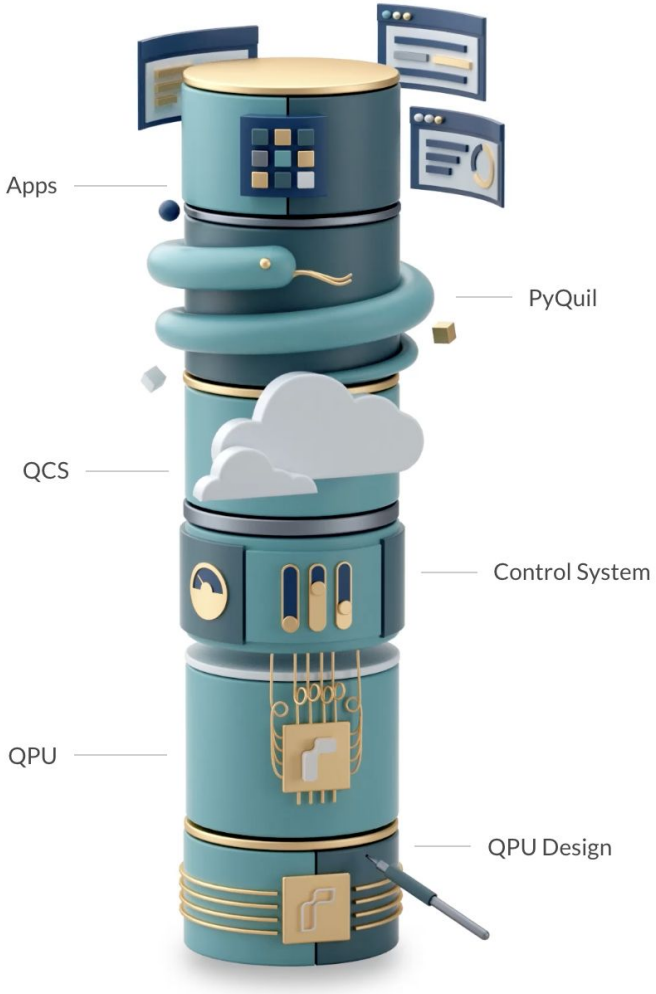
Quantum Information Summer School 2019, IBA

M. Sohaib Alam
Rigetti Computing
15 July, 2019



rigetti

The world's first **full-stack quantum computing** company.

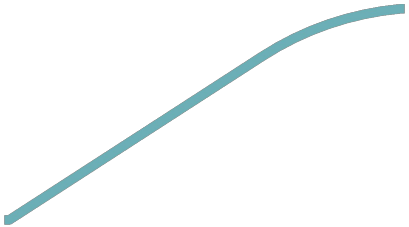
16-qubit QPUs currently operating on our cloud platform

100+ employees w/ $119M raised

Home of Fab-1, the world's first commercial quantum integrated circuit fab

Located in Berkeley, Calif. (R&D Lab) and Fremont, Calif.

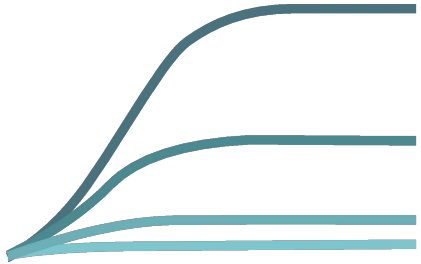
Classical computers have fundamental limits



Transistor scaling

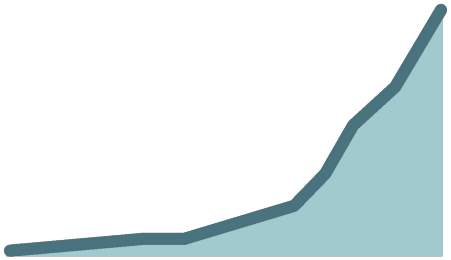
Economic limits with 10bn for next node fab

Ultimate single-atom limits



Returns to parallelization

Amdahl's law



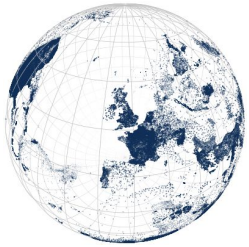
Energy consumption

Exascale computing project has its own power plant

Power density can melt chips

Why build a quantum computer?

Quantum computing power* scales exponentially with qubits
N bits can exactly simulate **log N qubits**

This compute unit....			
	Commodore 64	AWS M4 Instance 1 Million x Commodore 64	Entire Global Cloud 1 Billion x (1 Million x Commodore 64)
can exactly simulate:	**10 Qubits**	**30 Qubits**	**60 Qubits**

* More precisely ...

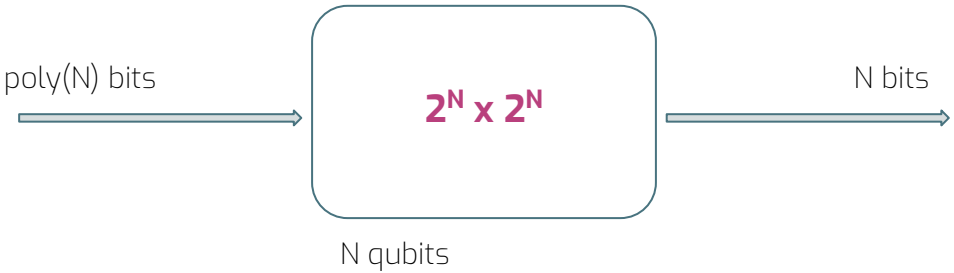
Why build a quantum computer?

For **N qubits** every time step (~100ns*) is an exponentially large **2^N x 2^N** complex **matrix multiplication**

