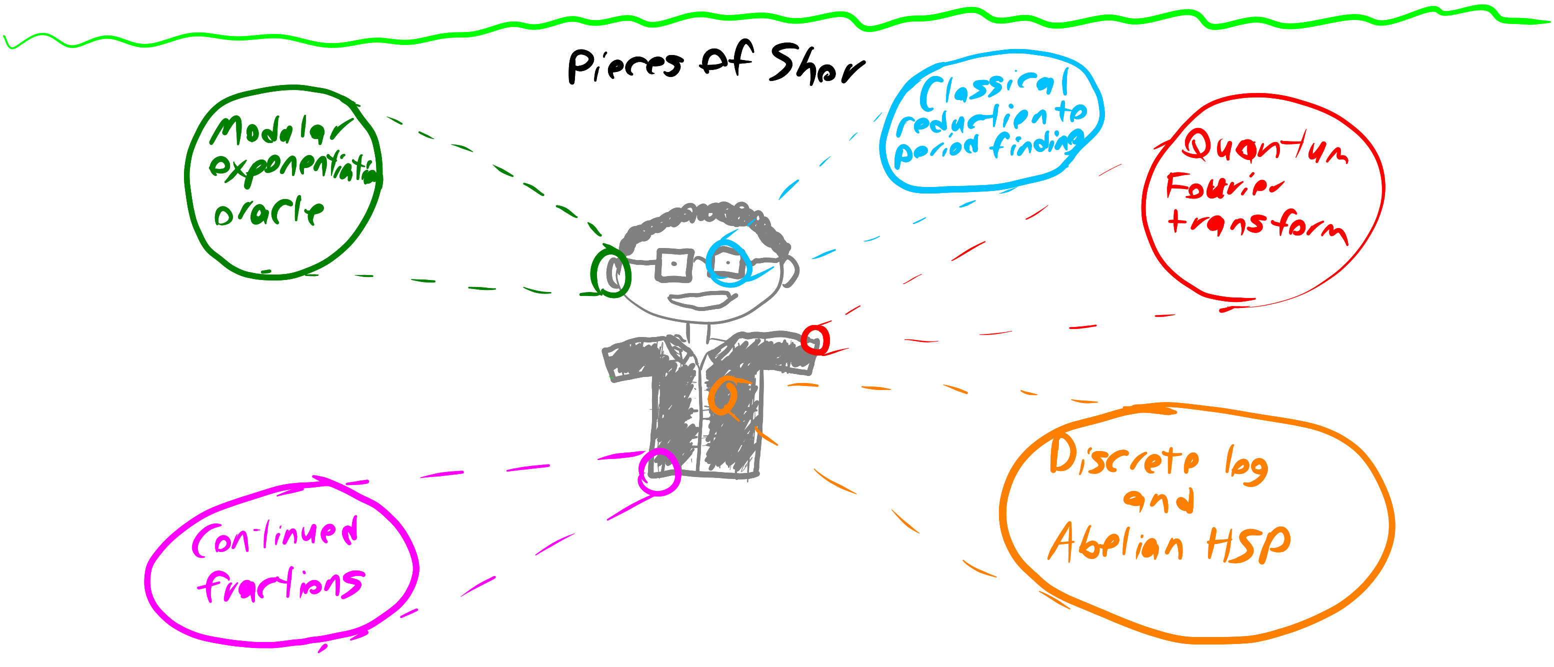
CMPT 476 Lecture 19

Shor's algorithm

(Not actually Shor, probably)

(Actually I met Shor in an elevator once. True story)

Last class we learned about Simon's algorithm which, while offering an exponential speed-up over any classical (Black box) algorithm, was mostly ignored because of the black box model and relatively pointless problem. However, it served as the inspiration for the perhaps most celebrated and famous quantum algorithm — Peter Shor's 1994 algorithm for factorizing integers in polynomial time. Today we start working up to Shor's algorithm, which involves many different pieces. Our first piece is the relationship to Simon's algorithm and the classical reduction of integer factorization to period finding.



