Shor's Algorithm

Quantum Computer Programming

Davide Carnemolla

Department of Mathematics and Computer Science

University of Catania

2022/2023

Integer Factorization

The problem

Definition

Given an integer N, the goal of the *integer factorization problem* is to find k primes $p_1^{e_1}, p_2^{e_2}, \ldots, p_k^{e_k}$ such that $N = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ with $e_i \geq 1 \quad \forall i \in \{1, \ldots, k\}$.

Example

Given $N = 143$ we know that the solution is the pair $(13, 11)$.

RSA

$(pk, sk) \leftarrow$ **keygen()**: given two primes p and q
let $n = pq$
choose e, d such that $e \cdot d \equiv 1 \bmod \phi(n)$
return $pk = (n, e)$, $sk = (n, d)$

$c \leftarrow \textbf{enc}_{pk}(m) = m^e \; mod \; n$

$m \leftarrow \textbf{dec}_{sk}(c) = c^d \; mod \; n$

RSA's Security

The RSA's Problem can be reduced to the Integer Factorization's Problem.

Solution in the classical model

Brute Force strategy

The brute force algorithm goes through all primes p up to $\sqrt{N}$ and checks whether p divides N.

Complexity

In the worst case, this would take time roughly , which is exponential in the number of digits $d = \log_2 N$.

Quadratic Sieve Algorithm

A more efficient algorithm, known as the quadratic sieve, attempts to construct integers a, b such that $a^2 - b^2$ is a multiple of N. Once such a, b are found, one checks whether $a \pm b$ have common factors with N.

Complexity

The quadratic sieve method has asymptotic runtime exponential in $\sqrt{d}$, where $d = \log_2 N$ is the number of digits of N.

GNFS

The General Number Field Sieve is the most efficient classical factoring algorithm. The main idea is the use of smooth numbers.

GNFS - Complexity

The number of digit of N is equals to $d = \log_2 N$. The algorithm's complexity can be simplified as follows

$$\mathcal{O}(exp(const \times d^{1/3}))$$

The Period Finding Problem

Definition of the problem

Given integers N and a, find the smallest positive integer r such that

$$a^r \equiv 1 \; mod \; N \iff N \mid a^r - 1$$

The number r is called the period of a modulo N.

Example

Suppose $N = 15$ and $a = 7$ then

$$7^2 \equiv 4 \; mod \; 15, \quad 7^3 \equiv 13 \; mod \; 15, \quad 7^4 \equiv 1 \; mod \; 15$$

That is, 7 has period 4 modulo 15.

$(N, a) \quad \rightarrow$  $\rightarrow \quad r$

period-finding machine

where r is the period of a modulo N.

From factoring to period finding

Assumption: N has only two distinct prime factors, $N = pq$.

Experiment

1. $a \xleftarrow{\$} [2, N-1]$ such that $gcd(N, a) = 1$
2. $r \leftarrow$ **period-finding-machine**(N, a)
3. go to 1 until r is even

Note

It can be shown that a significant fraction of all integers a have an even period, so on average one needs only a few repetitions.

Table of iterations

Suppose $N = 15$.

a	r	$gcd(15, a^{r/2} - 1)$	$gcd(15, a^{r/2} + 1)$
1	1		
2	4	3	5
4	2	3	5
7	4	3	5
8	4	3	5
11	2	5	3
13	4	3	5
14	2	1	15

We have found some pair (r, a) such that

1. r is even
2. r is the smallest integer such that $a^r - 1$ is a multiple of N

Also, we know that

$$a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$$

The above shows that $a^{r/2} - 1$ is not a multiple of N, otherwise the period of a would be $r/2$.

Assumption: $a^{r/2} + 1$ is not a multiple of N.

$$\implies a^{r/2} \pm 1 \text{ is not a multiple of } N\text{, but their product is.}$$

This is possible only if

$$p \text{ is a prime factor of } a^{r/2} - 1 \wedge q \text{ is a prime factor of } a^{r/2} + 1$$
(or vice versa)

We can thus find find p, q by computing

$$gcd(N, a^{r/2} \pm 1)$$

What we found: the unlucky case

When

$$a^{r/2} + 1 \text{ is a multiple of } N$$

we are in the unlucky case

$$\implies \text{ we give up and try a diffrent integer } a.$$

Fact

It can be shown that the unlucky integers a are not too frequent, so on average, only two calls to the period-finding machine are sufficient to factor N.