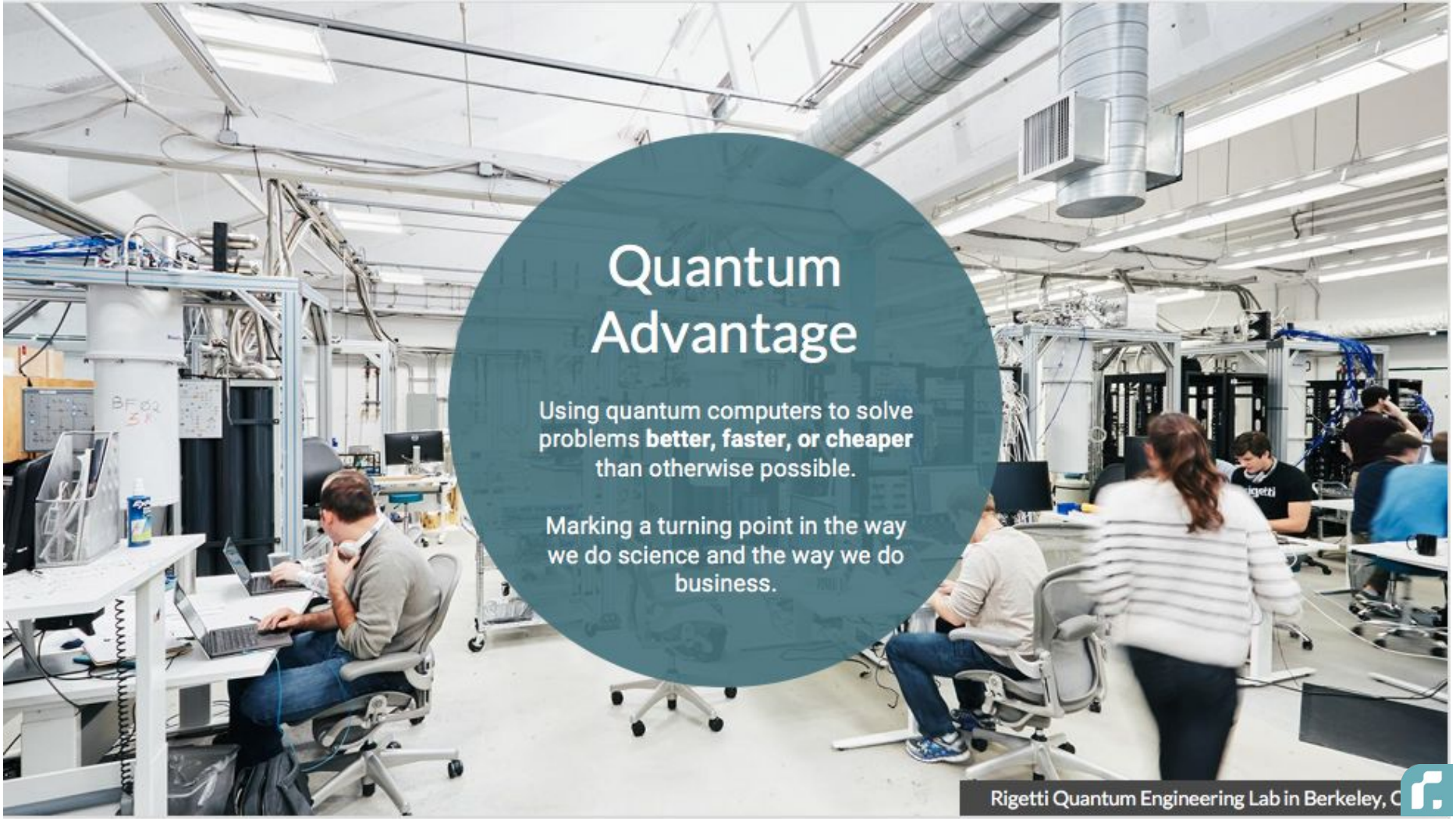
Crucial details:

- limited number of multiplications (hundreds to thousands) due to noise
- not arbitrary matrices (need to be easily constructed on a QC)
- small I/O, **poly(N)-bits in and N-bits out**

The “big-memory small pipe” mental model for quantum computing



* for superconducting qubit systems


Quantum Advantage
Using quantum computers to solve problems **better, faster, or cheaper** than otherwise possible.
Marking a turning point in the way we do science and the way we do business.
Rigetti Quantum Engineering Lab in Berkeley, C

Qubit States

Qubit: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1$

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

Measurement yields:

- ‘0’ with probability $|\alpha|^2$
- ‘1’ with probability $|\beta|^2$

Qubit States

Qubit: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1$

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

Bras $\langle 0| = (1,\ 0) \qquad \langle 1| = (0,\ 1) \qquad \langle \psi| = (\overline{\alpha}, \overline{\beta})$

Brackets
(Inner Product)

$$|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$$

$$\langle \phi|\psi\rangle = \overline{\gamma}\alpha + \overline{\delta}\beta = \overline{\langle \psi|\phi\rangle}$$

Qubit States

Qubit: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1$

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

Bras $\langle 0| = (1,\, 0) \qquad \langle 1| = (0,\, 1) \qquad \langle \psi| = (\overline{\alpha}, \overline{\beta})$

Normalization: $\langle \psi|\psi\rangle = 1 \quad \Rightarrow \quad |\alpha|^2 + |\beta|^2 = 1$

Qubit States

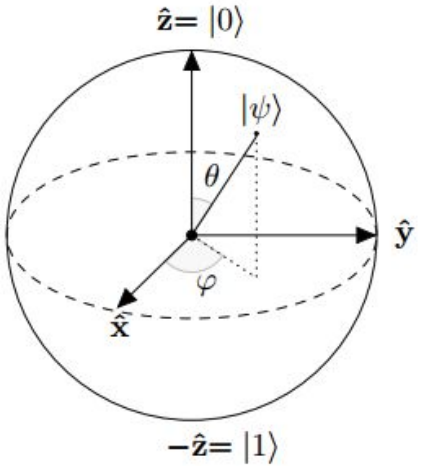
Qubit: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1$

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

Measurement yields:

- '0' with probability $|\langle 0|\psi\rangle|^2 = |\alpha|^2$
- '1' with probability $|\langle 1|\psi\rangle|^2 = |\beta|^2$

Qubit States: Bloch Sphere



$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$$

Qubit States: Multi-qubit states

Multiple qubits:

$$|\psi\rangle_{n-1} \otimes \ldots \otimes |\psi\rangle_2 \otimes |\psi\rangle_1 \otimes |\psi\rangle_0$$

Tensor product:

$$|\psi\rangle \otimes |\phi\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle)$$
$$= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle$$

Vector form:

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \otimes \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \\ \alpha_1 \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \alpha_0\,\beta_0 \\ \alpha_0\,\beta_1 \\ \alpha_1\,\beta_0 \\ \alpha_1\,\beta_1 \end{pmatrix}$$

Qubit States: Multi-qubit states

Associative:

$$(|\psi\rangle \otimes |\phi\rangle) \otimes |\gamma\rangle = |\psi\rangle \otimes (|\phi\rangle \otimes |\gamma\rangle)$$

Not commutative:

$$|\psi\rangle \otimes |\phi\rangle \neq |\phi\rangle \otimes |\psi\rangle$$

Qubit Operations

Unitary Operators:
(Gates)

$$UU^\dagger = U^\dagger U = I$$

$$|\psi\rangle \rightarrow |\psi'\rangle = U|\psi\rangle$$

Preserve inner product:

$$\langle\phi| \rightarrow \langle\phi'| = \langle\phi|U^\dagger$$

$$\langle\phi|\psi\rangle \rightarrow \langle\phi'|\psi'\rangle = \langle\phi|U^\dagger U|\psi\rangle = \langle\phi|\psi\rangle$$

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$	$a + b \in \mathbb{R}_+$ $a + b = 1$

Probability of 0

Probability of 1

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

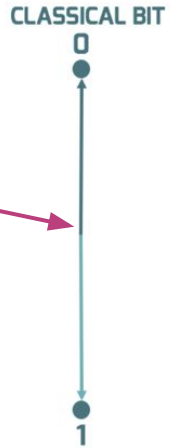
$|\alpha|^2$ = Probability of 0

$|\beta|^2$ = Probability of 1

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$



Bits vs. Probabilistic Bits vs. Qubits

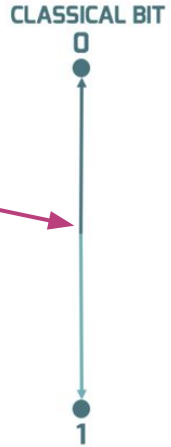
	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$

$$\text{qcoin} = \frac{1}{\sqrt{2}}\vec{0} + \frac{1}{\sqrt{2}}\vec{1}$$

$$\text{qcoin} = \frac{1}{\sqrt{2}}\vec{0} - \frac{1}{\sqrt{2}}\vec{1}$$

$$\text{qcoin} = \frac{1}{\sqrt{2}}\vec{0} - \frac{i}{\sqrt{2}}\vec{1} \quad \ldots$$

