3. The quantum Fourier transform, period finding, and Shor's factoring algorithm

Can we go beyond Fourier trafo on $\mathbb{Z}_2$ (to $\mathbb{Z}_N$, for $N \sim 2^n$)?

- What is the right transformation?
- Can it be implemented efficiently?
- What is it good for?

Further reading:
A. Ekert and R. Jozsa,
Quantum computation and Shor's factoring algorithm.
Rev. Mod. Phys **68**, 733 (1996)
https:///doi.org/10.1103/RevModPhys.68.733

a) The Quantum Fourier Transform

Discrete Fourier trafo (FT) on $\mathbb{C}^N$:

$$x = (x_0, \dots, x_{N-1}) \in \mathbb{C}^N$$

$$y = (y_0, \dots, y_{N-1}) \in \mathbb{C}^N$$

FT: $F: x \longmapsto y$ s.t. $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \, e^{2\pi i jk/N}$

Definition QFT

$$|j\rangle \longmapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i jk/N} |k\rangle$$

Observe:

$$\sum_j x_j |j\rangle \xrightarrow{\text{QFT}} \sum_{jk} x_j \, e^{2\pi i jk/N} |k\rangle = \sum y_k |k\rangle$$

i.e.: QFT acts as discrete FT on amplitudes!

Computational cost of classical FT:

- $O(N^2)$ operations.
- $N \sim 2^n \Rightarrow$ exponential in # of bits in N.
- Fast FT (FFT): only $O(N \log N)$, but still exponential!
- $O(N)$ is lower bound: minimal time to even just output y_k!

Will see: QFT can be implemented on a quantum state in $O(n^2)$ steps

$\rightarrow$ exponential speedup!

(But only useful if input is given as q. state!)

Step I: Rewrite QFT in binary

- Consider case $N = 2^n$.
- Write j etc. in binary:

$$j = j_1 j_2 j_3 \cdots j_n = j_1 \cdot 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0$$

- "Decimal" point notation:

$$0.j_\ell j_{\ell+1} \cdots j_n = \frac{1}{2} j_\ell + \frac{1}{4} j_{\ell+1} + \cdots + \frac{1}{2^{n-\ell+1}} j_n$$

Then:

$$|j\rangle \longmapsto \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i \, j \cdot k/2^n} |k\rangle \qquad \frac{k}{2^n} = 0.k_1 k_2 \cdots k_n$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{\ell=1}^{n} k_\ell 2^{-\ell} \right)} |k_1, \ldots, k_n\rangle$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \left[\bigotimes_{\ell=1}^{n} \left(e^{2\pi i j k_\ell 2^{-\ell}} |k_\ell\rangle \right) \right]$$

$$= \bigotimes_{\ell=1}^{n} \left[\frac{1}{\sqrt{2}} \sum_{k_\ell=0}^{1} e^{2\pi i j k_\ell 2^{-\ell}} |k_\ell\rangle \right]$$

$$= \bigotimes_{\ell=1}^{n} \frac{1}{\sqrt{2}} \left[|0\rangle + e^{2\pi i j 2^{-\ell}} |1\rangle \right] = \cdots$$

$$j \cdot 2^{-\ell} = \underbrace{j_1 j_2 \cdots j_{n-\ell}}_{\text{integer}} \cdot j_{n-\ell+1} \cdots j_n$$

$$e^{2\pi i (j \cdot 2^{-\ell})} = e^{2\pi i \cdot (\text{integer} + 0.j_{n-\ell+1} \cdots j_n)}$$

$$= e^{2\pi i \cdot 0.j_{n-\ell+1} \cdots j_n}$$

$$\cdots = \frac{|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_{n-1} j_n} |1\rangle}{\sqrt{2}} \otimes \cdots$$

$$\cdots \otimes \frac{|0\rangle + e^{2\pi i\, 0.j_1 j_2 \cdots j_n} |1\rangle}{\sqrt{2}}$$

