CSE491/596 Last lecture: Shor's Algorithm and Final Summation

Recall the Quantum Fourier Transform is just the Discrete Fourier Transform with exponential scaling:

5.2 Fourier Matrices

The next important family consists of the quantum Fourier matrices. Let ω stand for $e^{2\pi i/N}$, which is often called "the" principal Nth root of unity.

DEFINITION 5.2 The Fourier matrix $\boldsymbol{F}_N$ of order N is

$$\frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{N-2} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{N-3} \\ \vdots & & & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{N-2} & \omega^{N-3} & \cdots & \omega \end{bmatrix}$$

That is, $\boldsymbol{F}_N[i,j] = \omega^{ij \bmod N}$ divided by $\sqrt{N}$.

It is well known that $\boldsymbol{F}_N$ is a unitary matrix over the complex Hilbert space. This and further facts about $\boldsymbol{F}_N$ are set as exercises at the end of this chapter, including a running theme about its feasibility via various decompositions. For any vector $\boldsymbol{a}$, the vector $\boldsymbol{b} = \boldsymbol{F}_N\boldsymbol{a}$ is defined in our index notation by

$$\boldsymbol{b}(x) = \frac{1}{\sqrt{N}}\sum_{t=0}^{N-1}\omega^{xt}\boldsymbol{a}(t).$$

Since this is a Hermitian matrix, the inverse QFT simply conjugates all the entries, which is the same as using $\omega = e^{-2\pi i/N}$ instead.

Periodic Functions

The centerpiece of Peter Shor's algorithm detects a *period* in a function. Let

$$f : \mathbb{N} \to \{\, 0, 1, \ldots, M-1 \,\}$$

be a feasibly computable function. We are promised that there is a period r, meaning that, for all x,

$$f(x+r) = f(x).$$

The goal is to detect the period, that is, to determine the value of r. Actually, we need more than this promise. We also need that the repeating values

$$f(0), f(1), \ldots, f(r-1)$$

are all distinct. Some call this latter condition "injectivity" or "bijectivity." Possible relaxations of this condition are explored in the exercises, and overall its necessity and purpose are not fully understood.

The important example of a periodic function is **modular exponentiation**:

$$f_a(x) \;=\; a^x \bmod M.$$

Here a is a number in $\{0, 1, \ldots, M-1\}$ that is **relatively prime** to M. This means that a does not share a prime divisor with M. When $m = pq$ is the product of two different primes p and q, this simply means that a is not divisible by p or by q. If a and M did share a divisor p, then a^x would always be a multiple of p, and $a^x \bmod M$ is also a multiple of p because p divides M too. So you would not get all of the possible values modulo M. When a is relatively prime to M, what you always get is a number relatively prime to M. This is worth spelling out more than the text does:

Definition: $G_M \;=\; \{1\} \cup \{a : 1 \;<\; a \;<\; M \textit{ and a is relatively prime to } M\}$.

Theorem: G_M forms a **group** under multiplication.

When $M = pq$ is a product of two primes, the size of G_M is exactly $(p-1)(q-1)$. (The general name for the size of G_M is the **totient** function of M, devised by and often named for the mathematician Leonhard Euler.) The consequence of G_M being a group that we need is:

Corollary: For all $a \in G_M$ there is a positive integer r such that $a^r \equiv 1 \bmod M$.

The least such r is exactly the period of $f_a(x)$ that we want to find. It always divides $|G_M|$, so when $M = pq$ we get that r divides $(p-1)(q-1)$. You might think this should narrow down the possibilities, but:

- We don't actually get the value $m = (p-1)(q-1)$ factored for us---we don't even know m because we don't know how to factor $M \;=:\; pq$ to begin with.
- Compared to the number n of bits or digits of M, which is the complexity parameter we care about, the range of numbers less than m we might have to check is exponential in n.
- By the way, the number x in a^x can be exponential in n, so it looks like it takes too long to compute $f_a(x)$ to begin with. However, by **iterated squaring modulo** M we can compute the following values in $\widetilde{O}\left(n^2\right)$ time: $a_1 \;=\; a^2 \bmod M$, $a_2 \;=\; a_2^2 \bmod M \;=\; a^4 \bmod M$, $a_3 \;=\; a_2^2 \bmod M \;=\; a^8 \bmod M$, $a_4 \;=\; a_3^2 \bmod M \;=\; a^{16} \bmod M$, and so on up to $a_{n-1} \;=\; a_{n-2}^2 \bmod M \;=\; a^{n-1} \bmod M$. Then we need only multiply together those a_i such that x as a binary number includes 2^i. This needs only $2n$ multiplications and mod-M reductions of n-bit numbers, so it is doable in $\widetilde{O}\left(n^2\right)$ time using an $\widetilde{O}(n)$-time integer multiplication algorithm. (Or we can say $O\left(n^3\right)$ time using the simple multiplication algorithm. The **RSA cryptosystem** uses modular exponentiation too---and this time is largely why your credit card needed a chip.)

