Quantum Computation **CMU 15-859BB, Fall 2018**

Week 6 work: Oct. 11 — Oct. 18
12-hour week
Obligatory problems are marked with [**]

---

1. **[CCNOT.]** In class we showed how to simulate classical AND, OR, NOT, and FANOUT gates using only CSWAP gates (and ancillas that could be $|0\rangle$ or $|1\rangle$). Show how to do the same using only CCNOT gates (controlled-CNOT gates, aka Toffoli gates).

2. **[Controlled-Controlled-U.]** Let U be a d-dimensional qudit gate (i.e., a unitary $d \times d$ matrix). Define the "controlled-controlled-U" gate, which operates on 2 qubits and a qudit (call these the Control1 qubit, the Control2 qubit, and the Target qudit) as follows:

$$\begin{aligned}
|00\rangle \otimes |y\rangle &\mapsto |00\rangle \otimes |y\rangle\,, \\
|01\rangle \otimes |y\rangle &\mapsto |01\rangle \otimes |y\rangle\,, \\
|10\rangle \otimes |y\rangle &\mapsto |10\rangle \otimes |y\rangle\,, \\
|11\rangle \otimes |y\rangle &\mapsto |11\rangle \otimes (U\,|y\rangle).
\end{aligned}$$

(a) [$**$] How many rows/columns does this controlled-controlled-U gate have? (I.e., what is the dimension of space on which it operates?)

(b) [$**$] Prove that the gate is unitary.

(c) [$**$] Suppose V is a (unitary) qudit gate with $V^2 = U$. Show that controlled-controlled-U can be implemented as follows:

- Controlled-V on Control2 and Target.
- CNOT, with Control1 the control and Control2 the target.
- Controlled-$V^\dagger$ on Control2 and Target.
- CNOT, with Control1 the control and Control2 the target.
- Controlled-V on Control1 and Target.

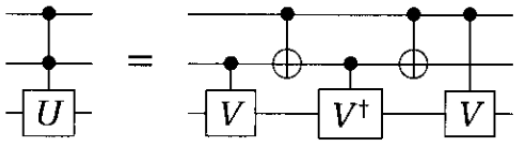

Figure 1: Building controlled-controlled-U

(Remark: Earlier we saw the 1-qubit $\sqrt{\text{NOT}}$ gate. If you take this to be V in the above construction, you see that CCNOT can be built out of 2-qubit gates. Interestingly, the classical CCNOT gate *cannot* be computed by any circuit of classical 2-bit gates!)

3. **[Deutsch's Algorithm.]** David really enjoys the fact that one can easily take a classical circuit computing a Boolean function F, and convert it into a quantum circuit which implements the same Boolean function when given "classical inputs" — but which also can accept quantum superpositions of classical inputs. For a whole bunch of small Boolean functions, David built a small quantum circuit that implements that function.

   Unfortunately, David forgot to label his quantum circuits, and now he forgets which one computes what! David runs across an old circuit $Q^{\pm}$ he built which evidently "sign-implements" some 1-bit Boolean function $F : \{0,1\} \to \{0,1\}$. That is[1], for each $x \in \{0,1\}$ it holds that $Q^{\pm} |x\rangle = (-1)^{F(x)} |x\rangle$ — but David doesn't know what F is. Of courses there are only four possibilities: $F(x) \equiv 0$, $F(x) \equiv 1$, $F(x) = x$, and $F(x) = \mathrm{NOT}(x)$. Let's call the first two possibilities "constant functions" and the second two possibilities "nonconstant functions".

   (a) [$**$] Show that it is possible for David to tell whether F is a constant function or a nonconstant function by just using $Q^{\pm}$ *once*. Specifically, you should describe a 1-qubit circuit, which may have various gates but only *one* instance of $Q^{\pm}$ in it. Your circuit start with a qubit initialized to $|0\rangle$, and should end with a measurement gate. And it should have the property that based on the measurement outcome, David can know with 100% certainty whether F is a constant function or a nonconstant function.

   (Hint: Use the good old Rotate, Compute, Rotate paradigm.)

   (b) [$**$] Show that if you have access to a classical circuit C computing one of the four possible F's, and you only run one bit through it, you cannot gain any information at all about whether F is a constant function or a nonconstant function.

[1] Let's ignore ancillas, as always.

4. **[Classical Query Complexity.]** Pythia is holding a strange contest. She is selling a sealed (classical) circuit C that computes some Boolean function $F : \{0,1\}^n \to \{0,1\}$. She promises that $F \not\equiv 0$; that is, there is at least one string x such that $F(x) = 1$. You can buy as many copies of C as you want, but each copy costs a drachma.