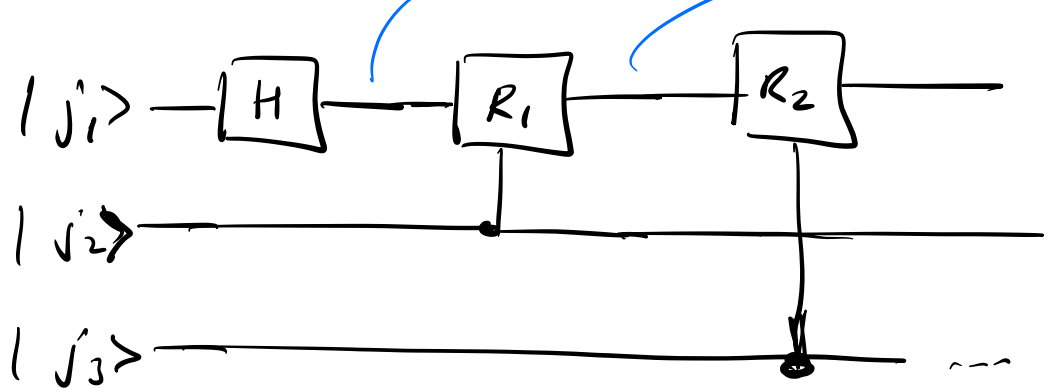
Step II: Implement this as a circuit.

Consider first only rightmost term:

$$\frac{|0\rangle + e^{2\pi i\, 0.j_1 j_2 \cdots j_n} |1\rangle}{\sqrt{2}} = \frac{|0\rangle + e^{2\pi i j_1/2}\, e^{2\pi i j_2/4}\, e^{2\pi i j_3/8} \cdots |1\rangle}{\sqrt{2}}$$

$(-1)^{j_1}$

$$|0\rangle + e^{2\pi i j_1/2} |1\rangle \qquad |0\rangle + e^{2\pi i j_1/2} e^{2\pi i j_2/4} |1\rangle$$



$$R_d = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & e^{2\pi i \cdot 2^{-(d+1)}} \end{pmatrix}$$

Action of gates:

$$H: |j_1\rangle \longmapsto |0\rangle + e^{2\pi i\, 0.j_1} |1\rangle$$

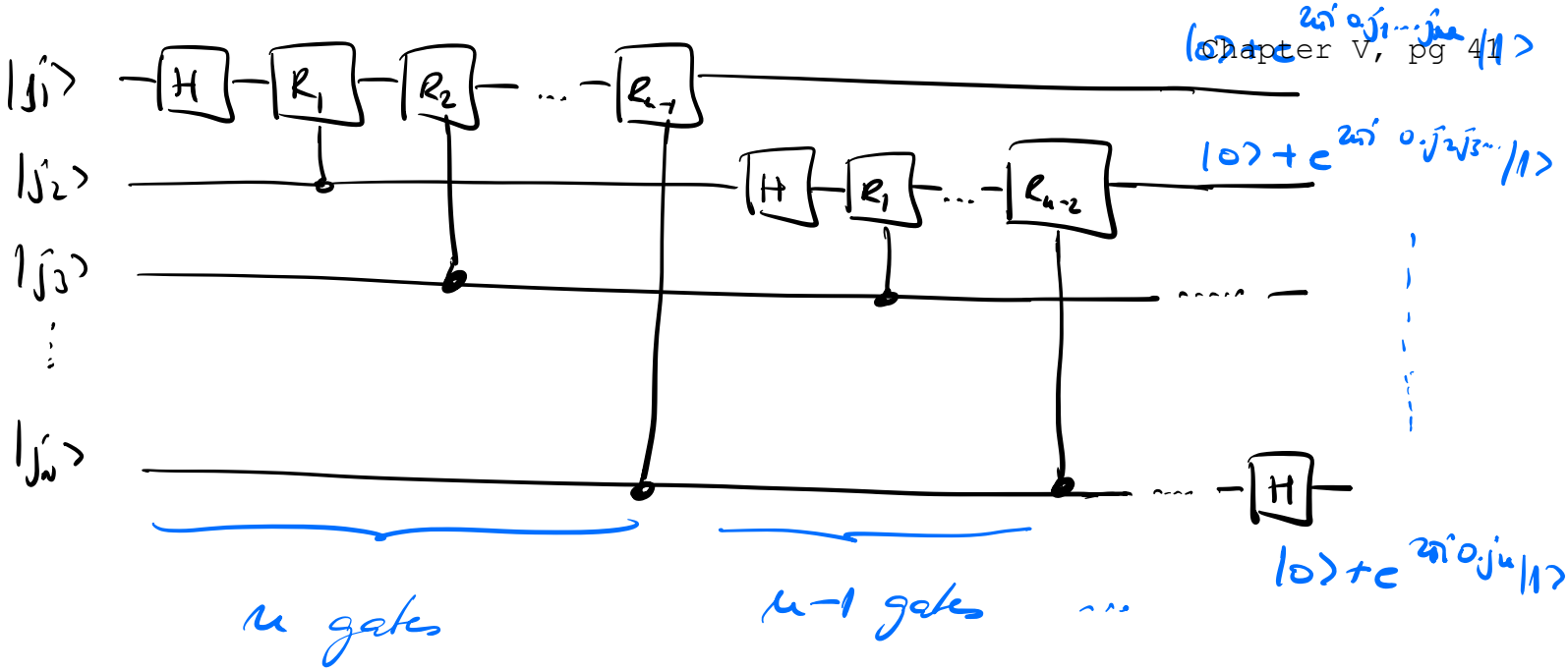
$$C\text{-}R_1: \left(|0\rangle + e^{2\pi i\, 0.j_1} |1\rangle\right)|j_2\rangle \longmapsto \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2} |1\rangle\right)|j_2\rangle$$

