IV. Quantum Computation

1. The circuit model

a) Classical computation

Task of class. computers:

Solve problems ≡ compute functions:

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

$$\underline{x} = (x_1, \dots, x_n) \longrightarrow f(x_1, \dots, x_n)$$

f depends on problem, $\underline{x}$ encodes instance of problem.

E.g.: Multiplication: $(a,b) \mapsto a \cdot b$

$$\underline{x} = (\underbrace{\underline{x}_1, \underline{x}_2}_{\text{binary enc. input}}) \Rightarrow f(\underline{x}) = \underbrace{\underline{x}_1 \cdot \underline{x}_2}_{\text{in binary}}$$

Factorization:

$\underline{x}$: integer; $f(\underline{x})$: list of prime factors (w/ suitable encoding)

Each problem is encoded by a family of functions

$f = f^{(n)}: \{0,1\}^n \to \{0,1\}^m$; $m = \text{poly}(n)$, $n \in \mathbb{N}$

Must be possible to "construct f systematically & efficiently"
(→ later!)

What ingredients do we need to compute a general function f?

(i) $f: \{0,1\}^n \to \{0,1\}^m$

$$f(\underline{x}) = (f_1(\underline{x}), f_2(\underline{x}), \ldots, f_m(\underline{x}))$$

$f_k: \{0,1\}^n \to \{0,1\}$

⇒ can restrict to Boolean functions $f: \{0,1\}^n \to \{0,1\}$.

(ii) Let $L = \{\underline{y} \mid f(\underline{y}) = 1\} = \{\underline{y}^1, \underline{y}^2, \ldots, \underline{y}^k\}$

Define $g_{\underline{y}}(\underline{x}) = \begin{cases} 0 & ; \underline{x} \neq \underline{y} \\ 1 & ; \underline{x} = \underline{y} \end{cases}$

$$f(\underline{x}) = g_{\underline{y}^1}(\underline{x}) \vee g_{\underline{y}^2}(\underline{x}) \vee \ldots \vee g_{\underline{y}^k}(\underline{x})$$

"$\vee$": logical "or": $0 \vee 0 = 0$, $0 \vee 1 = 1$, $1 \vee 0 = 1$, $1 \vee 1 = 1$

associative

⇒ $a \vee b \vee c = a \vee (b \vee c)$ etc!

(iii) $g_y(\underline{x}) = (y_1 = x_1) \wedge (y_2 = x_2) \wedge \dots$

"$\wedge$": logical "and": $1 \wedge 1 = 1$

otherwise $x \wedge y = 0$

(iv) $(y_i = x_i) = \begin{cases} x_i, & y_i = 1 \\ \neg x_i, & y_i = 0 \end{cases}$

"$\neg$": logical "not": $\neg 1 = 0$

$\neg 0 = 1$

$\Rightarrow$ $f(\underline{x})$ can be built from 4 predicates:

"and", "or", "not" gates, + copy gate $x \mapsto (x, x)$.

"Universal gate set".

(Note: In fact, either $\neg(x \wedge y)$ "nand" or $\neg(x \vee y)$ "nor", together w/ copy, are already universal!)

Circuit model of computation:

1 built from universal gate set (without "loops such in time").

Which problems can we solve efficiently w/ this model?

⇒ Problems where # gates (= # "time steps") scales nicely with n, i.e. as some polynomial $\text{poly}(n)$. (Class "P" of problems.)

Otherwise: hard problem.

Is a typical problem easy or hard?

$$f: \{0,1\}^n \to \{0,1\}$$

different f: $2^{(2^n)}$ ← # inputs

(on each input: 0 or 1)

But only $c^{\text{poly}(n)}$ circuits of length $\text{poly}(n)$

↑ # elementary gates

⇒ most f cannot be computed efficiently.

Note: Also require f to be a uniform family of circuits: can e.g. be generated by a computer program from input n in time $\text{poly}(n)$ for any n.

Does comp. power dep. on gate set?

→ No. By def., any universal gate set can simulate any other w/ constant overhead; same power!

What about other models of computation?

- CFG
- parallel computer
- Turing machine (tape based)
- Cellular automata

⋮ (other exotic models...)

⇒ All known "reasonable" models can simulate each other w/ poly(n) overhead ⇒ same comput. power!

Church-Turing-Thesis: All reasonable models of computation have the same computational power.

(Note: different variants; randomness? quantum ("strong" C-T-Thesis))

Will use circuit model to go to quantum systems:

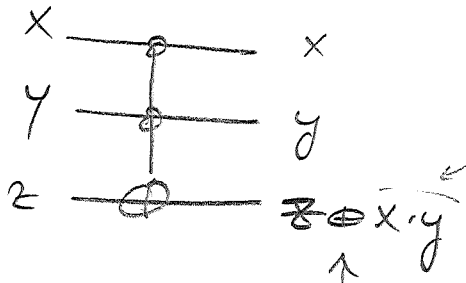
Gates → Unitaries

Rmk: Class. gates irreversible ⇔ unitaries reversible.

