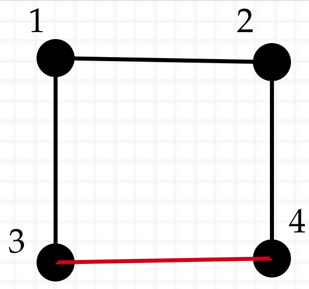
CSE439 Week 4: More Quantum Circuits

Graph-State Circuits

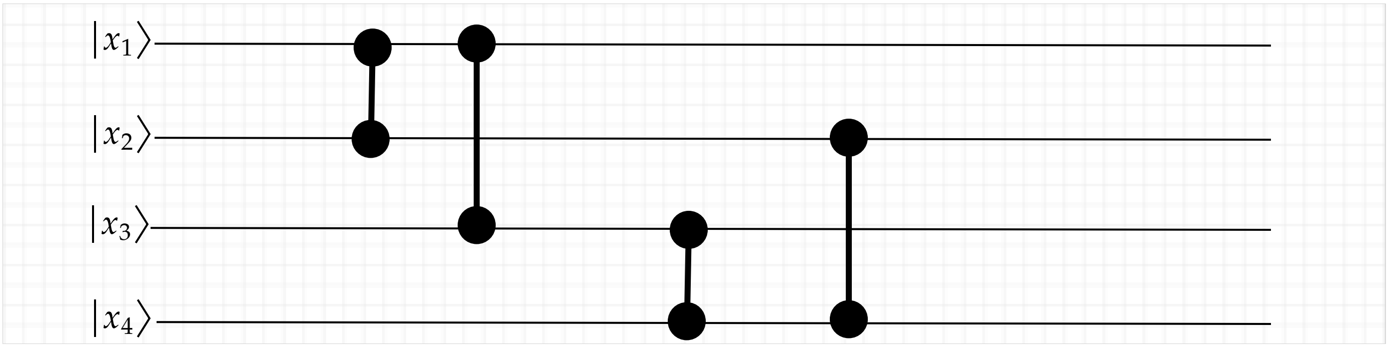
Now CZ gates are especially neat because they look like edges in a **graph** $G = (V, E)$, specifically an **undirected** graph because the gates are symmetric. Let's first see some examples of graphs. The *cycle graphs* C_k have k **vertices** (also called **nodes**) and k edges connecting them in a ring, for $k \geq 3$. The four-cycle graph has the following picture and **adjacency matrix**:



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad A' = \frac{1}{2}A \qquad A'' = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

Note: This differs from the text only in the labels 3 and 4. This makes it maybe easier to see that not only is A not unitary, it isn't even invertible: rows 2 and 3, and rows 1 and 4, are identical. But:

- A is a real symmetric matrix, so it is Hermitian.
- A' is a matrix of nonnegative entries each of whose rows and columns sums to 1, which makes it **doubly stochastic**. This is an analogue of "unitary" for classical probability.
- In fact, for any **regular** graph, meaning that all vertices have the same **degree** d, dividing the adjacency matrix by d always gives a doubly-stochastic matrix.
- We can in fact make a unitary matrix A'' by flipping the sign of the two 1s at lower right and dividing by $\sqrt{2}$ rather than by 2. This is, however, more of a coincidence than a general feature. The text shows that in the case of the regular *prism graph* ($n = 6$, $d = 3$), there is no sensible way to make it into a unitary matrix.
- The general way to encode graphs into quantum circuits via the $\mathbf{CZ}$ gate yield much bigger underlying matrices---and some surprises. Here we go:



When put on four qubits, the first gate gives the matrix $\mathbf{CZ} \otimes \mathbf{I} \otimes \mathbf{I}$, which we know how to build: replace every entry of the CZ matrix by the 4×4 identity matrix, to get the 16×16 matrix

$$\left|\begin{array}{c}0000\\0001\\0010\\0011\\0100\\0101\\0110\\0111\\1000\\1001\\1010\\1011\\1100\\1101\\1110\\1111\end{array}\right| : \quad \mathbf{CZ}\otimes\mathbf{I}\otimes\mathbf{I} = \begin{bmatrix}1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&-1&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1\end{bmatrix} = diag\left(\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\-1\\-1\\-1\\-1\end{bmatrix}\right)$$

At far left I've put the labels of the underlying coordinates by the sixteen basis strings of length 4. The point is that the -1 entries go in all the places where the first two bits of the string are 1 as shown in pink. This is because the first CZ gate is on the first two bits. Next, for the gate on qubits 1 and 3, we follow the same rule but for the coordinates where the first and third bit are 1:

$$\left|\begin{array}{c}0000\\0001\\0010\\0011\\0100\\0101\\0110\\0111\\1000\\1001\\1010\\1011\\1100\\1101\\1110\\1111\end{array}\right| : \quad diag\left(\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\-1\\-1\\1\\1\\-1\\-1\end{bmatrix}\right).$$

$$
= \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&-1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1
\end{bmatrix}
$$