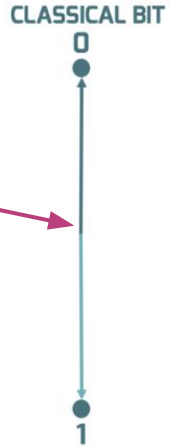


Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$

$$\vec{\text{qcoin}}(\theta) = \frac{1}{\sqrt{2}}\vec{0} + \frac{e^{i\theta}}{\sqrt{2}}\vec{1}$$

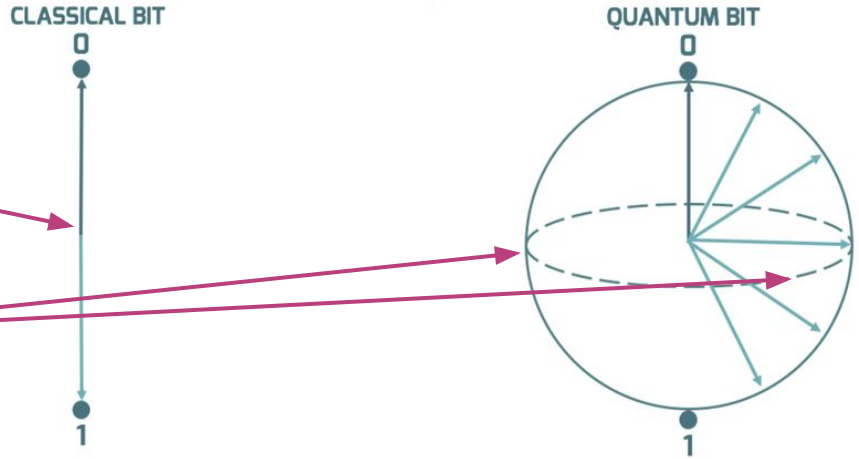


Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$

$$\vec{\text{qcoin}}(\theta) = \frac{1}{\sqrt{2}}\vec{0} + \frac{e^{i\theta}}{\sqrt{2}}\vec{1}$$



Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0,1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a+b \in \mathbb{R}_+$ $a+b=1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	

$$\vec{s} = \bigotimes_i^n b_i$$

Probability of bitstring x

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a+b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a+b=1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x \in \{0,1\}^n}$

$$\vec{s} = \bigotimes_{i}^{n} b_i$$

$$\vec{\psi} = \bigotimes_{i}^{n} \psi_i$$

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0,1\}$	Real vector $a+b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a+b=1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$

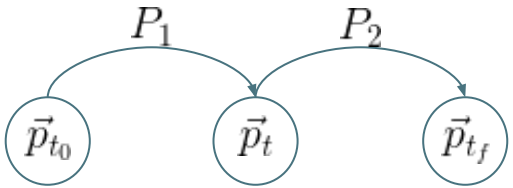
$$\vec{s} = \bigotimes_i^n b_i$$

$$\vec{\psi} = \bigotimes_i^n \psi_i$$

$|\alpha_x|^2$ = Probability of bitstring x

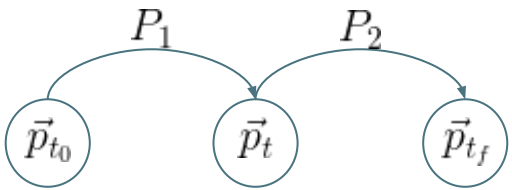
Bits vs. Probabilistic Bits vs. Qubits

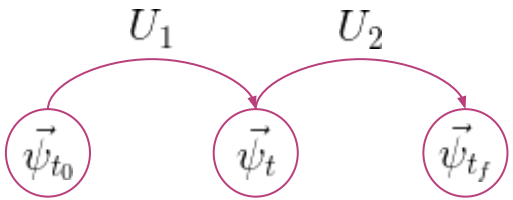
	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a+b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a+b=1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	



Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x \in \{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$





Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a+b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a+b=1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0,1\}$	Real vector $a+b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a+b=1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices

Sampling

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a + b \in \mathbb{R}_+$ $a + b = 1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x \in \{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices

Sampling

Born rule

$|\alpha_x|^2$ = Probability of bitstring x

Bits vs. Probabilistic Bits vs. Qubits

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a+b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a+b=1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices

Sampling

Born rule

Measurement

Qubit Operations: Pauli gates

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

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

Qubit Operations: Identity gate

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

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

Qubit Operations: Pauli-X (**NOT**) gate

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

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

Qubit Operations: Pauli-Y gate

$$Y|0\rangle = i|1\rangle \qquad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = i \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$Y|1\rangle = -i|0\rangle \qquad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = -i \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Qubit Operations: Pauli-Z gate

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

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

Qubit Operations: Hadamard gate

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

Qubit Operations: Hadamard gate

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

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

Qubit Operations: Multiple qubits

Examples:

$$I \otimes X \left(|0\rangle \otimes |0\rangle \right) = |0\rangle \otimes |1\rangle = I_1 X_0 |00\rangle = |01\rangle$$

$$X \otimes H \left(|0\rangle \otimes |0\rangle \right) = |1\rangle \otimes \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle \right) = X_1 H_0 |00\rangle = \frac{1}{2} \left(|10\rangle + |11\rangle \right)$$

Qubit Operations: Multiple qubits

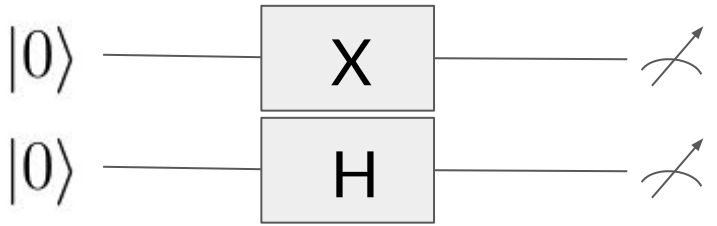
$$A \otimes B = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \otimes \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} = \begin{pmatrix} A_{00}[B] & A_{01}[B] \\ A_{10}[B] & A_{11}[B] \end{pmatrix}$$

$$= \begin{pmatrix} A_{00}B_{00} & A_{00}B_{01} & A_{01}B_{00} & A_{01}B_{01} \\ A_{00}B_{10} & A_{00}B_{11} & A_{01}B_{10} & A_{01}B_{11} \\ A_{10}B_{00} & A_{01}B_{01} & A_{11}B_{00} & A_{11}B_{01} \\ A_{10}B_{10} & A_{11}B_{11} & A_{11}B_{10} & A_{11}B_{11} \end{pmatrix}$$

Qubit Operations: Quantum Circuits

$$X|0\rangle = |1\rangle$$

$$X_1 H_0 |00\rangle$$



Quantum Programs

$$X|0\rangle = |1\rangle$$


```
from pyquil import Program, get_qc
from pyquil.gates import X

p = Program(X(0))
qc = get_qc('9q-generic-qvm')
results = qc.run_and_measure(p, trials=10)[0]

print (results)
```

```
[1 1 1 1 1 1 1 1 1 1]
```

