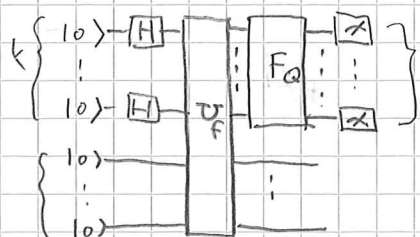


Recap from last time.

to know the period s of $f(r) = x^r \bmod N$
(for "good" x) which would give p, q ($N = pq$)



would give $c \frac{Q}{s}$ ($0 \leq c \leq 1$) $Q = 2^k$
with high prob.

U_f : XOR query $|zy\rangle \mapsto |z(y \oplus f(z))\rangle$

$$F_Q = \frac{1}{\sqrt{Q}} [w^{ij}]_{0 \leq i, j < Q} \quad w = e^{\frac{2\pi i}{Q}}$$

Fourier transform

we want concrete implementations of U_f , F_Q in
polynom. steps in terms of k ($\sim \log_2 Q \sim 2 \log_2 N$)
 $Q \sim N^2$

• impl. U_f : repeated squaring (classical)

fundamental block: square func. $sq: y \mapsto y^2$

to compute $f(z) = f\left(\sum_{0 \leq i < k} z^i z_i\right)$ ($z \in \{0, 1\}^k$)

$$\text{use } x^{1 \cdot z_0 + 2z_1 + \dots + 2^{k-1}z_{k-1}} = \underbrace{(1 \text{ or } x)}_{z_0=0,1} \cdot \underbrace{(1 \text{ or } x^2)}_{z_1=0,1} \cdot \dots \cdot \underbrace{(1 \text{ or } x^{2^{k-1}})}_{z_{k-1}=0,1}$$

$\Rightarrow$ enough to combine $\frac{k(k+1)}{2}$ many $sq(\sim)$.

together with k "if" and k XOR.

• Fourier transform (§ 20.1.1).

- recursive def. of F_Q

write F_Q as

$$\frac{1}{\sqrt{Q}} \begin{pmatrix} \overbrace{1 \ 1 \ \dots \ 1}^{2^{k-1}} & \overbrace{1 \ 1 \ \dots \ 1}^{2^{k-1}} \\ 1 & w^2 & \dots & 1 \\ 1 & w^{2^{k-2}} & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & w^{2^{k-2}} & \dots & w^{2^{k-1}} \end{pmatrix}$$

usual index from $i=0$ to $Q-1$

$j: (0 * \dots *) (1 * \dots *)$

$\bar{i}=1, \bar{j}=2 \leftrightarrow (010 \dots 0)$
 top-left corner: $\frac{1}{\sqrt{Q}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ \vdots & \omega^3 & \dots & \omega^{2^{k-1}-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^{k-1}-1} & \dots & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} F_{Q/2}$
 $\omega^2 = Q/2$ -th root of 1

= bottom-left corner

$\bar{i}=1, \bar{j}=1 \leftrightarrow (10 \dots 0)$
 $\bar{i}=2^{k-1}-1, \bar{j}=1$
 top-right corner: $\frac{1}{\sqrt{Q}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ \vdots & \omega^3 & \dots & \omega^{2^{k-1}-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^{k-1}-1} & \dots & 1 \end{bmatrix} \cdot \frac{1}{\sqrt{Q}}$

Prop 1. top-right corner = $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \omega & \dots & \omega^{2^{k-1}-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{2^{k-1}-1} & \dots & \dots & 1 \end{bmatrix} F_{Q/2}$
 $Q/2 = 2^{k-1} \rightarrow B$