   Pythia promises you fame and fortune if you can build a (classical) circuit S that outputs a string x such that $F(x) = 1$. The inputs to your circuit S are simply ancillas. You should think of each copy of C that you buy as a big gate that can be used in your circuit S. You are not allowed to "peer into the inner workings of C"; you can only incorporate it into your circuit as a "black box".

   (a) [$**$] Prove that you can attain fame and fortune by spending $2^n - 1$ drachmas. Actually, it would be too annoying for me to make you describe a circuit, so instead describe classical deterministic pseudocode that incorporates calls to a "subroutine" C. Your code should take no input, make at most $2^n - 1$ calls to C, and end by outputting an x such that $F(x) = 1$.

   (b) [$**$] Prove rigorously that it is impossible for you to attain fame and fortune without spending at least $2^n - 1$ drachmas. To be very precise, show that any circuit S incorporating fewer than $2^n - 1$ copies of C cannot be correct, in the sense that there exists an F for which S fails. (No wishy-washiness in your solution; it should be an airtight proof.)

   (c) [$**$] The famous "SAT" problem in theoretical computer science is equivalent to the following: Given as input the (description of) an n-input Boolean circuit C, with the promise that there is at least one string $x^* \in \{0,1\}^n$ such that $C(x^*) = 1$, output any string x such that $C(x) = 1$. If you were able to show that there is no classical algorithm solving this problem in fewer than $2^n - 1$ steps, then you would have shown $\mathsf{P} \neq \mathsf{NP}$, and real-world fame and fortune would truly be yours. How come your solution to part (b) does not achieve this?

5. **[Linear Algebra Modulo 2.]** The integers modulo 2 constitute what's called a "field" in mathematics: a set of numbers (namely 0 and 1) for which all the standard operations of plus, minus, times, and division-by-nonzero work as expected. This "field" is sometimes denoted $\mathbb{F}_2$. (Other examples of fields include the real numbers, the complex numbers, the rational numbers, and the integers modulo p whenever p is prime.) It's a wonderful fact that pretty much all of linear algebra works just as well when the underlying scalars come from any fixed field, like $\mathbb{F}_2$. The set of all n-dimensional vectors in this case is denoted $\mathbb{F}_2^n$. (You're most used to the cases $\mathbb{R}^n$ and $\mathbb{C}^n$, when the scalars are reals and complexes, respectively.)

   (a) [$**$] There's pretty much only one pitfall to watch out for: Show that the "dot product" in $\mathbb{F}_2^n$, namely the operation $\vec{u} \cdot \vec{v} = \sum_{i=1}^{n} u_i v_i$ (mod 2 of course), doesn't act like an "inner product", in the sense that it's perfectly possible to have $\vec{u} \cdot \vec{u} = 0$ even though $\vec{u} \neq \vec{0}$. Because of this, notions like "orthogonal basis" or "$\vec{u}$ and $\vec{v}$ are perpendicular" don't make as much sense in $\mathbb{F}_2^n$. (We still frequently use the dot product operation anyway, though.)

   (b) [$**$] Recall that a set of vectors $\vec{u}_1, \ldots, \vec{u}_k$ is said to be *linearly independent* if the only linear combination $c_1\vec{u}_1 + \cdots + c_k\vec{u}_k$ that equals $\vec{0}$ is the trivial one with $c_1 = c_2 = \cdots = c_k = 0$. As usual, the span (set of all linear combinations) of a set of k linearly independent vectors is called a *k-dimensional subspace*. Show that in $\mathbb{F}_2^n$, every k-dimensional subspace contains exactly 2^k vectors.

   (c) [$**$] As usual in linear algebra, one can study solutions of systems of linear equations like $Ax = 0$ (where $A \in \mathbb{F}_2^{m \times n}$ is a matrix and x is a vector of n unknowns) or, more generally, $Ax = b$ (where $b \in \mathbb{F}_2^n$ is a fixed right-hand side). The usual facts about Gaussian Elimination apply. Show that the set of solutions x to $Ax = 0$ forms a subspace of dimension equal to $n - r$, where r is the maximum size of a linearly independent set of rows of A.

   (d) [$**$] For a more general system $Ax = b$, prove that either there is *no* solution, or else there are 2^{n-r} solutions, where again, r is the maximum size of a linearly independent set of rows of A.

