Today's outline - February 07, 2023

- Universally approximating sets
- Classical reversible computations
- Implementing reversible operations
- Uniform superposition state
- Hamming distance & weight

Reading Assignment: Reiffel: 6.4; 7.1-7.3 Wong: 7.2

Homework Assignment #04:
due Tuesday, February 14, 2023

Homework Assignment #05:
due Tuesday, February 21, 2023

Universally approximating set of gates

The problem we encountered in trying to make a general unitary operator out of simple gates cannot be solved exactly, however the Solovay-Kitaev theorem states that there are finite sets of gates that can approximate any unitary transformation to arbitrary accuracy efficiently

If we desire accuracy to a level of 2^{-d}, there exists a polynomial $p(d)$ such that any single-qubit unitary transformation can be approximated to the desired accuracy by a sequence of no more than $p(d)$ gates

We want to find a finite set of gates that can approximate all single-qubit transformations so that with the addition of the C_{not}, we can prepare any unitary operator

Take the Hadamard and the C_{not} gates and add two phase gates $P_{\frac{\pi}{2}}$ and $P_{\frac{\pi}{4}}$

$$P_{\frac{\pi}{2}} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix} = |0\rangle\langle 0| + i|1\rangle\langle 1|, \quad P_{\frac{\pi}{4}} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} = |0\rangle\langle 0| + e^{i\frac{\pi}{4}}|1\rangle\langle 1| = e^{i\frac{\pi}{8}}T(-\frac{\pi}{8})$$

Making approximating set of gates

An arbitrary transformation can be viewed as a rotation of the qubit on the Bloch sphere by any amount

In order to make an approximating set to an arbitrary precision, we need to be able to combine the 4 gates to get as close as desired to any rotation, even irrational ones

Why do we only use rational rotations, those where for some integer m, $R^m = I$?

When working on the Bloch sphere, rational rotations around single axes are able to construct irrational rotations in three dimensions

$P_{\frac{\pi}{4}}$ is a rotation of $\frac{\pi}{4}$ about the z-axis of the Bloch sphere

$S = HP_{\frac{\pi}{4}}H$ is a rotation of $\frac{\pi}{4}$ about the x-axis

It can be shown that simply combining these two rational rotations $V = P_{\frac{\pi}{4}}S$ gives an irrational rotation

Making approximating set of gates (cont.)

Since V is irrational, any rotation can be approximated by a suitable power of V to within arbitrary precision

By transforming V as HVH we create an irrational rotation about an orthogonal axis and with these two, any operation can be constructed, recall that another way of generating an arbitrary transformation was

$$W = K(\delta)T(\alpha)R(\beta)T(\gamma)$$

$T(\alpha)$ rotates about the z-axis and $R(\beta)$ rotates about the y-axis

So we have that these 4 gates constitute a universally approximating set of gates

This is one possible set, others also exist

Classical reversible computations

Most classical computations are not reversible but reversible analogs can be made that are only slightly more computationally expensive

A reversible classical computation with n input and n output bits is just a permutation of the $N = 2^n$ bit strings

$$\pi : \mathbf{Z}_N \to \mathbf{Z}_N$$

An analogous quantum unitary transformation, U_π can be defined which simply reorders the basis elements

$$U_\pi : \sum_{x=0}^{N-1} a_x |x\rangle \mapsto \sum_{x=0}^{N-1} a_x |\pi(x)\rangle$$

A classical (non-reversible) computation on n input bits which produces m output bits maps the $N = 2^n$ input strings to the $M = 2^m$ output strings

$$f : \mathbf{Z}_N \to \mathbf{Z}_M$$
$$x \mapsto f(x)$$

This can be extended to a reversible function π_f which acts on $n+m$ which are partitioned into two registers

$$\pi_f : \mathbf{Z}_L \to \mathbf{Z}_L$$
$$(x, y) \mapsto (x, y \oplus f(x))$$

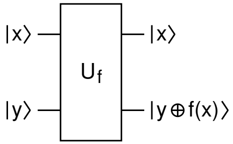
Reversible gates

$$\pi_f : \mathbf{Z}_L \to \mathbf{Z}_L$$
$$(x,y) \mapsto (x, y \oplus f(x))$$

π_f acts on the $L = 2^{n+m}$ bit strings, made up of an n bit input string x and an m bit output string y with the exclusive OR operator $\oplus$ so that when $y = 0$ initially the output is just f

