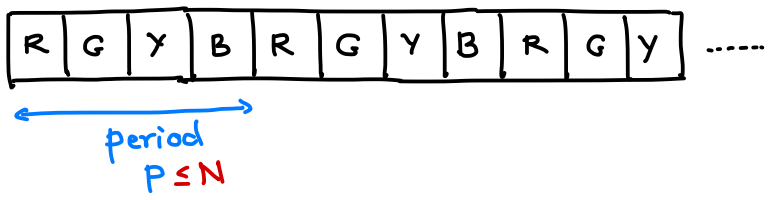
LECTURE 14 March 6th, 2025

PART II Fundamental Quantum Algorithms

Today RSA & Shor's Factoring Algorithm

RECAP Period-finding over integers

$f: \mathbb{Z} \to$ COLORS



Goal: Given query access to f, compute the period p

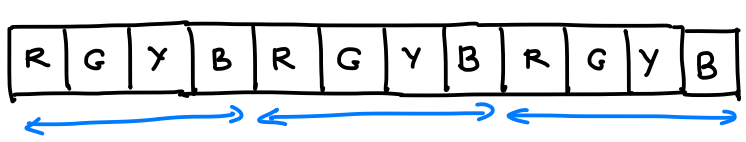
Last time Truncate the list to $Q = N^2$ elements

Then, we saw a quantum subroutine that gives us "clues" about the period

Key insight behind this was that Quantum Fourier Transform can extract "clues" about the period

How did this work?

- if the function mod Q was exactly periodic : p divides Q



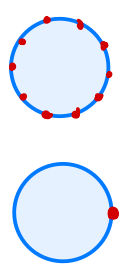
then the quantum subroutine outputs a random multiple of $\frac{Q}{P}$:= R (integer) — Also an integer

E.g. 5R, 10R, 20R, 3R, ...
35, 49, ...

This list and Q is known to the algorithm but not p
But if we take GCD of all these numbers we can figure out R and hence p with high probability

Here (when $\frac{Q}{P}$ is an integer)

$$\frac{1}{\sqrt{KQ}} \sum_{b=0}^{Q-1} \omega_Q^{bx} \underbrace{\left(\sum_{j=0}^{K-1} \omega_Q^{bjP} \right)}_{= \begin{cases} 0 & \text{if } b \neq \text{multiple of } \frac{Q}{P} \\ \text{OR} & \\ k & \text{if } b = \text{multiple of } \frac{Q}{P} \end{cases}} |b\rangle$$



- if the function is almost-periodic: p does not divide Q

Then the last piece may not be complete

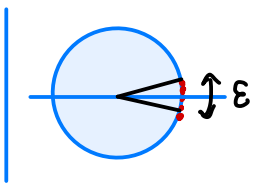
But the length of each piece is $p \leq N \leq \sqrt{Q}$, so this last piece is much smaller than the length of the array

Now, we will mostly see constructive interference if $k = \text{nearest-integer}\left(\text{multiple of } \frac{Q}{P}\right)$ (integer)

(when $\frac{Q}{P}$ is not an integer)

and destructive interference if $k \neq \text{nearest-integer}\left(\text{multiple of } \frac{Q}{P}\right)$

Basically, constructive interference occurs because:



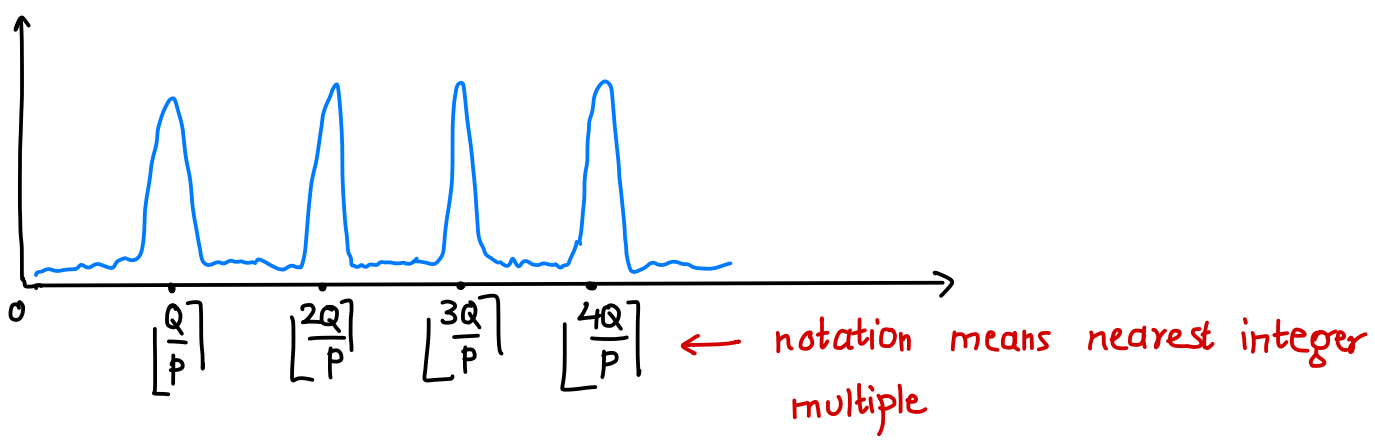
we sum over complex values $e^{i2\pi\varepsilon}$ where

