Shor’s Algorithm

Ben Prather
UIUC Algorithms Interest Group,
Sep 30, 2016

History

- Before/invented “quantum computing” as a popular field
- CS people largely ignored the field, a few physicists (Feynman, Deutsch) considered the general problem
- But factorization is the basis for cryptography, and breaking cryptography gets attention
- Shor’s paper was published in ‘94. The DOD hosted its first conference on quantum cryptography in ‘95, and the NSA put out a call for research in ‘96. Research has accelerated since

Overview

- Broadly, Shor's algorithm has two parts:

1. Reduce the factorization problem to finding the period of a function – a wrapper I will hereafter call the "factor-finder"
2. Efficiently find the period of integer functions via the quantum Fourier transform – the "period-finder"

Factor-finder: algorithm

1. Pick a random (relatively prime) number a < N, and find the period of $f(x) = a^x \mod N$, i.e. the smallest $r \mid f(x+r) = f(x)$. (Using the quantum Fourier transform to be discussed)
2. Repeat until r is even and $a^{r/2} \not\equiv -1 \pmod{N}$
3. Once this is true, N must at least one nontrivial factor

$$\gcd(a^{r/2}+1, N) \quad \text{or} \quad \gcd(a^{r/2}-1, N)$$

What?

- The integers coprime with N (that is, everything but its factors) form a finite, abelian group.
- Becuase of this, for a given member we can find the order (period) r such that $a^r \equiv 1 \pmod N$
- That is, starting at a and multiplying by itself modulo N, we will eventually reach a again
- N divides (is a factor of) (a^r – 1). This is a good start: find the order, and we've found something that shares factors with N.

Nontrivial Square Roots

- Now define $b \equiv a^{r/2} \pmod N$ (for r even)
- b must be a square root of 1 (mod N), but can't itself be 1 (otherwise the period would have been r/2)
- Further, let's require that b isn't -1 mod N (the other requirement in step 3)
- Now let's define $d = \gcd(b-1, N)$, which obviously divides N, and can be found quickly via the Euclidean algorithm
- Provided d ≠ 1,N this is our answer

Why d ≠ 1,N

- If d = N, then N divides b-1, and thus $b \equiv 1 \pmod N$, which we've said is false
- If d = 1, then by Bézout's identity there are u,v such that

$$(b-1)u + Nv = 1$$

$$(b^2-1)u + N(b+1)v = b+1$$

N divides the equation (since $b^2 - 1 = a^r - 1$), implying $b \equiv -1 \pmod N$, which again is false

- Thus d is a nontrivial divisor of N, and we are finished

A more constructive explanation

- When we define $b^2 \equiv 1 (\mod N)$
- Via the Chinese remainder theorem we can then say b satisfies one of

$$b_1 \equiv 1 \mod n_1 \equiv 1 \mod n_2$$
$$b_2 \equiv 1 \mod n_1 \equiv -1 \mod n_2$$
$$b_3 \equiv -1 \mod n_1 \equiv 1 \mod n_2$$
$$b_4 \equiv -1 \mod n_1 \equiv -1 \mod n_2$$

- The first and last solutions are 1 and -1, but the middle two are some other, nontrivial solution (i.e. nontrivial square roots of 1)

Constructive solution continued

- Having required that neither (b+1) nor (b-1) is zero, we can construct

$$b^2 - 1 = (b + 1)(b - 1) = cN$$

- And thereby say that at least one of b+1 or b-1 shares a nontrivial divisor with N

Note on prime-finder

- This whole thing relies on choosing a good starting number a. However, one can show (well, not me, but someone showed) that
  - Provided N has at least two *distinct* factors, and is not even
  - There is a greater than ½ probability of choosing the correct a, i.e. one for which r is even and $a^{r/2} \not\equiv -1 \pmod{N}$.
- These are the only conditions on Shor's algorithm as a whole

Period-finding: prepare the system

- Goal: find first $r \mid a^{x+r} \equiv a^x \mod N$
- We will need input and output registers capable of representing $Q = 2^q \geq N^2 > N \cdot r$ different numbers – i.e., q quantum bits long
- Initialize these to: $$|\psi\rangle = Q^{-1/2} \sum_{x=0}^{Q-1} |x\rangle$$
- And implement f(x) as a quantum function: $$\hat{f}\,|\psi\rangle = Q^{-1/2} \sum_{x=0}^{Q-1} |x, f(x)\rangle$$

Wait, “implement f?”

