

naive algorithm: divide by prime
numbers $\leq \sqrt{N}$

to factor $\sim 10^{100}$
 $\sqrt{10^{100}} \approx 10^{50}$

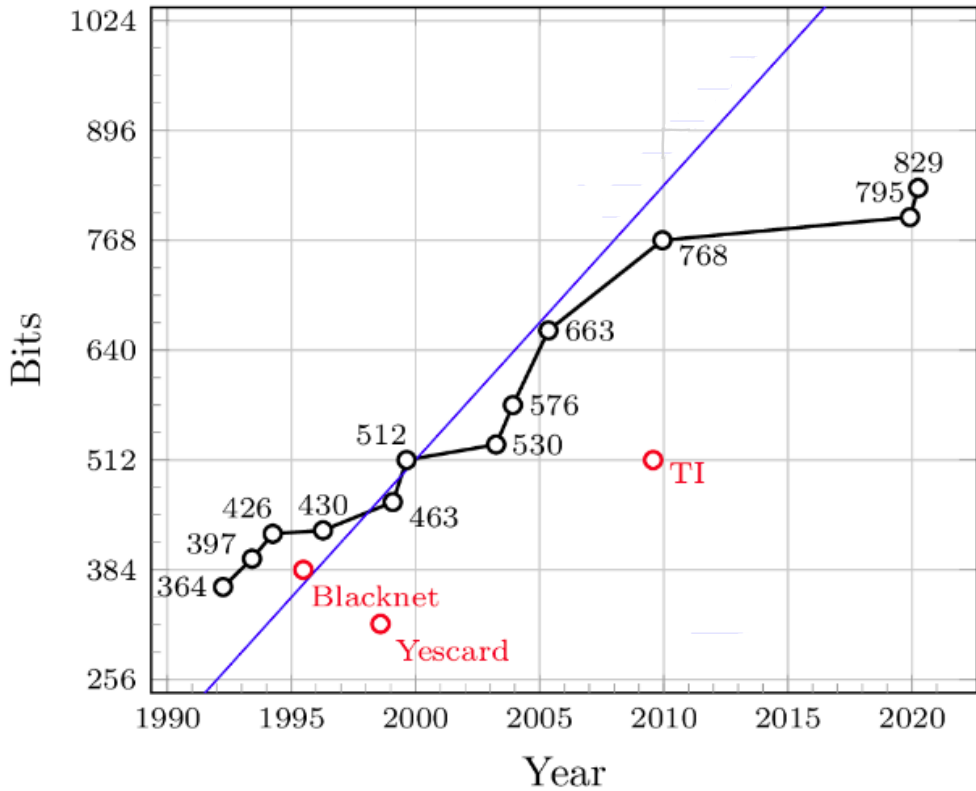
at 10^{-9} /sec takes 10^{41} sec

or since sec/year $\sim \pi \cdot 10^7$

that's $\sim 3 \cdot 10^{33}$ years 3.16

age of universe $\sim 10^{10}$ years

best algorithm $\sim 2^{n^{1/3}}$
 $n = \log_2 N$



RSA numbers factored

768 bits = 232 decimal digits

795 bits = 240 " "

829 bits = 250 " "

Quantum Physics

*[Submitted on 23 May 2019 (v1), last revised 5 Dec 2019 (this version, v2)]***How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits**