$\varepsilon \approx 0$ so the values are close to 1

destructive interference occurs because again the values almost cancel out

$$\frac{1}{\sqrt{KQ}} \sum_{b=0}^{Q-1} \omega_Q^{bx} \underbrace{\left(\sum_{j=0}^{K-1} \omega_Q^{bjP} \right)}_{:=\alpha_b} |b\rangle$$

If we plot $|\alpha_b|$ it now looks like (this is what matters for measurement)



If we measure, with high probability we will output an integer $b_1 = \lfloor \ell_1 \frac{Q}{P} \rceil$

Final thing that remains to do: if we get $b_1 = \lfloor \ell_1 \frac{Q}{P} \rceil$, $b_2 = \lfloor \ell_2 \frac{Q}{P} \rceil$, $b_3 = \lfloor \ell_3 \frac{Q}{P} \rceil$

E.g. $b_1 = \ell_1 \frac{Q}{P} + \frac{1}{100}$, $b_2 = \ell_2 \frac{Q}{P} - \frac{1}{10}$

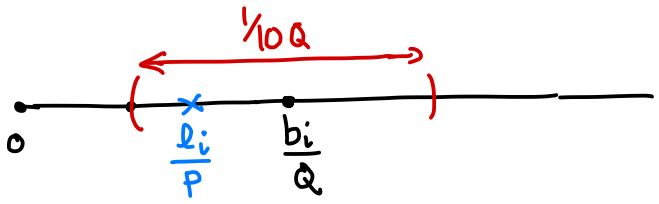
how do we find p?

Let us divide everything by Q & assume that the algorithm outputs rational numbers

Both known to algorithm → $\frac{b_1}{Q}, \frac{b_2}{Q}, \ldots \frac{b_T}{Q}$

Then, $\frac{b_i}{Q} = \frac{l_i}{P} \pm \frac{\varepsilon}{Q}$ where $|\varepsilon| \le \frac{1}{100}$ $\Rightarrow$ $\frac{l_i}{P}$ is close to the rational output of the algorithm

Pictorially



But there are infinitely many such rationals ! How do we find the one we are looking for ?

Note The rational we are looking for has a small denominator $p \le \sqrt{Q}$

How many such rationals are there ? Just one !

Claim Any two fractions with denominator $\le \frac{1}{\sqrt{Q}}$ must be at least $\frac{1}{Q}$ apart

Why ? $\left|\frac{l_1}{P_1} - \frac{l_2}{P_2}\right| = \frac{|l_1P_2 - l_2P_1|}{|P_1P_2|} \ge \frac{1}{Q}$

So, $\frac{l}{P}$ is the unique fraction with denominator $\le Q$ that is $\frac{1}{100Q}$ close to the known ratio $\frac{b}{Q}$

This can be found using a classical method called 'continued fractions'

We will explain continued fractions with an example

E.g. $0.25001 = \frac{25001}{100000} = \frac{1}{\frac{100000}{25001}} = \frac{1}{3 + \frac{24997}{25001}}$

$= \frac{1}{3 + \frac{1}{\frac{25001}{24997}}} = \frac{1}{3 + \frac{1}{1 + \frac{4}{24997}}} \approx \frac{1}{3+1} = \frac{1}{4}$

This converges very quickly to the correct rational and we can find $\frac{l_i}{P_i}$

But we still don't know whether the actual ratio came from $\frac{l_i}{P_i}$ or $\frac{2l_i}{2P_i}$ or $\frac{4l_i}{4P_i}$ ...

One can solve this by a similar trick we saw before

In particular, it turns out that :

The least common multiple of p_i's is the right value with high probability ■

Now, on to the main topic : How do we use period finding to factor integers ?

First, let us start with some motivation about why we want to factor large integers

The motivation is the RSA Cryptosystem

RSA Cryptosystem

Another widely-deployed cryptosystem is Diffie-Hellman invented in 1978

This was invented by Rivest, Shamir & Adleman in 1977

Widely used in practice and enables public-key encryption, digital signatures, ...