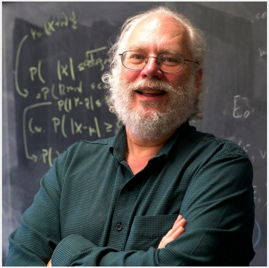
Table of iterations

Suppose $N = 15$.

a	r	$gcd(15, a^{r/2} - 1)$	$gcd(15, a^{r/2} + 1)$
1	1		
2	4	3	5
4	2	3	5
7	4	3	5
8	4	3	5
11	2	5	3
13	4	3	5
14	2	1	15

Shor's Algorithm

Shor's Algorithm is a quantum computer algorithm to solve the period-finding's problem.



It was developed in 1994 by the American mathematician Peter Shor.

Algorithm

1. $a \xleftarrow{\$} [2, N-1]$
2. Compute $K = gcd(N, a)$
3. **if** $K = 1$ **then**
4. $\quad r \leftarrow$ quantum-period-finding-subroutine(N, a)
5. $\quad$ **if** (r is odd $||$ $a^{r/2} \equiv -1 \ mod \ N$) **then**
   go to 1
6. $\quad$ **else**
   return $gcd(a^{r/2}+1, N), gcd(a^{r/2}-1, N)$
7. **else** go to 1

Where the gcd function is computed using the Euclidean Algorithm.

The main idea is to use the quantum phase estimation on the unitary operator

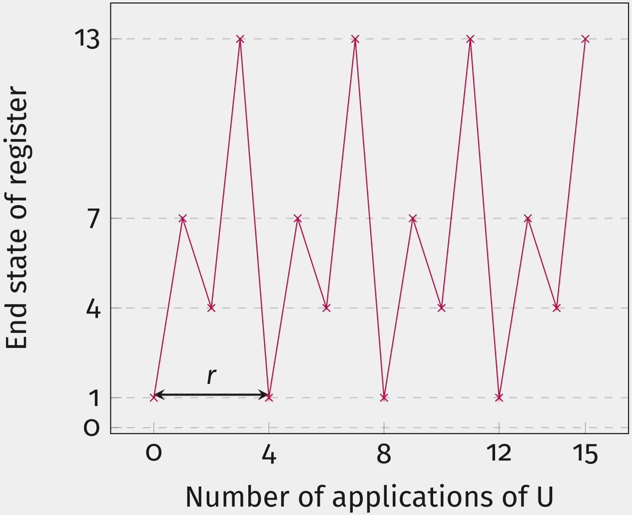
$$U|y\rangle \equiv |ay \bmod N\rangle$$

Example

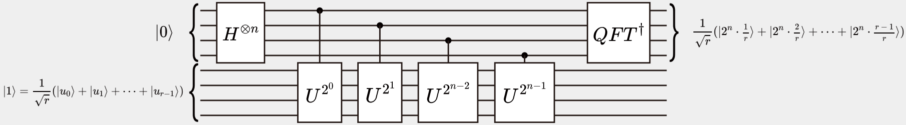
With $a = 7$ and $N = 15$

$$\begin{aligned} U|1\rangle &= |7\rangle \\ U^2|1\rangle &= |4\rangle \\ U^3|1\rangle &= |13\rangle \\ U^4|1\rangle &= |1\rangle \end{aligned}$$

Example



13
7
4
1
0
End state of register
r
0
4
8
12
15
Number of applications of U



$|0\rangle$
$|1\rangle = \frac{1}{\sqrt{r}}(|u_0\rangle + |u_1\rangle + \cdots + |u_{r-1}\rangle)$
$H^{\otimes n}$
U^{2^0}
U^{2^1}
$U^{2^{n-2}}$
$U^{2^{n-1}}$
$QFT^\dagger$
$\frac{1}{\sqrt{r}}(|2^n \cdot \frac{1}{r}\rangle + |2^n \cdot \frac{2}{r}\rangle + \cdots + |2^n \cdot \frac{r-1}{r}\rangle)$

