Quantum Algorithms

In this section we embark on more complex scores and explore the components it takes to construct real quantum algorithms. We go beyond simply defining entanglement and begin to use it in computation, in order to perform some well-known algorithms (with more to come in the future):

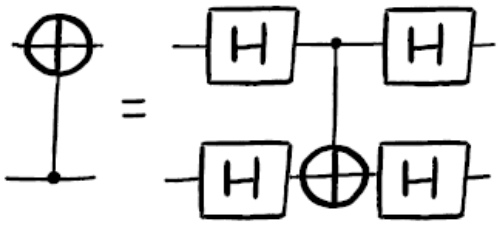
- Grover's algorithm
- Deutsch-Jozsa algorithm
- Phase estimation algorithm

Basic Circuit Identities and Larger Circuits

There are several facts about quantum circuits that can be used to express more complicated unitary transformations (https://en.wikipedia.org/wiki/Unitary_transformation), write circuits more concisely, or adapt circuits to experimental constraints.

Changing the direction of a CNOT gate

In the first example "CNOT (Reverse)," we consider how to implement a CNOT gate from control qubit 2 to target qubit 1 (notated $CNOT_{21}$) using a CNOT gate that acts in the opposite direction, from control qubit 1 to target qubit 2, $CNOT_{12}$. By multiplying the matrices for each gate, you can convince yourself that $(H \otimes H)CNOT_{12}(H \otimes H) = CNOT_{21}$.

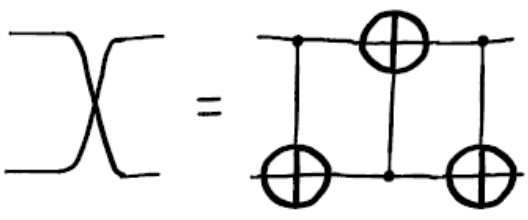


The Kronecker product (https://en.wikipedia.org/wiki/Kronecker_product) $H \otimes H$ in this equation is equal to a four-by-four matrix $\frac{1}{\sqrt{2}}\begin{pmatrix} H & H \\ H & -H \end{pmatrix}$. If you run the example, you will confirm that this combination of Hadamard and CNOT gates implements a CNOT gate in the opposite direction. The Pauli X (https://en.wikipedia.org/wiki/Quantum_gate#Pauli-X_gate) acts to invert the control qubit 2, and the result is $|11\rangle$ as expected for $CNOT_{21}$.

Swapping the states of qubits

Our second example, "Swap," demonstrates a building block that allows you to permute the information stored in a set of qubits. Suppose we want to exchange the states of a pair of qubits by implementing a SWAP gate on the pair. There is no SWAP gate in our basis, but we can construct one from three CNOT gates $SWAP_{12} = CNOT_{12}CNOT_{21}CNOT_{12}$.

To see that this is true, it is enough to look at what happens to each classical state 00, 01, 10, and 11. Let's consider 01. The first gate $CNOT_{12}$ does nothing since the control is 0. The second gate $CNOT_{21}$ flips the first qubit, so we have 11.

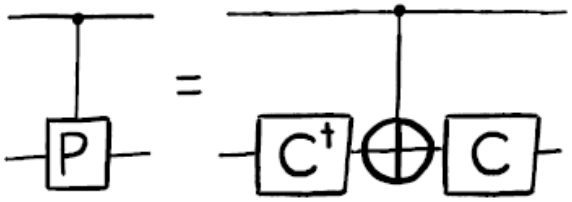


Finally, the last $CNOT_{12}$ flips the second qubit and we get 10. The "1" has moved from the second qubit to the first. The other cases can be worked out similarly. Now you can see this for yourself by running the "Swap" example below. Notice that we have used the "CNOT (Reverse)" identity to change the direction of the $CNOT_{21}$ gate, since this is necessary to run the example on the real device. Try deleting the Pauli X and placing a Pauli X on qubit 1 instead.

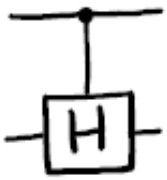
In the experimental device, not all qubits are connected to each other; therefore, some two-qubit gates cannot be applied directly. In the third example, "Swap Q0 with Q1," we show how to swap a pair of qubits that are not directly connected to each other but share a common neighbor (in this case Q2). The state $|+\rangle$, prepared by the first Hadamard gate on Q_0, is swapped into Q_1 by three successive SWAP gates.

Adding a control qubit to a gate

Some unitary transformations can be constructed exactly using gates in our basis. One set of transformations we use regularly are controlled-Pauli operations, where we apply a Pauli gate to a target qubit if a control qubit is $|1\rangle$. The CNOT gate is a controlled-X gate. Since we know that $HXH = Z$ and $SXS^\dagger = Y$, and furthermore that $HH = I$ and $SS^\dagger = I$, it is straightforward to construct circuits for controlled-Z and controlled-Y. This is illustrated in the following figure, where P is a Pauli gate X, Y, or Z, and C is a Clifford operation such as I, S, or H.

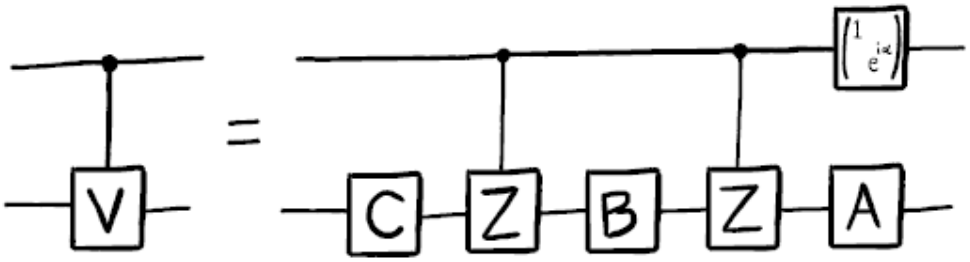


For a more involved example, let's add a control qubit to a Hadamard gate to implement a controlled-Hadamard operation:



It turns out that we can write down a circuit for a controlled-V operation if we can find three circuits A, B, and C such that $ABC = I$ and $e^{i\alpha}AZBZC = V$ [Barenco et al., 1995 (http://journals.aps.org/pra/abstract/10.1103/PhysRevA.52.3457?cm_mc_uid=43781767191014577577895&cm_mc_sid_50200000=1460741020)].

