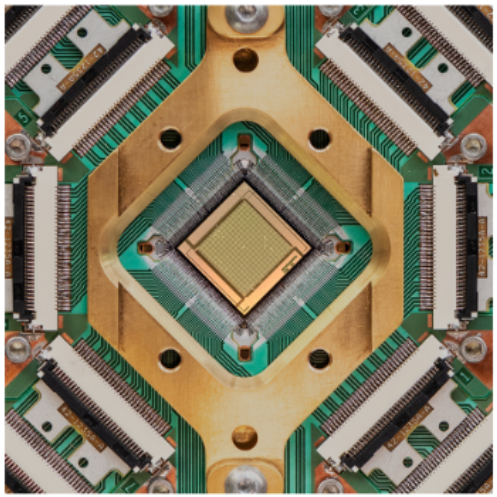
Largest machines (accurate as of 6 Jan)



1180 neutral atom qubits
Atom Computing



1121 superconducting qubits
IBM

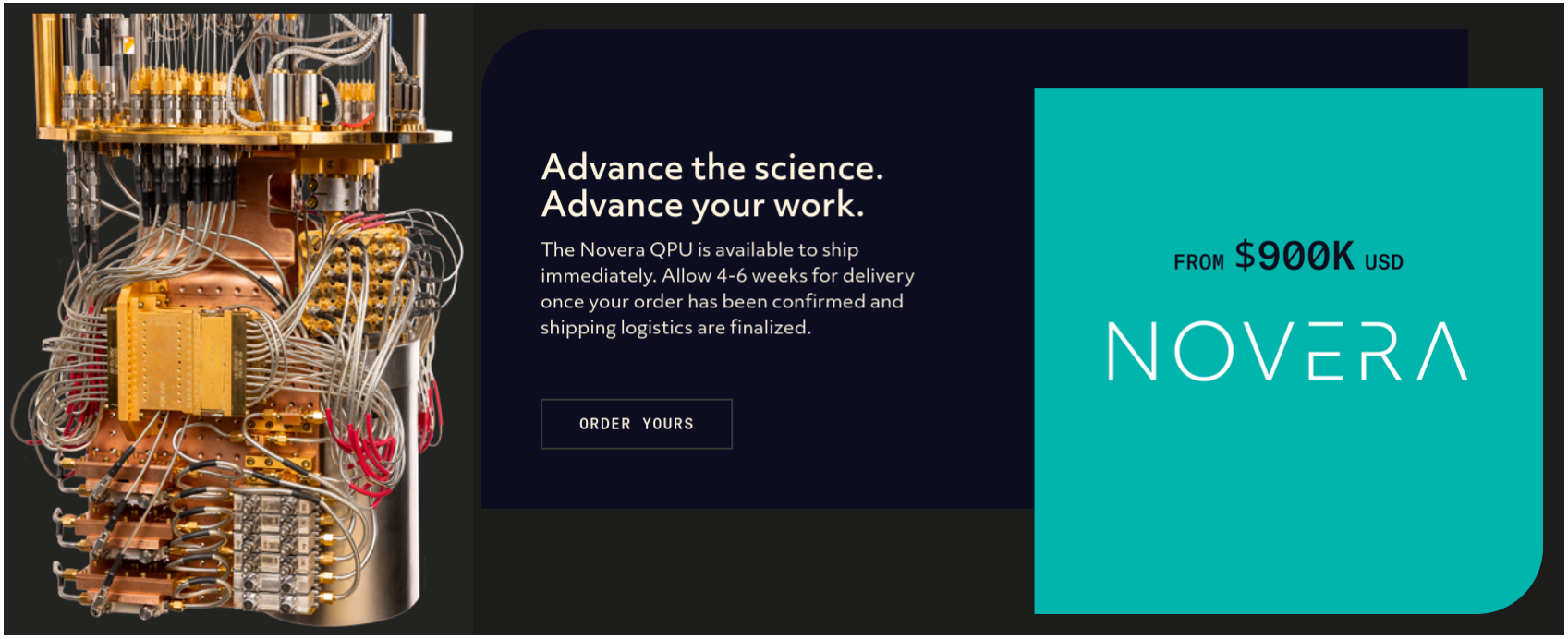


5000+ superconducting qubits
D-Wave

Images:
https://arstechnica.com/science/2023/10/atom-computing-is-the-first-to-announce-a-1000-qubit-quantum-computer/
https://techovedas.com/ibm-breaks-record-with-1121-qubit-quantum-processor-condor-takes-flight/
https://www.dwavesys.com/solutions-and-products/systems/

Quantum computers

You can literally buy one off-the-shelf*



*dilution refridgerator not included

Image: https://www.rigetti.com/novera

Quantum computers

Today's QCs are termed **noisy, intermediate scale quantum** (NISQ) devices.

Quantum Computing in the NISQ era and beyond

John Preskill

Institute for Quantum Information and Matter and Walter Burke Institute for Theoretical Physics, California Institute of Technology, Pasadena CA 91125, USA

Published: 2018-08-06, volume 2, page 79

Eprint: arXiv:1801.00862v3

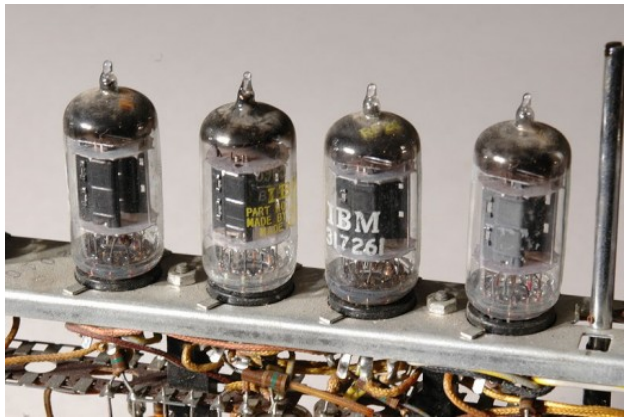
Doi: https://doi.org/10.22331/q-2018-08-06-79

Citation: Quantum 2, 79 (2018).