(Periodic functions)

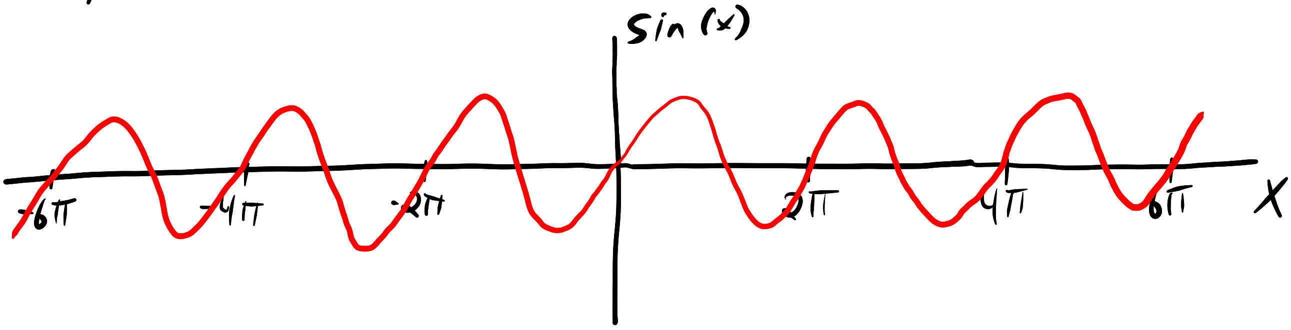
A function $f: A \to B$ is periodic if

$$f(x) = f(x+r) \quad \forall x \in A$$

for some $r \in A$ called the period of f.

Ex.

A classic periodic function is $\sin: \mathbb{R} \to \mathbb{R}$ with period 2π. As a function from $\mathbb{R}$ to $\mathbb{R}$, it's easy to visualize the period via a graph:

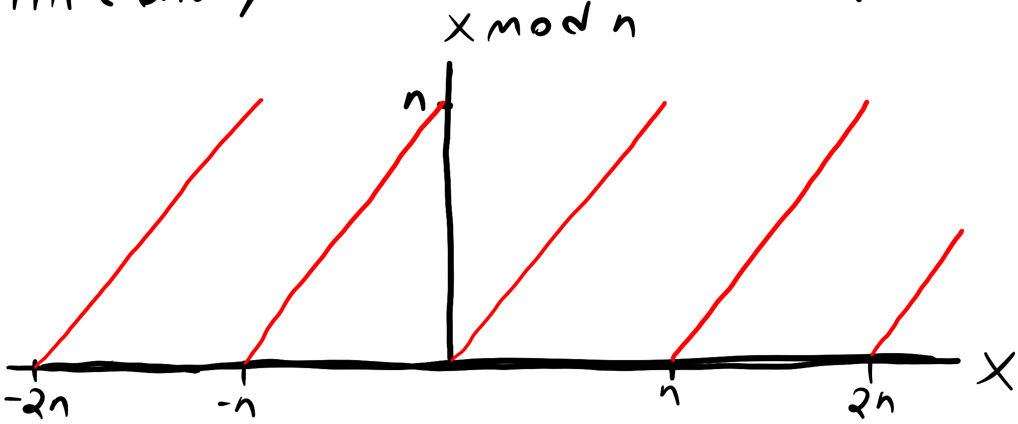


Another periodic function is $f: \mathbb{R} \to \mathbb{R}$ where

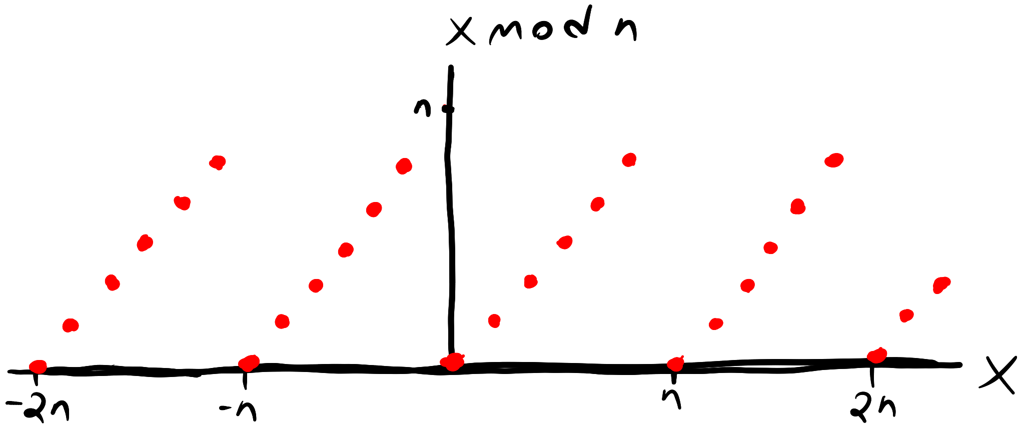
$$f(x) = x \bmod n$$

Recall that $x \equiv y \bmod n$ if $x = y + kn$ for $k \in \mathbb{Z}$. When we say $f(x) = x \bmod n$, we mean reduce $x \bmod n$ to $y \in [0, n)$

The graph of f is discontinuous unlike sin, but still conveys the intuition of repetition



If we restrict f to $f: \mathbb{Z} \to \mathbb{Z}$, we still have a periodic function, but it's less obvious visually



Recall that $\mathbb{Z}_2 = \{0,1\}$ are the integers mod 2 and $x \oplus y = x + y \bmod 2$ for $x, y \in \mathbb{Z}_2$. We can also add vectors over $\mathbb{Z}_2^n = \{0,1\}^n$ in the obvious way:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \oplus \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \oplus y_1 \\ x_2 \oplus y_2 \\ x_3 \oplus y_3 \end{bmatrix}$$

We say that $(\mathbb{Z}_2^n, \oplus)$ — the set $\mathbb{Z}_2^n$ equipped with the binary operator $\oplus : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ — is a group because

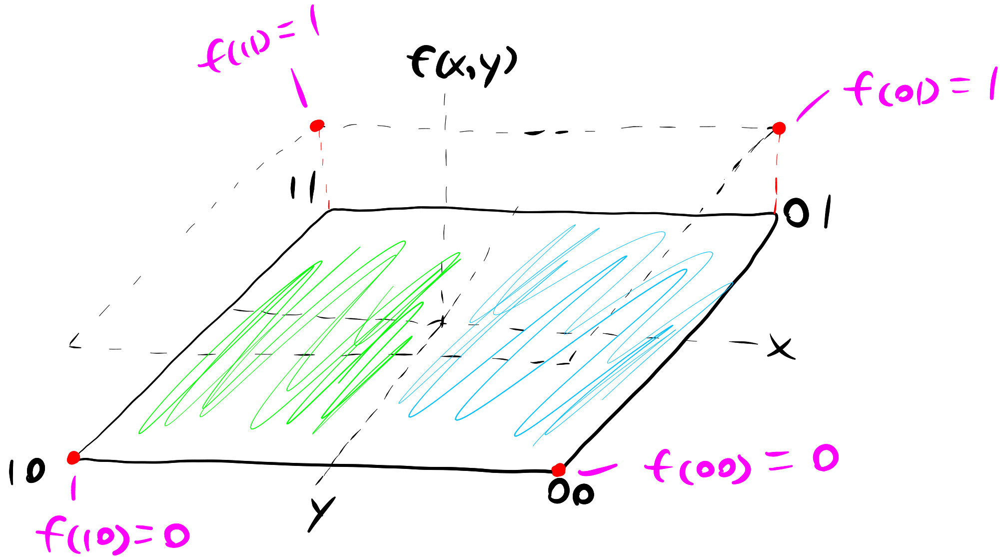
1. $\oplus$ is associative ——— $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
2. $0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ is neutral ——— $a \oplus 0 = a = 0 \oplus a$
3. Each a has an inverse $a^{-1} = a$ — $a \oplus a^{-1} = 0 = a^{-1} \oplus a$

What is a periodic function on $\mathbb{Z}_2^n$?