There is a general recipe for doing this, but we will just write down a solution for when $V = H$ that you can check for yourself: $A = e^{i3\pi/8} XSHTHS^\dagger$, $B = e^{-i\pi/8} SHT^\dagger HS^\dagger HSH$, $C = e^{-i\pi/4} HSH$, and $e^{i\alpha} = -i$. Combining these circuits as shown here



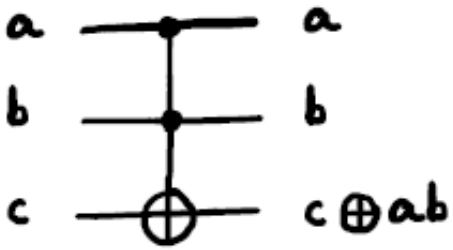
and making some simplifications, we get the result shown in the fourth example, "controlled-Hadamard." This example applies a Hadamard gate to the control qubit, Q_0, and then applies the controlled-Hadamard circuit from Q_0 to Q_2. This creates the output state $\frac{1}{\sqrt{2}}(|00\rangle + |1+\rangle) = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$. Try deleting the first Hadamard gate on Q_0 and replacing it with a bit-flip (X) to see what happens. Can you implement circuits for other controlled gates, such as a controlled-S?

Approximating a unitary transformation

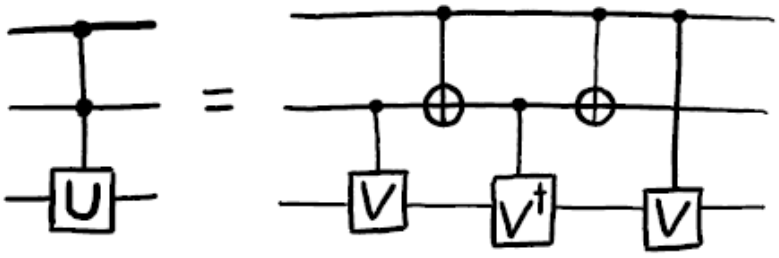
Most unitary transformations cannot be written exactly using the gates we have in our basis; but because our basis is a discrete universal set, it *is* possible to approximate any unitary transformation to any accuracy. Let's see an example of this. The $\sqrt{T}$ unitary transformation cannot be written exactly using our basis. Since $\sqrt{T}$ is a Z-rotation, the identity gate (which does nothing) is an approximate $\sqrt{T}$ gate, but not a very good one. The fifth example "Approximate sqrt(T)" gives a much better approximation using 17 Hadamard, S, and T gates. This example first puts the qubit Q_0 on the equator of the Bloch sphere using a Hadamard gate, then applies the 17 gate sequence. We use state tomography (https://en.wikipedia.org/wiki/Quantum_tomography) to observe the final state on the Bloch sphere. Had we applied an exact $\sqrt{T}$ gate, the final state would correspond to the point $(\frac{\sqrt{2+\sqrt{2}}}{2}, \frac{\sqrt{2-\sqrt{2}}}{2}, 0) \approx (0.92388, 0.38268, 0)$ on the Bloch sphere. How good is the 17 gate approximation? Arbitrarily good approximations exist, so can you find a better one? How might you use these circuits to construct an approximate controlled-T unitary transformation?

The Toffoli gate

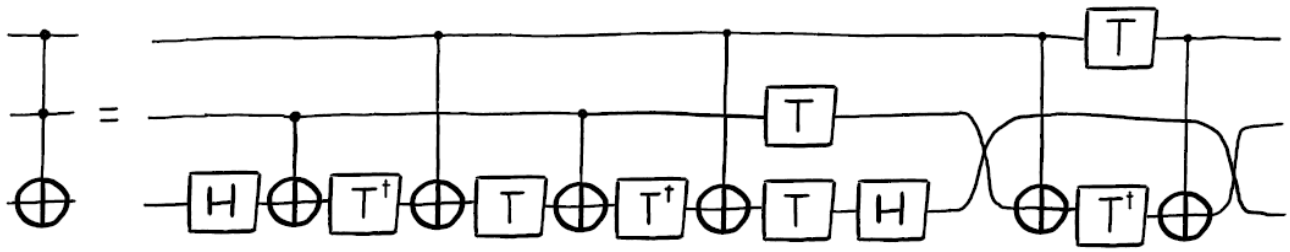
Our final examples, "Toffoli with flips" and "Toffoli state" demonstrate how to implement the reversible circuit equivalent of the (irreversible, classical) AND gate using gates from our basis. An AND gate has two inputs and one output, and outputs 1 if and only if both inputs are 1. One reason the AND gate is important for computation is that it is a non-linear transformation (a multiplication). Why do we say AND is irreversible? Notice that all three inputs 00, 01, and 10 result in an output of 0. If you see that the output of the gate is 0, you can't undo the gate because you don't know which of these three inputs gave you the result. Since there are three possible answers, even if you add another output qubit, you won't have enough information to undo the gate, since you must distinguish 3 cases, and there are only two choices for the state of the new qubit. However, it is possible to implement AND reversibly using 3 wires. This reversible AND gate is called the Toffoli gate (https://en.wikipedia.org/wiki/Toffoli_gate) $TOF|a,b,c\rangle = |a,b,(a \text{ AND } b) \text{ XOR } c\rangle$.