- All that means is design an operator such that

$$\hat{f}\,|x\rangle = |x, a^x \pmod N)\rangle$$

- The quantum circuit for modular exponentiation is similar to the classical algorithm for exponentiation by squaring
- Exponentiation requires O(n) multiplications and squarings in the number of digits
- And the fastest reversible multiplication algorithm requires O(n log(n) log(log(n))) (Schönhage-Strassen)

Period-finding: apply the qFt

- The quantum Fourier transform is just the discrete transform applied to a superposition of states. It maps each x like:

$$U_{QFT}\left|x\right\rangle = Q^{-1/2}\sum_{y=0}^{Q-1}\omega^{xy}\left|y\right\rangle \quad \text{where} \quad \omega = e^{\frac{2\pi i}{Q}}$$

- Thus on our state:

$$U_{QFT}\hat{f}\left|\psi\right\rangle = Q^{-1}\sum_{x=0}^{Q-1}\sum_{y=0}^{Q-1}\omega^{xy}\left|y, f(x)\right\rangle$$

Period-finding: apply the qFt

- We can reorder the sum so that the state reads

$$U_{QFT}\hat{f}\,|\psi\rangle = Q^{-1}\sum_{y=0}^{Q-1}\sum_{z}|y,z\rangle \sum_{x|f(x)=z}\omega^{xy}$$

Sum over range

Sum over multiplicity on range

Sum over (transformed) domain

- Breaking x into x_0 + rb, where x_0 is the first occurrence $f(x_0)=z$, and r is the period of f:

$$\sum_{x|f(x)=z}\omega^{xy} = \sum_{b=0}^{(Q-x_0-1)/r}\omega^{(x_0+rb)y} = \omega^{x_0 y}\sum_{b}\omega^{rby}$$

Period-finding: interpreting the result

- Since $\omega^{ry_0} = e^{2\pi i \frac{ry_0}{Q}} \approx 1$, $\frac{ry_0}{Q}$ will be nearly some integer c
- Taking the continued fraction expansion eventually yields integers d,s such that

$$\frac{y_0}{Q} \approx \frac{d}{s} \approx \frac{c}{r}$$

where $\left|\frac{y_0}{Q} - \frac{d}{s}\right| < \frac{1}{2Q}$ but $s < N$.

- This is our candidate for r! We can verify s or guess similar candidates, and start over if necessary

Notes on the period-finder

- f(x) must be implemented as a quantum function, which actually takes more gates than the quantum Fourier transform itself.
- Because of this, the circuits for period-finding also change for each choice of a: choose wrong, reconfigure the computer. Luckily there's a (1-1/8) = 87.5% chance of success after 3 iterations.

Implications

- RSA, Diffie-Hellman, and even elliptic-curve encryption algorithms assume that the factorization problem is exponentially hard – but a quantum computer would be able to recover users' secrets (factors) from public information (products) in only polynomial time in the key length
- There has been significant work on "post-quantum" algorithms, and quantum-resistant replacements for RSA, Diffie-Hellman, hashing, etc have been put forward. But adoption is slow (there are a lot of computers to change)
- Research in quantum computing, and (post-quantum and quantum-based) cryptography has increased steadily since.

References

- Original paper (clear, worth reading): here
- Wikipedia's explanation (notation I use): here
- Alternative, clearer explanation: here
- Scott Aaronson's popular explanation: here