FFT and Shor's algorithm for factoring

Lecture Notes for Quantum Computing

Lecturer: Dorit Aharonov
Scribe: Matan Ninio and Eyal Rozenman

March 25-30, 2001

Abstract

1. We define the Fourier transform over Z_N - the integers modulo N. We give an algorithm which calculates the Fourier transform over Z_N for $N = 2^n$, using a circuit which is polynomial in n, i.e. polylogarithmic in N. 2. We describe Shor's algorithm for factoring an integer - a problem which is thought to be hard for classical computers.

1 Fourier Transform

1.1 Definition

Let $\omega = e^{\frac{2\pi}{N}i}$ be an N^{th} root of unity. Then the Fourier transform FT_N of a basis vector $|k\rangle$ is defined as:

$$\mathrm{FT}_N(|k\rangle) = F_{N\times N} \cdot |k\rangle$$

where $F_{l,m} = \frac{1}{\sqrt{N}}\omega^{lm}$. Thus, by linearity:

$$\sum_k \alpha_k |k\rangle \quad \stackrel{FT_N}{\longrightarrow} \quad \sum_k \sum_j \alpha_k \frac{\omega^{jk}}{\sqrt{N}} |j\rangle$$

Proposition 1 *The FT transform is unitary.*

Proof: This is a direct calculation - we need to check that the columns of $F_{N\times N}$ form an orthonormal basis. Let u_i be the i-th column of F. Then

$$(u_i, u_j) = 1/N \sum_{l=1}^{N} \omega^{il}\overline{\omega^{jl}} = 1/N \sum_l \omega^{il}\omega^{-jl} = 1/N \sum_l \omega^{l(i-j)} = 1/N \sum_l \tilde{\omega}^l$$

where $\tilde{\omega} = \omega^{i-j}$. This sum is 1 when $\tilde{\omega} = 1$ and 0 when $\tilde{\omega}$ is any N^{th} root of unity other that 1 (to see this just calculate the geometric sum) which finishes the proof.

1.2 Fast Fourier Transform

Let $N = 2^n$. The Fourier transform can be represented by multiplication with an $N \times N$ matrix F. Thus naively, we would imagine that it requires an $O(N^2)$ sized circuit to compute the Fourier transform (this is generally how long it takes to compute a matrix-vector multiplication with dimensions $N \times N$ and N, respectively). However, with the Fast Fourier Transform (FFT) we can do it with only an $O(N \log N)$ sized circuit. Note the equality (which holds for $i < N/2$)

$$\begin{aligned}
(F_{N\times N}v)_i &= \sum_{j=0}^{N/2-1} \omega^{i(2j)} v_{2j} + \sum_{j=0}^{N/2-1} \omega^{i(2j+1)} v_{2j+1} \\
&= \sum_{j=0}^{N/2-1} (\omega^2)^{ij} v_{2j} + \sum_{j=0}^{N/2-1} \omega^i (\omega^2)^{ij)} v_{2j+1} \\
&= (FT_{N/2\times N/2} \cdot v_{even})_i + \omega^i (FT_{N/2\times N/2} \cdot v_{odd})_i \\
(F_{N\times N}v)_{i+N/2} &= \sum_{j=0}^{N/2-1} \omega^{(i+N/2)(2j)} v_{2j} + \sum_{j=0}^{N/2-1} \omega^{(i+N/2)(2j+1)} v_{2j+1} \\
&= \sum_{j=0}^{N/2-1} \omega^{Nj} (\omega^2)^{ij} v_{2j} + \sum_{j=0}^{N/2-1} \omega^{Nj+N/2} (\omega^2)^{i(j+1)} v_{2j+1} \\
&= (FT_{N/2\times N/2} \cdot v_{even})_i - \omega^i (FT_{N/2\times N/2} \cdot v_{odd})_i
\end{aligned}$$

This gives us the idea to work recursively. In order to do so, we need to reorder the basis vectors. Instead of looking at $v = v_0, v_1, \ldots v_{N-1}$, let us reorder it in the form $v_0, v_2, v_4, \ldots v_{N-2}, v_1, v_3, \ldots v_{N-1}$. How does the matrix