Quantum Programs

$$X|0\rangle = |1\rangle$$


```
from pyquil import Program, get_qc
from pyquil.gates import X, MEASURE

p = Program()
p.declare('ro', 'BIT', 1)
p.inst(X(0))
p.inst(MEASURE(0, 'ro'))
p.wrap_in_numshots_loop(shots=10)

qc = get_qc('9q-generic-qvm')
results = qc.run(qc.compile(p))

print (results)
```

```
[[1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]]
```

Quantum Programs

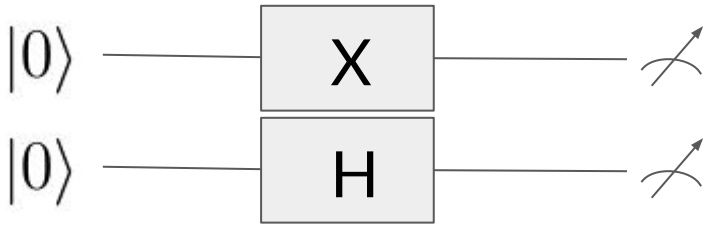
$$X|0\rangle = |1\rangle$$


```
from pyquil import Program, get_qc
from pyquil.gates import X, MEASURE

p = Program()
p.declare('ro', 'BIT', 1)
p.inst(X(0))
p.inst(MEASURE(0, 'ro'))
p.wrap_in_numshots_loop(shots=10)

qc = get_qc('9q-generic-qvm')
results = qc.run(qc.compile(p))

print (p)
```

```
DECLARE ro BIT[1]
X 0
MEASURE 0 ro[0]
```

Quantum Programs

$$X_1 H_0 \ |0\rangle_1 |0\rangle_0$$

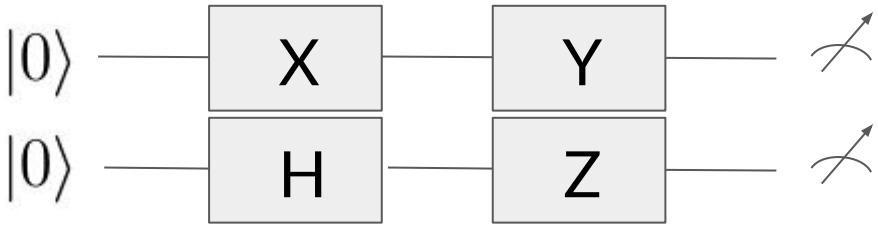



```
from pyquil import Program, get_qc
from pyquil.gates import X, H, MEASURE

p = Program()
ro = p.declare('ro', 'BIT', 2)
p.inst(H(0))
p.inst(X(1))
p.inst(MEASURE(0, ro[0]))
p.inst(MEASURE(1, ro[1]))
p.wrap_in_numshots_loop(shots=10)

qc = get_qc('9q-generic-qvm')
results = qc.run(qc.compile(p))

print (results)
```

```
[[0 1]
 [1 1]
 [0 1]
 [1 1]
 [1 1]
 [0 1]
 [1 1]
 [0 1]
 [0 1]
 [0 1]]
```

Quantum Programs

$$Y_1 Z_0 \, X_1 H_0 \; |0\rangle_1 |0\rangle_0$$




```
from pyquil import Program, get_qc
from pyquil.gates import X, Y, Z, H, MEASURE

p = Program()
ro = p.declare('ro', 'BIT', 2)
p += Program(H(0), X(1), Z(0), Y(1), MEASURE(0, ro[0]), MEASURE(1, ro[1]))
p.wrap_in_numshots_loop(shots=10)

qc = get_qc('9q-generic-qvm')
results = qc.run(qc.compile(p))

print (results)
```

```
[[1 0]
 [1 0]
 [0 0]
 [1 0]
 [1 0]
 [0 0]
 [1 0]
 [1 0]
 [0 0]
 [0 0]]
```

Quantum Dice

Goal: Create an N-sided dice using a quantum computer.

Quantum Dice

Question: What gate would we use?

$$H^{\otimes n}|0\rangle = \frac{1}{2^n}\sum_{z=0}^{n-1}|z\rangle$$

Quantum Dice

Question: How many qubits would we use?

$$2^n = N \Rightarrow n = log_2 N$$

Qubit Operations: Projections

Project a ket/bra along a given ket/bra via its corresponding **projection operator.**

For some $|\psi\rangle$ the corresponding projection operator is given by the **outer product**

$$P_\psi = |\psi\rangle\langle\psi| = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} (\overline{\alpha},\ \overline{\beta}) = \begin{pmatrix} |\alpha|^2 & \alpha\overline{\beta} \\ \beta\,\overline{\alpha} & |\beta|^2 \end{pmatrix}$$

e.g.

$$P_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},\ P_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},\ P_+ = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Qubit Operations: Controlled Operations

$$cU = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U$$

If qubit *1* is in the state |0> , apply *I* (identity) to qubit *0*
Else if qubit *1* is in the state |1>, apply *U* to qubit *0*

For example,

$$CNOT = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Entangled States

A state that cannot be written as a product state, i.e.

$$|\psi\rangle \neq |\xi\rangle \otimes |\phi\rangle$$

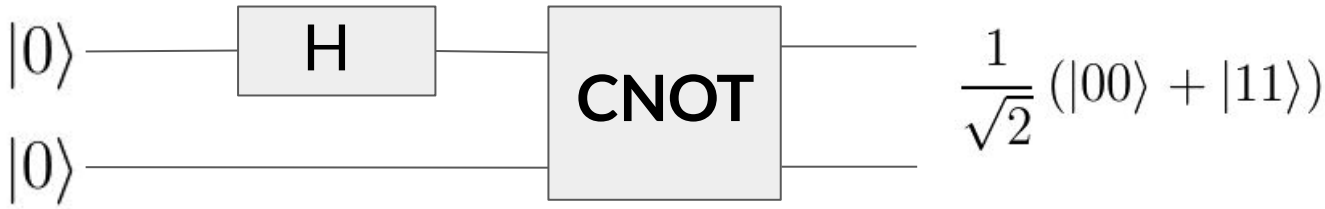
An example of a state that is not entangled:

$$\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

An example of a state that is entangled:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Bell State via CNOT




```
from pyquil import Program
from pyquil.gates import H, CNOT
from pyquil.api import WavefunctionSimulator

p = Program(H(1))
p += Program(CNOT(1, 0))
wfn = WavefunctionSimulator().wavefunction(p)

print (wfn)
```

```
(0.7071067812+0j)|00> + (0.7071067812+0j)|11>
```

Measurement in some arbitrary basis

Computational basis: $\{|0\rangle, |1\rangle\}$

Measurement yields:

- '0' with probability $\langle\psi|P_0|\psi\rangle = \langle\psi|0\rangle\langle 0|\psi\rangle = |\langle 0|\psi\rangle|^2$
- '1' with probability $\langle\psi|P_1|\psi\rangle = \langle\psi|1\rangle\langle 1|\psi\rangle = |\langle 1|\psi\rangle|^2$