Here is the product of all four gate matrices that we get. I've "properly" put the matrix for the first gate on the right now, but actually this doesn't matter---they are all diagonal matrices so they commute with each other. To multiply them, we can just multiply the entries in each of the sixteen rows. The blue 1s show cases where an even number of -1 entries multiplied to give $+1$:

$$
\begin{vmatrix}
0000\\0001\\0010\\0011\\0100\\0101\\0110\\0111\\1000\\1001\\1010\\1011\\1100\\1101\\1110\\1111
\end{vmatrix}
: \quad diag\left(\begin{bmatrix}1\\1\\1\\-1\\1\\1\\1\\-1\\1\\1\\1\\-1\\1\\1\\1\\-1\end{bmatrix}\right)
\cdot diag\left(\begin{bmatrix}1\\1\\1\\1\\1\\-1\\1\\-1\\1\\1\\1\\1\\1\\-1\\1\\-1\end{bmatrix}\right)
\quad diag\left(\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\-1\\-1\\1\\1\\-1\\-1\end{bmatrix}\right)
\cdot diag\left(\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\-1\\-1\\-1\\-1\end{bmatrix}\right)
= diag\left(\begin{bmatrix}1\\1\\1\\-1\\1\\-1\\1\\1\\1\\1\\-1\\1\\-1\\1\\1\\1\end{bmatrix}\right).
$$

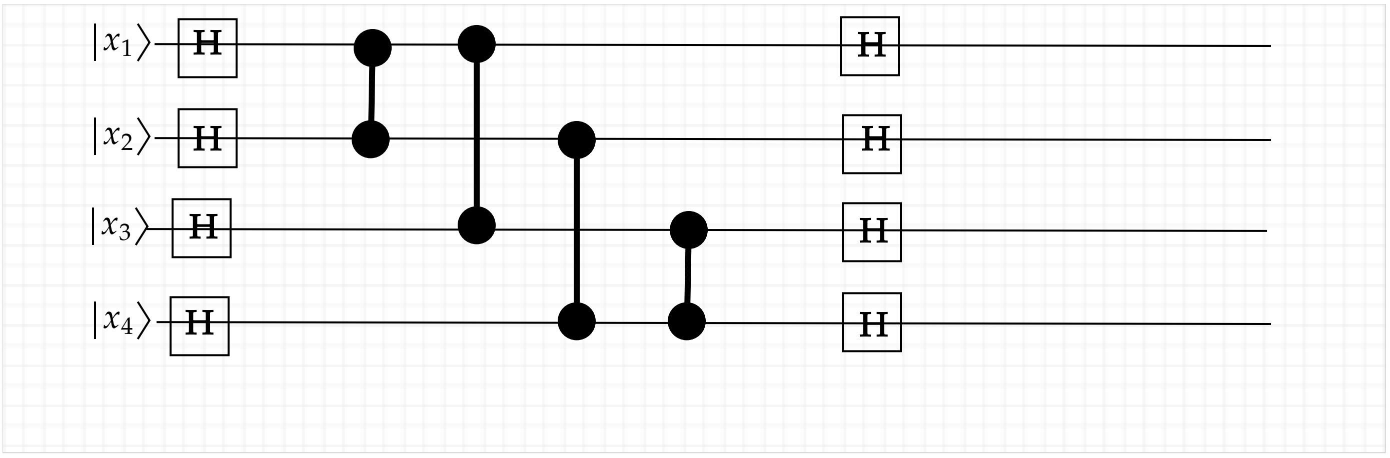
A word to the wise: The matrix for the fourth gate, which comes leftmost just above, is the tensor product $(\mathbf{I} \otimes \mathbf{I}) \times \mathbf{CZ}$. The matrices for the middle two gates, however, are technically not tensor products, because one identity comes "between the two arms" of the $\mathbf{CZ}$ gate. They are "morally" tensor products, though. The assigned exercise 4.11 makes a different case of this point. The rule about places with two particular 1s, however, applies in all cases. And the surviving -1 entries in the product at right mark four of the strings that gave exactly two 1s, the four corresponding to the edge set $E = \{(1,2),(1,3),(2,4),(3,4)\}$ of the graph.

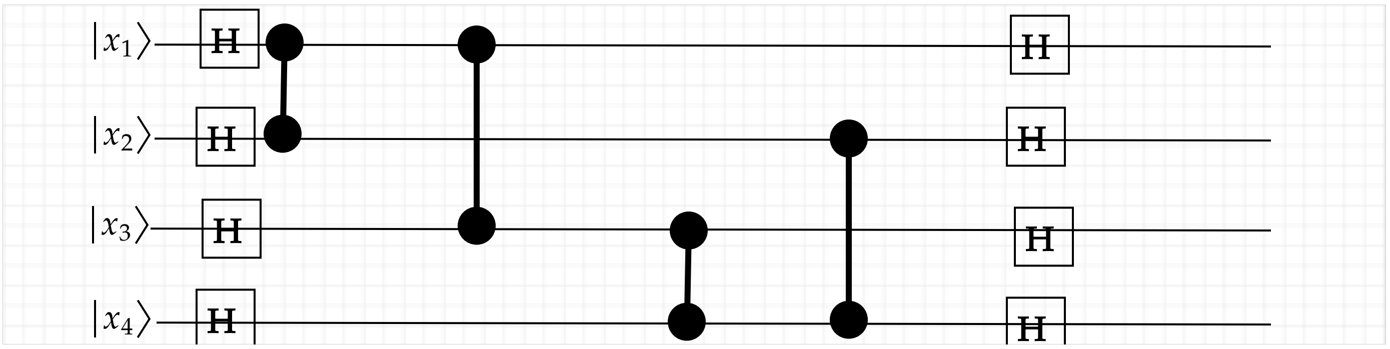
If we apply our diagonal matrix to the all-1 unit vector, here

$$[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]\frac{1}{4} = |{+}{+}{+}{+}\rangle,$$

then we get the column vector of the diagonal entries at right (again, divided by 4 to normalize it). Does that column vector faithfully preserve all information about the given graph? A question to ponder...