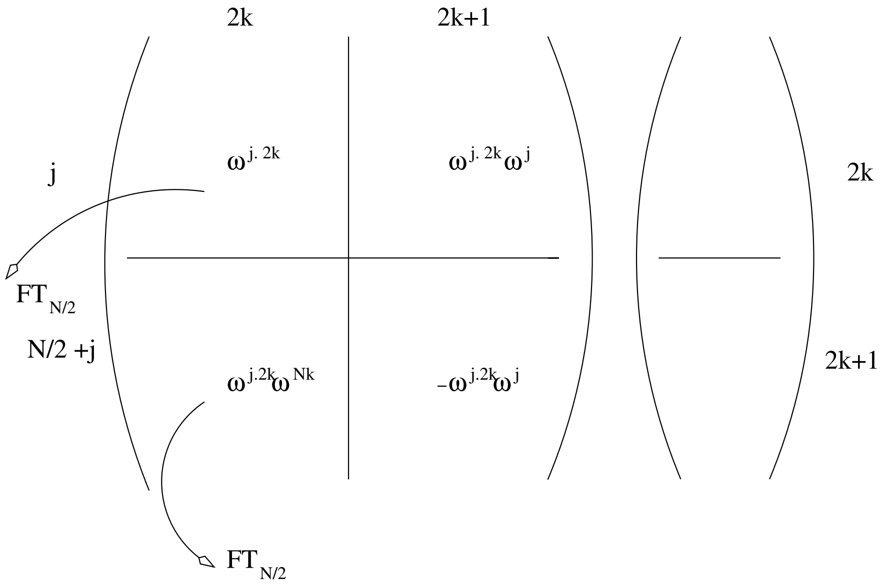


Figure 1: Splitting the Fourier Transform matrix

$F_{N\times N}$ look like in this basis? The reordering of the basis vectors changes F by reordering the columns of F, such that the even columns are now the first $N/2$ columns, and the odd ones are the last $N/2$ columns. Call the reordered matrix G, and split G to four sub-matrices where G_{11} will be the upper left quarter and G_{22} the lower right. Now we can calculate (see figure 1) that $(G_{11})_{ij} = (F_{N/2})_{ij}$, $(G_{21})_{ij} = (F_{N/2})_{ij}$, $(G_{12})_{ij} = \omega^j(F_{N/2})_{ij}$, and $(G_{22})_{ij} = -\omega^j(F_{N/2})_{ij}$. This representation instructs us how to calculate FT_N given a black box calculating $FT_{N/2}$ (see figure 2). If the size of the fast Fourier transform circuit is denoted by $S(N)$, then

$$S(N) \quad = \quad 2S(N/2) + O(N)$$

which gives $S(N) = O(N \log N)$.

1.3 Quantum Fourier transform circuit

We want to calculate the Fourier transform using a quantum circuit. Our input will now be a superposition $\sum_{k=0}^{N-1} \alpha_k |k\rangle$. This is a vector in $V^{\otimes n}$, where $n = \log_2 N$. So a quantum circuit calculating the Fourier transform