$\longrightarrow$ $f(x) = f(x \oplus s)$ where $s \in \mathbb{Z}_2^n$

This should be familiar: Simon's problem was to find s (the period) given f as above. So Simon's algorithm can be rephrased as period finding over $(\mathbb{Z}_2^n, \oplus)$!

Note that we mostly lose the intuitive visual notion of a periodic function here, except in simple cases like $f : \{0,1\}^2 \to \{0,1\}$ with period $s = 10$



period $s = 10$ → blue half-plane is repeated in the green half-plane

Now, $\oplus$ isn't the only addition we can define on $\mathbb{Z}_2^n$ — we can also interpret $a \in \mathbb{Z}_2^n$ as an n-bit integer $[a] \in \{0, 1, \ldots, 2^n - 1\} = \mathbb{Z}_{2^n}$ and define $+$ on $\mathbb{Z}_2^n$ as integer addition mod 2^n!

$$[a+b] = [a] + [b] \bmod 2^n$$

The resulting structure $(\mathbb{Z}_2^n, +) \simeq (\mathbb{Z}_{2^n}, +)$ is also a group.

Simon's algorithm consisted of two components:

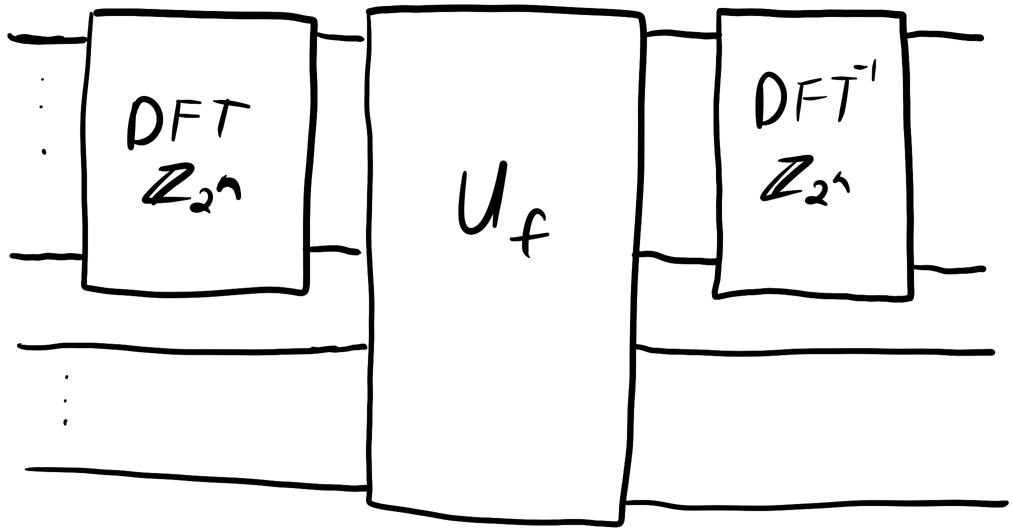
- An oracle for a periodic (on $(\mathbb{Z}_2^n, \oplus)$) function, and
- The discrete Fourier transform (DFT) on $(\mathbb{Z}_2^n, \oplus)$

$\underbrace{\text{discrete Fourier transform}}_{\text{THIS IS THE HADAMARD GATE } H^{\otimes n}}$

Shor took this idea and asked:

Can quantum computers be used to compute the period of a function $f: \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^n}$ (i.e. $\mathbb{Z}_2^n$ but with integer addition) instead?

The basic idea is to replace the DFT on $\mathbb{Z}_2$ with the DFT on $\mathbb{Z}_{2^n}$ and now we get the period of a function $f: \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^n}$!