It is not obvious how to build a Toffoli gate from gates in our basis [Barenco et al., 1995 (http://journals.aps.org/pra/abstract/10.1103/PhysRevA.52.3457?cm_mc_uid=43781767191014577577895&cm_mc_sid_50200000=1460741020)]. The main idea is illustrated in the following figure



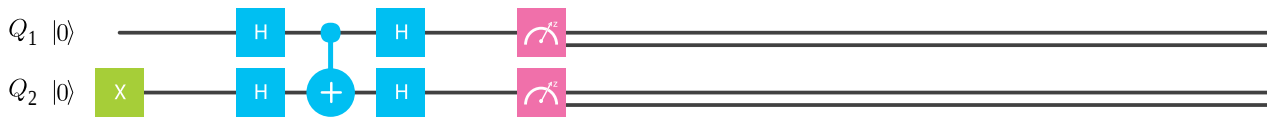
where $V = \sqrt{U}$. By tracing through each value of the two control qubits, you can convince yourself that a U gate is applied to the target qubit if and only if both controls are 1. Using ideas we have already described, you could now implement each controlled-V gate to arrive at some circuit for the doubly-controlled-U gate. It turns out that the minimum number of CNOT gates required to implement the Toffoli gate is 6 [Shende and Markov, 2009 (http://dl.acm.org/citation.cfm?id=2011799)]:



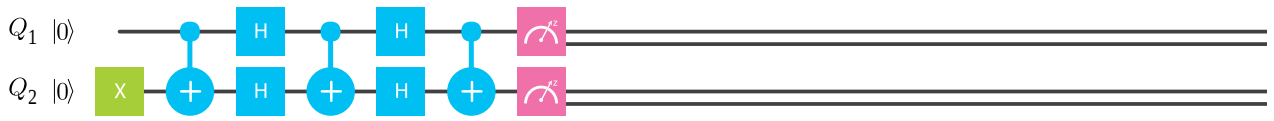
The "Toffoli with flips" example below allows you to choose an input state by changing the single-qubit gates at the beginning of the circuit. Right now they are set to Pauli X on qubits 0 and 1 and identity on qubit 2, so the default output is $|111\rangle$. You will notice that the example circuit uses 9 CNOT gates instead of the optimal number 6. This is because we wrote the circuit so it can run on the real device, which requires inserting a SWAP gate on qubits 1 and 2 near the end of the circuit. Note that we do not undo this swap, so if the input qubit labels are $0, 1, 2$, the output labels are actually $0, 2, 1$. This means, for example, that an input of 010 produces output 001 (not 010). Finally, the "Toffoli state" example demonstrates the creation of an interesting 3-qubit entangled state $SWAP_{1,2}TOF|++0\rangle_{0,1,2} = \frac{1}{2}(|000\rangle + |001\rangle + |100\rangle + |111\rangle)$ that encodes the truth table (https://en.wikipedia.org/wiki/Truth_table) of the Toffoli gate.

Using the Toffoli gate, it is possible to construct more complex circuits. A single Toffoli gate is sufficient to implement a modulo-4 addition operation between a pair of 2-bit registers or an increment-by-one operation from one 2-bit register to another. Can you find circuits for these operations and run them in the Quantum Composer?

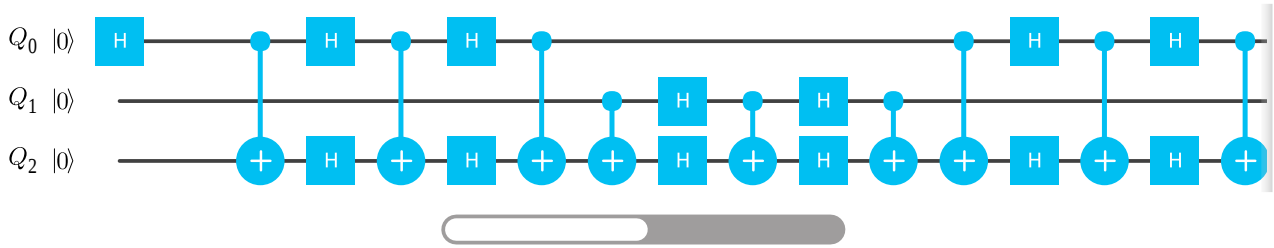
CNOT (Reverse)