A **graph-state circuit** conventionally includes an all-qubits Hadamard transform before and after it:



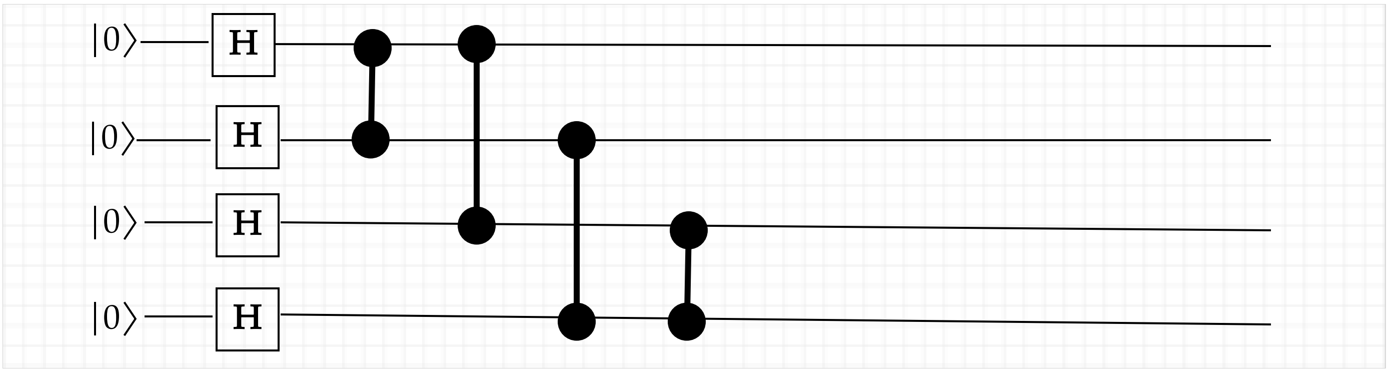


General Quantum Circuits and Computations

If there are n qubits, then the underlying matrices we get are $N \times N$ with $N = 2^n$. It is much harder to handle 2^n-sized stuff than n-sized stuff. Happily, we can always break the basic gates down to constant size---3 at most with the Toffoli gate in practice---and there are theorems that guarantee constant size gates working in general. One important case of using n single-qubit gates is the **Hadamard transform** $\mathbf{H} \otimes \mathbf{H} \otimes \cdots \otimes \mathbf{H}$ (n times), which can be abbreviated $\mathbf{H}^{\otimes n}$:

$$H^{\otimes 2} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, \quad H^{\otimes 3} = \frac{1}{2\sqrt{2}}\left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right]$$

We always have $H^{\otimes n}|0^n\rangle = |+\rangle^{\otimes n} = |+^n\rangle =$ the all-1 vector of length $N = 2^n$ divided by $\sqrt{N} = \sqrt{2^n} = 2^{n/2}$. Often this is the first step of a quantum circuit, for example:



Putting the same Hadamard transform also at the end creates what is called a **graph state circuit**; we will analyze them later.

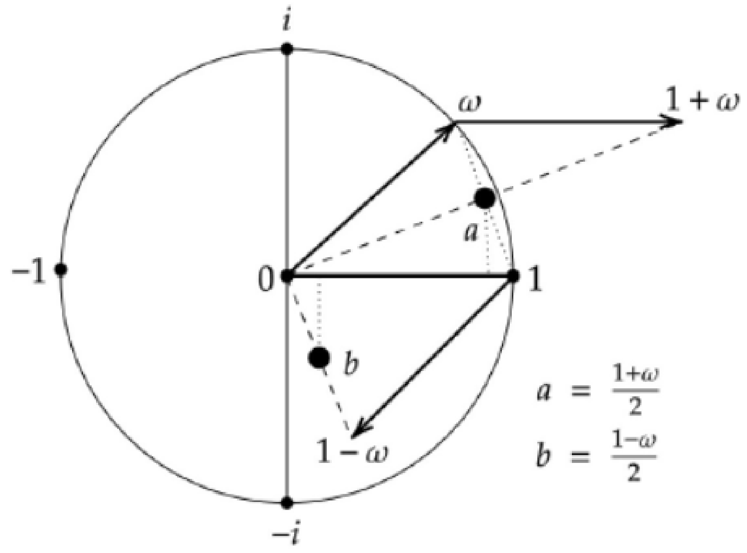
We will call an $N \times N$ matrix that arises from a single small gate---or a tensor product of small gates---a **succinct** matrix. Thus a **quantum computation** of **length** s is formally a composition of s succinct matrices applied to some input vector. The text draws allusion to a classical computation on a binary string x of length n, such as $x = 10100010$, say. The quantum circuit starts with input the basis state $|x\rangle = |10100010\rangle$. We could actually start with $|0^8\rangle$ but then **prepare** the state $|x\rangle$ by making the first column of the circuit be the tensor product

$$\mathbf{X} \otimes \mathbf{I} \otimes \mathbf{X} \otimes \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{X} \otimes \mathbf{I},$$

which has a NOT gate where x has a 1. This is why we often suppose ("without loss of generality") that the circuit starts with the all-zero basis vector.