Measurement in some arbitrary basis

Some other basis: $\{|0'\rangle = U|0\rangle, |1'\rangle = U|1\rangle\}$

Measurement yields:

- 0' with probability $\langle\psi|P_{0'}|\psi\rangle = \langle\psi|0'\rangle\langle 0'|\psi\rangle = \langle\psi|U|0\rangle\langle 0|U^\dagger|\psi\rangle = \langle\psi'|0\rangle\langle 0|\psi'\rangle = |\langle 0|\psi'\rangle|^2$
- 1' with probability $\langle\psi|P_{1'}|\psi\rangle = \langle\psi|1'\rangle\langle 1'|\psi\rangle = \langle\psi|U|1\rangle\langle 1|U^\dagger|\psi\rangle = \langle\psi'|1\rangle\langle 1|\psi'\rangle = |\langle 1|\psi'\rangle|^2$

$$|\psi'\rangle = U^\dagger|\psi\rangle$$

Measurement in some arbitrary basis

Measurement of $|\Psi\rangle$ in some basis $\{U|0\rangle, U|1\rangle\}$

= Measurement of $U^{\dagger}|\Psi\rangle$ in standard/computational basis $\{|0\rangle, |1\rangle\}$

Quantum Teleportation

Goal: Teleport a Qubit!

Quantum Teleportation

Scenario:

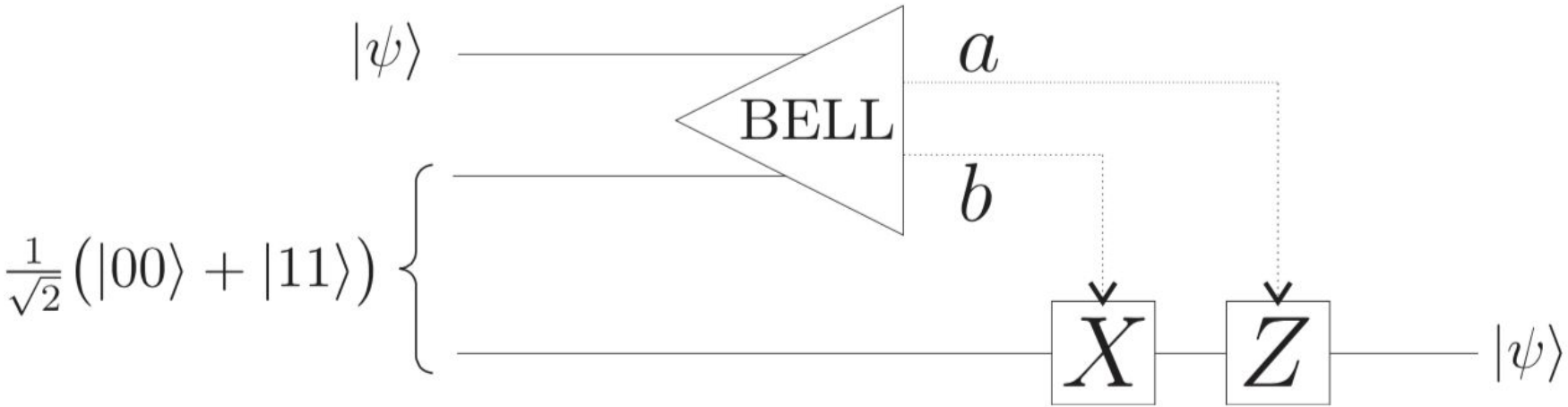
- Alice is in possession of a qubit $|\Psi>$, which she would like to teleport over to Bob, who is at some distant location.

Quantum Teleportation

Protocol:

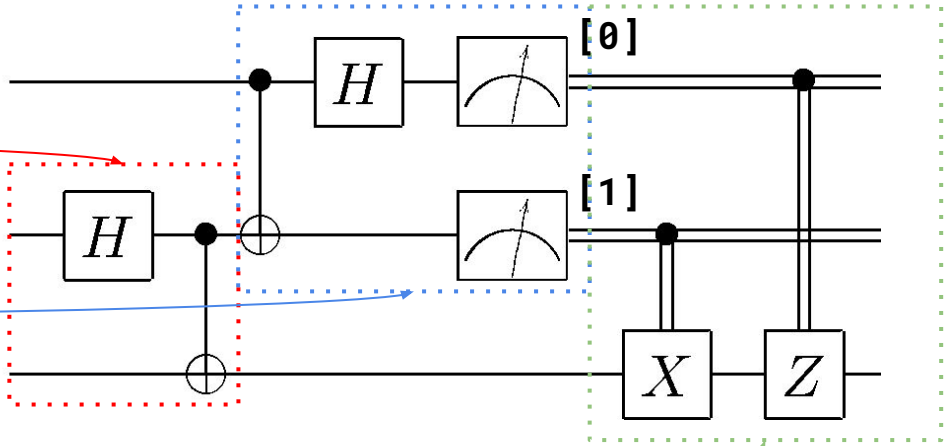
- Create a Bell state, giving one qubit each to Alice and Bob
- Have Alice measure both her qubits in the Bell basis, and send her results to Bob
- Have Bob *conditionally* apply gates to his qubits, based off Alice's measurements, to reconstruct the original qubit at his location

Quantum Teleportation



Quantum Teleportation

```
DEFCIRCUIT TELEPORT A q B:
    # Bell pair
    H         A
    CNOT      A B

    # Teleport
    CNOT      q A
    H         q
    MEASURE   q [0]
    MEASURE   A [1]

    # Classically communicate measurements
    JUMP-UNLESS @SKIP [1]
    X B
    LABEL @SKIP
    JUMP-UNLESS @END [0]
    Z B
    LABEL @END

# If Alice's qubits are 0 and 1
# and Bob's is 5
TELEPORT 0 1
```



Classical Control in pyQuil

```
from pyquil import Program
from pyquil.gates import I, X
from pyquil.api import WavefunctionSimulator

p = Program(X(0))
ro = p.declare('ro', 'BIT', 1)
p.measure(0, ro[0]).if_then(ro[0], Program(X(1)), Program(I(1)))
wfn = WavefunctionSimulator().wavefunction(p)

print (wfn)
```

```
(1+0j)|11>
```

Classical Control in pyQuil

```
from pyquil import Program
from pyquil.gates import I, X
from pyquil.api import WavefunctionSimulator

p = Program(I(0))
ro = p.declare('ro', 'BIT', 1)
p.measure(0, ro[0]).if_then(ro[0], Program(X(1)), Program(I(1)))
wfn = WavefunctionSimulator().wavefunction(p)

print (wfn)
```

```
(1+0j)|00>
```

Deutsch’s Algorithm

Goal: Given a function f: {0, 1} -> {0, 1}, determine whether it is constant (i.e. f(0) = f(1)) or balanced (i.e. f(0) != f(1)) in the minimum number of steps, assuming an oracle/black-box for the function.