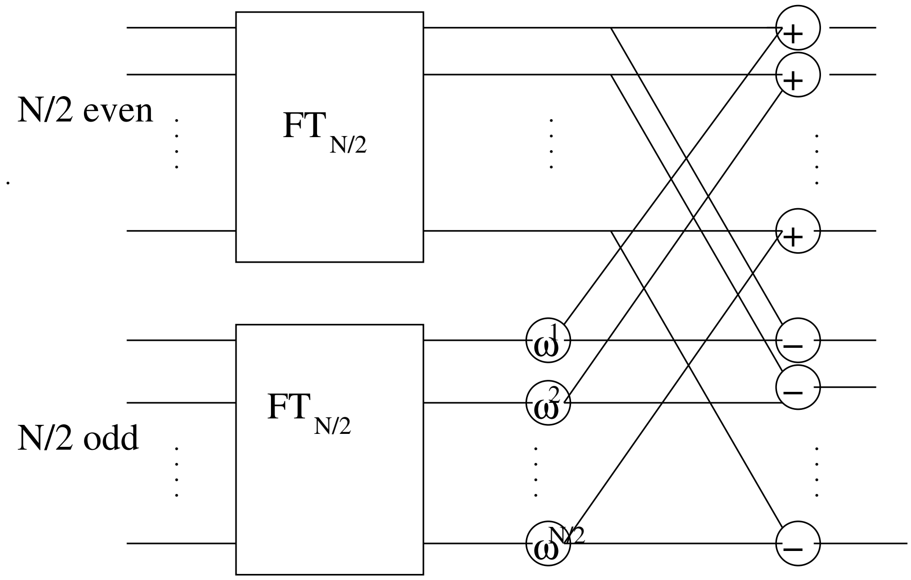


Figure 2: Fast Fourier transform circuit

needs only n input qubits, as opposed to N inputs needed by the classical circuits. We will derive a polynomial way to perform the Fourier transform over the superposition using polylog(N) gates, as opposed to the classical $Nlog(N)$, which is exponential in n. The price we pay is that the output of the circuit will again be a superposition $\sum_{k=0}^{N-1} \beta_k |k\rangle$; If we want to find the actual value of each Fourier coefficient, we will have to perform exponentially many measurements.

The circuit for QFFT will be a recursive one as well, using the same idea underlying the classical FFT. We apply a $QFFT_{n-1}$ to the $n-1$ most significant qubits. For a length-n string $k = k_1, k_1, \ldots k_n$, let $\hat{k}$ be the $n-1$ most significant bits of k. So our operation is: $|k\rangle \rightarrow \sum_{i=0}^{N/2-1} (FT_{N/2}|\hat{k}\rangle)_i \cdot |i\rangle \otimes |k_n\rangle$. To get the true value $(FT|k\rangle)_i$ we need to multiply $(FT_{N/2}|\hat{k}\rangle)_i|i\rangle$ by ω^i when $k_n = 1$. The trick is to do this bit by bit using a conditioned phase rotation. Let R_k be the phase shift :

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & w^{2^k} \end{pmatrix}$$

Then the following matrix will be controlled phase shift on two qubits, which rotates the second qubit only if the first one is 1 (we are using the basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$):

$$RC_k = \begin{pmatrix} I & 0 \\ 0 & R_k \end{pmatrix}$$

We take the output of our $QFFT_{N/2}$ and apply RC_i to qubit number i ($i < n$), controlled by the least significant bit (LSB) that is, k_n. We see that when $k_n = 1$ the result of $(QFFT_{N/2})_l$ is indeed multiplied by ω^l, since for every bit of l we applied the corresponding RC_k. It still remains to multiply the coefficients of $|j\rangle$ for $j \geq N/2$ by (-1) - again, only when $k_n = 1$. To achieve this we apply the Hadamard transform on the LSB. What happens when $k_n = 0$? Then $H(|0\rangle) = (|0\rangle + |1\rangle)/\sqrt{2}$. If we now interpret our LSB as the **MSB** of the output, we see that we get exactly what we need, since $FT_N(|k\rangle)_j = FT_N(|k\rangle)_{j+N/2} = FT_{N/2}(|\hat{k}\rangle)/2$, when $k_n = 0, j < N/2$. And the same when $k_n = 1$, where we get $H(|1\rangle) = (|0\rangle - |1\rangle)/\sqrt{2}$. In this case we already multiplied the j-th output by ω^j, and now the coefficients with $k_n = 1$ are also multiplied by (-1). Since we are using this bit as the MSB, then the values $(FT(|k\rangle)_j$ with $j \geq N/2$ are exactly those with $k_n = 1$, and exactly those are multiplied by (-1). So all in all we indeed satisfy the equation $FT_N(|k\rangle)_j = \omega^j FT_{N/2}(|\hat{k}\rangle)/2$, and $FT_N(|k\rangle)_{j+N/2} = -\omega^j FT_{N/2}(|\hat{k}\rangle)/2$.