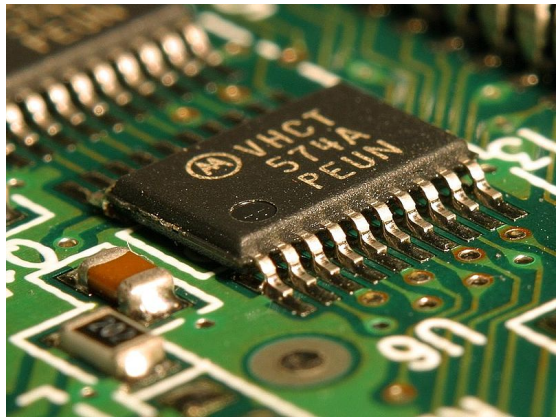
- Short coherence times
- High gate error rates (single-qubit, two-qubit operations)
- Limited qubit connectivity
- Challenging to scale up the hardware

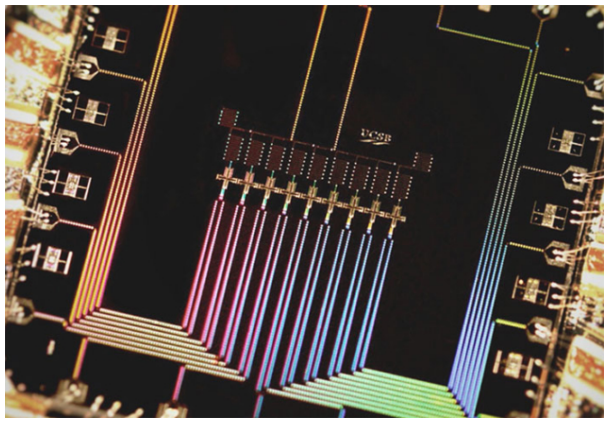
Quantum computing technologies

Too early to know which (combination?), if any, will win out.

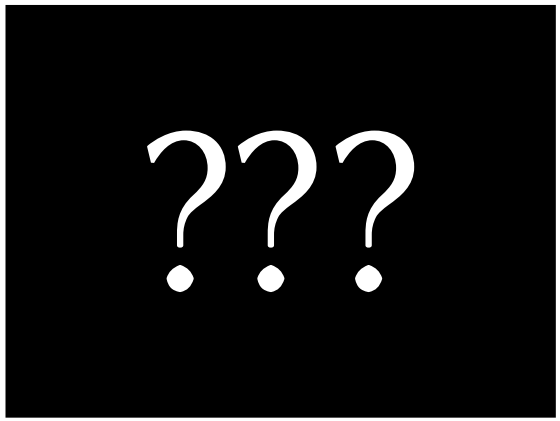


vs.





vs.

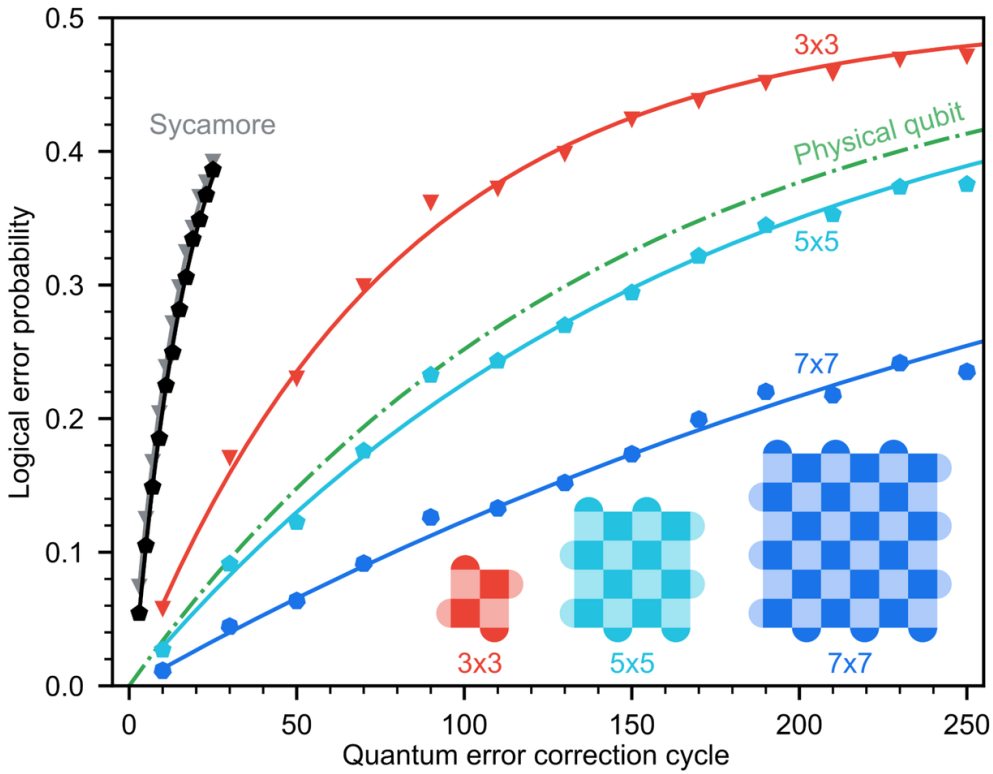


Images:
https://hubpages.com/business/What-Is-a-Transistor-and-Why-is-it-Important
https://en.wikipedia.org/wiki/Solid-state_electronics
https://physicsworld.com/a/google-gains-new-ground-on-universal-quantum-computer/

Quantum computing beyond the NISQ era

Solving problems "at scale" requires quantum error correction.

Numerous demonstrations of error-corrected qubits last year.
Example: Google's *Willow* (105 superconducting qubits).