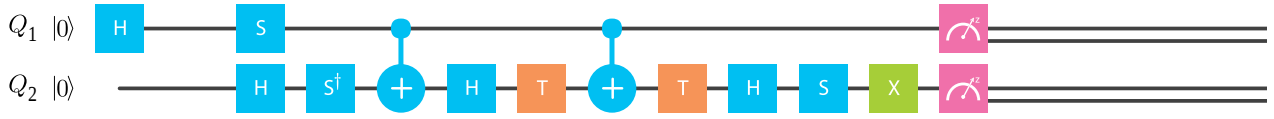
Swap



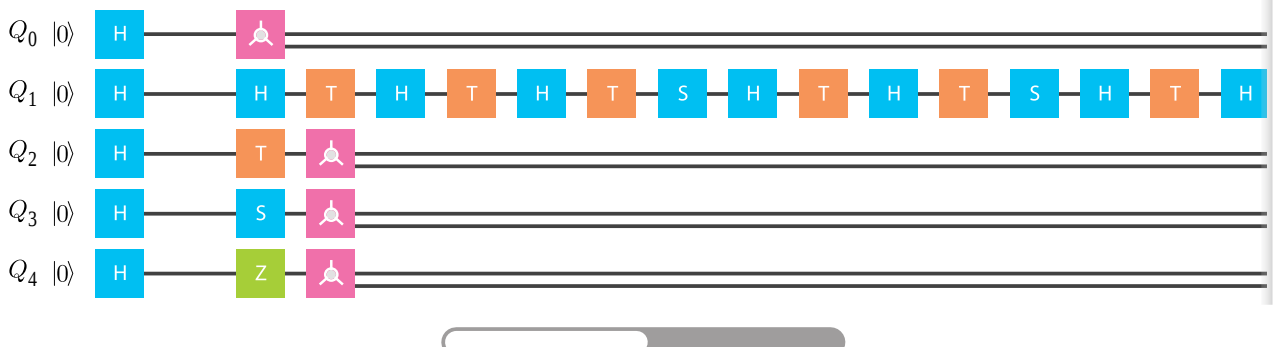
Swap Q0 and Q1



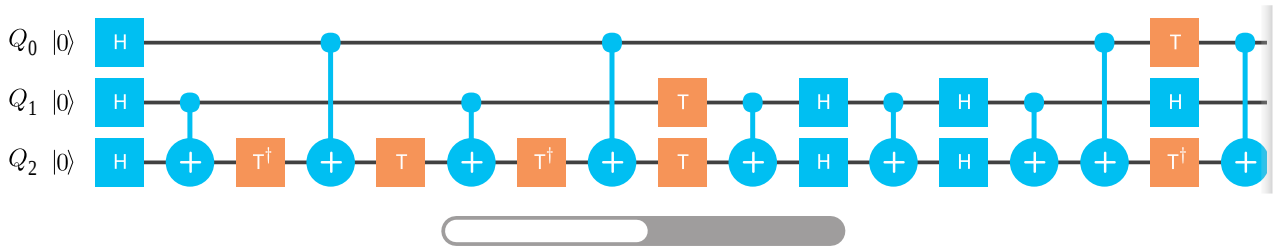
controlled-Hadamard



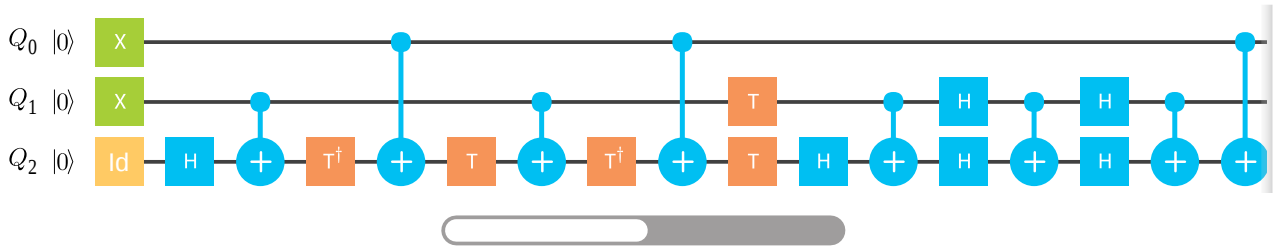
Approximate sqrt(T)



Toffoli state



Toffoli with flips



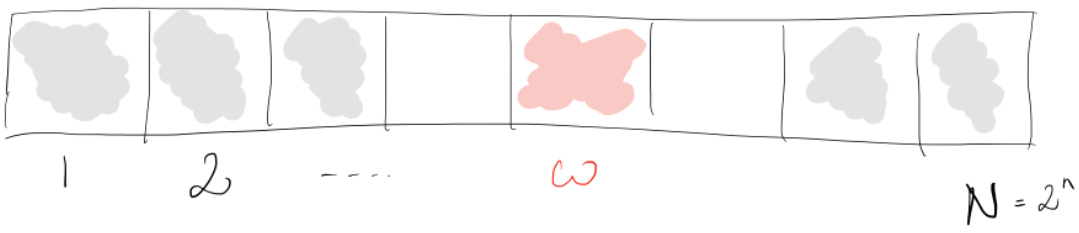
Grover's Algorithm

We are now in a good place to discuss our first quantum algorithms and subroutines. Let's begin with Grover's search algorithm (http://arxiv.org/abs/quant-ph/9605043) and the *amplitude amplification trick*.

You have likely heard that one of the many advantages a quantum computer has over a classical computer is its superior speed searching databases. Grover's algorithm demonstrates this capability. This algorithm can speed up an unstructured search problem quadratically, but its uses extend beyond that; it can serve as a general trick or subroutine to obtain quadratic run time improvements for a variety of other algorithms. This is called the amplitude amplification trick. But before we start the simulations, let's look at the unstructured search problem.

Unstructured search

Suppose you are given a large list of N items. Among these items there is one item with a unique property that we wish to locate; we will call this one the winner w. Think of each item in the list as a box of a particular color. Say all items in the list are gray except the winner w, which is pink.



To find the pink box -- the *marked item* -- using classical computation, one would have to check on average $N/2$ of these boxes, and in the worst case, all N of them. On a quantum computer, however, we can find the marked item in roughly $\sqrt{N}$ steps with Grover's amplitude amplification trick. It was proven (even before Grover's algorithm was discovered!) that this speedup is in fact the most we can hope for [Bennett, 1997 (http://arxiv.org/abs/quant-ph/9701001)]. A quadratic speedup is indeed a substantial

time-saver for finding marked items in long lists. Additionally, the algorithm does not use the list's internal structure, which makes it *generic;* this is why it immediately provides a quadratic quantum speed-up for many classical problems.

The Oracle

How will the list items be provided to the quantum computer? A common way to encode such a list is in terms of a function f which returns $f(x) = 0$ for all unmarked items x and $f(w) = 1$ for the winner. To use a quantum computer for this problem, we must provide the items in superposition to this function, so we encode the function into a unitary matrix called an *oracle*. First we choose a binary encoding of the items $x, w \in \{0,1\}^n$ so that $N = 2^n$; now we can represent it in terms of qubits on a quantum computer. Then we define the oracle matrix U_f to act on any of the simple, standard basis states $|x\rangle$ by

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle.$$

We see that if x is an unmarked item, the oracle does nothing to the state. However, when we apply the oracle to the basis state $|w\rangle$, it maps $U_f|w\rangle = -|w\rangle$. Geometrically, this unitary matrix corresponds to a reflection about the origin for the marked item in an $N = 2^n$ dimensional vector space.

Amplitude amplification

So how does the algorithm work? Before looking at the list of items, we have no idea where the marked item is. Therefore, any guess of its location is as good as any other, which can be expressed in terms of a quantum state called a *uniform superposition*:

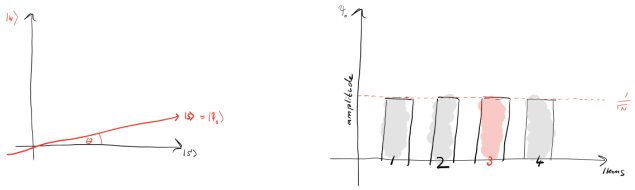
$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

If at this point we were to measure in the standard basis $\{|x\rangle\}$, this superposition would collapse, according to the fifth quantum law, to any one of the basis states with the same probability of $\frac{1}{N} = \frac{1}{2^n}$. Our chances of guessing the right value w is therefore 1 in 2^n, as could be expected. Hence, on average we would need to try about $N = 2^n$ times to guess the correct item.

Enter the procedure called amplitude amplification, which is how a quantum computer significantly enhances this probability. This procedure stretches out (amplifies) the amplitude of the marked item, which shrinks the other items' amplitude, so that measuring the final state will return the right item with near-certainty.