π_f is clearly reversible so there is a corresponding unitary transformation U_f such that

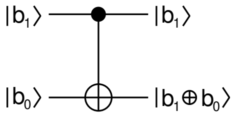
$$U_f : |x,y\rangle \to |x, y \oplus f(x)\rangle$$



The NOT gate is the simplest of the standard Boolean logic gates as it is already reversible

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

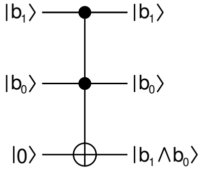
The CNOT gate, $C_{not} = \bigwedge_1 X$ requires two bits to be reversible, b_0 and b_1 and is basically an XOR gate



More classical gates

The reversible AND gate requires three bits and can be implemented with the Toffoli gate, $T = \bigwedge_2 X$

$$T|b_1, b_0, 0\rangle = |b_1, b_0, b_1 \wedge b_0\rangle$$
$$T|b_1, b_0, 1\rangle = |b_1, b_0, 1 \otimes b_1 \wedge b_0\rangle$$



Using the Toffoli gate it is possible to construct a complete set of Boolean gates and thus any combinatorial circuit

$$T|1, 1, x\rangle = |1, 1, \neg x\rangle$$
$$T|x, y, 0\rangle = |x, y, x \wedge y\rangle$$
$$T|1, x, y\rangle = |1, x, x \oplus y\rangle$$
$$T|x, y, 1\rangle = |x, y, \neg(x \wedge y)\rangle$$

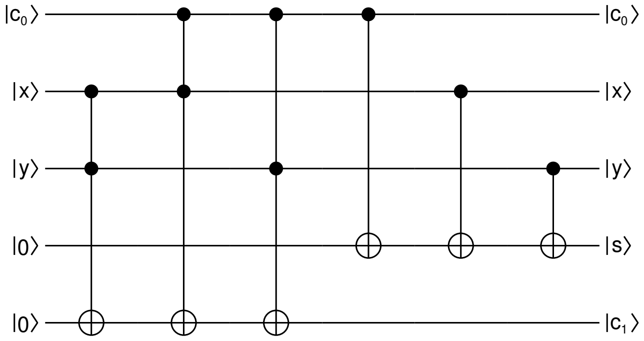
The Fredkin gate, a controlled swap can also be used to implement a complete set of Boolean functions

$$F = \bigwedge_1 S, \quad S : |xy\rangle \to |yx\rangle$$

$$F|x, 0, 1\rangle = |x, x, \neg x\rangle, \quad F|x, y, 1\rangle = |x, y \vee x, y \vee \neg x\rangle, \quad F|x, 0, y\rangle = |x, y \wedge x, y \wedge \neg x\rangle$$

One-bit full adder

The one-bit adder is an example of a reversible implementation with Toffoli and C$_{not}$ gates



In this one-bit adder, $|x\rangle$ and $|y\rangle$ are the input bits, $|c_0\rangle$ is the carry-in bit, $|s\rangle$ is the result, and $|c_1\rangle$ is the carry-out bit

https://tinyurl.com/y2avser4

Implementing reversible operations

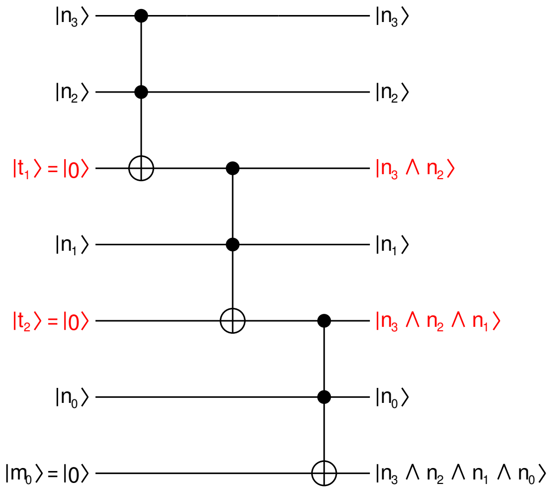
One way to implement reversible operations with a minimal number of extra bits is to use intermediate bits to hold temporary information during a computation