$$C\text{-}R_2: \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2}\right)|j_2\rangle|j_3\rangle \longmapsto \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2 j_3} |1\rangle\right)|j_2\rangle|j_3\rangle$$

$\vdots$ and so on.

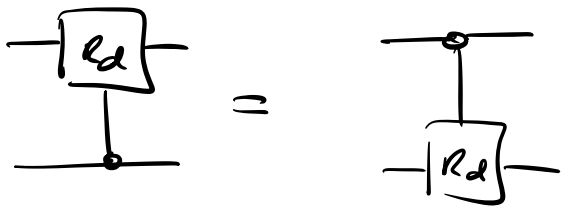
$\longrightarrow$ Outputs the n-th qubit of the QFT on 1st qubit.

Continue on this vein:



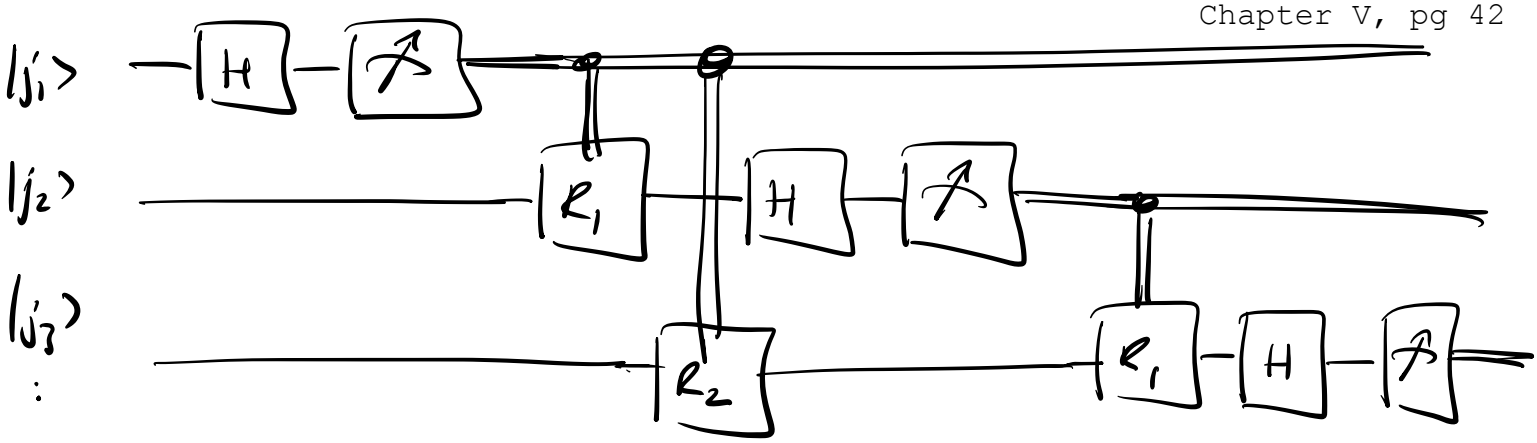
Gate count: $\frac{n(n+1)}{2} = O(n^2)$ gates!