Note: All our calculations were done using the input $|k\rangle$, and not a general superposition, but by linearity if our calculation is correct for every $|k\rangle$ then it is true for superpositions $\sum_{k=0}^{N-1} \alpha_k |k\rangle$ as well.

Figure 3 describes a circuit to calculate $QFFT_N$, using $QFFT_{N/2}$ as a black box. From its structure we can see that if $S(n)$ denotes the size of the quantum circuit to perform a Fourier transform over $\mathbb{Z}_{2^n}$, then $S(n) = S(n-1) + O(n)$. Hence, $S(n) = O(n^2) = (\log_2 N)^2$.

2 Shor's algorithm for factoring

Shor's algorithm is the most important algorithmic result in quantum computation. The algorithm builds on ideas that already appear in Deutsch and Jozsa's algorithm and in Simon's algorithm, and like these algorithms, the basic ingredient of the algorithm is the Fourier transform. The problem can be stated as follows:

Input: An integer N
Output: A non-trivial factor of N, if exists.

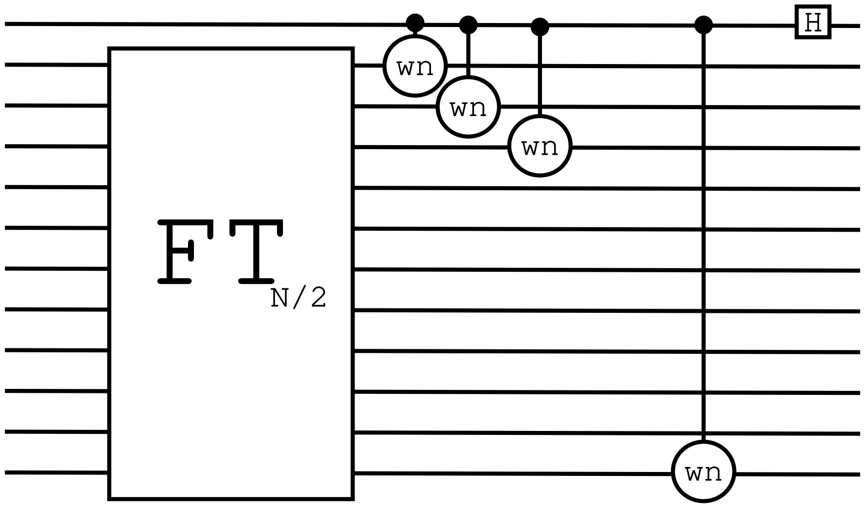


Figure 3: Quantum Fourier transform circuit

There is no proof that there is no polynomial classical factorization algorithm The problem is even not known to be NP-complete. However, factorization is regarded as hard, because many people have tried to solve it efficiently and failed. In 1994, Shor published a polynomial (in log(N)) quantum algorithm for solving this problem. In fact, Shor presented a quantum algorithm not for factoring, but for a different problem:

> **Order modulo N:**
> **Input:** An integer N, and Y coprime to N
> **Output:** The order of Y, i.e. the minimal positive integer r such that $Y^r = 1 \ mod \ N$.

The next section shows that in order to factor a number N it is enough to be able to find the order of x in $\mathbb{Z}_N$:

2.1 Reduction of factorization to order-finding

Theorem 1 *If there is a polynomial (in* $\log N$*) algorithm to solve order modulo* N*, then there is a polynomial algorithm to solve factorization.*

Proof: We show a (classical probabilistic) reduction from FACTORING to ORDER. That is, we assume we have a black box algorithm for finding the order of a given integer x (coprime to N), in $\mathbb{Z}_N$.