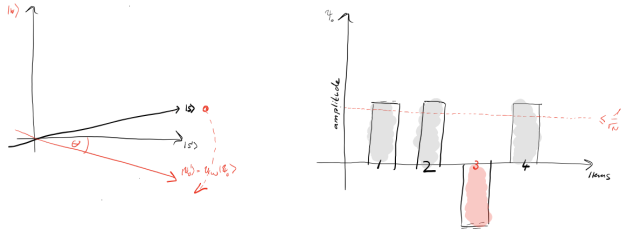
This algorithm has a nice geometrical interpretation in terms of two reflections, which generate a rotation in a two-dimensional plane. The only two special states we need to consider are the winner $|w\rangle$ and the uniform superposition $|s\rangle$. These two vectors span a two-dimensional plane in the vector space $\mathbb{C}^N$. They are not quite perpendicular because $|w\rangle$ occurs in the superposition with amplitude $N^{-1/2}$ as well. We can, however, introduce an additional state $|s'\rangle$ that is in the span of these two vectors, which is perpendicular to $|w\rangle$ and is obtained from $|s\rangle$ by removing $|w\rangle$ and rescaling.

step 0 The amplitude amplification procedure starts out in the uniform superposition $|s\rangle$. (The uniform superposition is easily constructed from $|s\rangle = H^{\otimes n}|0\rangle^n$, as was shown in a previous section.) At $t = 0$ the initial state is $|\psi_0\rangle = |s\rangle$.



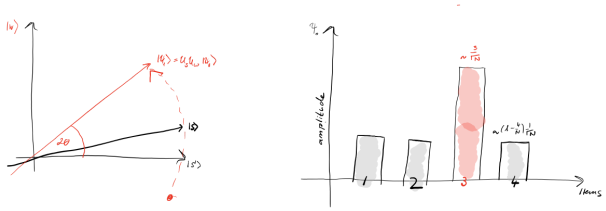
The left graphic corresponds to the two-dimensional plane spanned by $|w\rangle, |s\rangle$. The right graphic is a bar graph of the amplitudes of the state $|\psi_t\rangle$ for the case $N = 2^2 = 4$. The average amplitude is indicated by a dashed line.

step 1 We apply the oracle reflection U_f to the state $U_f|\psi_t\rangle = |\psi_{t'}\rangle$.



Geometrically this corresponds to a reflection of the state $|\psi_t\rangle$ about $-|w\rangle$. This transformation means that the amplitude in front of the $|w\rangle$ state becomes negative, which in turn means that the average amplitude has been lowered.

step 2 We now apply an additional reflection U_s about the state $|s\rangle$. In the bra-ket (https://en.wikipedia.org/wiki/Bra%E2%80%93ket_notation) notation this reflection is written $U_s = 2|s\rangle\langle s| - 1$. This transformation maps the state to $U_s|\psi_{t'}\rangle$ and completes the transformation $|\psi_{t+1}\rangle = U_s U_f |\psi_t\rangle$.

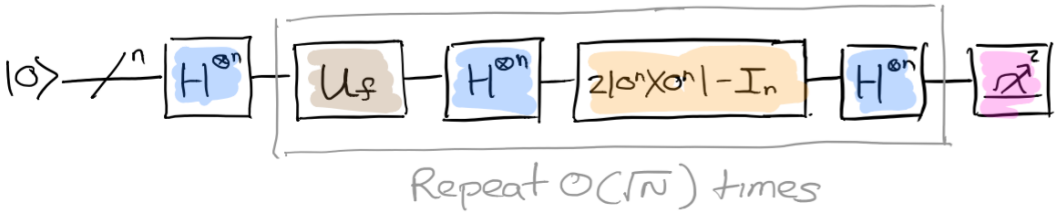


Two reflections always correspond to a rotation. The transformation $U_s U_f$ rotates the initial state $|s\rangle$ closer towards the winner $|w\rangle$. The action of the reflection U_s in the amplitude bar diagram can be understood as a reflection about the average amplitude. Since the average amplitude has been lowered by the first reflection, this transformation boosts the negative amplitude of $|w\rangle$ to roughly three times its original value, while it decreases the other amplitudes. We then go to **step 1** to repeat the application. This procedure will be repeated several times to zero in on the winner.

After t steps the state will have transformed to

$$|\psi_t\rangle = (U_s U_f)^t |\psi_0\rangle.$$

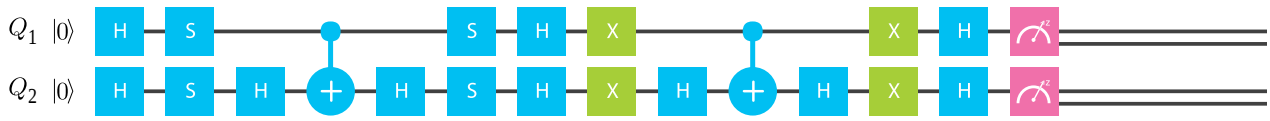
How many times do we need to apply the rotation? It turns out that roughly $\sqrt{N}$ rotations suffice. This becomes clear when looking at the amplitudes of the state $|\psi_t\rangle$. We can see that the amplitude of $|w\rangle$ grows linearly with the number of applications $\sim tN^{-1/2}$. However, since we are dealing with amplitudes and not probabilities, the vector space's dimension enters as a square root. Therefore it is the amplitude, and not just the probability, that is being amplified in this procedure.



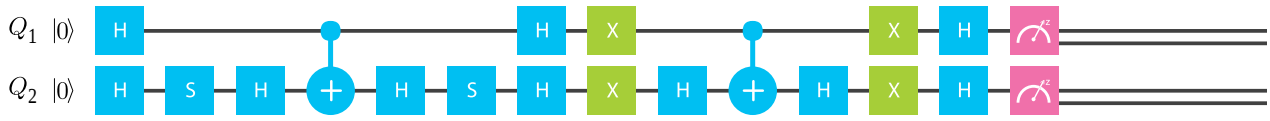
Example circuits

Let us now examine a simple example. The smallest circuit for which this can be implemented involves 2 qubits, i.e. $N = 2^2$, which means there are four possible oracles U_f, one for each choice of the winner. The first part of this example creates the uniform superposition. The second part tags the state with U_f, which is made from a control-Z gate, made from a CNOT (as described in the last section). The final part of the circuit performs U_s.