Deutsch's Algorithm

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

Deutsch's Algorithm

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

$$U_f \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = (-1)^{f(0)} \left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

Deutsch's Algorithm

(a) f(0) = f(1)

$$U_f \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \pm \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

(b) f(0) != f(1)

$$U_f \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \pm \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

Recall: $$H|0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right), \quad H|1\rangle = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right), \quad H^\dagger = H$$

Grover’s Search Algorithm

Goal: Given a function f: {0, 1}^n -> {0, 1}, find the bitstring $x \in \{0, 1\}^n$, such that f(x) = 1.

Grover's Search Algorithm

Assume a quantum black box

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

Grover's Search Algorithm

Prepare the query register as

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle = \frac{1}{\sqrt{N}}|\psi_{good}\rangle + \sqrt{\frac{N-1}{N}}|\psi_{bad}\rangle$$

where

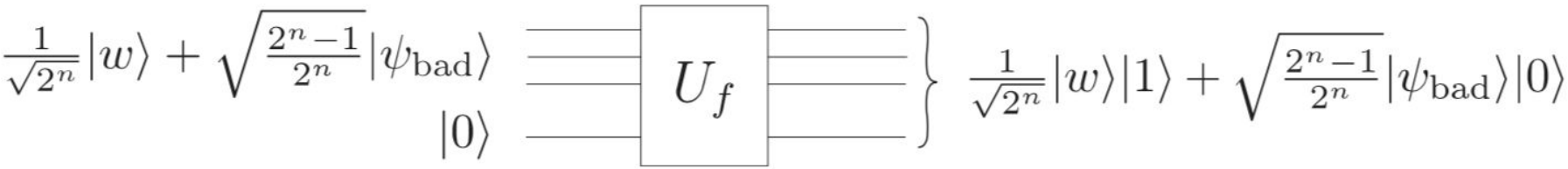
$$|\psi_{good}\rangle = |w\rangle$$

('w' is the bitstring to be found)

$$|\psi_{bad}\rangle = \frac{1}{\sqrt{N-1}}\sum_{x\in X_{bad}}|x\rangle$$

Grover's Search Algorithm

Oracle/Blackbox



Grover's Search Algorithm

Define a phase-shift operator

$$U_{\psi^\perp} : \begin{cases} |x\rangle \rightarrow -|x\rangle, & \langle x|\psi\rangle = 0 \\ |\psi\rangle \rightarrow |\psi\rangle \end{cases}$$

When |Ψ> is the equal-superposition state ('quantum dice'), we can write

$$U_{\psi^\perp} = 2|\psi\rangle\langle\psi| - I$$

Grover's Search Algorithm

Inversion about the mean:

$$U_{\psi^\perp} \sum_{i=0}^{2^n-1} \alpha_i |i\rangle = \sum_{i=0}^{2^n-1} (2\langle\alpha\rangle - \alpha_i)|i\rangle$$

Grover's Search Algorithm

Prepare target qubit as ZH|0> = HX|0> ~ (|0> - |1>) to get phase 'kick back' from applying oracle/blackbox

$$U_f|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = (-1)^{f(x)}|x\rangle \left(\frac{|0\rangle - |1\rangle}{2}\right)$$

