Introduction to Quantum Computing

Toni Bluher
Math Research Group, NSA

2018 Women and Mathematics Program

Goals of this Talk

- Give flavor for quantum computation
- Give historical perspective
- Emphasize similarity with “classical” computer programming

Welcome to the Quantum World!

- Quantum mechanics developed 1900-1920, explains and predicts natural phenomena at particle level.
- Polynomial-time quantum-mechanical processes take exponential time to simulate on a classical computer.
- Turning this around, quantum-mechanical systems, if designed cleverly, might solve math problems with better asymptotics than Turing machines!
- This led to notion of quantum computation (Feynman, Deutsch, ...) in 1970's and 1980's.

Quantum Simulation

The quantum-mechanical computation of one molecule of methane requires 10^{42} *grid points. Assuming that at each point we have to perform only 10 elementary operations, and that the computation is performed at the extremely low temperature* $T = 3 \times 10^{-3} K$*, we would still have to use all the energy produced on Earth during the last century.*

– R. P. Poplavskii, 1975

Quantum Computation

- Quantum particles with known polarization, spin, *etc.* play the role of zeroes or ones.
- Quantum operations simulate Turing machine operations such as XOR, AND, NOT. (Some caveats)
- Classical computer: n bits $\leftrightarrow$ 2^n values.

  Quantum computer: n bits $\leftrightarrow$ 2^n-dimensional over $\mathbb{C}$
- Large state space leads to massive parallelism, *e.g.* quantum computer finds $f(x)$ for all x simultaneously!
- Quantum Fourier transform (QFT) runs polynomial time!

Shor's Algorithm

- By computing all pairs $(x, f(x))$ and then applying the Quantum Fourier Transform (QFT), you learn the period of $f(x)$ in polynomial time.

- Shor's algorithm (1994) factors N by finding the period of $f(x) = g^x \pmod{N}$.

Shor’s algorithm

- Polynomial-time factoring and discrete log (if we had a quantum computer)
- Would break most public key cryptography
- Impetus for development of “quantum resistant cryptography” (NIST)

If computers that you build are quantum,
Then spies everywhere will all want 'em.
Our codes will all fail,
And they'll read our email,
Till we get crypto that's quantum,
and daunt 'em.

Jennifer and Peter Shor


Google
quantum computers will
quantum computers will **destroy bitcoin**
quantum computers will **never work**
quantum computers will **be possible**
quantum computers will **change everything**
quantum computers will **make your laptop**
quantum **computing** will **fail**
Business
Privacy

Shor's algorithm

- Shor's algorithm has several steps for factoring n-bit integer N:
  1. Create a superposition over all $0 \leq x < 2^{2n}$;
  2. Compute all pairs $(x, g^x \pmod{N})$;
  3. Use QFT to find period of $g^x \pmod{N}$;
  4. Factor N

Bits vs. Qubits

- *n*-bit classical register might consist of *n* [circuits?] and 0/1's are represented by [on/off?].
- *n*-qubit quantum register might consist of *n* [trapped ions?] and 0/1's are represented by [spin left/right?]
- Classical error correction is handled with error correction codes, at small additional cost.
- Quantum error correction is much more difficult, great cost.
- This talk: ignore error correction and just pretend all qubits are perfect.

Classical versus quantum integers

- An *n*-bit classical register holds an *n*-bit integer in binary, *e.g.* 11001.
- An *n*-bit quantum register holds a nonzero $\mathbb{C}$-linear combination of such:

$$\alpha_0|00000\rangle + \alpha_1|00001\rangle + \cdots + \alpha_{31}|11111\rangle.$$

- Abbreviate as $\sum \alpha_i|i\rangle$. This is called a *quantum integer* or *quantum state* or *superposition*.
- No physics experiment would distinguish $\sum \alpha_i|i\rangle$ from $\sum \lambda\alpha_i|i\rangle$, so they are equivalent.
- $(2^n - 1)$-dimensional projective space over $\mathbb{C}$.

Measurement

- Measurement of bit or qubit gives two possible values, 0 or 1. That is where the similarity stops.
- Measurement of a classical bit is deterministic.
- Measurement of a quantum bit is probabilistic, and causes the state to collapse to the part of the superposition that is consistent with the measurement.
- Think of quantum state as a good liar with a superposition of possible alibis. If you ask it a question, it comes up with an answer. Answers to future questions are consistent with that one, so you can't catch it in a lie.

Measurement (continued)

- Given $\sum \alpha_i |i\rangle$, suppose we measure the low qubit.
- Prob(measure 0) is $p_0 = \sum_{i\ even} |\alpha_i|^2 / A$ and Prob(measure 1) is $p_1 = \sum_{i\ odd} |\alpha_i|^2 / A$, where $A = \sum_i |\alpha_i|^2$.
- If 0 is measured, resulting state is $\sum_{i\ even} \alpha_i |i\rangle$.
- If 1 is measured, resulting state is $\sum_{i\ odd} \alpha_i |i\rangle$.

Measurement – example

- Begin with $ABC = |000\rangle - 2|100\rangle + |111\rangle$.
- Meas(C). $p_0 = (1+4)/(1+4+1) = 5/6$; $p_1 = 1/6$.
- Suppose you measure 0. New state is $|000\rangle - 2|100\rangle$. Now measure B, definitely get 0 and the state remains the same. Now measure A, get 0 with probability 1/5 (resulting in $|000\rangle$) or 1 with probability 4/5 (resulting in $-2|100\rangle = |100\rangle$).
- Exercise: Suppose you measure all the qubits of $\sum \alpha_i |i\rangle$, in any order. Then Prob(measure $|j\rangle$) is $|\alpha_j|^2 / \sum_i |\alpha_i|^2$.