By the last course module you'll know what everything in this plot is!

Image: https://research.google/blog/making-quantum-error-correction-work/

Quantum computing at scale

Some applications will require *millions* of qubits to run an algorithm on noisy qubits with full error correction.

RSA-2048	**Old estimates**				**Current estimates**			
p_g	n_ℓ	n_p	quantum resources	time	n_ℓ	n_p	quantum resources	time
10^{-3}	6190	19.20	1.17	1.46	8194	22.27	0.27	0.34
10^{-5}	6190	9.66	0.34	0.84	8194	8.70	0.06	0.15

Table 2. RSA-2048 security estimates. Here n_ℓ denotes the number of logical qubits, n_p denotes the number of physical qubits (in millions), time denotes the expected time (in hours) to break the scheme, and *quantum resources* (*quantum resources*) are expressed in units of megaqubitdays. The corresponding classical security parameter is 112 bits.

Image: V. Gheorghiu and M. Mosca, *A resource estimation framework for quantum attacks against cryptographic functions - recent developments.* Feb. 15 2021.

Quantum computing at scale

IBM Quantum roadmap (late 2024)

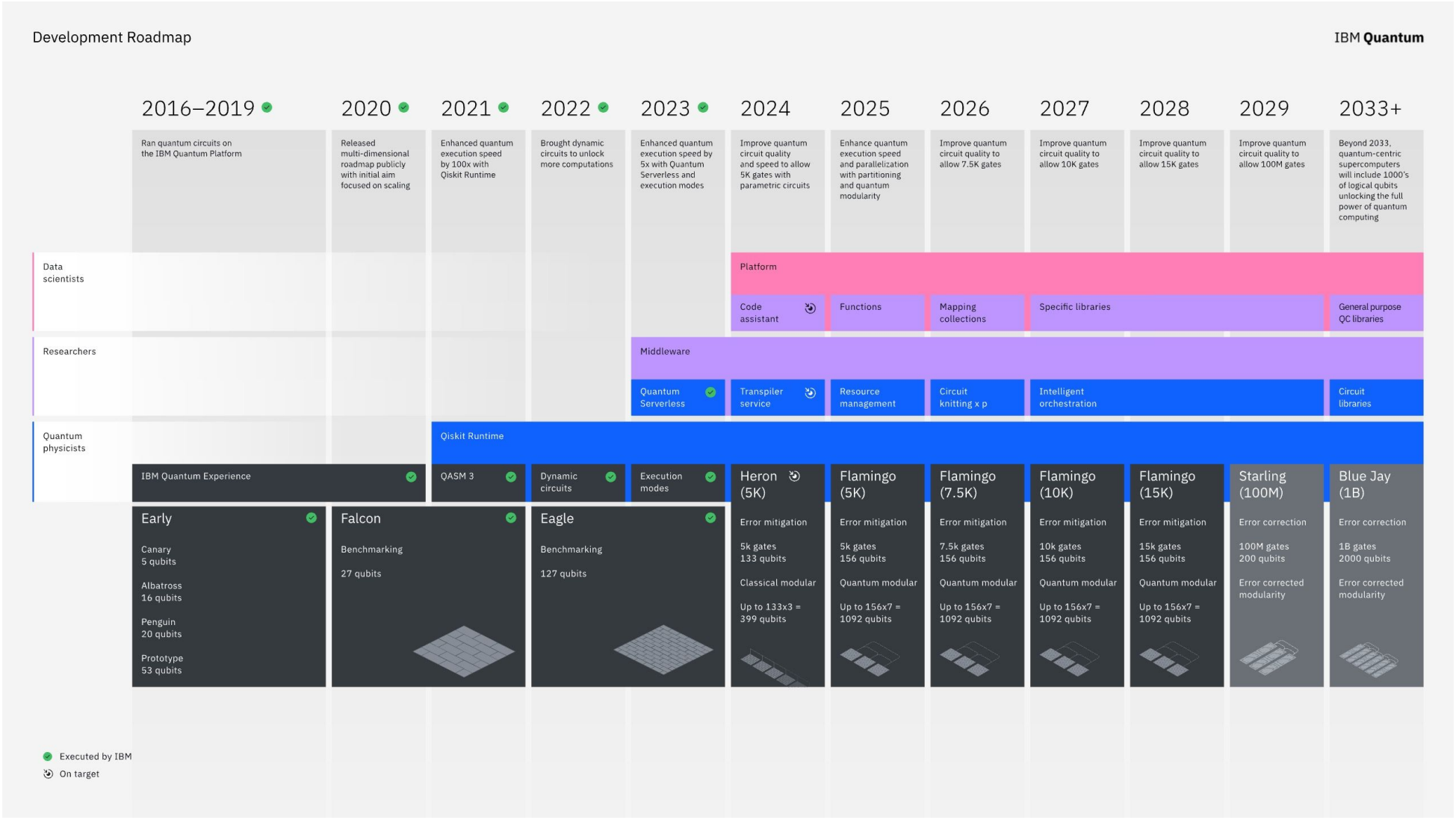
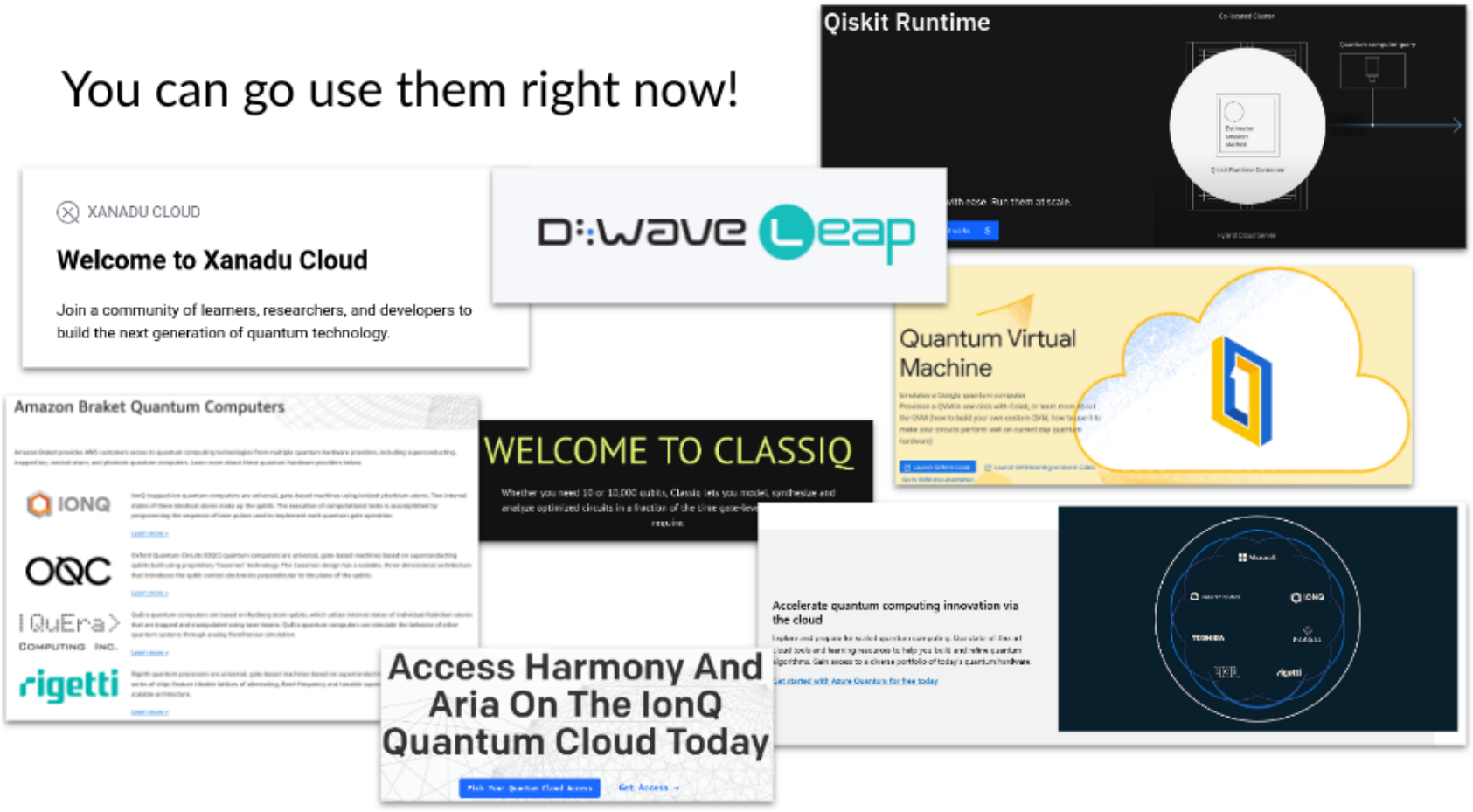