Nevertheless, if we *do* find the period r---for a "good" value a which we stand a fine chance of picking at random from G_M---then it was known long before Peter Shor found his algorithm in 1993 that we can go on to find p and q by classical efficient means.

Theorem: There is a classical randomized algorithm that, when provided a ***function oracle*** $g(M, a) =$ some integer multiple of the period of $f_a \bmod M$, finds a factor of M in expected polynomial time. That is, **Factoring** is in BPP^g.

The proof is the entire content of Chapter 12. Lipton and I bundled this up into a separate chapter so that instructors would have the freedom to skip it, as we'll do now. So we can focus on the task of finding r (or at least a multiple of r) via *quantum means*.

Shor's Theorem: **Factoring** is in BQP.

Steps of Shor's Algorithm

1. Given M, use classical randomness to guess a number a between 2 and $M-1$.
2. Use Euclid's algorithm to find $\gcd(a, M)$. If it gives a number $c > 1$, then "ka-ching!"---we got a divisor of M. Since both c and M/c are below $M/2$, we can recursively factor both of them.
3. If it gives $\gcd(a, M) = 1$, then we know $a \in G_M$. In the important $M = pq$ case, this had probability $\frac{(p-1)(q-1)}{pq}$ and so was pretty likely anyway. By the way, Euclid's algorithm also gives you a number b such that $ab = 1 \bmod M$. But it doesn't give you this b as a power of a (to wit, as $b = a^{r-1} \bmod M$), which is what you'd need to get r.
4. To give some slack, we choose a number $Q = 2^\ell \approx M^2$ and expand the domain of $f_a(x)$ to include x in the interval up to $Q-1$, not just up to $M-1$. The range is still 1 to $M-1$. So our domain is x in the range 0 to $2^\ell - 1$, which uses $\ell \approx 2n$ bits. This gives us quadratically many "ripples" of the period, which in turn helps the trigonometric analysis in the body of the proof.
5. The quantum circuit begins with q-many Hadamard gates, followed by a quantum implementation of the $n^{O(1)}$ classical gates needed to compute modular exponentiation. This produces the functionally superposed quantum state
$$\Phi_f = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^\ell} |x f_a(x)\rangle.$$
6. Apply the QFT (or its inverse) to the first ℓ qubits.
7. Then *measure* the whole result. Curiously, we ignore what happens in the "$f_a(x)$" portion of the circuit. The fact that those final n qubits were entangled with the first ℓ qubits is enough. So we let our output y be the first ℓ bits of the measured result over the binary standard basis.

My own quantum circuit simulator draws an ASCII picture of the Shor circuit, here for $M = 21 = 3*7$ (where I guessed $a = 5$), which gave $\ell = 9$ since $2^9 = 512$ is the next power of 2 after $M^2 = 441$:

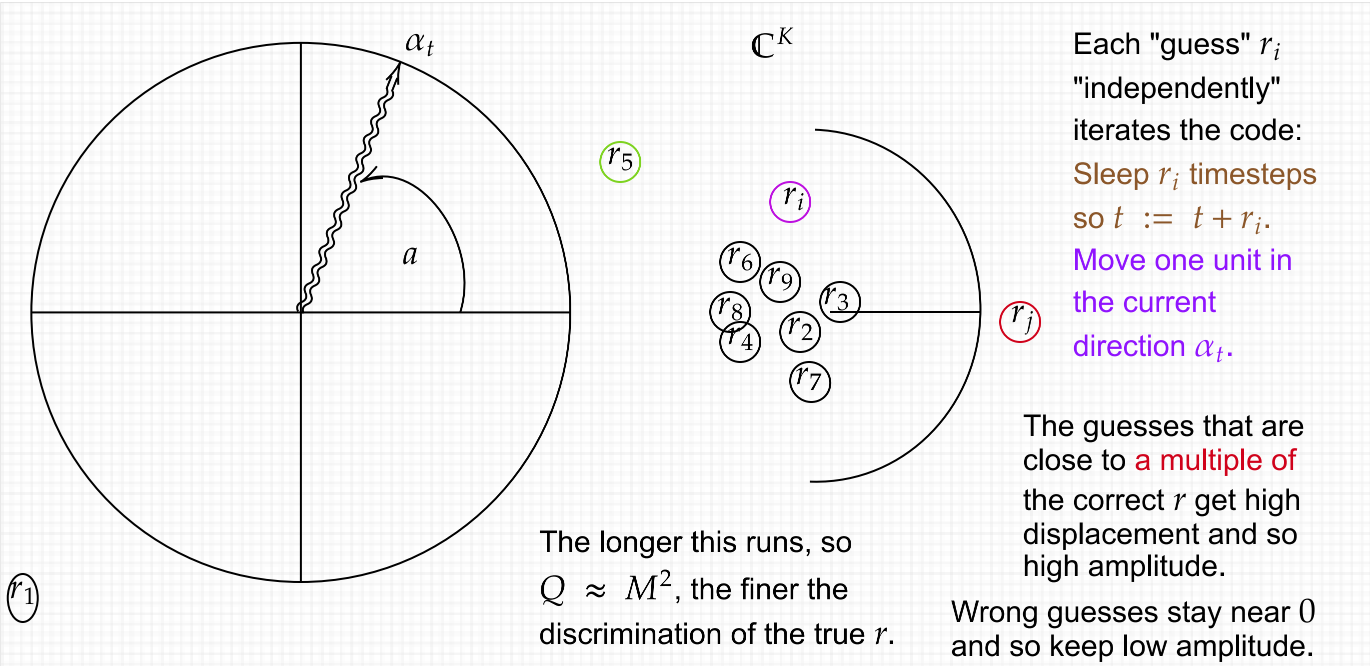
```
->--[H]--[MODEXP]--[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |        |
->--[H]------|-----[QFTI]--<?
              |
->-------[MODEXP]----------<?
              |
->-----------|-------------<?
              |
->-----------|-------------<?
              |
->-----------|-------------<?
              |
->-----------|-------------<?
```

But there isn't any more to the quantum circuitry than that. It's all simply: compute a giant functional superposition and apply QFT (or its inverse) to it.

The analysis establishes that with pretty good probability already in one shot, the output y reveals the period r by a followup classical means. And with initial good probability over the choice of a, the resulting value r unlocks the key to factoring M. We will focus on understanding why the measured y has much to do with the period r to begin with. Then basic point---which has been known for centuries---is that the Fourier transform converts *periodic data* to *peaked data*. Here is how the simple quantum circuit above applies this fact.

The Intuition (See also Scott Aaronson, https://www.scottaaronson.com/blog/?p=208)

Let r stand for the true period of f. Let a be any element of the group G_M of size $(p-1)(q-1)$. Then we will picture a as a "crazy clock" that jumps a units *counter*-clockwise at each time step.



With fairly high probability, measurement yields a multiple of r. The true r is the least of the multiples. It is individually the most likely value returned and is also returned with reasonable probability. A bad r might work anwyay. We can tell whether r works by seeing if the classical part gives us p or q, else we just try the quantum process again.