Grover's Search Algorithm

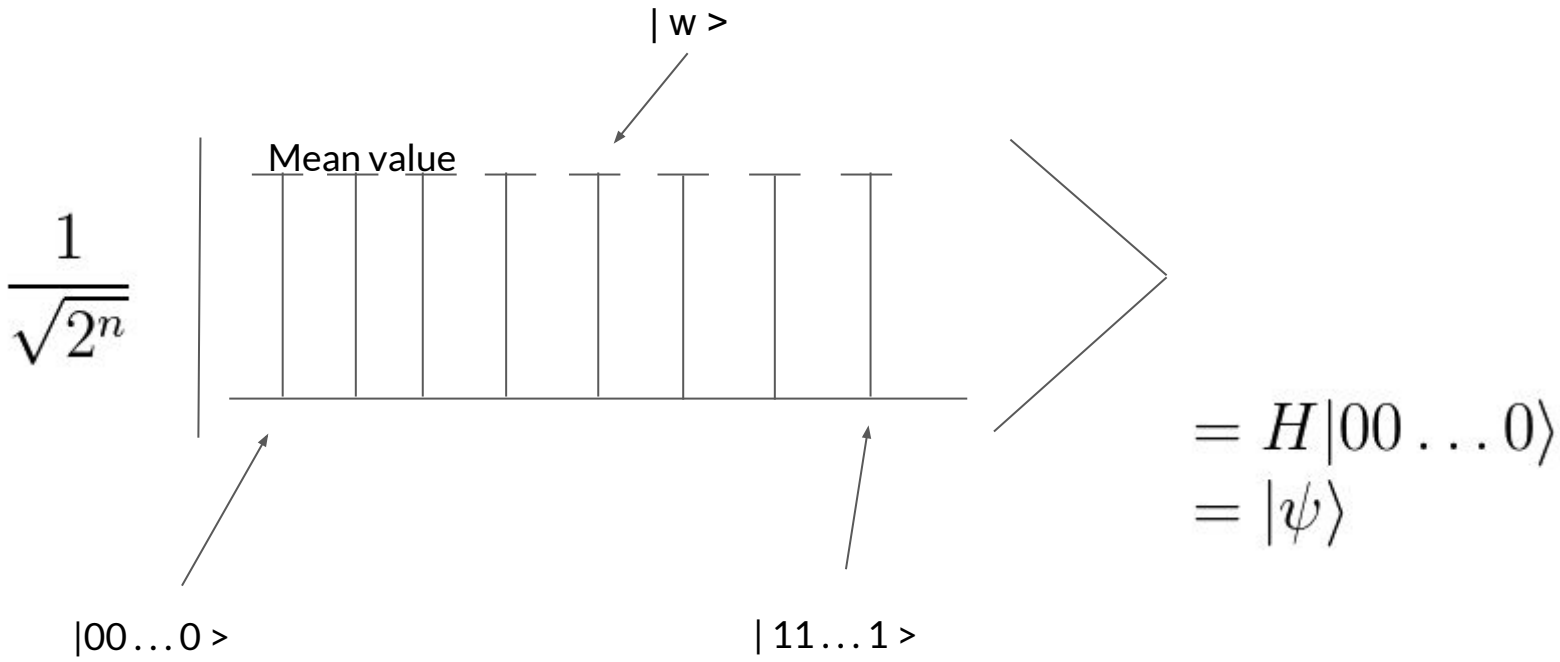
- Prepare the initial state

$$H^{\otimes n} \otimes (H_0 Z_0)|00\ldots0\rangle$$

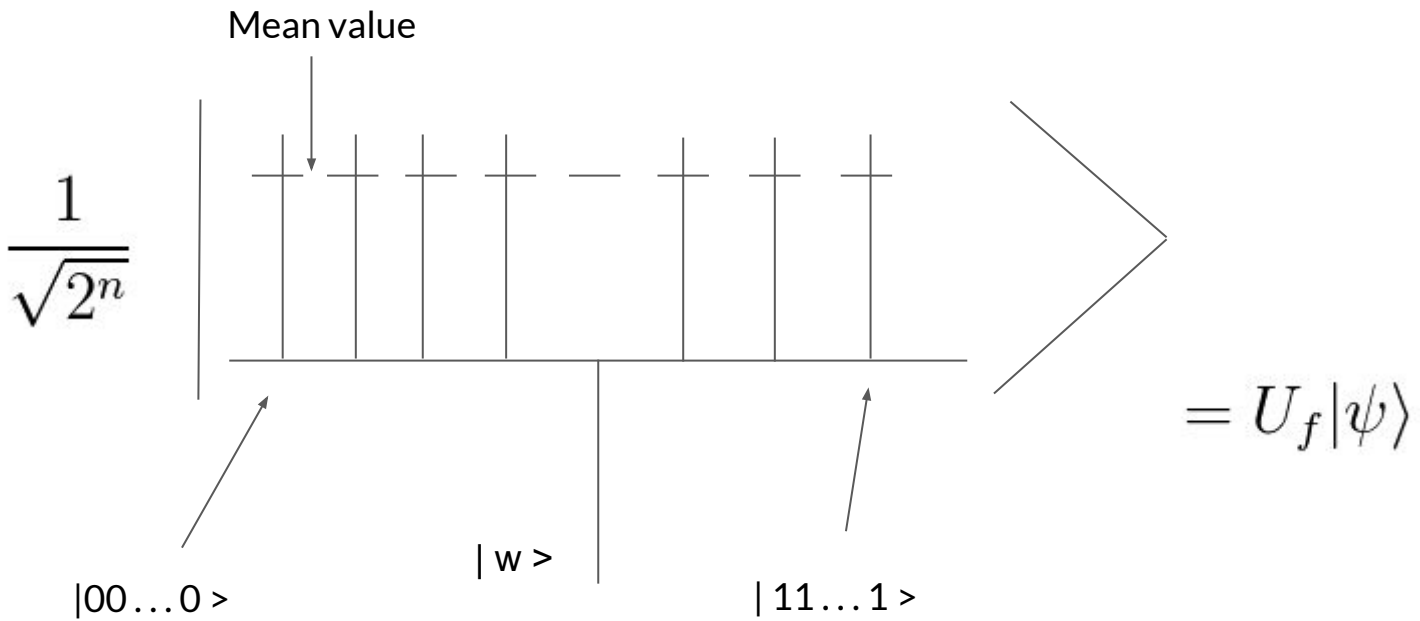
- Apply the Grover iterate O(√N) times

$$G = U_{\psi^\perp} U_f$$

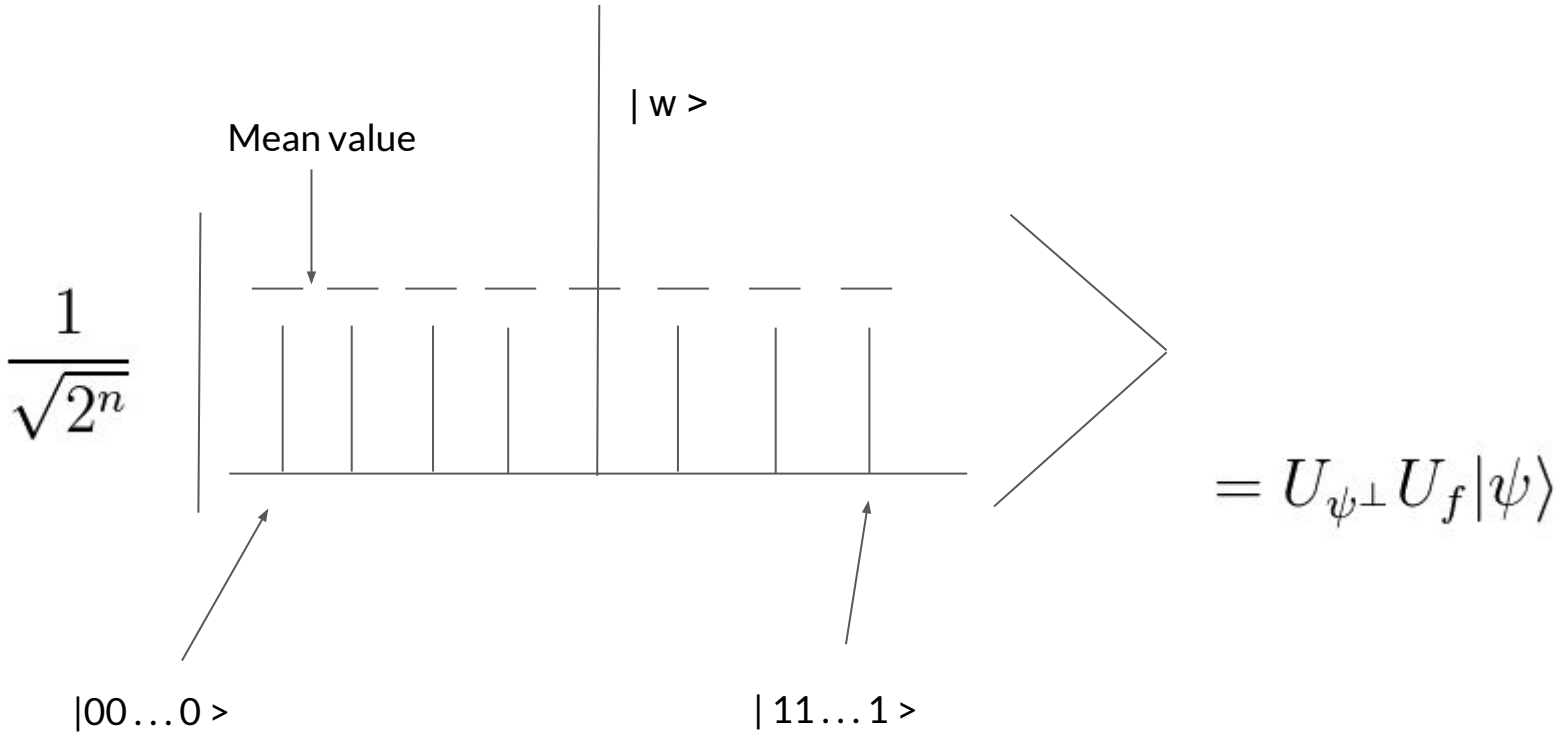
Grover's Search Algorithm



Grover's Search Algorithm



Grover's Search Algorithm



Variational Quantum Eigensolver

Goal: Find the ground state of some Hamiltonian, *H*.