Notes:

- Output qubits in reverse order (can re-order if needed: $n/2$ swaps).
-  ⇒ can flip C-R_d gates

Then, upper line acts as control in comp. basis.

⇒ If we measure directly after QFT in comp. basis, we can measure before the C-R_d gates & control them classically:



Only one-qubit gates needed (!!)

("Where is the quantum-ness?")

b) Period finding

Application of QFT: find period of a function?

(cf. Simon's algorithm)

Consider a periodic function f:

$$f: \{0,1\}^q \longrightarrow \{0,1\}^q,$$

understood as numbers $0, 1, \ldots, 2^q - 1$!

such that $\exists r > 0$ with

$$f(x) = f(x+r), \qquad 0 \leq x \leq 2^q - r - 1$$

and $f(x) \neq f(y)$ otherwise.

periodicity superfast across boundary.

Can we find r better than classically?

(i.e., with much less than $\sim r$ queries to f)

Consider the regime where $r \ll 2^n$

↑ will make this specific later.

Goal: superfactor at end. negligible.

Implement U_f on quantum computer as before:

$$U_f : |x\rangle_A |y\rangle_B \longmapsto |x\rangle_A |y \oplus f(x)\rangle_B$$

Algorithm:

(1) Hadamard on A, then U_f:

$$\frac{1}{2^{n/2}} \sum |x\rangle_A |0\rangle_B \xrightarrow{U_f} \frac{1}{2^{n/2}} \sum |x\rangle_A |f(x)\rangle_B$$

(2) Measure B register. For result $|f(x_0)\rangle_B$,

A collapses to

$$\frac{1}{\sqrt{k_0}} \sum_{k=0}^{k_0 - 1} |x_0 + kr\rangle$$

- here, $0 \leq x_0 < r$, and $\frac{2^4}{r} - 1 < k_0 \leq \frac{2^4}{r}$.

(3) Apply QFT:

$$\longmapsto \frac{1}{2^{4/2}\sqrt{k_0}} \sum_{k=0}^{k_0-1} \sum_{\ell=0}^{2^4-1} e^{2\pi i (x_0 + kr)\ell/2^4} |\ell\rangle_A$$

$$= \sum_{\ell=0}^{2^4-1} e^{2\pi i x_0 \ell/2^4} \underbrace{\sum_{k=0}^{k_0-1} \frac{1}{2^{4/2}\sqrt{k_0}} e^{2\pi i k r \ell/2^4}}_{=: \hat{a}_\ell} |\ell\rangle_A$$

$$=: a_\ell$$

(4) Measure in computational basis:

$|\hat{a}_\ell|^2$: probability to obtain outcome ℓ

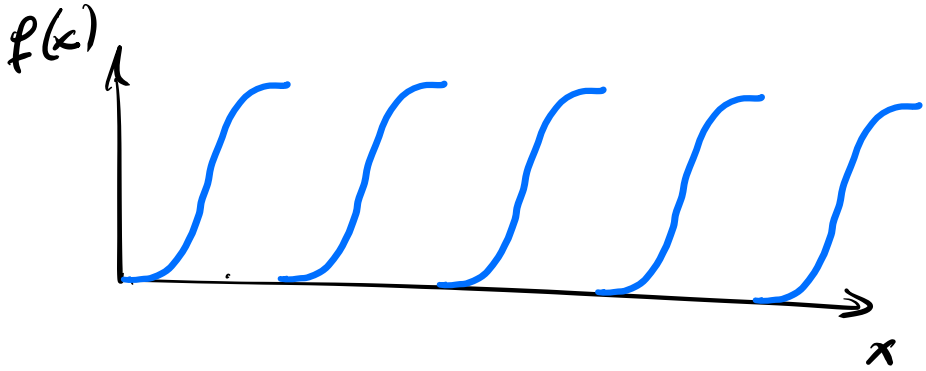
Intuitively: $\hat{a}_\ell \propto \sum_k e^{2\pi i k \left(r\ell/2^4\right)}$

peaked around points ℓ where $\frac{r\ell}{2^4}$ is close to an integer!

(→ will quantify this in a moment!)

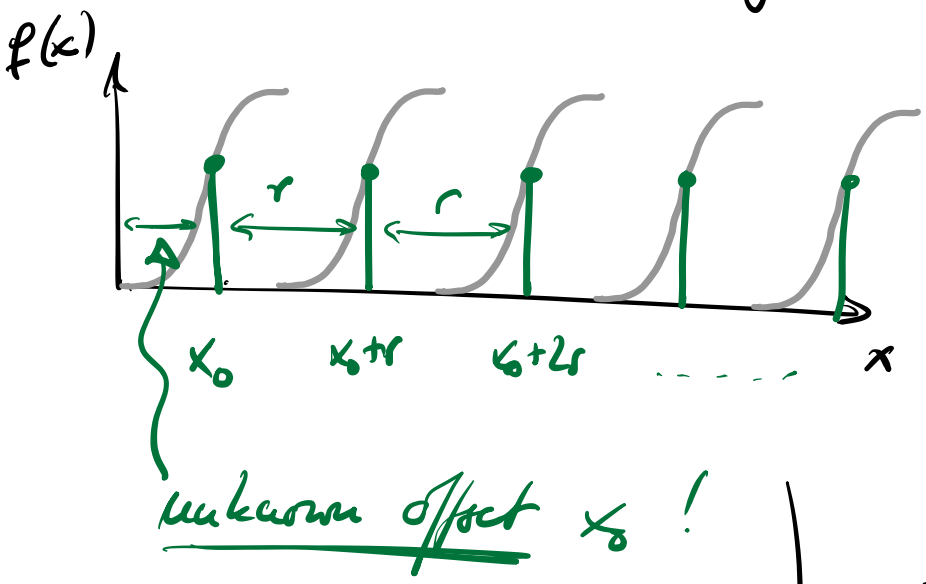
Intuitive picture:

(General features of Fourier transforms — valley quantum!)



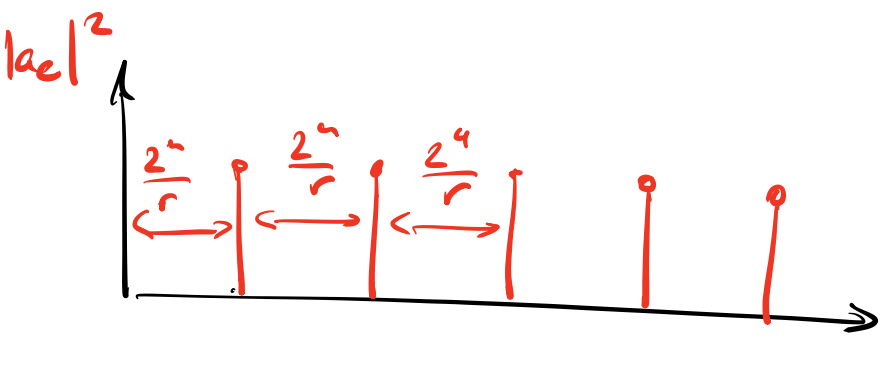
periodic function

after meas. of $B \longrightarrow x_0$



unknown offset x_0 !

Fourier trafo



unknown offset:

absorbed in phase of a_e!