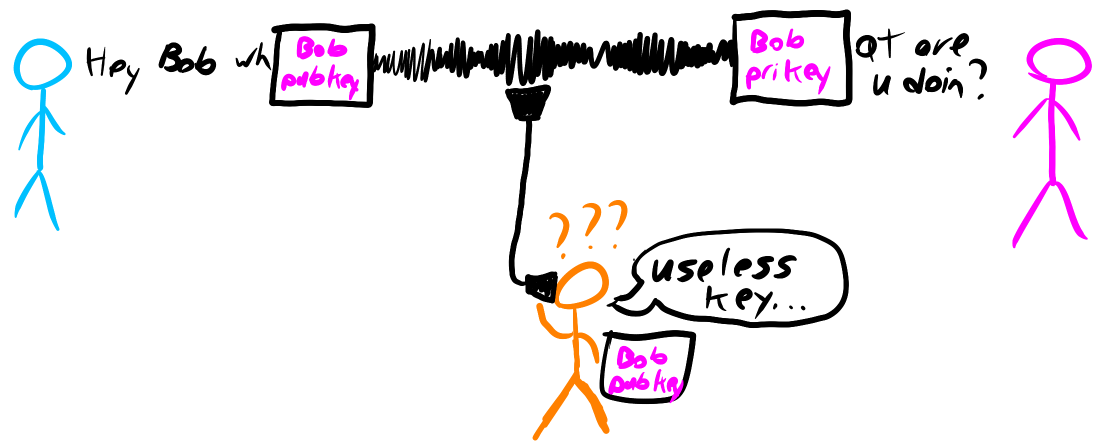


Easy, right?

First, let's back up and think about why we (or Peter Shor) would want to do such a thing...

(Integer factorization & RSA)

The internet is built on secure communication — you wouldn't want someone else to gain access to your DM's, Amazon purchase history, or CC information. To keep this information secret, it's usually encrypted so that if a third party gains access, they can't read your secrets. Public key cryptography allows others to send you secure communications asymmetrically by using your public key to encrypt the message, which can then only be decrypted with your private key which you (figuratively) keep under your mattress for safety.



Public key cryptosystems are built around problems which are easy to compute, but hard to invert, called a one-way function. RSA (Rivest, Shamir, Adleman) is one of the most widely used pubkey systems, based on the assumption that for two large primes p, q, computing $M = p \cdot q$ is easy, but factoring M into p and q is hard. In particular, Alice picks p & q, then publishes $M = p \cdot q$ and $e \in \mathbb{Z}$ as her public key, keeping p, q, and $d \in \mathbb{Z}$ such that

$$(A^e)^d \equiv A \bmod M, \quad A \in \mathbb{Z}$$

private. Bob can then send a message $A \in \mathbb{Z}$ to Alice like so, which requires p, q, or d to decrypt.

$$\text{encrypt}: A \longrightarrow A^e \bmod M$$
$$\text{decrypt}: A^e \bmod M \longrightarrow (A^e)^d \bmod M = A$$

The popular ssh protocol uses (among other pubkey systems) RSA to establish secure communication — so by breaking RSA I could steal your schoolwork

By the way, ssh until recently mostly used crypto based on the assumed hardness of computing Discrete logarithms — $\log_b a$ in some group G — like Diffie-Hellman, which Shor's algorithm also breaks!

Peter Shor knew a secret (not actually secret):

An efficient algorithm for period finding (over $\mathbb{Z}_2^n$) would allow factorizing integers or computing (certain) discrete logs efficiently

Before we tackle period finding, let's see how it can help factorize integers and break RSA.

(Integer factorization)

The problem of integer factorization is easy to state with only basic mathematics, but tackling it will require some basic group theory and number theory.

__Integer factorization problem__

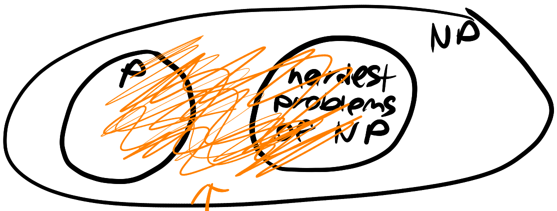
Input: a positive integer N represented in binary

Goal: Compute the unique prime factorization

$$N = p_1^{r_1} p_2^{r_2} \cdots p_k^{r_k}$$

where each p_i is prime and $r_i \in \mathbb{Z}^+$ ← positive integers

First, what is the classical (including probabilistic) complexity of integer factorization? As of now, it is not known whether IF (integer factorization) is in P or NP-hard.



possible locations of IF

Let $n = \log_2 N$ be the number of bits in N. The current best-known algorithm for IF — the Number field sieve — runs in time $\approx O(2^{\sqrt[3]{n}})$

So how does period finding relate?