Consider the four bit conjunction circuit

The temporary bits $|t_1\rangle$ and $|t_2\rangle$ hold the intermediate calculations but are discarded in the state they are left

This is wasteful of bits (qubits)

https://tinyurl.com/bpacjs6y



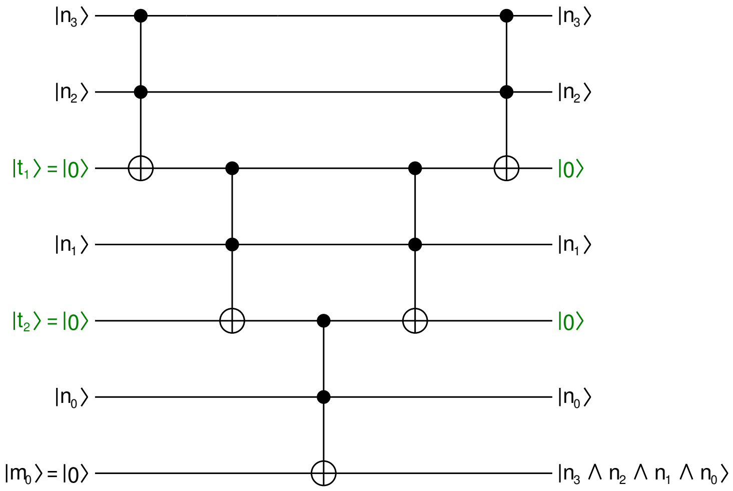
$|n_3\rangle$
$|n_2\rangle$
$|t_1\rangle = |0\rangle$
$|n_1\rangle$
$|t_2\rangle = |0\rangle$
$|n_0\rangle$
$|m_0\rangle = |0\rangle$
$|n_3\rangle$
$|n_2\rangle$
$|n_3 \wedge n_2\rangle$
$|n_1\rangle$
$|n_3 \wedge n_2 \wedge n_1\rangle$
$|n_0\rangle$
$|n_3 \wedge n_2 \wedge n_1 \wedge n_0\rangle$

A more reversible implementation

The challenge is to minimize the intermediate bits by reusing them which can be done by uncomputing them to their original state by reversing the operation applied to them

The temporary bits $|t_1\rangle$ and $|t_2\rangle$ hold the intermediate calculations but are restored to their inital states

https://tinyurl.com/2tpn54jn



The uniform superposition state

An important state of an n qubit system is the so-called uniform superposition state, $|s\rangle$ which can be written

$$|s\rangle = A\left[|0\dots00\rangle + |0\dots01\rangle + |0\dots10\rangle + \cdots + |1\dots11\rangle\right] = A\sum_{x=0}^{2^n-1}|x\rangle$$

The uniform superposition state must be normalized

$$1 = \langle s|s\rangle = |A|^2\sum_{x'=0}^{2^n-1}\sum_{x=0}^{2^n-1}\langle x'|x\rangle = |A|^2\sum_{x=0}^{2^n-1}\langle x|x\rangle = |A|^2 2^n \;\rightarrow\; A = \frac{1}{\sqrt{2^n}}$$

Since the individual states are orthogonal and normalized, the normalization constant is

What does this mean in practice for systems with 1-3 qubits?

$$n=1: \quad |s\rangle = \tfrac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$$

$$n=2: \quad |s\rangle = \tfrac{1}{\sqrt{4}}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right) = \tfrac{1}{\sqrt{4}}\left(|0\rangle + |1\rangle + |2\rangle + |3\rangle\right)$$

$$n=3: \quad |s\rangle = \tfrac{1}{\sqrt{8}}\left(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle\right)$$

$$= \tfrac{1}{\sqrt{8}}\left(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle\right)$$

