CS 593/MA 595 - Intro to Quantum Computation
Theoretical Homework 7

Due Wednesday, November 5 at 11:59PM (upload to Brightspace)

1. In this homework, we will explicitly build a (classical) circuit that calculates the modular exponentiation circuit in Shor's algorithm, with X, CNOT, and Toffoli gates (elementary gates). We will do this step-by-step, so this homework is long but easy.

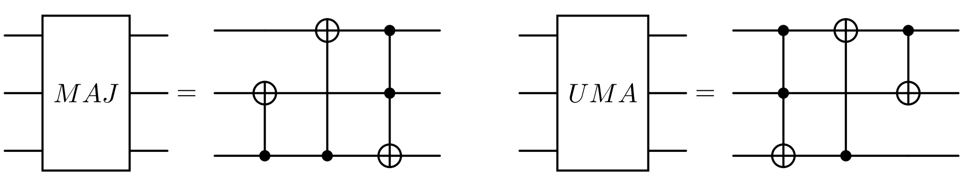
   Let's define some notations first. Let $[x]_n$ represent the binary representation of x stored in an n-bit register, and we always assume $0 \leq x < 2^n$ whenever x is in $[x]_n$. We write our goal in this homework as finding circuit $\mathrm{IME}_{m,n,a,N}$ with polynomial gates that satisfies

$$[k]_m[x]_n \xrightarrow{\mathrm{IME}_{m,n,a,N}} [k]_m[(a^k x)\%N]_n,$$

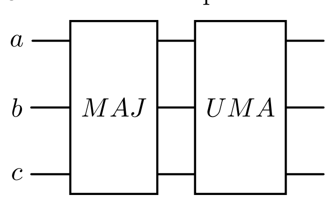
   which means that the circuit $\mathrm{IME}_{a,N}$ transforms a state with two registers of lengths m, n and values k, x to a state where the second register's value becomes $a^k x\%N$, with the help of ancillary bits. Here $a\%N$ represents the modular operation a mod N.

   Feel free to use the facts that the reverse and single-bit control of such a circuit can be done with constant factors: i.e., a k gate circuit R's single-bit controlled version $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes R$ can be realized with at most $3k$ gates, which can be proven by induction on k. Meanwhile, rearranging the order of bits (registers) is considered free and ancillary bits are also free to use. REMEMBER: always clean up your ancillary bits - ancillary bits are borrowed in value 0 and must be returned with value 0.

   (a) Let $s = (a + b + c)\%2$, and let gates MAJ and UMA be



   Calculate the outputs of



   (b) Construct circuit ADD_n (addition) with $O(n)$ elementary gates (equivalently, $O(n)$ MAJ and UMA gates) satisfying:

$$[b]_1[x]_n[y]_n \xrightarrow{\mathrm{ADD}_n} [b]_1[x]_n[(x + y + b)\%2^n]_n.$$

   (Hint: Make use of a carrier bit $s_i = (x_i + y_i + s_{i-1})\%2^n$. (a) gives the base case for the recursive construction.)

(c) Construct circuit NEG_n (negation), COPY_n (copying), and $\text{PREP}_{n,N}$ (preparation) with $O(n)$ elementary gates, where $0 \le N < 2^n$, satisfying

$$\begin{aligned}
[x]_n &\xrightarrow{\text{NEG}_n} [(-x)\%2^n]_n, \\
[x]_n[0]_n &\xrightarrow{\text{COPY}_n} [x]_n[x]_n, \\
[0]_n &\xrightarrow{\text{PREP}_{n,N}} [N]_n,
\end{aligned}$$

(d) With all gates introduced above, construct SUB_n (subtraction), CMP_n (comparison) with $O(n)$ elementary gates satisfying:

$$\begin{aligned}
[x]_n[y]_n &\xrightarrow{\text{SUB}} [x]_n[(y-x)\%2^n]_n, \\
[x]_n[y]_n[0]_1 &\xrightarrow{\text{CMP}} [x]_n[y]_n[x \le_? y]_1,
\end{aligned}$$

where $x \le_? y = \begin{cases} 0, & x > y \\ 1, & x \le y \end{cases}$.

(e) With all gates introduced above, construct $\text{MADD}_{n,N}$ (modular addition), where $0 \le N < 2^n$, with $O(n)$ elementary gates satisfying:

$$[x]_n[y]_n \xrightarrow{\text{MADD}_{n,N}} [x]_n[(x+y)\%N]_n.$$

(f) With all gates introduced above, construct $\text{MMUL}_{n,a,N}$ (modular constant multiplication), where $0 \le a < N < 2^n$, with $O(n^2)$ elementary gates satisfying:

$$[x]_n[0]_n \xrightarrow{\text{MMUL}_{n,a,N}} [x]_n[(ax)\%N]_n.$$

(g) With all gates introduced above, construct $\text{IMM}_{n,a,N}$ (in-place modular constant multiplication), where $0 \le a < N < 2^n$ and a, N are coprime, with $O(n^2)$ elementary gates satisfying:

$$[x]_n \xrightarrow{\text{IMM}_{n,a,N}} [(ax)\%N]_n.$$

You may assume that you know $0 \le a^{-1} < N$ such that $(a^{-1}a)\%N = 1$, due to Euclidean algorithm.

(h) With all gates introduced above, construct $\text{IME}_{m,n,a,N}$ (in-place modular constant exponentiation), where $0 \le a < N < 2^n$ and a, N are coprime, with $O(n^2 m)$ elementary gates satisfying:

$$[k]_m[x]_n \xrightarrow{\text{IME}_{m,n,a,N}} [k]_m[(a^k x)\%N]_n.$$

Now, we have an explicit construction for the classical arithmetic circuit with polynomial quantum gates to be used in Shor's discrete logarithm and factorization algorithm.