Gates

- A classical computer has NOT, XOR, and AND gates. One can also set a bit to 0 or 1.
- Quantum gates are always reversible, and they act on the whole superposition at once.

"Semi-classical" quantum gates

- NOT(A): $A \oplus := 1$
- CNOT(A, B): $B \oplus := A$ (analogue of XOR)
- TOFF(A, B, C): $C \oplus := A \mathbin{\&} B$ (analogue of AND)

- TOFF is much more costly than CNOT, which is much more costly than NOT.
- Input qubits are required to be distinct, *e.g.* $A \oplus := A$ is not allowed.

Semi-classical quantum gates – example

- Begin with $ABC = |000\rangle - 2|100\rangle + |111\rangle$.
- NOT(B): $ABC = |010\rangle - 2|110\rangle + |101\rangle$
- $C \oplus := B$: $ABC = |011\rangle - 2|111\rangle + |101\rangle$
- $A \oplus := B \mathbin{\&} C$: $ABC = |111\rangle - 2|011\rangle + |101\rangle$
- Exercise: if we repeat the above commands in the reverse order, we get back to the original state.

  Hint: Each semi-classical quantum gate has order 2, *i.e.* , is equal to its own inverse.

Example: implementing controlled swap

- CSWAP(A, B, C): If $A = 1$, then Swap(B, C).
- e.g. $ABC = |111\rangle - 2|011\rangle + |101\rangle$ is sent to $|111\rangle - 2|011\rangle + |110\rangle$.
- Want to implement CSWAP only with NOT, CNOT, and TOFFOLI.
- It's tempting to measure A, but that would collapse the state!
- Instead, try this: $B \oplus := C$; $C \oplus := A \,\&\, B$; $B \oplus := C$.
- Exercise: Show those three commands implement CSWAP.

Example: mod 4 addition

- Quantum algorithm for $(a, b) \mapsto (a, a + b \pmod 4)$:

$$
\begin{array}{cc}
c_1 = a_0 b_0 & \\
a_1 & a_0 \\
b_1 & b_0 \\
\hline
a_1 \oplus b_1 \oplus c_1 & a_0 \oplus b_0
\end{array}
$$

- Initial: $A_1 A_0 B_1 B_0 = \sum c_{ab} |a, b\rangle$, $a = a_1 a_0$, $b = b_1 b_0$

$$
\begin{array}{lll l}
B_1 & \oplus:= & A_0 \ \& \ B_0 & (B_1 = b_1 \oplus c_1) \\
B_1 & \oplus:= & A_1 & (B_1 = a_1 \oplus b_1 \oplus c_1) \\
B_0 & \oplus:= & A_0 & (B_0 = a_0 \oplus b_0)
\end{array}
$$

Hadamard gate

- $H(|0\rangle) = |0\rangle + |1\rangle, \qquad H(|1\rangle) = |0\rangle - |1\rangle.$
- $H^2 = I.$
- Example: If $ABC = |000\rangle - 2|100\rangle + |111\rangle$ then applying Hadamard to A yields

$$(|000\rangle + |100\rangle) - 2(|000\rangle - |100\rangle) + (|011\rangle - |111\rangle)$$
$$= -|000\rangle + 3|100\rangle + |011\rangle - |111\rangle.$$

- Exercise: Apply Hadamard a second time and verify you get back to the original.
- Verify that the code $H(A)$; $A \oplus := B$; $H(A)$ takes $\sum_{a,b\in\{0,1\}} c_{ab}|ab\rangle$ to $\sum_{a,b\in\{0,1\}} (-1)^{ab} c_{ab}|ab\rangle$.

Creating a superposition

- To create a superposition, begin with all-0's: $ABC = |000\rangle$. Then apply H to each qubit.

- $H(A)$: $|000\rangle + |100\rangle$
- $H(B)$: $(|000\rangle + |010\rangle) + (|100\rangle + |110\rangle)$
- $H(C)$: $(|000\rangle + |001\rangle) + (|010\rangle + |011\rangle) + (|100\rangle + |101\rangle) + (|110\rangle + |111\rangle) = \sum_{i=0}^{7} |i\rangle$

Shor's algorithm

- To factor n-bit integer N:
  1. Create a superposition over all $0 \leq x < 2^{2n}$;
  2. Compute all pairs $(x, g^x \pmod{N})$;
  3. Use QFT to find period of $g^x \pmod{N}$;
  4. Factor N
- To do step 1: Start with $3n$ qubits all equal to 0. Apply H to first $2n$ qubits. Results in $\sum_{0 \leq i < 22n} |i\rangle|0\rangle_n$.
- In second step, quantum gates will be applied to compute $g^x \pmod{N}$.

References

- Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary Edition, Cambridge University Press, 2010.

  Affectionately called "Mike and Ike"; has been a standard reference for almost two decades.

- Eleanor Rieffel and Wolfgang Polak, *Quantum Computing – A Gentle Introduction*, MIT Press, 2011. (A great book!)
- Peter W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing, 26(5): 1484–1509, 1997.
- Andrew Childs, online lectures and lecture notes, https://www.cs.umd.edu/∼amchilds/qa/qa.pdf