Craig Gidney, Martin Ekerå

We significantly reduce the cost of factoring integers and computing discrete logarithms in finite fields on a quantum computer by combining techniques from Shor 1994, Griffiths–Niu 1996, Zalka 2006, Fowler 2012, Ekerå–Håstad 2017, Ekerå 2017, Ekerå 2018, Gidney–Fowler 2019, Gidney 2019. We estimate the approximate cost of our construction using plausible physical assumptions for large-scale superconducting qubit platforms: a planar grid of qubits with nearest-neighbor connectivity, a characteristic physical gate error rate of 10^{-3} , a surface code cycle time of 1 microsecond, and a reaction time of 10 microseconds. We account for factors that are normally ignored such as noise, the need to make repeated attempts, and the spacetime layout of the computation. When factoring 2048 bit RSA integers, our construction's spacetime volume is a hundredfold less than comparable estimates from earlier works (Fowler et al. 2012, Gheorghiu et al. 2019). In the abstract circuit model (which ignores overheads from distillation, routing, and error correction) our construction uses $3n + 0.002n \lg n$ logical qubits, $0.3n^3 + 0.0005n^3 \lg n$ Toffolis, and $500n^2 + n^2 \lg n$ measurement depth to factor n -bit RSA integers. We quantify the cryptographic implications of our work, both for RSA and for schemes based on the DLP in finite fields.

Dec 2009: 768 bits, 2000 core years

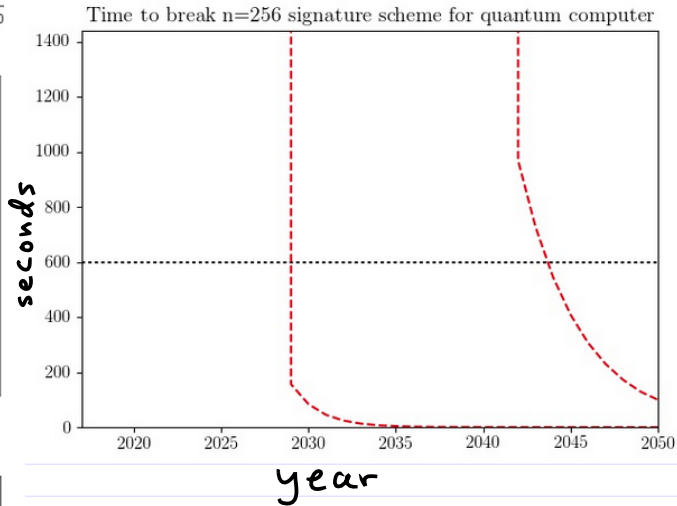
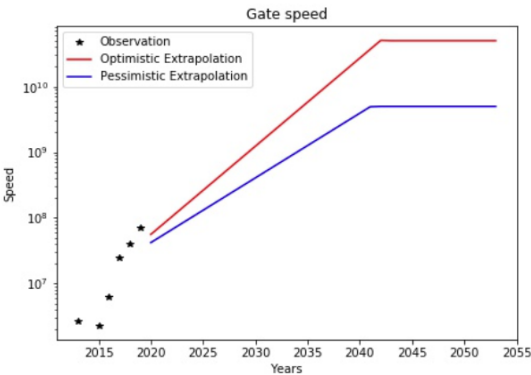
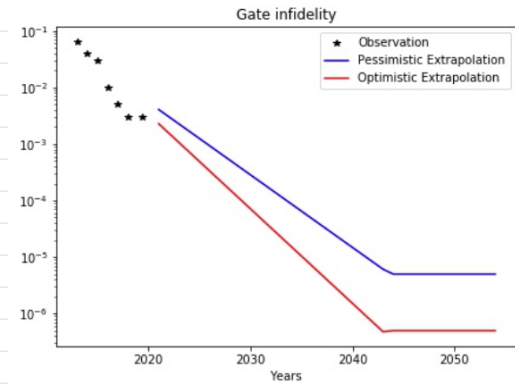
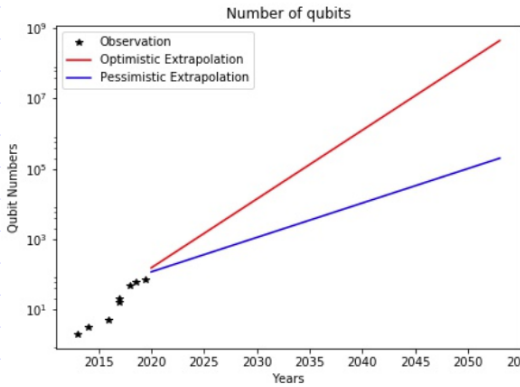
better algorithms:

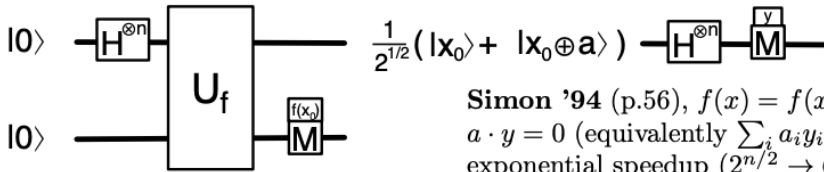
Dec 2019: 795 bits took 900 core years
[Intel Xeon Gold 2.1 GHz CPUs]

Feb 2020: 829 bits took 2700 core years

⇒ 2048 bits would take $>10^{10}$ x more

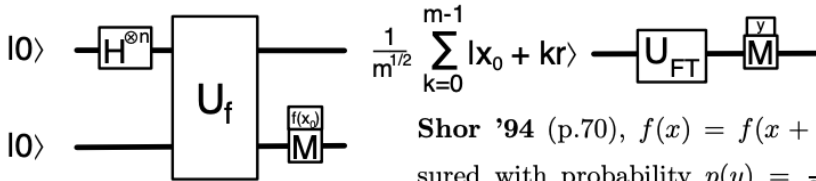
(and
arXiv: 1710.10377)





$$\frac{1}{\sqrt{2}} (|x_0\rangle + |x_0 \oplus a\rangle)$$

Simon '94 (p.56), $f(x) = f(x \oplus a)$, measured y has $a \cdot y = 0$ (equivalently $\sum_i a_i y_i = 0 \pmod{2}$), exponential speedup ($2^{n/2} \rightarrow O(n)$) to determine a



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

Shor '94 (p.70), $f(x) = f(x + r)$, resulting y is measured with probability $p(y) = \frac{1}{2^m} \left| \sum_{k=0}^{m-1} e^{2\pi i k r y / 2^n} \right|^2$, gives $|y - 2^n/r| < 1/2$ with $p > .4$, sufficient to determine

period r via partial fraction expansion, exponential speedup ($n2^n, \exp(n^{1/3}) \rightarrow O(n^3)$).

(Note: replaces $H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{0 \leq y < 2^n} e^{i\pi x \cdot y} |y\rangle$ with $U_{FT}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{0 \leq y < 2^n} e^{2\pi i x y / 2^n} |y\rangle$.)

Practical application is $f(x) \equiv b^x \pmod{N}$, where $b \equiv a^c \pmod{N}$ is an encrypted message, from which d' , satisfying $cd' \equiv 1 \pmod{r}$, can be calculated, and d' recovers unencrypted message $a \equiv b^{d'} \pmod{N}$ (in contrast to using d , with $cd = 1 \pmod{(p-1)(q-1)}$, where $N = pq$ and r divides $(p-1)(q-1) = |G_{pq}|$).

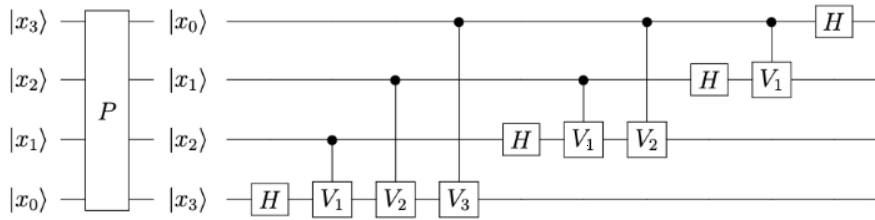
$$H^{\otimes n} |x\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{0 \leq y < 2^n} e^{\pi i x \cdot y} |y\rangle_n$$

