09. Quantum Information Theory, Part II

I. Quantum Computation

- *General Goal*: To use the inaccessible arbitrarily large amount of information encoded in qubits to perform computations in "quantum parallel" (*i.e.*, in record time!).

- *Initial (modest) Goal*: To compute all possible values of a function f in a single computation.

- *First Question*: Can classical computations be done using qubits instead of classical bits?
  - *Can transformations on qubits be defined that reproduce the transformations on bits that are needed to implement a classical computer.*

Classical Computation Using Bits

To implement a classical computer, it suffices to have an AND transformation and a NOT transformation on classical bits defined by the following:

$$\begin{aligned} 0 \; AND \; 0 &= 0 \\ 0 \; AND \; 1 &= 0 \\ 1 \; AND \; 0 &= 0 \\ 1 \; AND \; 1 &= 1 \end{aligned} \qquad \begin{aligned} NOT \; 0 &= 1 \\ NOT \; 1 &= 0 \end{aligned}$$

- *AND takes two input bits and produces one output bit.*
- *NOT takes one input bit and produces one output bit.*

- *Initial problem*: Transformations on qubits are *reversible*: the number of input qubits *always* must equal the number of output qubits.

> *Why?* *Qubit transformations are operators on vector spaces. And an operator defined on an n-dim vector space (e.g., n-qubit space) that acts on n-dim vectors (e.g., n qubits) can only spit out n-dim vectors.*

Solution: The "Controlled-controlled-NOT" CC_{NOT} operator.

- Changes the third target qubit if the first two control qubits are $|1\rangle|1\rangle$, and leaves it unchanged otherwise.

$$CC_{NOT}|0\rangle|0\rangle|0\rangle = |0\rangle|0\rangle|0\rangle \qquad CC_{NOT}|0\rangle|1\rangle|1\rangle = |0\rangle|1\rangle|1\rangle \qquad CC_{NOT}|1\rangle|1\rangle|0\rangle = |1\rangle|1\rangle|1\rangle$$

$$CC_{NOT}|0\rangle|0\rangle|1\rangle = |0\rangle|0\rangle|1\rangle \qquad CC_{NOT}|1\rangle|0\rangle|0\rangle = |1\rangle|0\rangle|0\rangle \qquad CC_{NOT}|1\rangle|1\rangle|1\rangle = |1\rangle|1\rangle|0\rangle$$

$$CC_{NOT}|0\rangle|1\rangle|0\rangle = |0\rangle|1\rangle|0\rangle \qquad CC_{NOT}|1\rangle|0\rangle|1\rangle = |1\rangle|0\rangle|1\rangle$$

$$CC_{NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad |0\rangle|0\rangle|0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |0\rangle|0\rangle|1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \ldots \qquad |1\rangle|1\rangle|1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Claim: CC_{NOT} implements AND and NOT on qubits.

- To implement NOT, act with CC_{NOT} on a 3-qubit state in which the first two qubits are $|1\rangle|1\rangle$: $CC_{NOT}|1\rangle|1\rangle|x\rangle = |1\rangle|1\rangle|NOT\ x\rangle$.
- To implement AND, act with CC_{NOT} on a 3-qubit state in which the last qubit is $|0\rangle$: $CC_{NOT}|x\rangle|y\rangle|0\rangle = |x\rangle|y\rangle|x\ AND\ y\rangle$.

- *So*: Any classical computation can be done using qubits instead of bits.
  - *In particular*: Any classical function that takes n input bits and produces k output bits can be implemented using arrays of primitive CC_{NOT} "gates".

How to Construct a Qubit-Based Function Calculator

- Let $|x\rangle_{(n)}$ represent n input qubits that encode the number x.
- Let $|0\rangle_{(k)}$ represent k qubits $|0\rangle$ (the output register).
- Let $|f(x)\rangle_{(k)}$ represent k output qubits that encode the number $f(x)$.
- Define an operator U_f that acts on $(n+k)$ qubits in the following way:

$$U_f|x\rangle_{(n)}|0\rangle_{(k)} = |x\rangle_{(n)}|f(x)\rangle_{(k)}.$$

- *Now*: Feed U_f a *superposition* of all possible numbers x it can take as input.
- *Result*: A superposition of all possible values of the function in a *single* computation!