So a superposition of the states in this cycle $|u_0\rangle$ would be an eigenstate of U:

$$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |a^k \ mod \ N\rangle$$

Example with $a = 7, N = 15$

$$\begin{aligned} |u_0\rangle &= \frac{1}{2}\left(|1\rangle + |7\rangle + |4\rangle + |13\rangle\right) \\ U|u_0\rangle &= \frac{1}{2}\left(U|1\rangle + U|7\rangle + U|4\rangle + U|13\rangle\right) \\ &= \frac{1}{2}\left(|7\rangle + |4\rangle + |13\rangle + |1\rangle\right) = |u_0\rangle \end{aligned}$$

Case u_1

A more interesting eigenstate could be one in which the phase is different for each of these computational basis states.
For instance,

$$|u_1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k}{r}} |a^k \; mod \; N\rangle$$

$$U|u_1\rangle = e^{\frac{2\pi i}{r}} |u_1$$

Example

Example $a = 7, N = 15$

$$|u_1\rangle = \frac{1}{2}\left(|1\rangle + e^{-\frac{2\pi i}{4}}|7 + e^{-\frac{4\pi i}{4}}|4\rangle + e^{-\frac{6\pi i}{4}}|13\rangle\right)$$

$$U|u_1\rangle = \frac{1}{2}\left(|7\rangle + e^{-\frac{2\pi i}{4}}|4 + e^{-\frac{4\pi i}{4}}|13\rangle + e^{-\frac{6\pi i}{4}}|1\rangle\right)$$

$$U|u_1\rangle = e^{\frac{2\pi i}{4}} \cdot \frac{1}{2}\left(|e^{-\frac{2\pi i}{4}}|7 + e^{-\frac{4\pi i}{4}}|4\rangle + e^{-\frac{6\pi i}{4}}|13\rangle + e^{-\frac{8\pi i}{4}}|1\rangle\right)$$

$$U|u_1\rangle = e^{\frac{2\pi i}{4}}|u_1\rangle$$

We can see that r appears in the denominator of the phase. This is interesting. Generalizing the idea, we obtain

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi isk}{r}} |a^k \; mod \; N\rangle$$

$$U|u_s\rangle = e^{\frac{2\pi is}{r}} |u_s\rangle$$

We now have a unique eigenstate for each integer value of s where

$$0 \leq s \leq r - 1$$

Fact

If we sum up all the eigenstates, the different phases cancel out all computational basis states except $|1\rangle$

$$\frac{1}{r}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle$$

If we apply the Quantum Phase Estimation on U using the state $|1\rangle$, we will measure a phase

$$\phi = \frac{s}{r}$$

where s is the random number between 0 and $r-1$.

Using the Quantum Phase Estimation we can find the phase ϕ but not r. We can find r using the Continued fractions algorithm.

Definition

Let $a_0, a_1, \ldots, a_k \in \mathbb{Z}$ such that $\forall i \in \{1, \ldots, k\} a_i \geq 0$. Then the expression

$$r = a_0 + \frac{1}{a_1 + \frac{1}{\cdots + \frac{1}{a_k}}}$$

is called the **continued fraction rapresentation** of the rational number r and is denoted shortly as $r = [a_0, a_1, \ldots, a_k]$

Continued Fractions Algorithm

Input: $\phi, e = 0.0001$
Output: s,r

1. $A \leftarrow [\lfloor\phi\rfloor]$
2. **while** $\phi - \lfloor\phi\rfloor > e$ **then**
   $\phi \leftarrow 1/(\phi - \lfloor\phi\rfloor)$
   append $\lfloor\phi\rfloor$ to A
3. $p \leftarrow [0, 1], q \leftarrow [1, 0]$
4. **for each** *it* in A **do**
   append $p[len(p)] * it + p[len(p) - 1]$ **to** p
   append $q[len(q)] * it + q[len(q) - 1]$ **to** q
5. **return** $p[len(p)], q[len(q)]$

Example

Example $r = \frac{5}{3}$

$$r = [1, 1, 2] = 1 + \frac{1}{1 + \frac{1}{1+\frac{1}{2}}}$$

$$P = [0, 1, 1, 2, 5] \implies p = 5 \qquad Q = [1, 0, 1, 1, 3] \implies q = 3$$

Complexity

The number of digit of N is equals to $d = \log_2 N$. The Shor's Algorithm complexity can be simplified as follows

$$\mathcal{O}(const \times d^3)$$