Image: https://www.ibm.com/quantum/blog/ibm-quantum-roadmap-2025

Quantum computing in the cloud

You can go use them right now!



But what would you do with them?

Course learning outcomes

Core goal: **learn how to program quantum computers** in a hands-on, software-focused setting.

- Describe the societal importance and implications of quantum computing
- Explain the theory and principles behind gate-model quantum computing
- Outline and describe the operation of core quantum algorithms
- Implement basic and research-level quantum algorithms using Python and PennyLane

In this course you will implement everything you learn!

Covered in this course

Divided into 5 modules:

1. Basic elements of quantum computation
2. Oracle-based algorithms, complexity, and quantum resources
3. Quantum Fourier transform-based algorithms
4. Quantum information theory
5. Quantum error correction

Covered in this course

Divided into 5 modules:

1. Basic elements of quantum computation
2. Oracle-based algorithms, complexity, and quantum resources
3. Quantum Fourier transform-based algorithms
4. Quantum information theory
5. Quantum error correction

Differences from previous years:

- Module 5 is brand new; module 4 bigger than previous years
- Hamiltonian simulation covered in a hands-on assignment only; removed variational algorithm content
- Hands-on activities done in tutorial section

Not covered in this course

- How the hardware actually works
- Non-gate-model quantum computing
  - adiabatic quantum computing
  - quantum annealing
  - measurement-based quantum computing
  - continuous-variable quantum computing
- Other topics: complexity theory, variational algorithms, physics simulation, tensor networks, and more

Logistics

My info:

Office:	KAIS 3043
Office hrs:	M 13:00-14:00, Th 14:30-15:30, or by appointment. Open-door policy after 12:00 weekdays
Email:	`olivia@ece.ubc.ca`
GitHub:	`glassnotes`