↖ bitwise inner prod

$$U_{FT} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{0 \leq y < 2^n} e^{2\pi i x y / 2^n} |y\rangle_n$$

↖ arithmetic multiplication

permutes



$$\frac{1}{2^2} \sum_{0 \leq y < 2^4} e^{2\pi i xy / 2^4} |y\rangle$$

What about U_f for the function $f(x) = b^x \bmod N$

$x = 11_{10} = 1011_2$ $b^{11} = b^8 \cdot b^2 \cdot b^1$, so calculate

$b^1, b^2, b^4, b^8, b^{16}, b^{32}, b^{64}, \dots \bmod N$

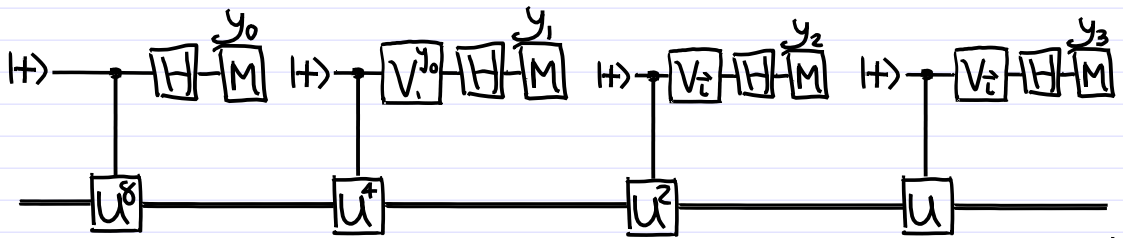
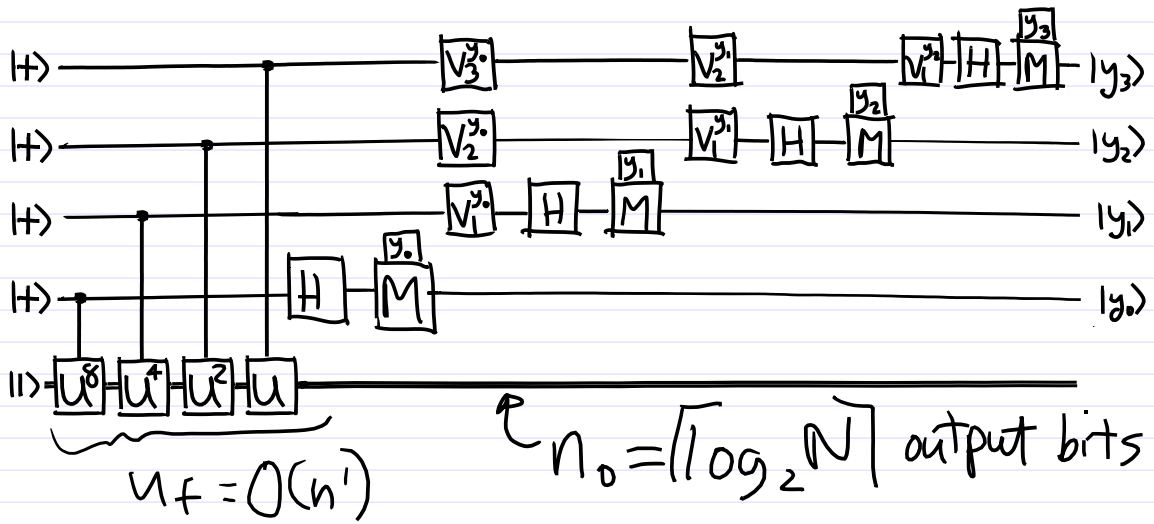
$$X = X_{n-1} X_{n-2} \dots X_1 X_0$$

$$b^X = b^{X_{n-1} 2^{n-1} + X_{n-2} 2^{n-2} + \dots + X_1 \cdot 2 + X_0}$$

$$= \prod_j (b^{2^j})^{X_j}$$

number of powers to calculate only grows logarithmically in x .