The $\mathbf{Z}$ and $\mathbf{CZ}$ gates are the heads of an important family of basic gates having to do with rotations of

phase, which is a curious but definitely physical property. When a complex number $x + iy$ is rewritten in polar form as $re^{i\theta}$, the angle θ is the phase. The magnitude is r, so when $r = 1$ we have a unit complex number. Note that i itself is the same as $e^{i\pi/2}$ since $\frac{\pi}{2}$ means 90° phase. Then $i^2 = e^{i\pi} = -1$ and if we put $\omega = e^{i\pi/4}$ then $\omega^2 = i$. In Cartesian coordinates, $\omega = \frac{1+i}{\sqrt{2}}$. Here is some more geometry:



The vector $\mathbf{u} = [a, b]^T$ is a funky unit vector. To see that it is a unit vector, note that

$$||\mathbf{u}||^2 = \langle \mathbf{u}, \mathbf{u} \rangle = \mathbf{u}^* \cdot \mathbf{u} = a^*a + b^*b = \left(\frac{1+\overline{\omega}}{2}\right)\left(\frac{1+\omega}{2}\right) + \left(\frac{1-\overline{\omega}}{2}\right)\left(\frac{1-\omega}{2}\right).$$

In polar form, the complex conjugate of $e^{i\theta}$ is always $e^{-i\theta} = e^{i(2\pi-\theta)}$, so $\overline{\omega} = e^{i7\pi/4} = \omega^7$. In Cartesian coordinates,

$$\frac{1+\omega}{2} = \frac{1}{2}\left(1 + \frac{1+i}{\sqrt{2}}\right) = \frac{\sqrt{2}+1+i}{2\sqrt{2}} \quad \text{and} \quad \frac{1-\omega}{2} = \frac{1}{2}\left(1 - \frac{1+i}{\sqrt{2}}\right) = \frac{\sqrt{2}-1-i}{2\sqrt{2}}$$

So

$$\frac{1+\overline{\omega}}{2} = \frac{1}{2}\left(1 + \frac{1-i}{\sqrt{2}}\right) = \frac{\sqrt{2}+1-i}{2\sqrt{2}} \quad \text{and} \quad \frac{1-\overline{\omega}}{2} = \frac{1}{2}\left(1 - \frac{1-i}{\sqrt{2}}\right) = \frac{\sqrt{2}-1+i}{2\sqrt{2}}.$$

Then

$$\left(\frac{1+\overline{\omega}}{2}\right)\left(\frac{1+\omega}{2}\right) = \frac{1}{8}\left(\sqrt{2}+1+i\right)\left(\sqrt{2}+1-i\right) = \frac{1}{8}\left[\left(\sqrt{2}+1\right)^2+1\right] = \frac{1}{8}\left(2+1+2\sqrt{2}+1\right) = \frac{2+\sqrt{2}}{4}$$

and

$$\left(\frac{1-\overline{\omega}}{2}\right)\left(\frac{1-\omega}{2}\right) = \frac{1}{8}\left(\sqrt{2}-1-i\right)\left(\sqrt{2}-1+i\right) = \frac{1}{8}\left[\left(\sqrt{2}-1\right)^2+1\right] = \frac{1}{8}\left(2+1-2\sqrt{2}+1\right) = \frac{2-\sqrt{2}}{4}.$$

These squared values add to 1 as promised, so $\mathbf{u} = [a, b]^T$ is a unit vector. How do we get it? Here is the start of an infinite family of gates:

$$\mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} 1 & 0 \\ 0 & \omega \end{bmatrix}, \quad \mathbf{T}_{\pi/8} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{bmatrix}.$$

The controlled versions to go with $\mathbf{CZ}$ are $\mathbf{CS}$, $\mathbf{CT}$, etc. They, too, are symmetric---indeed, all of these gates are controlled phase shifts conditioned on the basis-state 1 of all of the (one or two) qubits involved. (Here I must note global inconsistency and confusion in notation, especially about rotations, which we will try to resolve when we cover the **Bloch Sphere** next week.)

Now we have all the background we need to read **quantum circuits**. Lecture will go on to illustrate them, both out of section 4.5 and (the same examples) on QC web applets.

The Quantum Fourier Transform

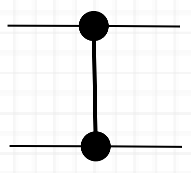
Super-tiny angles are in the definition of the QFT itself. For any n, it takes $\omega_n = e^{2\pi i/N}$ where $N = 2^n$. With $n = 3$, the matrix together with its quantum coordinates is:

$$\left[\begin{array}{c|cccccccc}
 & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\hline
000 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
001 & 1 & \omega & i & i\omega & -1 & -\omega & -i & -i\omega \\
010 & 1 & i & -1 & -i & 1 & i & -1 & -i \\
011 & 1 & i\omega & -i & \omega & -1 & -i\omega & i & -\omega \\
100 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
101 & 1 & -\omega & i & -i\omega & -1 & \omega & -i & i\omega \\
110 & 1 & -i & -1 & i & 1 & -i & -1 & i \\
111 & 1 & -i\omega & -i & -\omega & -1 & i\omega & i & \omega
\end{array}\right]
=
\left[\begin{array}{c|cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & \omega & \omega^2 & \omega^3 & -1 & \omega^5 & \omega^6 & \omega^7 \\
2 & 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\
3 & 1 & \omega^3 & \omega^6 & \omega & -1 & \omega^7 & \omega^2 & \omega^5 \\
4 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
5 & 1 & \omega^5 & \omega^2 & \omega^7 & -1 & \omega & \omega^6 & \omega^3 \\
6 & 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\
7 & 1 & \omega^7 & \omega^6 & \omega^5 & -1 & \omega^3 & \omega^2 & \omega
\end{array}\right]$$