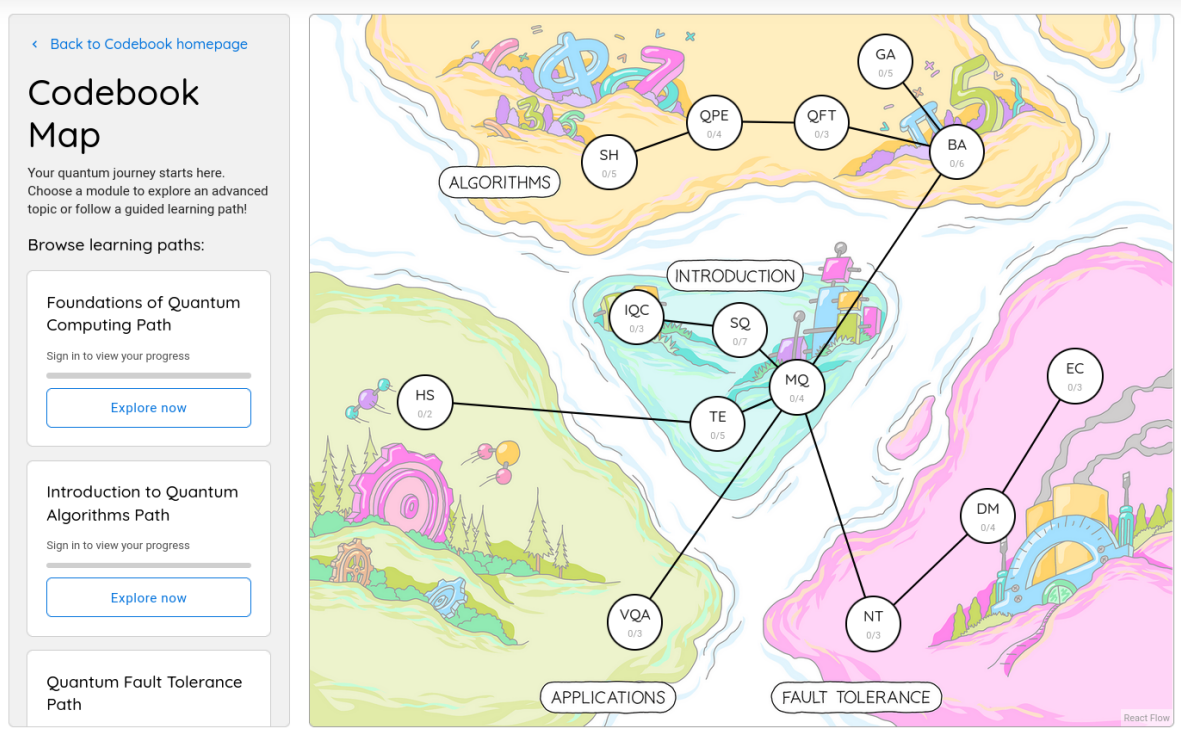
TA info: see syllabus and Piazza

All lecture slides/demos posted on course GitHub:

`https://github.com/glassnotes/CPEN-400Q`

Textbook

Primary resource: **free**, **online** PennyLane Codebook. Use to learn content, PennyLane, and do practice programming exercises.



https://pennylane.ai/codebook

Secondary (optional) resource: *Introduction to Quantum Computation and Quantum Information*, Nielsen and Chuang.

Assignments and grading

Six components:

- 15% technical assignments
- 5% “quantum literacy” assignments
- 10% in-class quizzes
- 30% in-class midterm exam
- 30% final group project
- 10% final oral interview

Assignments and grading, cntd.

15% technical assignments ($\sim$8-10):

- Done on PrairieLearn
- Mix of pen-and-paper and programming problems
- Includes hands-on group activities from tutorials
- May discuss with classmates but what you submit must be your own work (and cite your collaborators!)

Purpose is to gain deeper understanding of content from class, and apply it to new situations.

"Assignment 0" available; for practice only (review of linear algebra, Python, NumPy, and introduction to PrairieLearn).

Assignments and grading, cntd.

5% quantum literacy assignments (2-4):

- Done individually, no collaboration unless otherwise stated
- Activities may include:
  - critical analysis of media articles or videos
  - small creative writing projects
  - group discussions or debates

Purpose is to explore the ethical and societal implications around construction and use of quantum computers, and help dispel hype.

You may submit one assignment (of either type) up to 24hrs late, with advance notice.

Assignments and grading, cntd.

10% quizzes (10):

- Starting next Monday; done at start of class in PrairieLearn
- Done individually, but open book/notes/docs
- Includes math and/or programming problems and conceptual questions; based on previous week's lecture material

Purpose is for reviewing content. Schedule is in syllabus.

Your lowest quiz grade will be automatically dropped.

Assignments and grading, cntd.

30% midterm:

- Wednesday 29 January during lecture
- closed-book and closed-notes
- pen-and-paper only; no programming

Purpose is to test knowledge of core content before studying advanced algorithms.

You must pass the midterm to pass the course

- can retry in form of oral exam (multiple attempts allowed)
- highest achievable score from retakes is 50%

Assignments and grading, cntd.

30% final project:

- Replicate results of a research paper
- Work in groups of 4
- Three components:
  - Fully-documented software implementation
  - Companion report
  - 15-20 minute in-class presentation
- Use a different quantum programming framework if you like

More details (rubric, grading, papers, etc.) after midterm.

Assignments and grading, cntd.

10% final exam:

- oral "exit interview" (10-20 minutes) during exam period
- technical and conceptual questions about course content
- questions about your group's project

You must pass the exam to pass the course (can retry once for max score of 50%).

Academic misconduct and AI usage policies

The following will lead to a grade of 0 and/or a misconduct investigation:

- copying the work of classmates
- written content that closely paraphrases external sources
- usage of generative AI tools for all course components, unless indicated otherwise

Exceptions:

- can use, e.g., ChatGPT for rephrasing of **already written** text in literacy assignments (must submit your logs)
- if your IDE has, e.g., Copilot integration, you must acknowledge as source in assignment submission

Module 1: basic elements of quantum computation

Module 1 learning outcomes

- perform quantum computations using Dirac notation and matrix algebra
- express quantum computations using quantum circuits
- program quantum circuits in PennyLane
- use the Bloch sphere to represent states and the action of quantum operations
- list and define the core set of elementary quantum operations
- define and give examples of entangled states
- compute the result of measurements on one or more qubits in multiple bases
- compute the expectation value of an observable
- distinguish between non-orthogonal states with generalized quantum measurements
- use Bell basis measurements to implement superdense coding and teleportation

Today

Learning outcomes:

1. Define quantum computing, and explain its societal importance
2. Define a *qubit* conceptually and mathematically
3. Outline the mathematical structure of the key components of quantum algorithms (states, gates, measurements)

Bits

All computation in our computers today is done with bits.