Two Steps:

1. Prepare as input a superposition of all possible numbers x that can be encoded in n bits:

   (i) Start with an n-qubit state $|0\rangle_1|0\rangle_2\cdots|0\rangle_n$

   (ii) Now apply a Hadamard transformation to each qubit:

$$
\begin{aligned}
&\left(H_1 \otimes H_2 \otimes \cdots \otimes H_n\right)|0\rangle_1|0\rangle_2\cdots|0\rangle_n \\
&= \left(\sqrt{\tfrac{1}{2}}\right)^n \left\{\left(|0\rangle_1 + |1\rangle_1\right)\left(|0\rangle_2 + |1\rangle_2\right)\cdots\left(|0\rangle_n + |1\rangle_n\right)\right\} \\
&= \left(\sqrt{\tfrac{1}{2}}\right)^n \left\{\underbrace{|0\rangle_1|0\rangle_2\cdots|0\rangle_n} + |0\rangle_1|0\rangle_2\cdots|1\rangle_n + \cdots \underbrace{|1\rangle_1|1\rangle_2\cdots|1\rangle_n}\right\} = \left(\sqrt{\tfrac{1}{2}}\right)^n \sum_{x=0}^{2^n-1}|x\rangle_{(n)}
\end{aligned}
$$

The first term encodes the binary number for 0.

Each term in between is the binary number for each number between 0 *and* $2^n - 1$.

The last term encodes the binary number for $2^n - 1$.

So the entire sum is a superposition that encodes all numbers x *such that* $0 \le x < 2^n$.

Two Steps:

2. Now attach a k-qubit output register $|0\rangle_{(k)}$ and apply U_f.

$$U_f\left(\sqrt{\tfrac{1}{2}}\right)^n \sum |x\rangle_{(n)}|0\rangle_{(k)} = \left(\sqrt{\tfrac{1}{2}}\right)^n \sum U_f|x\rangle_{(n)}|0\rangle_{(k)} = \underbrace{\left(\sqrt{\tfrac{1}{2}}\right)^n \sum |x\rangle_{(n)}|f(x)\rangle_{(k)}}$$

A superposition of all possible values $f(x)$, for $0 \leq x < 2^n$, of the function f. And we've effectively calculated them all with just a *single* application of U_f.

- *The Catch*: None of these values of f is accessible until we make a measurement!

The Task for Quantum Algorithm construction

- Given a problem, first construct an appropriate superposition of solutions. Then manipulate the superposition so that the relevant terms aquire high probability.

Example: *Shor's Factorization Algorithm (1994)*

- Factors large integers into primes in *polynomial* time.

> - *Polynomial time* ⇒ the time needed to factor an integer increases exponentially as the number of digits increases.
> - *Exponential time* ⇒ the time needed to factor an integer increases as a power of the increase in number of digits of the integer.

- To factor integer N, current classical algorithms require $10^{4(\log N)^{1/3}}$ steps.
- The largest numbers capable of such factorization have ~150 (base 10) digits.

Why is fast prime factorization important?

- Classical RSA Encryption:
  - *public encryption key* = product pq of two (very large) primes.
  - *private decryption key* = p, q separately
  - *Thus*: Factorizing pq (in your lifetime) would let you break RSA encryption (standard encryption for web transactions).

Two essential facts underlie Shor's algorithm:

(a) Factorizing a large integer is equivalent to determining the period r of an associated periodic function $f(x+r) = f(x)$.

(b) A *discrete Fourier transform* maps a function $g(x)$ of period r on the domain $(0,\, 2^n-1)$ to a function $G(c)$ which has (approximately) non-zero values only at multiples of $2^n/r$.

Protocol

- By Fact (a), to factorize a given large integer, suppose we've determined that we need to find the period r of an appropriate periodic function $f(x)$.

Step 1

- Construct a superposition of all possible solutions of $f(x)$ for $0 \le x < 2^n$.

$$U_f \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle_{(n)} |0\rangle_{(k)} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle_{(n)} |f(x)\rangle_{(k)}$$

Our Good Friend the qubit-based function calculator!

Step 2

- Measure $f(x)$; *i.e.*, compute *one* value of it, say $f(x_0)$.

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle_{(n)}|f(x)\rangle_{(k)} \xrightarrow[\textit{collapse}]{} C\sum_{x=0}^{2^n-1}g(x)|x\rangle_{(n)}|f(x_0)\rangle_{(k)}$$

where $g(x) = 1$ for $x = x_0 + kr$, and zero otherwise (for k an integer).

- *Note*: The output register now has a single term $|f(x_0)\rangle_{(k)}$, but the input register $|x\rangle_{(n)}$ is still in a superposition of all those values of x for which $f(x) = f(x_0)$.

There are $2^n/r$ such values.

- *Also*: $g(x)$ has the same period r as $f(x)$, since $g(x) = g(x_0 + kr)$.

- *So*: To find the period of $f(x)$, we now need to find the period of $g(x)$.