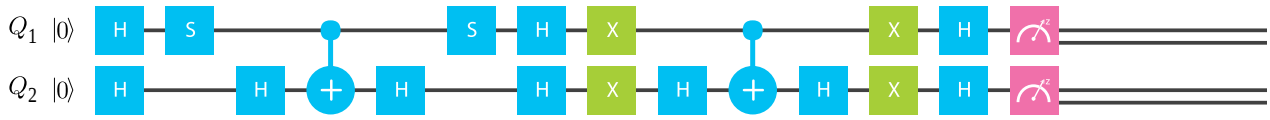
Grover N=2 A=00



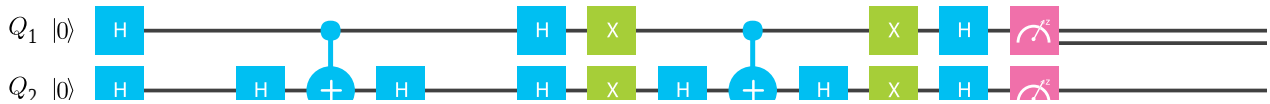
Grover N=2 A=01



Grover N=2 A=10



Grover N=2 A=11



Deutsch-Jozsa Algorithm

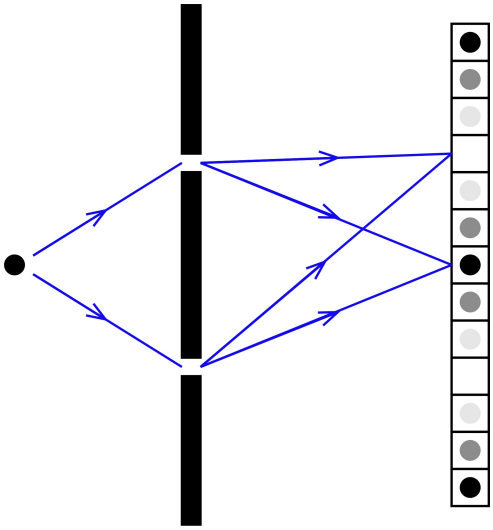
The Deutsch-Jozsa algorithm was the first to show a separation between the quantum and classical difficulty of a problem. This algorithm demonstrates the significance of allowing quantum amplitudes to take both positive and negative values, as opposed to classical probabilities that are always non-negative.

The Deutsch-Jozsa problem is defined as follows. Consider a function $f(x)$ that takes as input n-bit strings x and returns 0 or 1. Suppose we are promised that $f(x)$ is either a **constant** function that takes the same value $c \in \{0,1\}$ on all inputs x, or a **balanced** function that takes each value 0 and 1 on exactly half of the inputs. The goal is to decide whether f is constant or balanced by making as few function evaluations as possible. Classically, it requires $2^{n-1}+1$ function evaluations in the worst case. Using the Deutsch-Jozsa algorithm, the question can be answered with just one function evaluation. In the quantum world the function f is specified by an oracle circuit U_f (see the previous section on Grover's algorithm, such that $U_f|x\rangle = (-1)^{f(x)}|x\rangle$).

To understand how the Deutsch-Jozsa algorithm works, let us first consider a typical interference experiment: a particle that behaves like a wave, such as a photon, can travel from the source to an array of detectors by following two or more paths simultaneously. The probability of observing the particle will be concentrated at those detectors where most of the incoming waves arrive with the same phase.

Imagine that we can set up an interference experiment as above, with 2^n detectors and 2^n possible paths from the source to each of the detectors. We shall label the paths and the detectors with n-bit strings x and y respectively. Suppose further that the phase accumulated along a path x to a detector y equals $C(-1)^{f(x)+x\cdot y}$, where

$$x \cdot y = \sum_{i=1}^{n} x_i y_i$$



is the binary inner product and C is a normalizing coefficient. The probability to observe the particle at a detector y can be computed by summing up amplitudes of all paths x arriving at y and taking the absolute value squared:

$$\mathbf{Pr}(y) = |C \sum_x (-1)^{f(x)+x\cdot y}|^2$$

Normalization condition $\sum_y \mathbf{Pr}(y) = 1$ then gives $C = 2^{-n}$. Let us compute the probability $\mathbf{Pr}(y = 0^n)$ of observing the particle at the detector $y = 0^n$ (all zeros string). We have $\mathbf{Pr}(y = 0^n) = |2^{-n} \sum_x (-1)^{f(x)}|^2$

If $f(x) = c$ is a constant function, we get $\mathbf{Pr}(y = 0^n) = |(-1)^c|^2 = 1$. However, if $f(x)$ is a balanced function, we get $\mathbf{Pr}(y = 0^n) = 0$, since all the terms in the sum over x cancel each other.

We can therefore determine whether f is constant or balanced with certainty by running the experiment just once.

Of course, this experiment is not practical since it would require an impossibly large optical table! However, we can simulate this experiment on a quantum computer with just n qubits and access to the oracle circuit U_f. Indeed, consider the following algorithm:

Step 1. Initialize n qubits in the all-zeros state $|0, \ldots, 0\rangle$.
Step 2. Apply the Hadamard gate H to each qubit.
Step 3. Apply the oracle circuit U_f.
Step 4. Repeat Step 2.
Step 5. Measure each qubit. Let $y = (y_1, \ldots, y_n)$ be the list of measurement outcomes.