Lemma 1 *A solution to the equation $x^2 \equiv 1(modN)$ with $x \neq -1, +1$ gives a factor of N.*

This is true since this equality implies $(x+1)(x-1) = mN$ for some integer m, and both $1 < x-1, x+1 < N-1$. This means that either $gcd(x-1, N) \neq 1, N$ or $gcd(x-1, N) \neq 1, N$, as N divides $(x+1)(x-1)$. This proves the lemma. Now pick a random (nonzero) element of $\mathbb{Z}_N$. If $gcd(x, N) > 1$, then we have found a factor of N. If not, find the order r of x in Z_N. Assume we are lucky, and r is even and also $y := x^{r/2} \neq -1, +1$. Then y is a nontrivial solution to the equation $y^2 \equiv 1(modN)$, which gives us a factorization of N by the previous lemma. What is the probability that we are lucky?

Lemma 2 *Let $N = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \dots \cdot p_m^{\alpha_m}$, where p_i are distinct primes, N is odd and not prime (i.e. $m > 1, p_i \neq 2$). Choose $x \in Z_N$ randomly. Then*

$$Prob(ORD_{\mathbb{Z}_N}(x) \textit{ is even, and } x^{r/2} \neq -1, +1) \geq 1 - 2^{-m} \geq 1/2$$

We shall not prove this lemma (proof can be found in Nielsen-Chuang.) This shows that a randomly chosen x will give a factor of x using the procedure described above with probability $\geq 1/2$. This can, as usual, be amplified by repeating this process several times. We have thus proved our theorem.

2.2 The quantum algorithm for order finding

Again we will work with two registers. The first will hold a number between 1 to Q. (Q will be fixed later: it is much larger than N, but still polynomial in N.) The second register will carry numbers between 1 to N. Hence the two registers will consist of $O(\log(\mathrm{N}))$ qubits

Shor's Algorithm

$|\vec{0}\rangle \otimes |\vec{0}\rangle$

Apply Fourier Transform over Z_Q on the first register

$$\Downarrow$$

$\frac{1}{\sqrt{Q}} \sum_{l=0}^{Q-1} |l\rangle \otimes |\vec{0}\rangle$

Call subroutine which computes $|l\rangle|d\rangle \longmapsto |l\rangle|d \oplus Y^l \ mod \ N\rangle$

$$\Downarrow$$

$\frac{1}{\sqrt{Q}} \sum_{l=0}^{Q-1} |l\rangle \otimes |Y^l mod N\rangle$

Measure second register.

$$\Downarrow$$

$\frac{1}{\sqrt{A}} \sum_{l=0|Y^l=Y^{l_0}}^{Q-1} |l\rangle \otimes |Y^{l_0}\rangle = \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |jr + l_0\rangle \otimes |Y^{l_0}\rangle$

Apply Fourier Transform over Z_Q on the first register

$$\Downarrow$$

$\frac{1}{\sqrt{Q}} \sum_{k=0}^{Q-1} \left(\frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} e^{2\pi i (jr+l_0)k/Q} \right) |k\rangle \otimes |Y^{l_0}\rangle$

$$\Downarrow$$

Measure first register. Let k_1 be the outcome.
Approximate the fraction $\frac{k_1}{Q}$ by a fraction with denominator smaller than N, using the (classical) method of continued fractions.
If the denominator d doesn't satisfy $Y^d = 1 mod \ N$, throw it away.
Else call the denominator r_1.

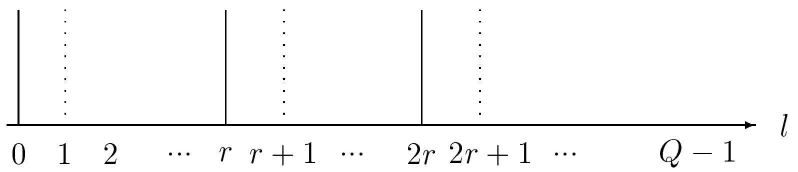
$$\Downarrow$$

Repeat all previous steps poly(log(N)) times to get r_1, r_2,..
Output the minimal r.