Procedure:

- Prepare some trial state $|\Psi; \theta>$
- Calculate expectation value $<\Psi; \theta|H|\Psi; \theta>$
- Find the values of θ that will minimize the above

Quantum Approximate Optimization Algorithm

Goal: Given binary constraints over bitstrings

$$z \in \{0,1\}^n$$

$$C_\alpha(z) = \begin{cases} 1 & \text{if } z \text{ satisfies the constraint } \alpha \\ 0 & \text{if } z \text{ does not} \end{cases}$$

Find the bitstring that maximizes the objective function

$$\text{argmax}_z C(z) = \text{argmax}_z \sum_{\alpha=1}^{m} C_\alpha(z)$$

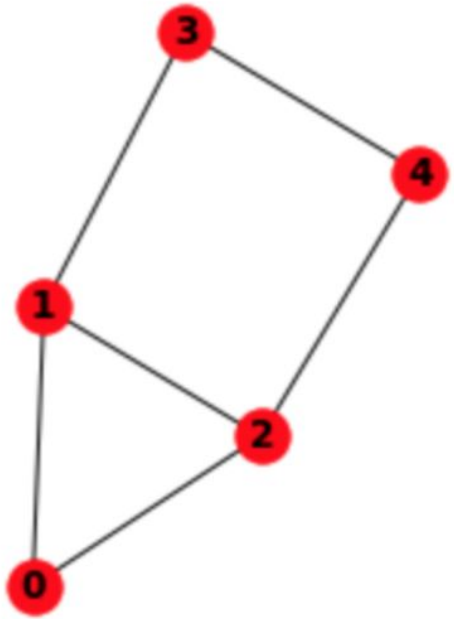
Quantum Approximate Optimization Algorithm

MaxCut problem:

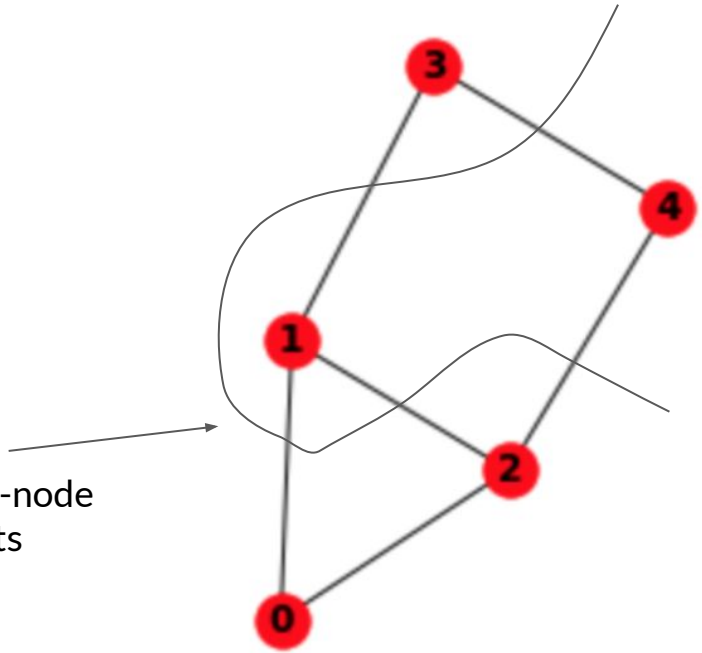
Given some undirected graph with arbitrary (non-negative) weights, find a partition $(S, \overline{S})$ of the graph's nodes (a 'cut' of the graph) that maximizes the weights along the cut

$$\sum_{i \in S,\, j \in \overline{S}} w_{ij}$$

Quantum Approximate Optimization Algorithm



Quantum Approximate Optimization Algorithm



MaxCut for simple 5-node graph with all weights either 0 or 1.

MaxCut solution (as a bitstring):

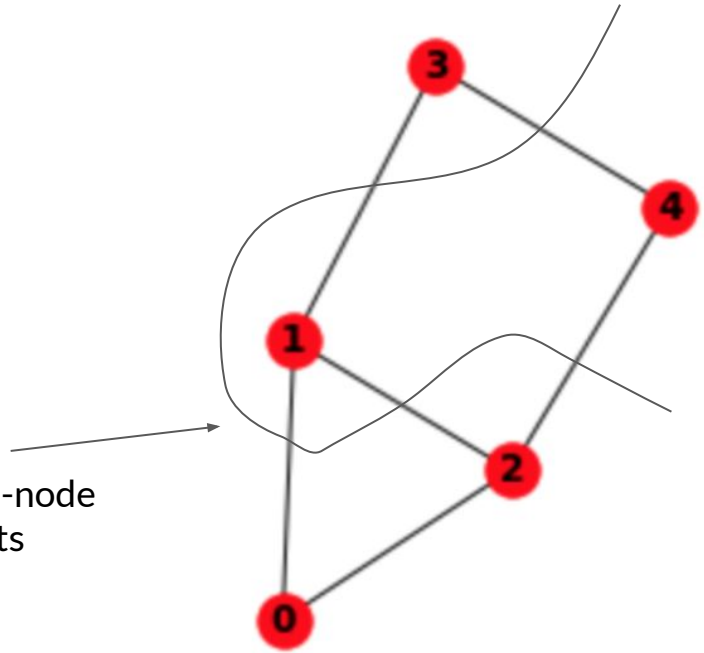
01001 OR 10110

On a quantum computer:

$$\frac{1}{\sqrt{2}}\left(|01001\rangle + |10110\rangle\right)$$

(ideally)

Quantum Approximate Optimization Algorithm



MaxCut for simple 5-node graph with all weights either 0 or 1.

MaxCut objective function:

$$\sum_{i \in S,\, j \in \overline{S}} w_{ij}$$

On a quantum computer:

$$\frac{1}{2} \sum_{i,\, j \in V} w_{ij} \frac{1 - Z_i Z_j}{2}$$

Quantum Approximate Optimization Algorithm

- Prepare equal-superposition as initial state $|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle$
- Define the ‘mixer’ operator $B = \sum_{j=1}^{n} X_j, \quad U(B, \beta) = e^{-i\beta B}$
- Define the ‘cost’ operator $C = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \frac{1 - Z_i Z_j}{2}, \quad U(C, \gamma) = e^{-i\gamma C}$
- Apply $U(B, \beta_p)U(C, \gamma_p) \ldots U(B, \beta_1)U(C, \gamma_1)|s\rangle$ and sample/measure

Quantum Approximate Optimization Algorithm

```
import numpy as np
from pyquil import Program
from pyquil.api import WavefunctionSimulator
from pyquil.paulis import sI, sX, sY, sZ, exponential_map

angle = np.pi / 8
pauli_sum = sX(1) * sY(0) + sI(1) * sY(0)

p = Program()
for ps in pauli_sum:
    p += exponential_map(ps)(angle)

wfn_sim = WavefunctionSimulator()
wfn = wfn_sim.wavefunction(p)

print (wfn)
```

(0.8535533906+0j)|00> + (0.3535533906+0j)|01> + (-0.1464466094+0j)|10> + (0.3535533906+0j)|11>

Noise and Quantum Computation

‘Pure’ quantum states

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

evolve via Unitary operations

$$|\psi\rangle \rightarrow |\psi'\rangle = U|\psi\rangle \qquad UU^\dagger = U^\dagger U = I$$

Noise and Quantum Computation

More generally, quantum states are described by “Density Matrix”

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

evolving via Kraus operations (“quantum channel”)

$$\rho \to \sum_i K_i \rho K_i^\dagger, \qquad \sum_i K_i^\dagger K_i = I$$

Noise and Quantum Computation

For example,

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

Not to be confused with

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

Noise and Quantum Computation

Example of quantum channel/set of Kraus operators/noise model:

$$\{\sqrt{p}X, \sqrt{1-p}Z\}$$

Quantum state passing through the channel/experiencing the noise transforms to:

$$\rho \rightarrow pX^{\dagger}\rho X + (1-p)Z^{\dagger}\rho Z$$

Thanks, and keep in touch!

sohaib@rigetti.com

Join our Slack channel: rigetti-forest.slack.com