The key lies in Euler's theorem

(Euler's theorem)

Let a and N be coprime. Then

$$a^{\varphi(N)} \equiv 1 \mod N$$

where $\varphi(N)$ is the number of numbers in $[0, N-1]$ coprime to N.

Now, given a number N which we want to factor, the function

$$f(x) = a^x \mod N \quad (x \text{ an integer})$$

is periodic if a is coprime to N, since

$$f(x + \varphi(N)) = a^x a^{\varphi(N)} \equiv a^x \mod N$$

So knowing the period r of f tells us something about N, namely that

$$a^r \equiv 1 \mod N$$

Now, observe that $a^r - 1 \equiv 0 \mod N$, or $a^r - 1 = kN$, $k \in \mathbb{Z}$

Then $(\sqrt{a^r} + 1)(\sqrt{a^r} - 1) = kN$ which almost makes progress towards factorizing N, but we have a few problems:

1. If r is odd then $\sqrt{a^r}$ is not an integer
2. If $\sqrt{a^r} + 1$ or $\sqrt{a^r} - 1$ is a multiple of N, we factorized k, not N
3. If neither of the above cases hold, then one of $(\sqrt{a^r} + 1)$ or $(\sqrt{a^r} - 1)$ is only gauranteed to share a non-trivial factor with N

Let's ignore 1 & 2 for now and focus on 3. Given two integers a & b that share a factor i.e.

$$a = MN, \quad b = LN$$

how do we find N? That's the greatest common divisor of a & b, $GCD(a,b)$!

(Euclid's algorithm)

Let $a > b$ be integers. Then

$$GCD(a,b) = GCD(b,r)$$

where r is the remainder of a/b, i.e.

$$a = qb + r, \text{ where } r < b$$

Euclid's algorithm finds the GCD in time $O(n^2)$ for n-bit numbers.

So

(1) If we know some a which is coprime to N,

(2) and the period r (or order) of $a \bmod N$ is even ($r = 2s$)

(3) and neither $a^s + 1$ nor $a^s - 1$ is a multiple of N,

then we can (in poly-time) compute a non-trivial factor M such that $N = MK$.

Setting aside the assumptions, if we repeat with M and K, how many of these factorizations do we need to get to the level of primes? $O(n)$!

$$N = p_1 p_2 \cdots p_k \quad k \in O(n)$$

$$M = p_{i_1} p_{i_2} \cdots p_{i_\ell} \qquad K = p_{j_1} p_{j_2} \cdots p_{j_p}$$

$$\vdots \qquad\qquad \vdots$$

$$p_1 \quad p_2 \quad p_3 \quad \cdots \quad p_{k-2} \quad p_{k-1} \quad p_k$$

at most k splits

We've shown that we can factorize in poly-time if our assumptions hold, but they're **Big** assumptions.

For complicated number theory reasons, If N is

(4) • odd, and

(5) • not a prime power, i.e. $N = p^n$ for prime p,

then for any a coprime to N, the probability (2) and (3) are satisfied is at least $\frac{1}{2}$.

Now to deal with (4), keep dividing N (at most $O(n)$ times) by 2 until it's odd. To deal with (5), it suffices to observe that prime powers can be factored in polynomial time (Assignment question).

Our final (non-quantum) piece is to find multiple integers a coprime with N. Note that a & N are coprime if and only if $GCD(a, N) = 1$. So, an algorithm to find or factor N is:

pick a random number $a \in [0, N-1]$.
If $GCD(a, N) = 1$, then return a,
else return $GCD(a, N)$ which is a non-trivial factor of N.

That was complicated. Let's recap.

(Classical algorithm reducing IF to period finding)