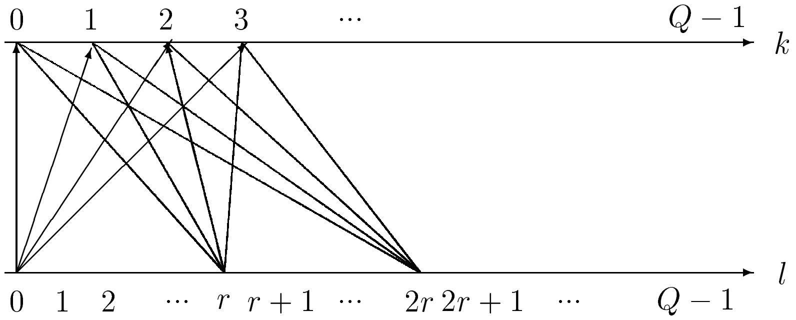
Let us now understand how this algorithm works. In the second step of the algorithm, all numbers between 0 and $Q-1$ are present in the superposition, with equal weights. In the third step of the algorithm, they are separated to sets, each has periodicity r. This is done as follows: there are r possible values written on the second register: $a \in \{Y^0, Y^1, ...Y^{r-1}\}$. The third state can thus be written as:

$$\frac{1}{\sqrt{Q}}\left((\sum_{l=0|Y^l=Y}^{Q-1} |l\rangle \otimes |Y\rangle) + (\sum_{l=0|Y^l=Y^2}^{Q-1} |l\rangle \otimes |Y^2\rangle) + ... + (\sum_{l=0|Y^l=Y^r}^{Q-1} |l\rangle \otimes |Y^r = 1\rangle) \right)$$

Note that the values l that give $Y^l = a$ have periodicity r: If the smallest such l is l_0, then $l = l_0 + r, l_0 + 2r, ..$ will also give $Y^l = a$. Hence each term in the brackets has periodicity r. Each set of l's, with periodicity r, is attached to a different state of the second register. Before the computation of Y^l, all l's appeared equally in the superposition. Writing down the Y^l on the second register can be thought of as giving a different "color" to each periodic set in $[0, Q-1]$. Visually, this can be viewed as follows:



The measurement of the second register picks randomly one of these sets, and the state collapses to a superposition of l's with periodicity r, with an arbitrary shift l_0. Now, how to obtain the periodicity? The first idea that comes to mind is to measure the first register twice, in order to get two samples from the same periodic set, and somehow deduce r from these samples. However, the probability that the measurement of the second register yields the same shift in two runs of the algorithm, i.e. that the same periodic set is chosen twice, is exponentially small. How to gain information about the periodicity in the state without simply sampling it? This is done by the Fourier transform. To understand the operation of the Fourier transform, we use a diagram again:



Each edge in the diagram indicates that there is some probability amplitude to transform from the bottom basis state to the upper one. We now measure the first register, to obtain k. To find the probability to measure each k, we need to sum up the weights coming from all the j's in the periodic set.

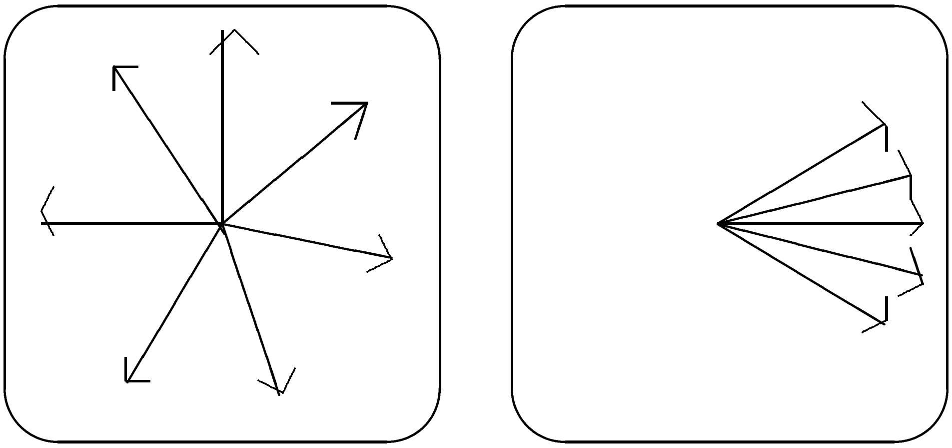
$$\mathrm{Prob(k)} = |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}} \mathrm{e}^{2\pi \mathrm{ik(jr+l_0)/Q}}|^2 = |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}} (\mathrm{e}^{2\pi \mathrm{ikr/Q}})^{\mathrm{j}}|^2 \tag{1}$$