$\Rightarrow$ can determine multiple of $\frac{2^n}{r}$ by measuring ℓ (How to get r? Later!)

Detailed analysis of $|a_\ell|^2$:

How much total weight is on all $|a_\ell|^2$ with

$$\ell = \frac{2^n}{r} \cdot s + \delta_s \; ; \quad \delta_s \in \left(-\tfrac{1}{2}; \tfrac{1}{2}\right] ; \quad s = 0, \dots, r-1$$

(i.e. only those ℓ which are closest to $\frac{2^n}{r} \cdot s$

$\rightarrow$ from those, we can uniquely infer $\frac{2^n}{r} \cdot s$.)

Then, $\hat{a}_\ell = \frac{1}{2^{n/2}\sqrt{k_0}} \sum_{k=0}^{k_0-1} e^{2\pi i k \underbrace{\left(s + \frac{r}{2^n}\delta_s\right)}_{\equiv r\ell/2^n}}$

$$= \frac{1}{2^{n/2}\sqrt{k_0}} \; \frac{e^{2\pi i \frac{r}{2^n}\delta_s k_0} - 1}{e^{2\pi i \frac{r}{2^n}\delta_s} - 1}$$

... since $\frac{2^n}{r} - 1 < k_0 \leq \frac{2^n}{r}$, and $r \ll 2^n$:

$$\frac{k_0 r}{2^n} = 1 - \varepsilon \; , \quad 0 \leq \varepsilon < \frac{r}{2^n} \ll 1.$$

$$= \frac{1}{2^{4/4}\sqrt{k_0}} \; \frac{e^{2\pi i \delta_s (1-\varepsilon)} - 1}{e^{2\pi i \frac{r}{2^4}\delta_s} - 1}$$

$\sin x \geq \frac{x}{\pi/2}$ in relev. interval

$$\Rightarrow \quad |a_e|^2 = \frac{1}{2^4 k_0} \left(\frac{\sin\left(\pi \delta_s (1-\varepsilon)\right)}{\sin\left(\frac{\pi r}{2^4}\delta_s\right)} \right)^2$$

$\sin x \leq x$

$$\geq \frac{1}{2^4 k_0} \; \frac{\frac{\pi^2 \delta_s^2 (1-\varepsilon)^2}{\pi^2/4}}{\frac{\pi^2 r^2}{(2^4)^2}\,\delta_s^2}$$

$$= \frac{4}{\pi^2}\,\frac{1}{r}\; \frac{(1-\varepsilon)^2}{\frac{k_0 r}{2^4}}$$

$\geq 1-\varepsilon$

$$= \frac{4}{\pi^2}\,\frac{1}{r}\,(1-\varepsilon) \approx \frac{4}{\pi^2}\,\frac{1}{r}$$

(can be easily made more quantitative, using $\varepsilon < \frac{r}{2^n}$!)

since $s = 0, \ldots, r-1$: Total probability that $\left| \ell - \frac{2^n}{r} s \right| \leq \frac{1}{2}$ for one such s: $p \geq \frac{4}{\pi^2} \approx 0.4$

With sufficiently high probability — we will see that we can <u>check</u> success and thus repeat until we succeed! — we obtain an ℓ s.th. $\ell = \frac{2^n}{r} s + \delta_s$, and thus,

$$\frac{\ell}{2^n} \approx \frac{s}{r},$$

where s is chosen uniformly at random.

Since $r \ll 2^n$ (if chosen suitably), there is only <u>one</u> such ratio $\frac{s}{r}$ with $\left| \ell - \frac{2^n}{r} s \right| \leq 1/2$, and it can be found efficiently.

(See further reading.)

If s and r are co-prime, i.e. $\gcd(r,s) = 1$, we can infer r from $\frac{s}{r}$. This happens with large enough probability. (At least all prime $2 \le s < r$ will do, and the density of primes scales as $\sim 1/\log N$, i.e. the probability that s is prime is roughly

$$p(s \text{ prime}) \ge 1/\log(s_{max}) = 1/\log(2^n/r) \ge \frac{1}{n} :$$

at most a linear number of tries will thus suffice.