Calculate by successive squares



With "recycling"
 total qubits
 $\sim 2-3 n_0$

$$|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

U implements multiplication by $b \pmod N$ and
 $U_f |x\rangle_n |0\rangle_{n_0} = |x\rangle_n |b^x \pmod N\rangle_{n_0}$

Euclidean Algorithm

$$\text{GCD}(60, 40) = \text{GCD}(40, 20) = 20$$

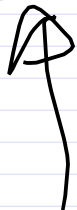
$$\text{GCD}(200, 60) = \text{GCD}(60, 20) \rightarrow 20$$

$$\text{Ex: } (62, 42) \rightarrow (42, 20) \rightarrow$$

$$(f, c) \rightarrow (c, f \bmod c)$$

$\rightarrow \dots$

$$(60, 7) \rightarrow (7, 4) \rightarrow (4, 3) \rightarrow (3, 1)$$



no
Common
divisor

How do we use this to
calculate inverse
 $7^{-1} \pmod{60}$?

$$\begin{aligned} 1 &= 4 - 3 = 4 - (7 - 4) \\ &= 2 \cdot 4 - 7 \\ &= 2(60 - 8 \cdot 7) - 7 \\ &= 2 \cdot 60 - 17 \cdot 7 \end{aligned}$$

$$7^{-1} \pmod{60} = -17 = 43$$

$$43 \cdot 7 = 301 \pmod{60} = 1 \quad \checkmark$$

output bits $n_0 = \lceil \log_2 N \rceil$ (round up)

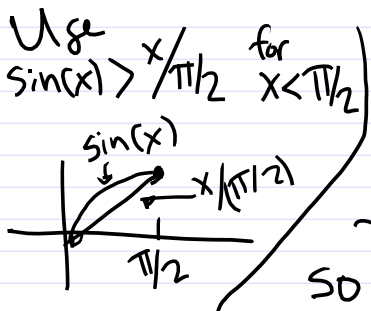
$n = 2n_0$ input bits to ensure many periods:
i.e., m large

e.g. N with 500 digits, $n_0 \approx 1700$,

input bits $n = 2n_0 \approx 3400$

Will want $y = y_j = j \frac{2^n}{r} + \delta_j$ $\delta_j \leq \frac{1}{2}$

$$P(y_j) = \frac{1}{2^{n/m}} \frac{\sin^2(\pi \delta_j m r / 2^n)}{\sin^2(\pi \delta_j r / 2^n)} \approx \frac{1}{r} \frac{\sin^2 \pi \delta_j}{(\pi \delta_j)^2}$$



$$\geq \frac{1}{r} \frac{4}{\pi^2}$$

using $m \approx 2^n/r \gg 1$

$\sim r$ different values of j ,
so total prob of measuring y within

$1/2$ of $2^n/r$ is $> 4/\pi^2 \approx 0.405$.

[more refined estimate $\Rightarrow > 90\%$]

To extract j/r reliably from $y=2^n/r$
 need $2^n \gg 2^{n_0} \approx N$. Why? :

we have $|y - j \frac{2^n}{r}| < 1/2$ with high prob

$$\Rightarrow \left| \frac{y}{2^n} - j/r \right| < \frac{1}{2^{n+1}} \leq \frac{1}{2N^2}$$

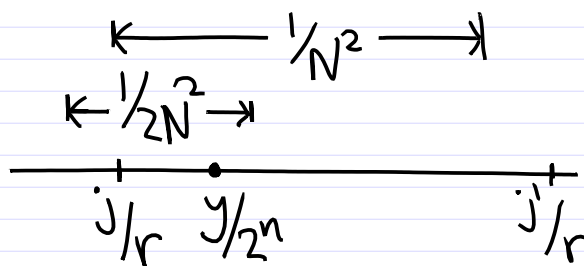
and want

hence $n=2n_0$

If there were some other j'/r' , then $|j/r - j'/r'| > 1/N^2$,

since $r, r' < N$ and $\left| \frac{a}{b} - \frac{c}{d} \right| > \frac{1}{bd}$ for integers a, b, c, d

so picture looks like this:



any other j'/r'
 is far enough away
 that j/r is uniquely
 specified by $y/2^n$
 (up to common factors in j, r)