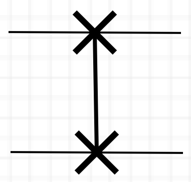
$$\mathbf{QFT}[i, j] = \omega^{ij}$$

Two other 2-qubit gates and their matrix and circuit representations are:

$$\mathbf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$



$$\mathbf{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

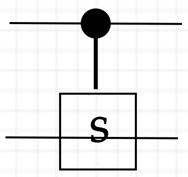


The $\mathbf{CZ}$ gate is symmetric: note that its results on $|01\rangle$ and on $|10\rangle$ are the same. So are the $\mathbf{CS}$ and

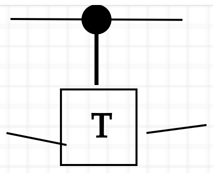
$\mathbf{CT}$ gates, which have i and $\omega = e^{i\pi/4} = \sqrt{i}$ in place of the -1. For a general $r \times r$ matrix A, $\mathbf{C}A$ is the $2r \times 2r$ matrix given in block form as $\begin{bmatrix} I & 0 \\ 0 & A \end{bmatrix}$. The circuit convention is to put a black dot on the **control** qubit line and a vertical line extending to A in a box the **target** line(s). Most applets make you do that with $\mathbf{CZ}$ as well as $\mathbf{CS}$ and $\mathbf{CT}$, but it is good to remember that these three (and further ones with roots of ω at bottom right) are symmetric.

Two other 2-qubit gates and their matrix and circuit representations are:

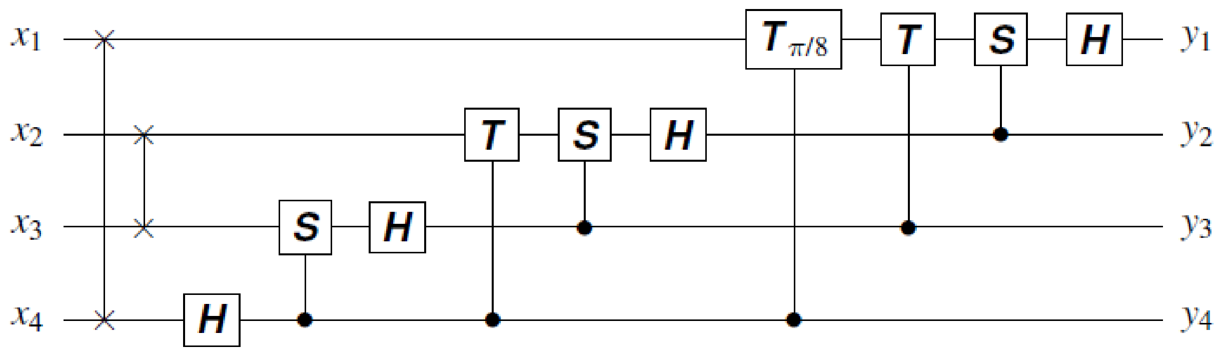
$$\mathbf{CS} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}$$



$$\mathbf{CT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \omega \end{bmatrix}$$

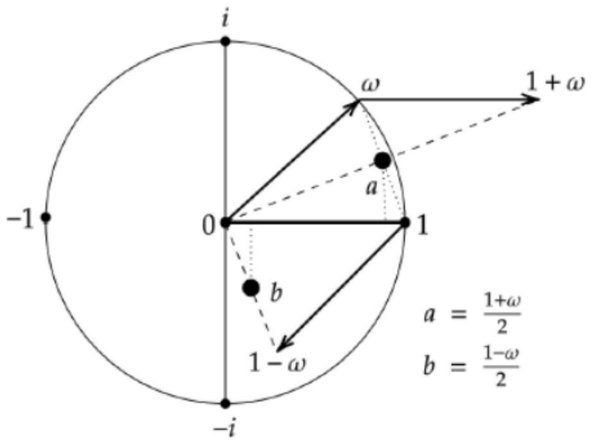


Continuing the idea of the progression $\mathbf{CZ}$, $\mathbf{CS}$, $\mathbf{CT}$,... to finer angles leads into the general construction of $O(n^2)$-sized circuits of basic gates for the n-qubit **Quantum Fourier Transform** (**QFT**). The usual recursive way to build it via $O(n^2)$ unary and binary gates uses controlled rotations by exponentially tiny angles. This is already evident from the four-qubit illustration in the textbook (where the two gates on the left are :

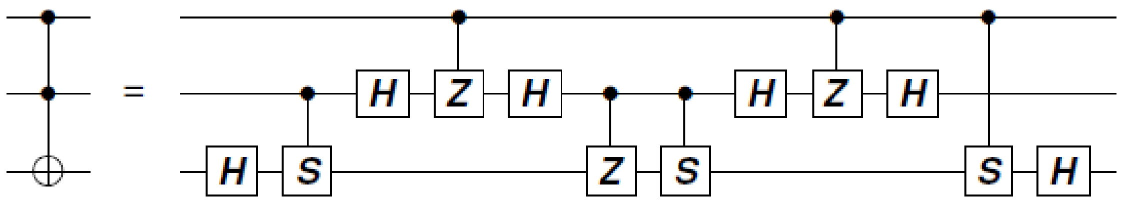


Here $T_{\pi/8} = \begin{bmatrix} 1 & 0 \\ 0 & \omega' \end{bmatrix}$ with $\omega' = e^{i\pi/8}$ not $\omega = e^{i\pi/4}$ as with the T-gate. So ω' has a phase angle one-sixteenth of a circle. For $n = 5$ the next bank uses $1/32$, then $1/64$, and soon the angles would be physically impossible so the gates could never be engineered.