The run of my simulator on $M = 21$ and $a = 5$ succeeded on the second try:

```
About to do try 1 of sampling QFT applied to 1010101011010010100 with status now PROBS_ENUMERA
Sampling with status PROBS_ENUMERATED:
Base probability for conditionals: 0.166015625000
Current: 0 with probability 0.083007813 on rolling 0.325191374; last 0 prob = 0.500000000
Current: 00 with probability 0.055282593 on rolling 0.563273639; last 0 prob = 0.665992647
Current: 001 with probability 0.027659269 on rolling 0.559076137; last 0 prob = 0.499674899
Current: 0010 with probability 0.027418884 on rolling 0.941772811; last 0 prob = 0.991309060
Current: 00101 with probability 0.027183985 on rolling 0.139894580; last 0 prob = 0.008567052
Current: 001010 with probability 0.026380861 on rolling 0.938149097; last 0 prob = 0.970455980
Current: 0010101 with probability 0.025648040 on rolling 0.595421001; last 0 prob = 0.02777850
Current: 00101010 with probability 0.020074378 on rolling 0.114898273; last 0 prob = 0.7826866
Current: 001010101 with probability 0.018908726 on rolling 0.791199151; last 0 prob = 0.058066
sampled output vector: 00101010110100
time cost: 1.23308 milliseconds.

Measured 001010101 as 85 giving 0.166015625
Fractional approximation is 1/6
; Possible period is 6
; Unable to determine factors, we'll try again.
Let's take a free random crack at it without the QFT application...
Fractional approximation is 2/3
; Odd denominator, trying to expand by 2.
; Possible period is 6
; Unable to determine factors, we'll try again.

About to do try 2 of sampling QFT applied to 1010101011010010100 with status now PROBS_ENUMERA
Sampling with status PROBS_ENUMERATED:
Base probability for conditionals: 0.166015625000
Current: 1 with probability 0.083007813 on rolling 0.527169932; last 0 prob = 0.500000000
Current: 10 with probability 0.055282593 on rolling 0.051374227; last 0 prob = 0.665992647
Current: 100 with probability 0.027623324 on rolling 0.277237177; last 0 prob = 0.499674899
Current: 1000 with probability 0.027576410 on rolling 0.189192738; last 0 prob = 0.998301645
Current: 10000 with probability 0.027567765 on rolling 0.562397971; last 0 prob = 0.999686499
Current: 100000 with probability 0.027564179 on rolling 0.523783427; last 0 prob = 0.999869929
Current: 1000000 with probability 0.027562462 on rolling 0.694951445; last 0 prob = 0.99993770
Current: 10000000 with probability 0.027561612 on rolling 0.646817553; last 0 prob = 0.9999691
Current: 100000000 with probability 0.027561188 on rolling 0.353241189; last 0 prob = 0.999984
sampled output vector: 10000000010100
time cost: 1.2329 milliseconds.

Measured 100000000 as 256 giving 0.500000000
Fractional approximation is 1/2
; Possible period is 2
; Success: 21 = 3 * 7
Success after 2 xy sample(s) plus 2 QFT sample(s).
```