We find that f is a constant function if y is the all-zeros string. Why does this work? Recall that the Hadamard gate H maps $|0\rangle$ to the uniform superposition of $|0\rangle$ and $|1\rangle$. Thus the state reached after Step 2 is $2^{-n/2} \sum_x |x\rangle$, where the sum runs over all n-bit strings. The oracle circuit maps this state to $2^{-n/2} \sum_x (-1)^{f(x)} |x\rangle$. Finally, let us

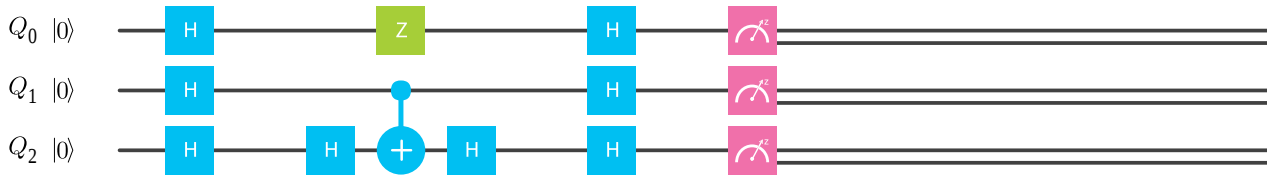
apply the layer of Hadamards at Step 4. It maps a basis state $|x\rangle$ to a superposition $2^{-n/2}\sum_y(-1)^{x\cdot y}|y\rangle$. Thus the state reached after Step 4 is $|\psi\rangle = \sum_y \psi(y)|y\rangle$, where $\psi(y) = 2^{-n}\sum_x (-1)^{f(x)+x\cdot y}$.

This is exactly what we need for the interference experiment described above. The final measurement at Step 5 plays the role of detecting the particle. As was shown above, the probability to measure $y = 0^n$ at Step 5 is one if f is a constant function and zero if f is a balanced function. Thus we have solved the Deutsch-Jozsa problem with certainty by making just one function evaluation.

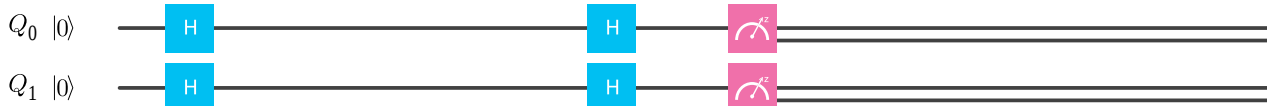
Example circuits

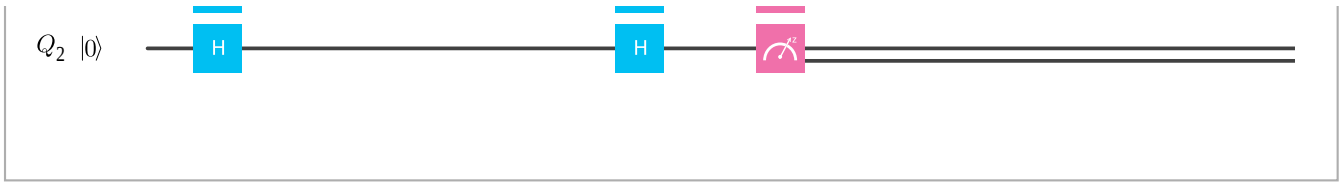
Suppose $n = 3$ and $f(x) = x_0 \oplus x_1 x_2$. This function is balanced since flipping the bit x_0 flips the value of $f(x)$ regardless of x_1, x_2. To run the Deutsch-Jozsa algorithm we need an explicit description of the oracle circuit U_f as a sequence of quantum gates. To this end we need a Z_0 gate such that $Z_0|x\rangle = (-1)^{x_0}|x\rangle$ and a controlled-Z gate $CZ_{1,2}$ such that $CZ_{1,2}|x\rangle = (-1)^{x_1 x_2}|x\rangle$. Using basic circuit identities (see the *Basic Circuit Identities and Larger Circuits* section), one can realize the controlled-Z gate as a CNOT sandwiched between two Hadamard gates.

DJ N=3 Example



DJ N=3 Constant



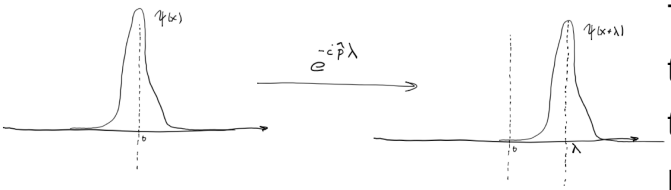

Q_2 $|0\rangle$
H
H

Quantum Phase Estimation

Quantum phase estimation is one of the most important subroutines in quantum computation. It serves as a central building block for many quantum algorithms and implements a measurement for essentially any Hermitian operator (https://en.wikipedia.org/wiki/Hermitian_adjoint). Recall that a quantum computer initially only permits us to measure individual qubits. If we want to measure a more complex observable, such as the energy described by a Hamiltonian H, we resort to quantum phase estimation. The routine prepares an eigenstate of the Hermitian operator in one register and stores the corresponding eigenvalue in a second register.

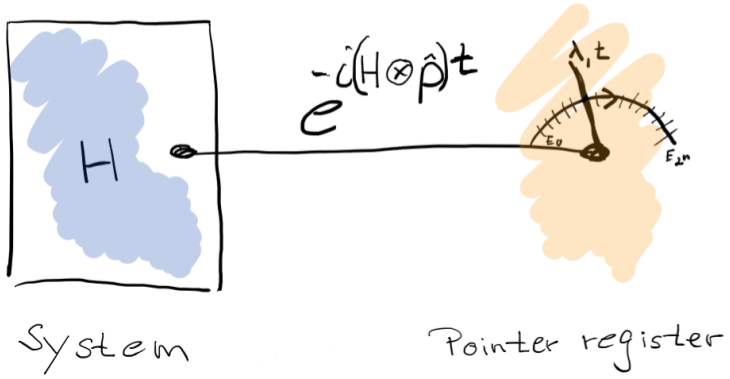
John von Neumann's measurement scheme

Quantum phase estimation is a discretization of von Neumann's prescription to measure a Hermitian observable $H = \sum_j E_j |\psi_j\rangle\langle\psi_j|$. The scheme that von Neumann envisioned is the following. We consider a quantum system that supports the observable H, which we want to measure. We assume that we are only able to measure simpler observables, in our case single qubits, or as in the original setting the location of a single particle. It is therefore our goal to reduce the measurement of the complex observable H to a measurement of the simpler observable, e.g. the location. This simple observable is then referred to as the pointer. To map the complex observable on to the simpler one we'll make use of a convenient observation from quantum mechanics. It is known that the momentum operator $\hat{p}$ generate shifts for single particles.



That is, if we apply the unitary $\exp(-i\hat{p}\lambda)$ to some wave packet $\psi(x)$, then this wave packet will be shifted by λ in the positive direction.