Continued fraction,
 write any number as $a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots}}}$
 $= [a_0; a_1, a_2, a_3, \dots]$

$$\pi = 3.1415926536\dots = 3 + \frac{1}{7 + \frac{1}{15 + \frac{1}{1 + \frac{1}{292 + \dots}}}}$$

$$\frac{1}{.14159\dots} = 7.06251330542\dots$$

$$\frac{1}{.0625\dots} = 15.9965945\dots$$

$$\frac{1}{.9965\dots} = 1.003417099\dots$$

$$\frac{1}{.003\dots} = 292.64\dots$$

$$\pi = (3; 7, 15, 1, 292, \dots)$$

partial sums =

$$3, \frac{22}{7}, \frac{333}{106}, \frac{355}{113}, \dots, \dots$$

$$\frac{355}{113} = \underline{3.14159292\dots}$$

Known in 5th c. AD.
 to Chinese (Tsu)

Some continued fractions

Golden mean:

$$\varphi = \frac{1 + \sqrt{2}}{2} = 1.618\dots = [1; \overline{1}]$$

$$\sqrt{2} = [1; \overline{2}]$$

$$\sqrt{3} = [1; \overline{1, 2}]$$

$$e = [2; 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, \dots]$$

...

$$\varphi \text{ satisfies } \varphi - 1 = \frac{1}{\varphi}$$

$$\sqrt{2} \text{ satisfies } x - 1 = \frac{1}{1+x}$$

...

Grover's Algorithm

Search.

Given N items, 1 "marked"

look at k of them

prob = k/N of finding it.

Quantum: "look" at $\sim \sqrt{N}$

e.g. Database

$$\text{or } p = m^2 + n^2 \quad 44 + 1$$

$$\sim \sqrt{p/2}, \quad \sqrt{p}$$

$$f(x) = \begin{cases} 0 & x \neq a \\ 1 & x = a \end{cases} \leftarrow \text{marked item}$$

$$U_f |x\rangle_n |y\rangle_1 = |x\rangle |y \oplus f(x)\rangle_1 \quad N=2^n$$

"phase
kickback"

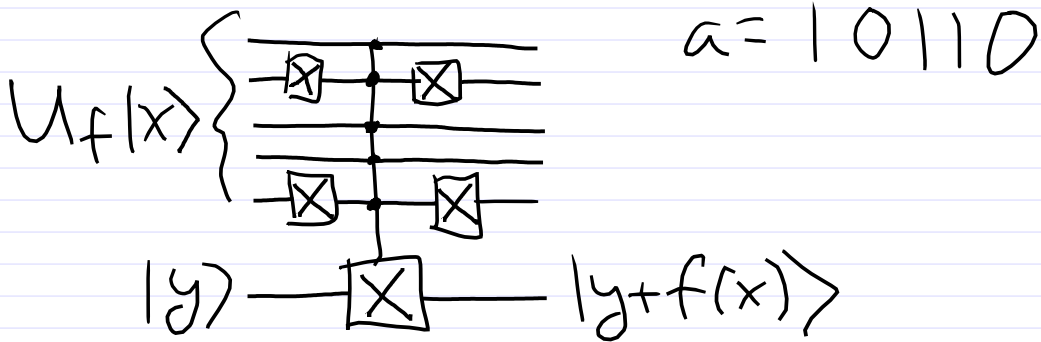
$$\begin{array}{ccc} |x\rangle & \text{---} & |x\rangle \\ |1\rangle & \text{---} \boxed{H} & (-1)^{f(x)} H |1\rangle \end{array} \quad \left[U_f \right]$$