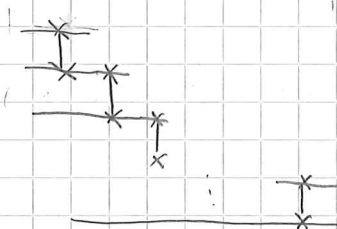
Proof (i, j') -entry is $(F_Q)_{i, j}$ with $j = 1 + 2j'$
 $(0 \leq i, j' < 2^{k-1})$
 $\Rightarrow \frac{1}{\sqrt{Q}} \omega^{i \cdot (1+2j')} = \frac{1}{\sqrt{Q}} \omega^i \cdot (\omega^2)^{j'} = \frac{1}{\sqrt{Q}} \omega^i \cdot (F_{Q/2})_{i, j'}$

Prop 2. bottom-right corner = $-\frac{1}{\sqrt{2}} B F_{Q/2}$

Proof (i', j') -entry is $(F_Q)_{i, j}$ with $i = 2^{k-1} + i'$
 $j = 1 + 2j'$
 $\Rightarrow \frac{1}{\sqrt{Q}} \omega^{(2^{k-1} + i') \cdot (1+2j')} = \frac{1}{\sqrt{Q}} \underbrace{(\omega^{Q/2})^{(1+2j')}}_{-1} \cdot \omega^{i' \cdot (1+2j')}$

Prop 3. $|z_0 z_1 \dots z_{k-1}\rangle \mapsto |z_1 z_2 \dots z_{k-1} z_0\rangle$

is a prod. of swaps



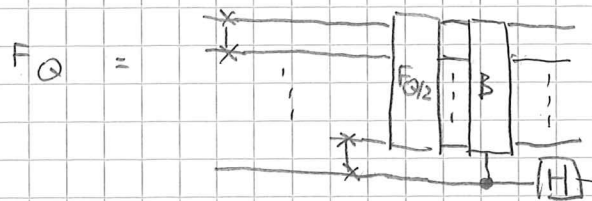
So far:

$F_Q = \begin{bmatrix} F_{Q/2} & B F_{Q/2} \\ F_{Q/2} & -B F_{Q/2} \end{bmatrix}$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} F_{Q/2} & B F_{Q/2} \\ F_{Q/2} & -B F_{Q/2} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{Q/2} & I_{Q/2} \\ I_{Q/2} & -I_{Q/2} \end{bmatrix} \begin{bmatrix} I_{Q/2} & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} F_{Q/2} & 0 \\ 0 & F_{Q/2} \end{bmatrix}$$

$I_{Q/2} \otimes I$ controlled B $F_{Q/2} \otimes I_2$

Summary:

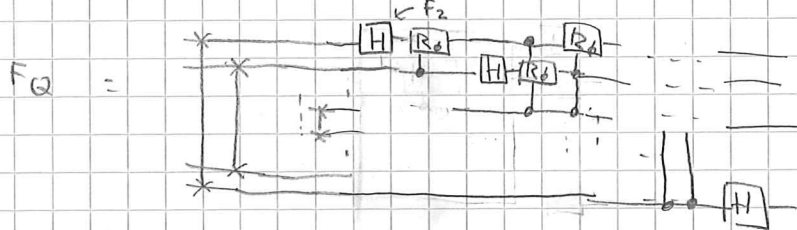


$B =$ product of rotations

$$|z_0 \dots z_{k/2-1}\rangle \xrightarrow{\omega^{z_0 z_1 \dots z_{k/2-1}}} |z_0 \dots z_{k/2-1}\rangle$$

$$= \left((1 \text{ or } \omega) \right)_{z_0=0,1} \left((1 \text{ or } \omega^2) \right)_{z_1=0,1} \dots \left((1 \text{ or } \omega^{2^{k/2-1}}) \right)_{z_{k/2-1}=0,1} |z_0 \dots z_{k/2-1}\rangle$$

working out recursion



So we can implement Shor's algorithm with $\sim k^2$ fundamental gates and square functions

Application of Shor's algorithm

structure of finite commutative groups

e.g. $(\mathbb{Z}/p)^\times$ p : prime for discrete logarithm problem $a^x = b \pmod{p}$

↳ Diffie-Hellman encryption

but not finite (non-commutative) groups in general

↳ lattice-based cryptography