For $\mathbf{QFT}_N$ we raise ω_N with its tiny phase to exponentially many different powers. How can this possibly be feasible? Leonid Levin among others raised this objection. Here are several answers:



- Basic gates can fabricate quantum states having finer phases. This is already hinted by the diagram in the case of $\mathbf{HTH}$. Try composing $\mathbf{HTHT^*H}$ and $\mathbf{HTHT^*HTHT^*H}$. The *Solovay-Kitaev theorem* enables approximating operators with exponentially fine angles by polynomially many gates of phases that are multiples of ω.
- The Toffoli and Hadamard gates by themselves, which have phases only $+1$ and -1, can simulate the real parts and imaginary parts of quantum computations separately via binary code, in a way that allows re-creating all measurement probabilities. (This is undertaken in exercises 7.8--7.14 with a preview in the solved exercise 3.8.)
- The **CNOT** and Hadamard gates do not suffice for this, even when the so-called "phase gate" $\mathbf{S} = \mathbf{T}^2 = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ is added. The Pauli $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ gates and also $\mathbf{CZ}$ can be built from these, but quantum circuits of these gates can be simulated in deterministic ("classical") polynomial time. However, $\mathbf{CS}$ suffices to build the Toffoli gate, per the diagram below (which is also a presentation option). So Hadamard + $\mathbf{CS}$ is a universal set using only quarter phases.
- The signature application of the QFT, which is *Shor's algorithm* showing that factoring belongs to BQP, may only require coarsed-grained approximations to $\mathbf{QFT}_N$. Indeed, the above theorem about Hadamard and Toffoli gates implies that they can efficiently represent the *acceptance outcomes* of any quantum circuit---though not its complex amplitudes. This extends to the replacement of Toffoli by Controlled-$\mathbf{S}$ owing to the equation we have seen:



For these reasons, $\mathbf{QFT}_N$ is considered feasible even though $N = 2^n$ is exponential. Not every $N \times N$ unitary matrix U is feasible---the Solovay-Kitaev theorem relies on U having a small exact formulation to begin with. But if we fix a finite **universal gate set** (such as $\mathbf{H+T+CNOT}$, $\mathbf{H+Tof}$, or $\mathbf{H+CS}$ above) and use only matrices that are compositions and tensor products of these gates, then we can use the simple gate-counting metric as the main complexity measure.

Outputs and Measurements

There are various conventions about what it means for a family $[C_n]$ of quantum circuits to compute a function f on $\{0,1\}^*$, where f is an ensemble of functions f_n on $\{0,1\}^n$ and each C_n computes f_n. I like supposing that $f(x)$ is coded in $\{0,1\}^r$ where r depends only on n and giving C_n r-many output qubits separate from the n input qubits, plus some number m of ancilla qubits. (It is traditional, IMHO weirdly, to consider that the primordial input is always $\mathbf{0}^n$ and that for any other x, **NOT** gates are prepended onto the circuit for those lines i where $x_i = 1$.)

For *languages*, this means that the yes/no verdict comes on qubit $n+1$. Many references say to measure line 1 instead. (Using a swap gate between lines 1 and $n+1$ can show these conventions to be equivalent, but I prefer reserving lines 1 to n for *potential* use of the "copy-uncompute" trick, which is covered in section 6.3.) Even for languages, however, one evidently cannot get the most power if you need always to rig the circuit so that on any input $x \in \{0,1\}^n$, the output line always has a (standard-)basis value, i.e., is $\mathbf{0}$ with certainty or is $\mathbf{1}$ with certainty. Instead, one must **measure** it, whereupon the value $\mathbf{0}$ is given with some probability p, $\mathbf{1}$ with probability $1 - p$.

The math of measurements (at least of the kind of *pure states* we get in completely-specified circuits) is simple. At the end we have a quantum state Ψ of $n+r+m$ qubits, counting the output and any ancilla lines. It "is" a vector $(v_1, v_2, \dots v_Q) \in \mathbb{C}^Q$ where $Q = 2^{n+r+m}$. Numbering $\{0,1\}^{n+r+m}$ in canonical order as $z_1, \dots, z_S$, an **all-qubits measurement** gives any z_j with probability $|v_j|^2$. If we focus on just the r output lines, then any $y \in \{0,1\}^r$ occurs with probability

$$\sum_{j:\, z_j \text{ agrees with } y \text{ on the } r \text{ output lines}} |v_j|^2.$$

When $r = 1$ and $y = \mathbf{0}$ the sum is over all binary strings z_j that have a 0 in the corresponding places. To simplify the notartion, let p_x denote the probability of measuring 1 on the output qubit line. The notion of uniformity is similar to that for ordinary Boolean circuits: it means that C_n can be written down in $n^{O(1)}$ (classical) time. We can finally define:

Definition: A language L belongs to BQP if there is a uniform family $[C_n]$ of polynomial-sized quantum circuits such that for all n and inputs $x \in \{0,1\}^n$,

$$\begin{aligned} x \in L &\implies p_x \geq 3/4; \\ x \notin L &\implies p_x \leq 1/4. \end{aligned}$$

[Do HTH and (HTHT)* examples of getting higher probabilities, and maybe contrast with graph states in the multi-qubit case.]