— See further reading.)

Once we have used this to obtain a guess for r, we can <u>test</u> whether $f(x) = f(x+r)$, and repeat until success!

$\Rightarrow$ Efficient algorithm for period finding.

$\sim O(n)$ applications of f required!

c) Application: Factoring Algorithm

Factoring: Given $N \in \mathbb{N}$ (not prime), find

$$f \in \mathbb{N},\ f \neq 1,\ \text{such that}\ f \mid N.$$

↑ "f divides N"

(Note: Primality of N can be checked efficiently.)

This can be solved efficiently if we have an efficient method for period finding!

Sketch of algorithm:

(1) Select a random a, $2 \leq a < N$.

If $\gcd(a, N) > 1 \Rightarrow$ done, $f = \gcd(a, N)$!

↑ eff. computable!

Thus: Assume $\gcd(a, N) = 1$.

(2) Denote by r the smallest $x > 0$ such that

$$a^x \bmod N = 1.$$

– that is, the period of

$$f_{N,a}(x) := a^x \bmod N$$

r is called the <u>order of a mod N</u>.

(<u>Note</u>: Some $z > 1$ s.t. $a^z \bmod N = 1$ must exist since

$\exists x, y \in \{1, \dots, N\}$: $a^x \equiv a^y \bmod N$ (counting pigeonholes)

$\Rightarrow a^x \left(1 - a^{y-x}\right) \equiv 0 \bmod N$

$\Rightarrow N \mid \left(a^x \left(1 - a^{y-x}\right)\right)$

$\gcd(a,N)=1$ $\Rightarrow N \mid \left(1 - a^{y-x}\right)$

$\Rightarrow a^{y-x} \equiv 1 \bmod N$ ∎)

Recall: "Efficient" means "polynomial in # of digits of N"

Furthermore, $f_{N,a}(x)$ can be <u>computed efficiently</u>:

Why $x = x_{m-1} 2^{m-1} + x_{m-2} 2^{m-2} + \dots$,

$$a^x \bmod N = \underbrace{\left(a^{(2^{m-1})}\right)^{x_{m-1}} \cdot \left(a^{(2^{m-2})}\right)^{x_{m-2}} \cdot \ldots}_{\text{eff. computable via repeated squaring mod } N:} \bmod N$$

$$a \mapsto a^2 \bmod N \mapsto \underbrace{a^4 \bmod N}_{\equiv (a^2 \bmod N)^2 \bmod N} \mapsto \ldots,$$

by doing "mod N" in each step the numbers don't require an exp. number of digits:

$O(n)$ multiplications of n-digit numbers.

$\Rightarrow$ r can be <u>found efficiently</u> with a <u>quantum computer</u>!

(3) Assume for now <u>r even</u>:

$$a^r \bmod N = 1$$

$$\Leftrightarrow N \mid (a^r - 1),$$

$$\Leftrightarrow N \mid (a^{r/2} + 1)(a^{r/2} - 1)$$

However, we also know that $N \nmid (a^{r/2} - 1)$,

since otherwise $a^{r/2} \bmod N = 1$ ↯ (does not divide)

$\Rightarrow$ either $N \mid a^{r/2} + 1$

or N has non-trivial common factors with both $a^{r/2} \pm 1$.

$\Rightarrow 1 \neq f := \gcd(N, a^{r/2}+1) \mid N$

$\Rightarrow$ found a non-trivial factor f of N!

$\Rightarrow$ Algorithm will succeed as long as

(i) r even

(ii) $N \nmid (a^{r/2}+1)$

This can be shown to happen with prob. $\geq 1/2$ for a random choice of a (see further reading)

- unless either N is even

(can be checked efficiently),

or $N = p^k$, p prime

(can also be checked efficiently by factoring roots; there are only $O(\log(N))$ roots which one has to check!)

- and in both cases, this gives a non-trivial factor!

$\Rightarrow$ efficient Quantum Algorithm for Factoring.

"Shor's algorithm"