The scheme now assumes that we can apply the unitary evolution $\exp(-iH \otimes \hat{p}t)$ to both the system and the pointer register as illustrated in the following picture



This picture essentially describes von Neumann"s measurement scheme. We now follow the steps and first adjoin an ancilla -- the pointer -- which is a continuous quantum variable initialized in the state $|0\rangle$ (the origin), so that the system+pointer is initialized in the state $|\psi\rangle|0\rangle$, where $|\psi\rangle$ is the initial state of the system. Then we evolve according to the new Hamiltonian $K = H \otimes \hat{p}$ for a time t, so the evolution is given by

$$e^{-itH\otimes\hat{p}} = \sum_{j=1}^{2^N} |\psi_j\rangle\langle\psi_j| \otimes e^{-itE_j\hat{p}}.$$

We now observe the action of this measurement apparatus. Suppose that $|\psi\rangle$ is an eigenstate (https://en.wikipedia.org/wiki/Introduction_to_eigenstates) $|\psi_j\rangle$ of H, we find that the system evolves to $e^{-itH\otimes\hat{p}}|\psi_j\rangle|0\rangle = |\psi_j\rangle|x = tE_j\rangle$. A measurement of the position of the pointer with sufficiently high accuracy will provide an approximation to E_j .

The quantum algorithm

To carry out the above operation efficiently on a quantum computer, we discretize the pointer using r qubits, replacing the continuous quantum variable with a 2^r-dimensional space, where the computational basis states $|z\rangle$ of the pointer represent the basis of momentum eigenstates of the original continuous quantum variable. The label z is the binary representation of the integers 0 through $2^r - 1$. In this representation, the discretization of the momentum operator becomes

$$\hat{p} = \sum_{j=1}^{r} 2^{-j} \frac{\mathbb{I} - \sigma_j^z}{2}.$$

With this normalization $\hat{p}|z\rangle = \frac{z}{2^r}|z\rangle$. Now the discretized Hamiltonian $K = H \otimes \hat{p}$

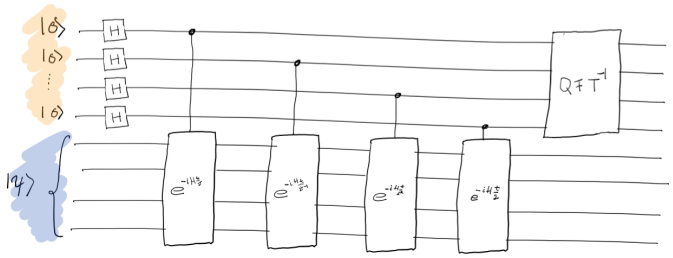
is a sum of terms involving at most $k+1$ qubits, if H is a Hamiltonian involving terms of at most k qubits. Thus we can simulate the dynamics of K using standard methods. In terms of the momentum eigenbasis the initial (discretized) state of the pointer is written $|x=0\rangle = \frac{1}{2^{r/2}} \sum_{z=0}^{2^r-1} |z\rangle$. This state can be prepared efficiently on a quantum computer by first initializing the qubits of the pointer in the state $|0\rangle \cdots |0\rangle$ and applying an (inverse) quantum Fourier transform (https://en.wikipedia.org/wiki/Quantum_Fourier_transform). Since we have a very simple initial state, the Fourier transform can be represented by a product of Hadamard matrices. The discretized evolution of the system+pointer now can be written

$$e^{-itH\otimes\hat{p}} |\psi_j\rangle|x=0\rangle = \frac{1}{2^{r/2}} \sum_{z=0}^{2^r-1} e^{-iE_j zt/2^r} |\psi_j\rangle z\rangle.$$

Performing an inverse quantum Fourier transform on the pointer leaves the system in the state $|\psi_j\rangle \otimes |\phi\rangle$, where

$$|\phi\rangle = \sum_{x=0}^{2^r-1} \left(\frac{1}{2^r} \sum_{z=0}^{2^r-1} e^{\frac{2\pi i}{2^r}\left(x - \frac{E_j t}{2\pi}\right) z} \right) |x\rangle,$$

which is strongly peaked near the location $x = \left\lfloor \frac{E_j t}{2\pi} \right\rceil$. To ensure that there are no overflow errors we need to choose $t < \frac{2\pi}{\|H\|}$. (We assume here, for simplicity, that $H \geq 0$.) It is easy to see that actually performing the simulation of K for $t=1$ is a product of r simulations of the evolution according to $\frac{1}{2^r} H \otimes \frac{\mathbb{I} - \sigma_k^z}{2}$ for $1, 2, 2^2, \ldots, 2^{r-1}$ units of time, respectively. This results in the general circuit for quantum phase estimation:

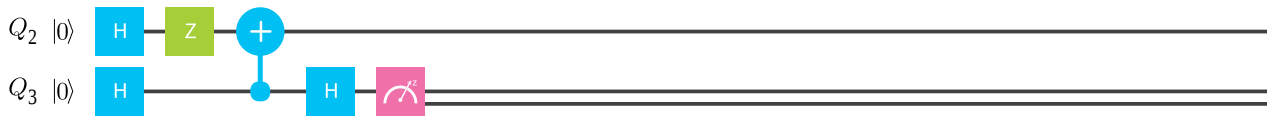


In order to implement the full circuit on a quantum computer, we still need to decompose the controlled unitaries $e^{-iH\frac{t}{2^k}}$ as well as the inverse quantum Fourier transform denoted by QFT^{-1} into our elementary gates.

Example circuit

The example below demonstrates quantum phase estimation for a toy single-qubit Hamiltonian σ^x acting on qubit Q_2. Qubit Q_3 serves as a pointer system. In this example the quantum Fourier transform on the pointer system is equivalent to the Hadamard gate H on Q_3. The discretized evolution of the system+pointer system is described by the CNOT gate. The final measurement outcome on the pointer qubit Q_3 is 0 or 1 depending on whether Q_2 is prepared in the $+1$ or -1 eigenstate of σ^x. In this example, qubit Q_2 is initialized in a state $ZH|0\rangle$ which is -1 eigenvector σ^x. Accordingly, the measurement outcome is 1.

Phase Estimation Circuit (-)



Phase Estimation Circuit (+)