Computing Functions

Let us view the 4-qubit Hadamard transform as a big matrix:

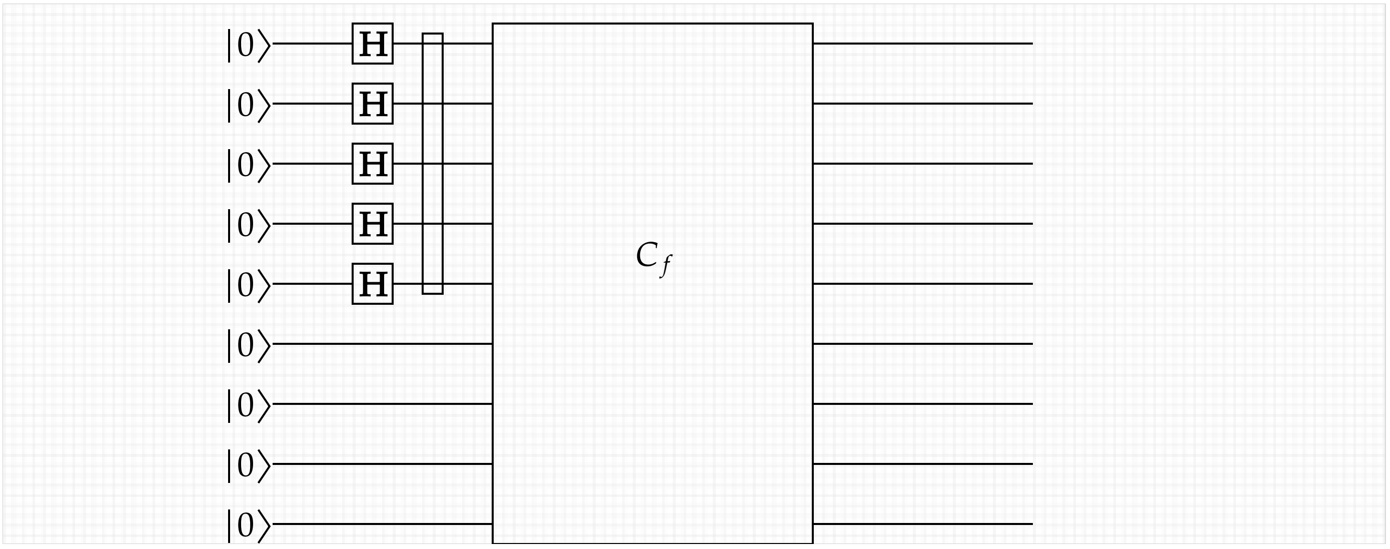
$$\left[\begin{array}{ccccccccccccccccc}
\mathbf{H} & 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\
0000 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0001 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
0010 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
0011 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 \\
0100 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
0101 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
0110 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
0111 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1000 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1001 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1010 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1011 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1100 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1101 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1110 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1111 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1
\end{array}\right]$$

$$\mathbf{H}[u,v] \;=\; (-1)^{u \bullet v}$$

We have argued that the Hadamard transform is feasible: it is just a column of n Hadamard gates, one on each qubit line. There is, however, one consequence that can be questioned. We observed that a network of Toffoli gates suffices to simulate any Boolean circuit C (of NAND gates etc.) that computes a function $f : \{0,1\}^n \to \{0,1\}^r$. The Toffoli network C_f actually computes the reversible form

$$F(x_1,\,\dots,x_n,a_1,\,\dots,a_r) \;=\; (x_1,\,\dots,x_n,a_1 \oplus f(x)_1,\,\dots,a_r \oplus f(x)_r).$$

The matrix $\mathbf{U_f}$ of C_f is a giant permutation martrix in the 2^{n+r} underlying coordinates. Yet if the Boolean circuit C has s gates, then we reckon that C_f costs $O(s)$ to build and operate. Now build the following circuit, which is illustrated with $n = 5$ and $r = 4$:



What this circuit piece computes is the **functional superposition** of f, defined as