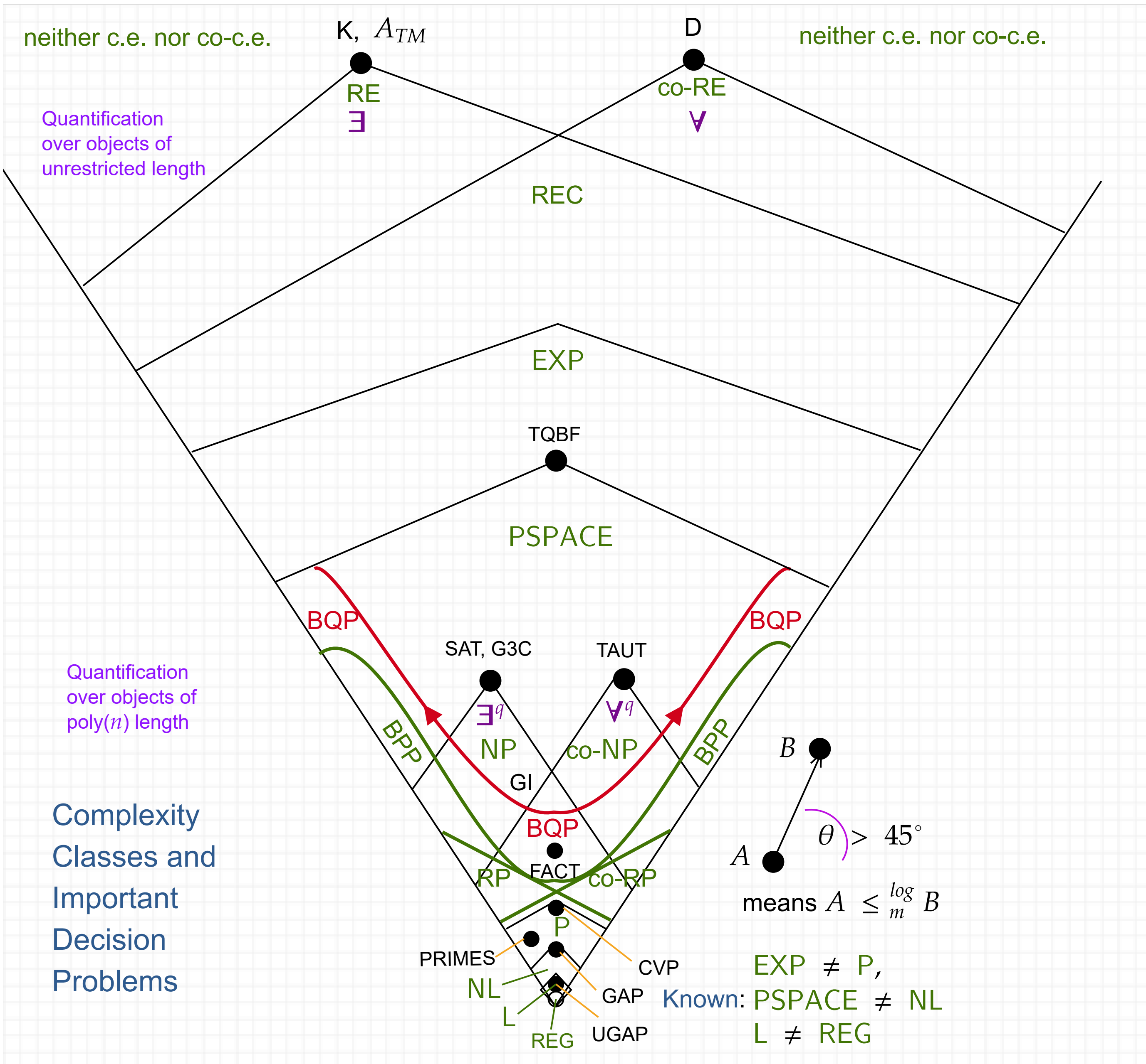
[For the curious, but officially FYI: extra details of Shor's and Grover's algorithms are on the course webpage.]

A Final Look Back at Complexity

Shor's algorithm classifies **FACTORING** as belonging to BQP. Another problem in NP that is strongly not believed to be complete (because of a sense in which it "almost" belongs to NP $\cap$ co - NP) is **Graph Isomorphism**: the set of representations of pairs of graphs (G_1, G_2) of the same number n of nodes such that there is a function g from the vertex set V_1 of G_1 onto the vertex set V_2 of G_2 such that, for all $u, v \in V_1$, (u, v) is an edge in G_1 if and only if $(g(u), g(v))$ is an edge in G_2. Nor is this language, which is called **GI** for short, known to be in BQP either. Since BQP is closed downward under $\leq_m^p$, it follows that none of the NP-complete problems is known to belong to BQP---and here there is a strong belief that they are not.

Here are three more facts that round out the known memberships of major languages and shapes of complexity classes:

- For any reasonable space function $s(n) = \Omega(\log n)$, NSPACE$[s(n)]$ is closed under complements. In particular, NL = co - NL and NLBA = co - NLBA. This theorem was proved independently by Robert Szelepcsenyi and Neil Immerman in 1987-88.
- The set **PRIMES** was finally classified into deterministic polynomial time in 2002 by Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. We say that it was **de-randomized** from RP $\cap$ co-RP into P.
- The graph-accessibility problem for undirected graphs G, which is called **UGAP** , was de-randomized from randomized logspace to L in 2004 by Omer Reingold.


neither c.e. nor co-c.e.
K, A_{TM}
D
neither c.e. nor co-c.e.
RE
∃
co-RE
∀
Quantification over objects of unrestricted length
REC
EXP
TQBF
PSPACE
BQP
BQP
SAT, G3C
TAUT
$\exists^q$
$\forall^q$
Quantification over objects of poly(n) length
BPP
NP
co-NP
BPP
GI
B
Complexity Classes and Important Decision Problems
BQP
θ > 45°
A
FACT
RP
co-RP
means $A \leq^{log}_{m} B$
P
PRIMES
CVP
NL
GAP
L
REG
UGAP
Known: EXP ≠ P, PSPACE ≠ NL, L ≠ REG