$$U_f (|x\rangle \otimes H|1\rangle) = (-1)^{f(x)} (|x\rangle \otimes H|1\rangle)$$

$$V(x) |x\rangle_n = (-1)^{f(x)} |x\rangle_n = \begin{cases} |x\rangle & x \neq a \\ -|a\rangle & x = a \end{cases}$$

$$V |\psi\rangle = |\psi\rangle - 2|a\rangle \langle a|\psi\rangle$$

$$V_f = \underline{1} - 2|a\rangle \langle a| \quad \text{embodies } f$$



$$|\varphi\rangle = H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{0 \leq x < 2^n} |x\rangle$$

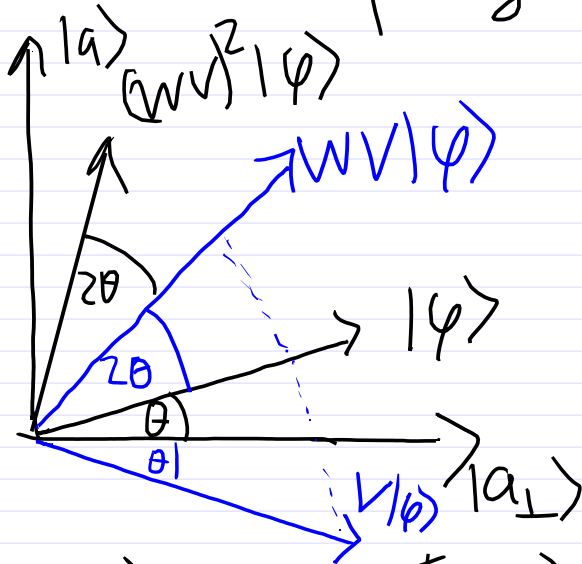
also need

$$W = 2|\varphi\rangle\langle\varphi| - 1$$

inverts states orthogonal to $|\varphi\rangle$

$$V = \underline{1} - 2|a\rangle\langle a|$$

work in 2d space generated by $|a\rangle, |\psi\rangle$



$$\langle a | \psi \rangle = \cos(\pi/2 - \theta) = \sin \theta = \frac{1}{2^{n/2}} \\ \approx \theta \approx \frac{1}{\sqrt{N}}$$

WV is a rotation (by what angle?)
by 2θ

$$\text{(or } V = \begin{pmatrix} 1 & \\ & -1 \end{pmatrix} \quad W = R_\theta V R_{-\theta} \quad R_\theta = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \\ WV = R_\theta V R_{-\theta} V = R_{2\theta} \text{)}$$

Apply $(WV)^l$ $\theta \approx \frac{1}{2^{n/2}}$

with $(2l+1)\frac{1}{2^{n/2}} = \frac{\pi}{2}$

gives close as possible to $|a\rangle$

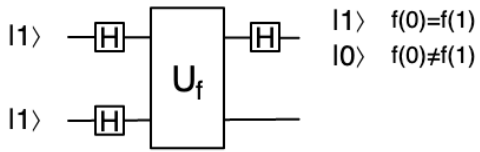
$$l \approx \frac{\pi}{4} 2^{n/2} = \frac{\pi}{4} \sqrt{N}$$

$$p(a) = |\langle a | (WV)^l | \varphi \rangle|^2$$

$$= \sin^2(2l+1)\theta = \sin^2\left(\frac{2l+1}{2^{n/2}}\right)$$

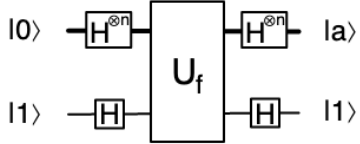


1.