The Walsh-Hadamard transformation

From the $n = 1$ example it is clear that $|s\rangle \equiv H|0\rangle$

Thus the uniform superposition state can be generated by applying the Hadamard transformation to each of the n qubits

$$H_n \otimes H_{n-1} \otimes \cdots \otimes H_1 |0_n 0_{n-1} \dots 0_1\rangle = \frac{1}{\sqrt{2^n}} \left[(|0_n\rangle + |1_n\rangle)(|0_{n-1}\rangle + |1_{n-1}\rangle) \cdots (|0_1\rangle + |1_1\rangle) \right]$$

$$= \frac{1}{\sqrt{2^n}} \left[|0\dots00\rangle + |0\dots01\rangle + |0\dots10\rangle + \cdots + |1\dots11\rangle \right] = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = |s\rangle$$

This transformation, applying H to each of the qubits is called the Walsh-Hadamard transformation and can be written

$$W = H \otimes H \otimes \cdots \otimes H$$

In shorthand notation the application of W is written

$$W|0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle, \qquad N = 2^n$$

Hamming distance

A useful concept is the Hamming distance, $d_H(x, y)$ which is defined as the number of bits in which two bit strings x and y differ

The Hamming weight is the number of bits in which a bit string x differs from a string which is all zeroes, $d_H \equiv d_H(x, 0)$

Operations on two bit strings, x and y, are defined as:

$x \cdot y$	the number of common bits with a value of 1
$x \oplus y$	the bitwise exclusive-OR of the strings
$x \wedge y$	the bitwise AND of the strings
$x \oplus 11\dots1 = \neg x$	the bit string that flips all the bits in x

Several identities arise from these definitions

$$x \cdot y = d_H(x \wedge y)$$

$$x \cdot y + x \cdot z =_2 x \cdot (y \oplus z)$$

$$(x \cdot y) \bmod 2 = \tfrac{1}{2}\left(1 - (-1)^{x \cdot y}\right)$$

$$d_H(x \oplus y) =_2 d_H(x) + d_H(y)$$

$$\sum_{x=0}^{2^n-1} (-1)^{x \cdot y} = \begin{cases} 2^n & y = 0 \\ 0 & y \neq 0 \end{cases}$$

Bit string identities

$$x \cdot y = d_H(x \wedge y) \qquad (x \cdot y) \bmod 2 = \tfrac{1}{2}\left(1 - (-1)^{x \cdot y}\right)$$

$$x \cdot y + x \cdot z =_2 x \cdot (y \oplus z) \qquad d_H(x \oplus y) =_2 d_H(x) + d_H(y)$$

If we have $x = 1011$, $y = 0111$, and $z = 0010$ these identities can be easily checked

$$x \cdot y = 1011 \cdot 0111 = 2 \qquad d_H(x \wedge y) = d_H(0011) = 2$$

$$\begin{aligned}(x \cdot y) \bmod 2 &= (1011 \cdot 0111) \bmod 2 \\ &= 2 \bmod 2 = 0\end{aligned} \qquad \begin{aligned}\tfrac{1}{2}\left(1 - (-1)^{x \cdot y}\right) &= \tfrac{1}{2}\left(1 - (-1)^{2}\right) \\ &= \tfrac{1}{2}(1 - 1) = 0\end{aligned}$$

$$\begin{aligned}x \cdot y + x \cdot z &= 1011 \cdot 0111 + 1011 \cdot 0010 \\ &= 2 + 1 = 3 =_2 1\end{aligned} \qquad \begin{aligned}x \cdot (y \oplus z) &= 1011 \cdot (0111 \oplus 0010) \\ &= 1011 \cdot 0101 = 1\end{aligned}$$

$$\begin{aligned}d_H(x \oplus y) &= d_H(1011 \oplus 0111) \\ &= d_H(1100) = 2 =_2 0\end{aligned} \qquad \begin{aligned}d_H(x) + d_H(y) &= d_H(1011) + d_H(0111) \\ &= 3 + 3 = 6 =_2 0\end{aligned}$$