The state of a bit is *either* 0 or 1.

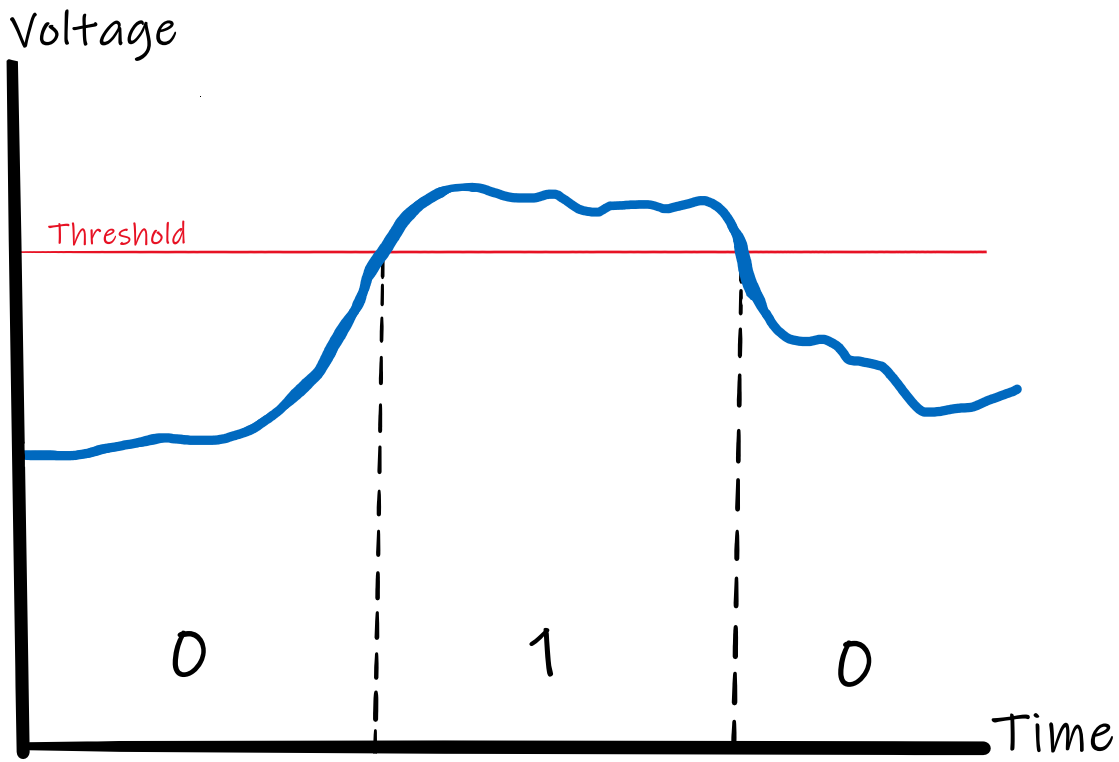
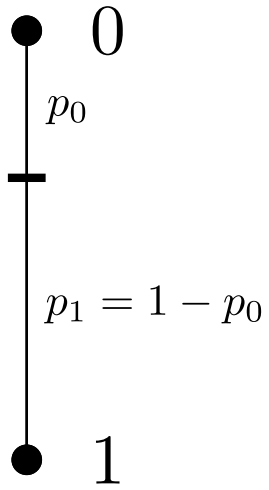


Image: PennyLane Codebook (All About Qubits)

Probabilistic bits

The value of a bit can be represented by a probability distribution.



Example: A coin flip: Heads (1) or Tails (0), each equally likely.

Example: Is it raining? Yes (1) or No (0) with some probability.

Probabilistic bits

Probability distributions can evolve over time.

Exercise:

- If my laptop is connected to `ubcsecure` now, the probability it will still be connected in an hour is 80%
- If it isn't connected now, the probability it will be connected in an hour is 60%

My laptop is currently connected (I think). What is the probability that I give up and buy an ethernet adapter in three hours?

$\approx 80\%$

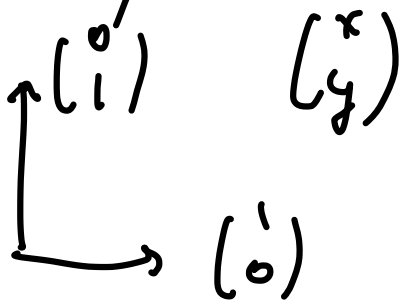
~ 0.488

. tree diagram

Probabilistic bits

Represent the system as a 2-dimensional real-valued **vector** $(\mathbb{R}^2)$

connection status $= \begin{pmatrix} \text{prob. connected} \\ \text{prob. not connected} \end{pmatrix}$



Elements necessarily sum to 1. More compactly:

$$\vec{s} = \begin{pmatrix} p_c \\ 1-p_c \end{pmatrix}$$

$$= \begin{pmatrix} p_c \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1-p_c \end{pmatrix}$$

$$= p_c \begin{pmatrix} 1 \\ 0 \end{pmatrix} + (1-p_c) \begin{pmatrix} 0 \\ 1 \end{pmatrix} = p_c \vec{c} + (1-p_c)\vec{n}$$

basis vectors

We call $\vec{c}$ and $\vec{n}$ basis vectors.