How does it work ?

Suppose you want to send a secret message (such as a credit card number) to Amazon

Now you and Amazon haven't agreed to a secret key, so how can you do it so that no adversary can decode your message but Amazon can

- Amazon generates
  - two large prime numbers p & q, set $N = pq$
  - random prime number $e \in \{1, \ldots, (p-1)(q-1)\}$
  - computes an integer d such that $de = 1 \bmod (p-1)(q-1)$
- Amazon publishes public key $= (e, N)$ for everyone to see
  and keeps secret key $= d$ hidden
- Now, if you want to send a message $x \in \{1, \ldots, N-1\}$ to Amazon

  your browser sends $m = x^e \bmod N$ (e, N) is public

- To decode, Amazon computes $m' = (x^e)^d \bmod N$
$= x^{de} \bmod N$
$= x \bmod N$ by Fermat's Little Theorem

Now Amazon knows your credit card number x

- Why can't an adversary decode m as well?

A: They don't have d!

But if they could factor $N = p \cdot q$ they could compute it and break cryptosystems

This is why factoring is such an important problem

Factoring Given $N = p \cdot q$ where p and q are primes ← if we can solve this case, we can also factor other numbers as well

find p, q

Trivial Algorithm: check all numbers $1, 2, \ldots \sqrt{N}$ to see if they divide N

time $= \sqrt{N} = 2^{\frac{\log N}{2}}$ where $\log N$ = # bits in N

If $N = 1024$, this is 2^{256} which would take billions of years

Sieve-based Algorithm takes time $2^{(\log N)^{1/3}}$ which is still impractical unless you have huge amount of resources

But if we have 2048-bit integers, everything we have is impractical even after 50 years of efforts!

Shor's Factoring Algorithm ← One of the most important developments in quantum computing

Invented by Peter Shor in 1993 taking inspiration from Simon's Algorithm

KEY IDEA Reduce to period-finding / order-finding over integers to get "clues" about factors

Use classical post-processing to extract factors

Key point to remember is that in period-finding, we are given query access to a function but we can implement this query access very efficiently for this problem of order finding

Reducing Factoring to Order Finding

Input : N

- Pick a random number $a \in \{1, 2, \ldots N-1\}$
  - If $GCD(a,N) \neq 1 \Rightarrow$ we have found a factor (although this is very unlikely)
  - If $GCD(a,N) = 1 \Rightarrow$ compute the order of x mod N, call it by invoking the order-finding subroutine

Order Finding Find smallest integer r s.t. $a^r = 1 \bmod N$

Basically, the function $f(x) = a^x \bmod N$ is periodic

$$1 = a^0 \bmod N$$
$$\ldots = a^1 \bmod N$$
$$\vdots$$
$$\ldots = a^r \bmod N$$
$$1 = a^{r+1} \bmod N$$

- If r is odd, we repeat the above
- Otherwise, r is even & $x^r - 1 = 0 \bmod N$

$$\Leftrightarrow \underbrace{(x^{r/2}+1)}_{=a}\underbrace{(x^{r/2}-1)}_{=b} = 0 \bmod N$$

If $x^{r/2}+1$ & $x^{r/2}-1$ are multiples of N, repeat again

Otherwise, we get lucky since $ab = h \cdot N = hpq$

Since a, b are not multiples of N, computing $GCD(a,N)$ or $GCD(b,N)$ will give a nontrivial divisor of N

Needs $O(\log N)^2$ gates

How far away is Shor's Algorithm?

Practically, for factoring 2048-bit number, we need 4096 ideal qubits which needs 20 million noisy qubits with overhead for error correction

Total time $\cong$ 8 hours

IBM / Google / etc. have a roadmap to 1 million qubits by 2030

What will replace RSA?

Diffie-Hellman? Also, broken by Shor's algorithm
which can be generalized to any abelian group

NIST (National Institute for Standards & Technology) just concluded a multi-year competition to find a post-quantum cryptosystem to replace RSA

↑ you don't have a quantum computer but your adversary does

The winners are ....

1. Crystals - KUBER → For encryption
2. Crystals - Dilithium
3. FALCON
4. SPHINCS+

→ For digital signatures

First three are based on lattices, where the goal is to find short / close vectors in a high dimensional lattice

It is believed that short/close vector problems are hard even for quantum computers

The evidence for this is not conclusive & it will take time to build confidence in these new cryptosystems

In fact, SPHINCS+, which was based on elliptic curves is already broken by classical computers!