Hence, in order to compute the probability to measure each k, we need to evaluate a geometrical series. Alternatively the geometric series is a sum over unit vectors in the complex plane.

Exact periodicity: Let us assume for a second *exact periodicity*, i.e. that r divides Q exactly. Then $A = Q/r$. In this case, the above geometrical series is equal to zero, unless $e^{2\pi ikr/Q} = 1$. Thus we measure with probability 1 only $k's$ such that $kr = 0 \ mod \ Q$. This is where destructive interference comes to play: only "good" k's, which satisfy $kr = 0 \ mod \ Q$, remain, and all the others cancel out. Why are such k's "good"? We can write $kr = mQ$, for some integer m, or $k/Q = m/r$. We know Q, and we know k since we have measured it. Therefore we can reduce the fraction k/Q. If m and r are coprime the denominator will be exactly r which we are looking for! The probability for all "good" k's is the same, so m is chosen randomly between 0 to $r-1$. By the prime number theorem, there are approximately $r/log(r)$ primes smaller than r. Repeating the experiment a large enough number of times we will with very high probability eventually get m prime, i.e. coprime to r.

Imperfect periodicity: In the general case, r does not divide Q, and this means that the picture is less clear. "Bad" k's do not completely cancel out.

We distinguish between two types of k's, for which the geometrical series of vectors in the complex plain looks as follows:



In the left case, all vectors point in different directions, and they tend to cancel each other. This will cause destructive interference, which will cause the amplitude of such k's to be small. In the right case, all vectors point almost to the same direction. In this case there will be constructive interference of all the vectors. This happens when $e^{2\pi ikr/Q}$ is close to one, or when $kr\ mod\ Q$ is close to zero. This means that with high probability, we will measure only $k's$ which satisfy an *approximate* criterion $kr \approx 0\ mod\ Q$. In particular, consider k's which satisfy:

$$-r/2 \leq kr \bmod Q \leq r/2 \tag{2}$$

There are exactly r values of k satisfying this requirement, because k runs from 0 to $Q-1$, therefore kr runs from 0 to $(Q-1)r$, and this set of integers contains exactly r multiples of Q. Note, that for such k's all the complex vectors lie in the upper half of the complex plane, so they are instructively interfering. Now the probability to measure such a k is bounded below, by choosing the largest exponent possible:

$$\mathrm{Prob(k)} = |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}}(\mathrm{e}^{2\pi\mathrm{ikr/Q}})^{\mathrm{j}}|^2 \geq |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}}(\mathrm{e}^{\mathrm{i}\pi\mathrm{r/Q}})^{\mathrm{j}}|^2$$

$$= \frac{1}{QA}|\frac{1-e^{\pi irA/Q}}{1-e^{i\pi r/Q}}|^2 = \frac{1}{QA}|\frac{sin(\frac{\pi rA}{2Q})}{sin(\frac{\pi r}{2Q})}|^2 \approx \frac{4}{\pi^2 r}$$

Where the approximation is due to the fact that Q is chosen to be much larger than $N > r$, therefore the sinus in the enumerator is close to 1 with negligible correction of the order of r/Q. In the denominator we use the approximation sin(x) $\approx$ x for small x, and the correction is again of the order of r/Q. The probability to measure any k which satisfies 2 is approximately $4/\pi^2$, since there are r such k's.