Step 3

- Act on the input register with a *quantum Fourier transformation*:

$$C\sum_{x=0}^{2^n-1} g(x)|x\rangle_{(n)}|f(x_0)\rangle_{(k)} \xrightarrow[\text{quantum FT}]{} C'\sum_{c=0}^{2^n-1} G(c)|c\rangle_{(n)}|f(x_0)\rangle_{(k)}$$

  where $G(c)$ is the *discrete Fourier transform* of $g(x)$.

- By Fact (b), $G(c)$ is non-zero only for $c = j2^n/r$, for integer j.
- *Which means*: The right hand side can be written as

$$C' \sum_{j=0}^{r-1/2^n} G(j\,\tfrac{2^n}{r})|j\,\tfrac{2^n}{r}\rangle_{(n)}|f(x_0)\rangle_{(k)}$$

Step 4

- Conduct a measurement on the input register:

$$C' \sum_{j=0}^{r-1/2^n} G(j\,\tfrac{2^n}{r})|j\,\tfrac{2^n}{r}\rangle_{(n)}|f(x_0)\rangle_{(k)} \xrightarrow[\text{collapse}]{} |j\,\tfrac{2^n}{r}\rangle_{(n)}|f(x_0)\rangle_{(k)}$$

- This produces a value of $j2^n/r$.
- We can now calculate (or approximate closely) the value of r.

II. Interpretive Issues.

(1) *How are quantum computers different from classical computers?*

Claim: Apart from *hardware* differences (quantum 2-state systems *vs.* classical 2-state systems), the essential difference between a quantum computer and a classical computer is that the former are ideally much more *efficient* than the latter.

- A quantum computer can compute anything that a classical computer can.
  - *Recall*: *Any computation implemented using bits can be implemented using qubits.*

- A classical computer can compute anything that a quantum computer can.
  - *Any computation implemented using qubits can be implemented using bits and a probabilistic algorithm.*
  - *Intuitively: There are probabilistic classical 2-state systems that can simulate the output of quantum 2-state systems, (although perhaps not as efficiently).*

(2) *Is quantum information different from classical information?*

<u>*Information*</u> = What is produced by an information source that is required to be reproducible at the receiver if the transmission is to be counted a success.

<u>*Two Types of Information Source*</u> (Timpson 2008)

I. Classical information source

- <u>*Abstractly*</u>: Produces letters from a set $\{a_1, a_2, ..., a_n\}$ with probabilities $p_i = p(a_i)$.
- Messages = sequences of letters. <u>*Ex*</u>: $a_7 a_3 a_4 ...$
- <u>*Concretely*</u>: Produces *physical systems* (e.g., on-off switches) in *classical states* $\{a_1, a_2, ..., a_n\}$.
- Output = sequence of classical states. <u>*Ex*</u>: $a_7 a_3 a_4 ...$

II(a). Quantum information, Pure Source

- Produces *physical systems* (e.g., electrons) in *"pure" quantum states* $\{|a_1\rangle, |a_2\rangle, ..., |a_n\rangle\}$.
- Output = sequence of quantum pure states. <u>*Ex*</u>: $|a_7\rangle|a_3\rangle|a_4\rangle ...$

II(b). Quantum information, Entanglement Source

- Produces *physical systems* (*i.e.*, electrons) in *entangled quantum states* which include other systems inaccessible to the source.
- Output = sequence of quantum entangled states.

Example:

$B = \{B_1, B_2, ...\} = \{\textit{electrons produced by source}\}$

$A = \{A_1, A_2, ...\} = \{\textit{electrons entangled with source electrons}\}$

$C = \{C_1, C_2, ...\} = \{\textit{"target" electrons at receiver}\}$

- *Suppose*: Electron B_i is produced at source in an entangled state $|\psi\rangle_{A_iB_i}$ with electron A_i.
- *Goal*: To reproduce this entangled state at receiver, but between A_i and C_i: $|\psi\rangle_{A_iC_i}$.
- *In general*: If source produces sequence of states $|\psi\rangle_{A_iB_i}|\psi'\rangle_{A_jB_j}|\psi''\rangle_{A_kB_k}\cdots$, then successful transmission occurs if receiver reproduces sequence of states $|\psi\rangle_{A_iC_i}|\psi'\rangle_{A_jC_j}|\psi''\rangle_{A_kC_k}\cdots$.

- *Upshot*: No *fundamental* difference between classical and quantum information (just a difference in types of *sources*).

- *Moreover*: Recall the *Shannon Entropy* for classical information:

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i)$$

  - Specifies the minimal number of bits required to encode the output of a classical information source (*Shannon's 1948 Noiseless Coding Theorem*).

- The *von Neumann Entropy*:

$$S(\rho) = -\mathrm{Tr}\rho \log_2 \rho$$

  - ρ = the state associated with the output of a quantum information source.
  - Specifies the minimal number of qubits required to encode the output of a quantum information source (*Schumacher 1995*).