LECTURE 14 October 5^{th}, 2023

PART II Fundamental Quantum Algorithms

Today RSA & Shor's Factoring Algorithm

RECAP Period-finding over integers

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

R	G	Y	B	R	G	Y	B	R	G	Y	......

period
$p \le N$

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

R	G	Y	B	R	G	Y	B	R	G	Y	B

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

E.g. $5R,\ 10R,\ 20R,\ 3R, \ldots$
$35, 49, \ldots$

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

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

R	G	Y	B	R	G	Y	B	R	G	Y

Then the last piece may not be complete
But the length of each piece is $p \le N \le \sqrt{Q}$, so this last piece is much smaller than the length of the array

Because of this errors can be handled and the subroutine gives us

$\lfloor lR \rceil$ where l is random and $R = \frac{Q}{p}$

$\lfloor x \rceil \equiv$ Nearest integer to a real number x

If we run this many times

we get $b_1 = \lfloor \frac{l_1 Q}{p} \rceil$, $b_2 = \lfloor \frac{l_2 Q}{p} \rceil$, $b_3 = \lfloor \frac{l_3 Q}{p} \rceil$

E.g. $b_1 = \frac{l_1 Q}{p} + \frac{1}{100}$, $b_2 = l_2 \frac{Q}{p} - \frac{1}{10}$

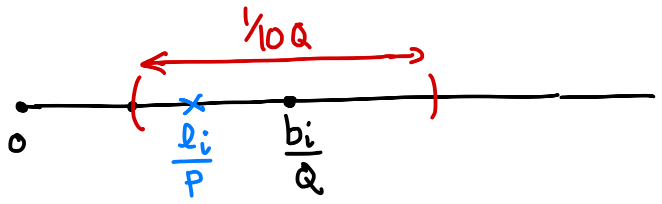
How do we find p?

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

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

Then, $\frac{b_i}{Q} = \frac{l_i}{p} \pm \frac{\varepsilon}{Q}$ where $|\varepsilon| \leq \frac{1}{100}$ $\Rightarrow$ $\frac{l_i}{p}$ 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 $p \leq \sqrt{Q}$

How many such rationals are there? Just one!

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

Why? $\left| \frac{l_1}{p_1} - \frac{l_2}{p_2} \right| = \frac{|l_1 p_2 - l_2 p_1|}{|p_1 p_2|} \geq \frac{1}{Q}$

So, $\frac{l}{p}$ is the unique fraction with denominator $\leq Q$ 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. $0.25001 = \frac{25001}{100000} = \cfrac{1}{\frac{100000}{25001}} = \cfrac{1}{3 + \frac{24997}{25001}}$

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

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 perioding 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
$$m' = (x^e)^d \bmod N$$
$$= x^{de} \bmod N$$
$$= x \bmod N \quad \text{by Fermat's Little Theorem}$$

  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, find p, q ← if we can solve this case, we can also factor other numbers as well

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

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

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

$\boxed{\text{KEY IDEA}}$ 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 $a \in \{1, 2, \dots, N-1\}$
  - 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 $x \bmod N$, 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

$$
\begin{aligned}
1 &= a^0 \bmod N \\
\dots &= a^1 \bmod N \\
&\vdots \\
\dots &= a^r \bmod N \\
1 &= a^{r+1} \bmod N
\end{aligned}
$$

- 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 +

(2–4) → 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!

$\boxed{\text{NEXT WEEK}}$ Projects

& when we resume, Quantum search with Grover's Algorithm