Probabilistic bits

- Connected now, 80% prob. connected in one hour
- Not connected now, 60% prob. connected in one hour

How does this affect our basis vectors?

$$\vec{c} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \xrightarrow{1 \text{ hour}} \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix} \Rightarrow \vec{c} \to 0.8\vec{c} + 0.2\vec{n}$$

$$\vec{n} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \xrightarrow{1 \text{ hour}} \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \Rightarrow \vec{n} \to 0.6\vec{c} + 0.4\vec{n}$$

More compactly,

$$\vec{s} \longrightarrow P\vec{s} = \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix} \vec{s}$$

Probabilistic bits

Determine what happens one, two, and three hours from now

$$\vec{s}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

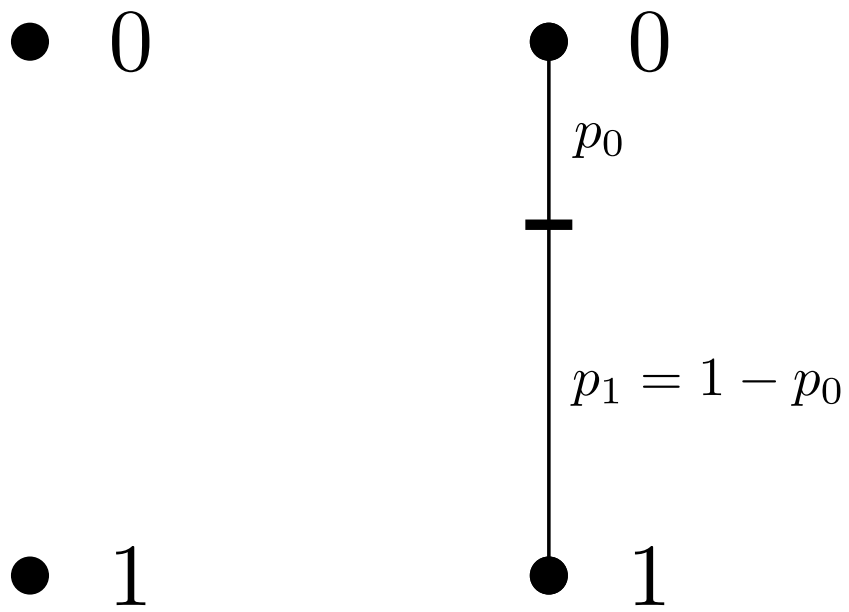
$$\vec{s}_1 = P\vec{s}_0 = \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix}$$

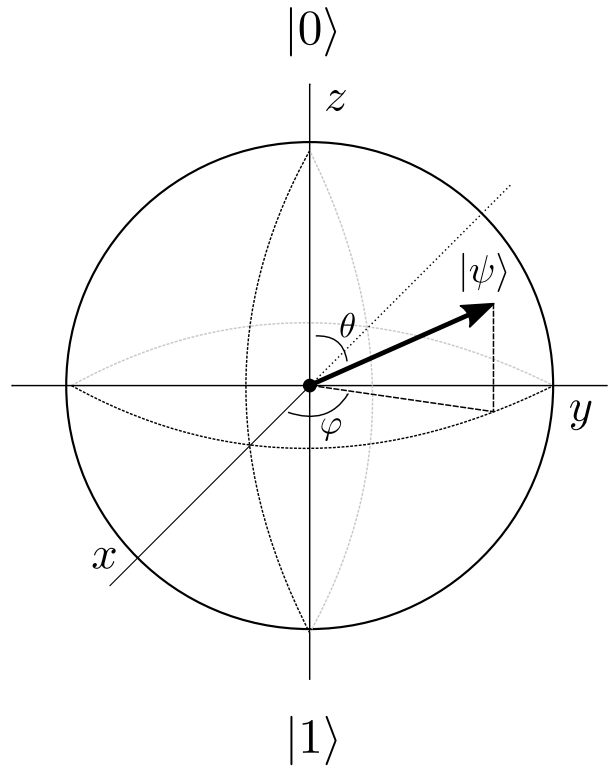
$$\vec{s}_2 = P\vec{s}_1 = P^2 s_0 = \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix}\begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 0.76 \\ 0.24 \end{pmatrix}$$

$$\vec{s}_3 = \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix}\begin{pmatrix} 0.76 \\ 0.24 \end{pmatrix} = \begin{pmatrix} 0.752 \\ 0.248 \end{pmatrix}$$

Qubits

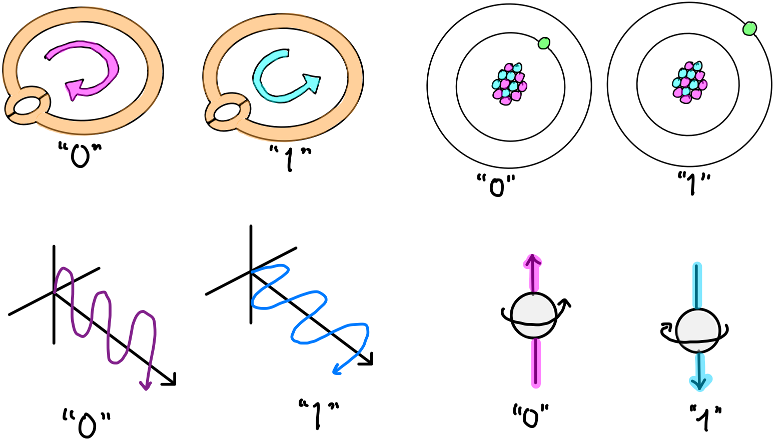
Extension of probabilistic bits to 2-level quantum systems:





Qubits

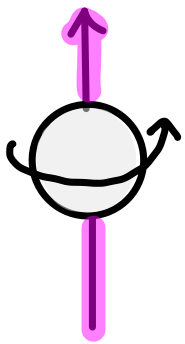
Quantum computers work by manipulating **qubits**.



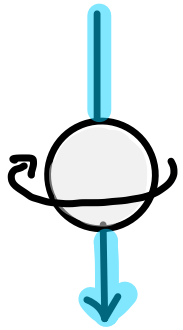
Mathematical representation of qubits

The vector space where qubits live is called **Hilbert space**.