Why are such k's "good"? Given an integer k which satisfies the criterion 2, we can find r with reasonably high probability. Note that for "good" k's, there exists an integer m such that:

$$|\frac{k}{Q} - \frac{m}{r}| \leq \frac{1}{2Q}.$$

Remember that Q is chosen to be much larger than N, say $Q \geq N^2$. This means that $\frac{k}{Q}$, a fraction with denominator $\geq N^2$, can be approximated by $\frac{m}{r}$, a fraction with denominator smaller than N, to within $\frac{1}{N^2}$

Lemma 3 *There is at most one fraction m/r that approximates a fraction k/Q with error*

$$|\frac{k}{Q} - \frac{m}{r}| \leq \frac{1}{2Q}.$$

When $r < N$, $Q > N^2$

Proof: This is true since any two non-equal fractions a/b and c/d with denominators $b, d \leq N$ satisfy

$$|\frac{a}{b} - \frac{c}{d}| = |\frac{ad - cb}{bd}| \geq \frac{1}{N^2}.$$

So no two different fractions (with denominator smaller than N) can approximate any real number whatsoever with error as small as $1/2Q$ - which is smaller that $1/(2N^2)$

Given k/Q, the approximating fraction, $\frac{m}{r}$, can be found efficiently, using the method of **continued fractions**: Let α be some real number. Define $a_0 = \lfloor \alpha \rfloor$. So a_0 is some approximation of α, with error $x = \alpha - a_0$. Now write

$$\alpha = a_0 + x = a_0 + \frac{1}{(1/x)} = a_0 + \frac{1}{\lfloor (1/x) \rfloor + \{(1/x)\}}$$

here $\{(1/x)\} = x - \lfloor x \rfloor$ is the fractional part of x. Now define $a_1 = \lfloor x \rfloor$, and a second approximation to α would be

$$\alpha \approx a_0 + \frac{1}{a_1}$$

We may now proceed recursively. a_1 was an approximation to $1/x$, with error $x_1 = \{(1/x)\}$. Again look at $1/x_1$, and approximate it by $a_2 = \lfloor (1/x_1) \rfloor$. This gives another approximation to α:

$$\alpha \approx a_0 + \frac{1}{a_1 + \frac{1}{a_2}}$$

We are generating a sequence of rational approximations to α. Call the n-th approximation $CF_n(\alpha) = [a_0, a_1, \ldots, a_{n-1}]$. We quote a few properties of this sequence

Lemma 4 *1. For every real number α the sequence of rationals $CF_n(\alpha)$ converges to α as n tends to infinity.*

2. If $CF_n(\alpha) = p_n/q_n$ where $gcd(p_n, q_n) = 1$, then $CF_n(\alpha)$ is the best approximation to α by a rational number with denominator $\leq q_n$.

3. If $\alpha = p/q$ is rational, then its continued fraction expansion is finite, That is, $CF_n(\alpha) = \alpha$ for some finite n. Furthermore, the length of the expansion satisfies $n = O(\log q, \log p)$

4. If α is rational, and $|\frac{p}{q} - \alpha| < 1/q^2$ for some other rational p/q, then $CF_k(\alpha) = p/q$ for some k.

We now turn back to the algorithm. Recall that we have a fraction $\frac{k}{Q}$ as the (random) output of the algorithm, and that $|\frac{k}{Q} - \frac{m}{r}| \leq \frac{1}{2Q}$. We can now use the above lemma (since $Q \geq N^2$ and $N > r$) to see that if we calculate the sequence of continued fraction expansions for $\frac{k}{Q}$ then at some point $\frac{m}{r}$ will appear. We can also see that by the lemma, we don't have to calculate too many elements of the sequence - $\log k$ are enough. So we have a polynomial algorithm for reconstructing r from $\frac{k}{Q}$. Well, not precisely. Again, it might be the case that m and r are not coprime, and the number we find will be the denominator of the reduced fraction of $\frac{m}{r}$. In this case the number will fail the test $Y^r = 1$ which is included in Shor's algorithm, and it will be thrown away. Fortunately, the probability for m to be coprime to r is large enough: it is greater than $1/log(r)$. We repeat the experiment until this happens.
This concludes Shor's algorithm.