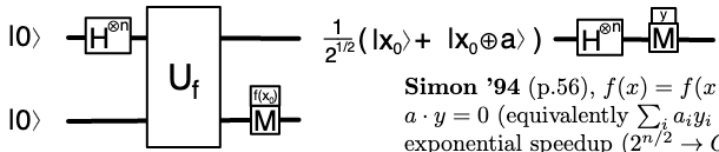
Deutsch '92 (p.44), factor of 2 speedup to determine whether or not 1bit→1bit function $f(x)$ is constant

2.



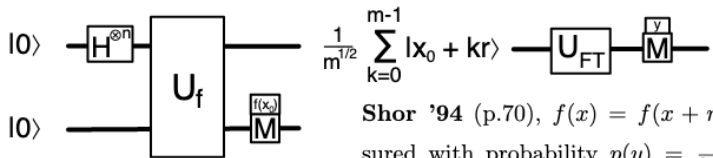
Bernstein-Vazirani '93 (p.52), $f(x) = a \cdot x \equiv \oplus_i a_i x_i$, factor of n speedup to determine a

3.



Simon '94 (p.56), $f(x) = f(x \oplus a)$, measured y has $a \cdot y = 0$ (equivalently $\sum_i a_i y_i = 0 \pmod{2}$), exponential speedup ($2^{n/2} \rightarrow O(n)$) to determine a

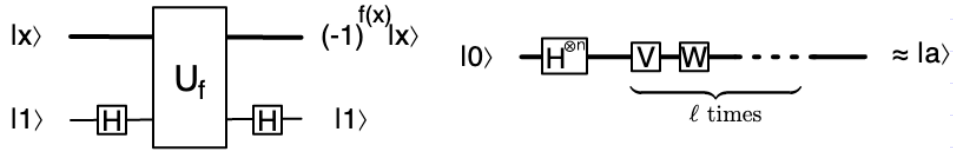
4.



Shor '94 (p.70), $f(x) = f(x + r)$, resulting y is measured with probability $p(y) = \frac{1}{2^{2n}} \left| \sum_{k=0}^{m-1} e^{2\pi i k r y / 2^n} \right|^2$, gives $|y - 2^n/r| < 1/2$ with $p > .4$, sufficient to determine

period r via partial fraction expansion, exponential speedup ($n2^n, \exp(n^{1/3}) \rightarrow O(n^3)$).
 (Note: replaces $\mathbf{H}^{\otimes n} |x\rangle = \frac{1}{2^{n/2}} \sum_{0 \leq y < 2^n} e^{i\pi x \cdot y} |y\rangle$ with $\mathbf{U}_{FT} |x\rangle = \frac{1}{2^{n/2}} \sum_{0 \leq y < 2^n} e^{2\pi i x y / 2^n} |y\rangle$.)
 Practical application is $f(x) \equiv b^x \pmod{N}$, where $b \equiv a^c \pmod{N}$ is an encrypted message, from which d' , satisfying $cd' \equiv 1 \pmod{r}$, can be calculated, and d' recovers unencrypted message $a \equiv b^{d'} \pmod{N}$ (in contrast to using d , with $cd = 1 \pmod{(p-1)(q-1)}$, where $N = pq$ and r divides $(p-1)(q-1) = |G_{pq}|$).

5.



Grover '96 (p.90), $f(x) = 1$ only for (m) marked value(s) $x = a$, uses "phase kickback" to express \mathbf{U}_f in terms of $\mathbf{V} = \mathbf{1} - 2|a\rangle\langle a|$, and $\mathbf{W} = 2|\phi\rangle\langle\phi| - \mathbf{1} = \mathbf{H}^{\otimes n} (2|0\rangle\langle 0| - \mathbf{1}) \mathbf{H}^{\otimes n}$ is easily constructed. Applying $\ell \approx \frac{\pi}{4} \frac{2^{n/2}}{\sqrt{m}}$ times gives probability $p(a) \approx 1 - O(m/2^n)$, for square-root speedup ($2^n/m \rightarrow \sqrt{2^n/m}$).