$\mathcal{H} = \mathbb{C}^2$



"0"



"1"

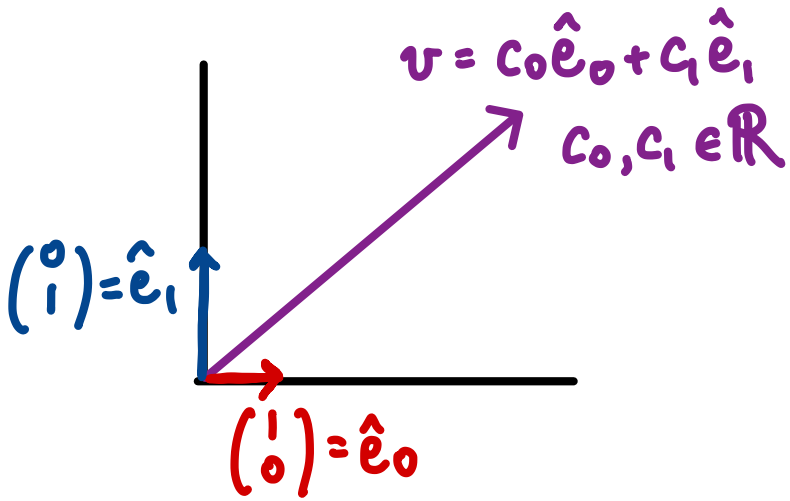
$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

These two vectors together form the **computational basis**.

The notation $|\cdot\rangle$ is called Dirac, or **bra-ket notation**. $|\cdot\rangle$ is a ket.

Mathematical representation of qubits

Qubit states are linear combinations of these basis states.



$\Rightarrow$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$
$$\alpha, \beta \in \mathbb{C}$$

The coefficients α and β can be complex, and are called **amplitudes**. "Valid" qubit state vectors have unit length:

$$|\alpha|^2 + |\beta|^2 = 1 \quad = \alpha\alpha^* + \beta\beta^*$$

Such states are called **normalized**.

Mathematical representation of qubits

Exercise: is $|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}e^{0.2i}|1\rangle$ a valid quantum state?

α

$|\alpha|^2 = \alpha\alpha^* = \frac{1}{\sqrt{3}} \cdot \frac{1}{\sqrt{3}} = \frac{1}{3}$

β

$|\beta|^2 = \sqrt{\frac{2}{3}}e^{0.2i} \cdot \sqrt{\frac{2}{3}}e^{-0.2i} = \frac{2}{3}$

$\rightarrow |\alpha|^2 + |\beta|^2 = 1$

Mathematical representation of qubits

A qubit in a linear combination

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

is in a **superposition** of the two basis states.

Note: a qubit in superposition isn't *"in both states simultaneously"*!

Using qubits for computation

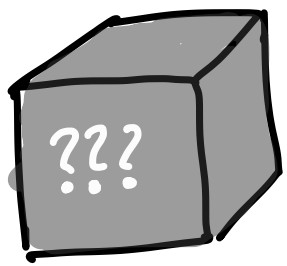
1. **Prepare** qubits in a **superposition**
2. Apply **operations** that **entangle** the qubits and manipulate the amplitudes
3. **Measure** qubits to extract an answer
4. Profit

...how do we do this?

Manipulating qubits

Manipulating a qubit changes its state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad |\alpha|^2 + |\beta|^2 = 1$$



$$|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle \qquad |\alpha'|^2 + |\beta'|^2 = 1$$

We need a mechanism for sending valid quantum state vectors to other valid quantum state vectors.

Manipulating qubits: unitary operations

Single-qubit states are manipulated by 2×2 unitary matrices. A matrix U is unitary if

$$U = \begin{pmatrix} * & * \\ * & * \end{pmatrix} \qquad UU^\dagger = I = U^\dagger U$$

$$I \Rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The $\dagger$ means "conjugate transpose":

$$U^\dagger = (U^*)^T$$

dagger

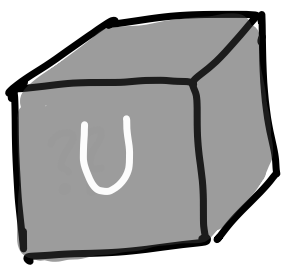
complex conjugate

Unitary operations, or **gates**, are **reversible**: if we apply U, we can undo it by applying $U^\dagger$.

Manipulating qubits: unitary operations

Unitary operations preserve the length of vectors, i.e., *maintain the normalization of qubit states*.

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ $\qquad |\alpha|^2 + |\beta|^2 = 1$



$UU^\dagger = U^\dagger U = \mathbb{1}$

$|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$ $\qquad |\alpha'|^2 + |\beta'|^2 = 1$

Manipulating qubits: unitary operations

Unitary operations act *linearly* on superpositions.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \longrightarrow U|\psi\rangle = U(\alpha|0\rangle + \beta|1\rangle)$$
$$= \alpha \cdot U|0\rangle + \beta \cdot U|1\rangle$$

This is a key contributor to the power of quantum computing!

Exercise: The matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

is unitary. Determine its action on the basis states. $\Rightarrow$ flips

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \quad \Rightarrow \text{NOT}$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

Example: X

Exercise: What does X do to the state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

$X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle$

X: NOT, Pauli X, bit flip ...

Example: H

Perhaps the most important unitary in quantum computing is the Hadamard gate (H):

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

This gate creates a *uniform superposition*:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle \right) = |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle \right) = |-\rangle$$

Example: H

What if we apply it twice?

$H^2|0\rangle = ?$

$H^2|1\rangle = ?$

} will see Wednesday!

For next time

Tomorrow's tutorial: TA office hour (meet your TA; get help installing PennyLane or configuring programming environment)

Content:

- Getting started with PennyLane
- Quantum circuits
- More unitary operations

Action items:

1. Set up Python programming environment with PennyLane v0.39 (go to tutorial or office hours if you need help)
2. See instructions on Canvas to access PrairieLearn and Piazza
3. Do assignment 0 for practice / review

Recommended reading:

- Codebook modules IQC and SQ.
- Nielsen and Chuang 1.1-1.2