6. **[Real Quantum Computing.]** So far in the course complex amplitudes have almost never arisen, even though they *may*, according to the laws of quantum mechanics. Almost all of our favorite quantum gates (NOT, H, Z, "Rotations", CNOT, SWAP, CSWAP, CCNOT) are represented by unitary matrices with real entries, meaning they never create states with complex (non-real) amplitudes. We briefly mentioned the 1-qubit *phase gate* P, whose matrix is $\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$; i.e., it sends $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $i\,|1\rangle$.[2] It's a fact that if you want the ability to generate all complex unitary transformations, it's sufficient to: (i) be able to generate all real unitary transformations; and, (ii) be able to apply the P gate.

   For this problem, prove that "real amplitudes are sufficient for universal quantum computation". Specifically, show how to take any quantum circuit that has T gates from the set $\{\text{real unitaries}, \text{measurements}, P\}$ and "simply" convert it to an equivalent[3] quantum circuit with no P gates (and hence where all internal states have only real amplitudes). Your new circuit should have just one extra qubit, and it should still have T gates, though some of these gates might act on more qubits than they previously did. (Hint: $(a+bi)\,|x\rangle$ versus $a\,|x\rangle\,|0\rangle + b\,|x\rangle\,|1\rangle$.)

[2] One more popular gate with complex entries is the "Y" gate, which operates on 1 qubit via the matrix $\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$

[3] With respect to measurement outcomes, assuming inputs are initialized to unentangled $|0\rangle$'s and $|1\rangle$'s.

7. **[The Birthday Attack.]**

(a) [**] Let $1 \leq k \leq n$ be integers. Suppose that every time you press a button, a computer prints out a random integer between 1 and n (inclusive). Show that if you press the button $k+1$ times, the probability that all the integers you see are distinct is

$$p_{k,n} = (1 - 1/n)(1 - 2/n)(1 - 3/n) \cdots (1 - k/n).$$

(b) [**] Produce high quality plots of $p_{k,n}$ vs. k for each $n = 10^i$, $i = 1 \ldots 6$. In each case, let k range from $.1\sqrt{n}$ to $4\sqrt{n}$.

(c) Prove that the function $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = 2^x$ is convex. (This means that if you connect the points $(a, f(a))$ and $(b, f(b))$ on the graph of f by a straight line segment, the graph is below that straight line between $x = a$ and $x = b$. It's also equivalent to the fact that just the midpoint value $f(m)$ (for $m = \frac{a+b}{2}$) is below the midpoint of the line segment, $\frac{f(a)+f(b)}{2}$. You can prove the convexity claim either with calculus, or else by proving the midpoint statement using the more lowbrow inequality $(A/\sqrt{2} - B/\sqrt{2})^2 \geq 0$ for well-chosen A and B.)

(d) Deduce that if $0 \leq x \leq 1$, then $1 + x \geq 2^x$.

(e) Deduce that if $0 \leq x \leq 1$, then $1 - x \leq 2^{-x}$.

(f) Deduce that $p_{k,n} \leq 2^{-\frac{k(k+1)}{2n}}$.

(g) Deduce that if $k \geq 4\sqrt{n}$, then $p_{k,n} \leq 2^{-8}$.

(h) Show that if $x_1, x_2 \geq 0$, then $(1 - x_1)(1 - x_2) \geq 1 - x_1 - x_2$.

(i) Show that $p_{k,n} \geq 1 - \frac{k(k+1)}{2n}$. Deduce that $p_{k,n} \geq 1 - \frac{k^2}{n}$.

(j) Show that if $k \leq .1\sqrt{n}$, then $p_{k,n} \geq .99$.

(k) [**] Dan goes to the hardware store and buys $N/2$ differently colored buckets of paint (N is even). He also buys N balls and an opaque urn. He groups the balls into $N/2$ pairs, and paints each pair a different color. He then puts all the balls into the urn.

You come along and start pulling balls out of the urn, at random. Show that there is some absolute constant $c > 0$ such that the following is true: If you pull out fewer than $c\sqrt{N}$ balls, then the probability of you getting two balls of the same color is at most .01.

8. **[The Probability of Coprimality.]** Suppose that A and B are chosen independently and uniformly at random from the set $\{1, 2, \ldots, S\}$.

   (a) For P a fixed prime, show that $\mathbf{Pr}[A, B \text{ both divisible by } P] \leq 1/P^2$.

   (b) Show that
   $$\mathbf{Pr}[\text{GCD}(A, B) \neq 1] \leq \sum_{\text{primes } P} 1/P^2.$$

   (c) Show that
   $$\sum_{\text{primes } P} 1/P^2 \leq .99,$$
   thereby concluding that "The probability that two random integers are coprime is at least 1%." (In fact, you don't need calculus; e.g., there's an elementary proof that $\sum_{\text{primes } P \geq 5} 1/P^2 \leq 2/5$, and hence $\sum_{\text{primes } P} 1/P^2 \leq .77\ldots$)