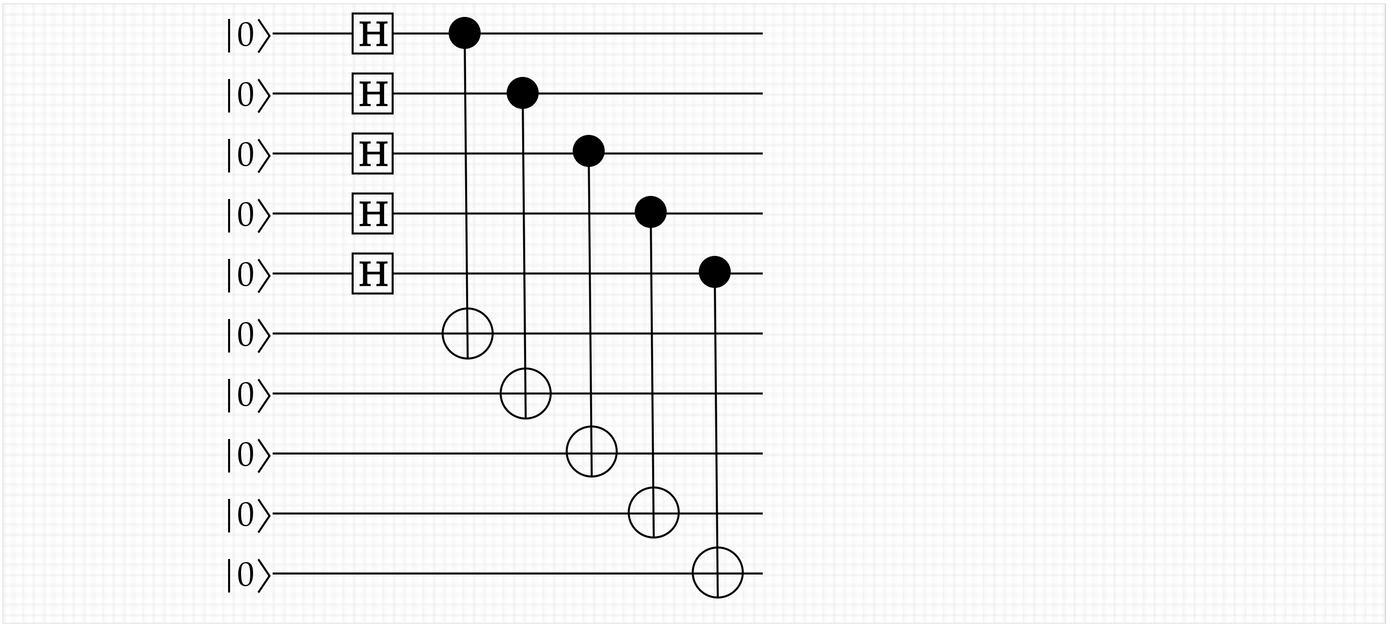
$$|\Phi_f\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle.$$

The juxtaposition of two kets really is a tensor product. This sum has exponentially many terms. It seems to preserve an exponential amount of information: the entire truth table of the Boolean function $f(x)$ over all arguments $x \in \{0,1\}^n$. However:

- f is not an arbitrary or "random" function: it is computed by a small circuit of s NAND gates.
- We cannot actually extract an exponential amount of information from $|\Phi_f\rangle$. If we measure it using the standard basis, we get our argument x back again plus r bits of some sampled function value. Measuring it in a different basis does not increase the information yield (this is part of **Holevo's Theorem**).

Nevertheless, the question remains of whether some exponential amount of "effort" must go in to the creation of $|\Phi_f\rangle$, instead of just $O(n)$ for the Hadamard transform plus $O(s)$ for the circuit.

Let's ask this where the circuit is just a bunch of $\mathbf{CNOT}$ gates. On five qubits,



computes the functional superposition

$$\frac{1}{\sqrt{32}} \sum_{x \in \{0,1\}^5} |x\rangle|x\rangle.$$

This is not the same as $|{+}{+}{+}{+}{+}\rangle \otimes |{+}{+}{+}{+}{+}\rangle$, because that is the equal superposition over all basis states for 10-bit binary strings, including all the cases of $|xy\rangle$ where the binary strings x and y of length 5 are different. An analogy is that for any set A of two or more elements, the Cartesian product of A with itself includes ordered pairs (x, y) with $x, y \in A$ but $x \neq y$, whereas the functional superposition is like the diagonal of the Cartesian product, namely $\{(x, x) : x \in A\}$. The functional superposition is entangled, just as we first saw in the case $n = 1$.

If we replace the five $\mathbf{H}$ gates by a subcircuit that prepares a general 5-qubit state

$$|\phi\rangle \;=\; a_0|00000\rangle + a_1|00001\rangle + \;\cdots\; + a_{30}|11110\rangle + a_{31}|11111\rangle,$$

then the five $\mathbf{CNOT}$ gates produce

$$D\big(|\phi\rangle\big) \;=\; a_0|0000000000\rangle + a_1|0000100001\rangle + \;\cdots\; + a_{30}|1111011110\rangle + a_{31}|1111111111\rangle.$$

This is not the same as $|\phi\rangle \otimes |\phi\rangle$, whose terms have coefficients $a_i a_j$ for all i and j. IMHO the notation $|\phi\rangle|\phi\rangle$ or $|\phi\phi\rangle$ can be unclear about what is meant, though I've freely used $|{+}{+}\rangle$ etc. as above. When $|x\rangle$ is a basis element in the basis used for notation, then there is no difference: both $|x\rangle \otimes |x\rangle$ and $D\big(|x\rangle\big)$ have the single term $|xx\rangle$ with coefficient $1 = 1^2$.

Feasible Diagonal Matrices (section 5.4)

We can continue the progression $\mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, $\mathbf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$, by

$$\mathbf{CCZ} = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & -1 \end{bmatrix}, \mathbf{CCCZ} = diag([1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1]),$$

and so forth. These are examples of a different kind of conversion of a Boolean function f besides the reversible form called F or C_f above. This is the matrix G_f defined for all indices u, v by

$$G_f[u,v] \;=\; \begin{cases} 0 & \text{if } u \neq v \\ -1 & \text{if } u = v \;\wedge f(u) = 1 \\ 1 & \text{if } u = v \;\wedge f(u) = 0 \end{cases}.$$

The above are G_{AND} for the n-ary AND function. The G stands for "Grover Oracle", though here I would rather emphasize that it is a concretely feasible operation. This ultimately leads to a theorem whose statement doesn't appear until chapter 6:

Theorem (6.2): If f is computable by a Boolean circuit with s gates, thgen G_f can be computed by a quantum circuit of $O(s)$ gates.

When $s \;=\; s(n)$ is polynomial in n, this makes a big contrast to G_f being a 2^n-sized diagonal matrix.