Can we fit even class. comput. in this picture?

Classical computation can be turned reversible:

Toffoli gate:



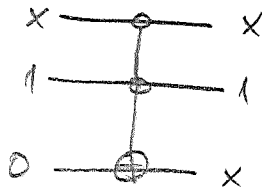
$x, y, z \in \{0,1\}$

sum mod 2 = "XOR"

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

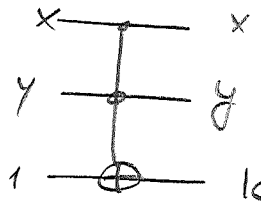
→ reversible!

→ can simulate and/or/not/copy using ancillas, e.g.:



COPY

;



⇒ reversible univ. gate (using ancillas)

Any $f(x)$ can be computed reversibly:

$$f^R : (x, y) \longmapsto (x, f(x) \oplus y)$$

($\oplus$ ← bitwise XOR.)

- possibly w/ ancillas – w/out changing anything else.

(Idea: Compute f reversibly w/ ancillas, XOR result w/ y, and "un-compute" everything – clean ancillas.
– Can be optimized to use few ancillas! → Prakash)

⇒ Everything can be computed reversibly.

But: 3-bit gate needed (→ HW)

5) Quantum Circuits

Model of quantum computation:

- System consists of qubits (or d-its): tensor prod. structure
- Choose universal gate set

$$S = \{U_1, \ldots, U_k\} \text{ of "small" (i.e. few-qubit) gates}$$

⇒ Small (poly-size) circuits.

- Input: Classical input $|x_1\rangle|x_2\rangle \dots |x_n\rangle$

  in computational basis.

- "Output": Measure all qubits at the end in the "computational basis" $\{|0\rangle, |1\rangle\}$.

  (Note: Other measurements do not help: can use unit. gate set to do any POVM.

  - Not meas. gubits = tracing out = meas. and ignoring result.
  - Meas. at earlier time: can be postponed until end (no difference!)

)

Which gate set should we choose?

- Continuum of gates: much more rich!
- turns out: 1+2 qubit gate sufficient (unlike classical!)
- Almost all 2-qubit-gates are universal by themselves (in an appropr. sense, if involved phases incommensurate)

Our choice:

(i) 1-qubit rotations about x & z axis:

$$R_x(\phi) = e^{-i X \phi/2} \quad ; \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \; ; \; X^2 = I$$

$$R_z(\phi) = e^{-i Z \phi/2} \quad ; \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \; ; \; Z^2 = I$$

For $\pi^2 = I$: $e^{i\pi\phi/2} = \cos\phi/2 \, I + i \sin\phi/2 \, \pi$

$$\rightarrow R_x(\phi) = \begin{pmatrix} \cos\phi/2 & -i\sin\phi/2 \\ -i\sin\phi/2 & \cos\phi/2 \end{pmatrix};$$

$$R_z(\phi) = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{+i\phi/2} \end{pmatrix}$$

$\Rightarrow$ Can generate any rotation $U \in SU(2)/\mathbb{Z}_2 \cong SO(3)$
(→ Euler angles!)

$$U = R_x(\alpha) R_z(\beta) R_x(\gamma)$$

(ii) one 2-qubit gate (almost all are ok...)

Typ. we choose "controlled-NOT"

$$CNOT = \quad x \to x, \quad y \to x \oplus y \quad = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$\Rightarrow$ flips y iff $x=1$: Classical gate!

Turns out: Univ. gate set can be used to exactly create any n-qubit gate. (Of course not eff.: there are "even more" n-qubit unitaries than n-bit functions.)

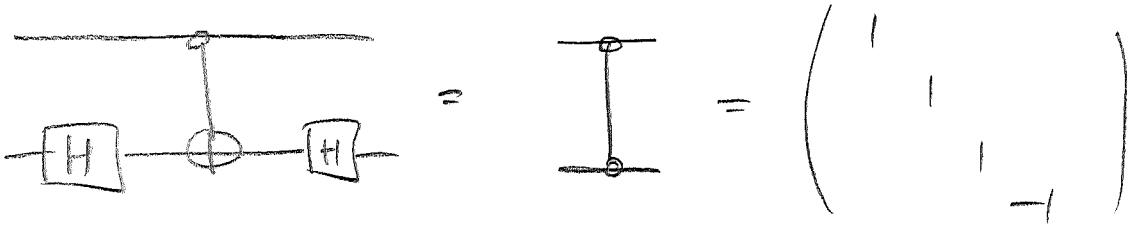
Some important gate sets + identities: ($\rightarrow$ HW):

Hadamard gate: $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$; $H = H^\dagger$

$$H R_x(\phi) H = R_z(\phi)$$

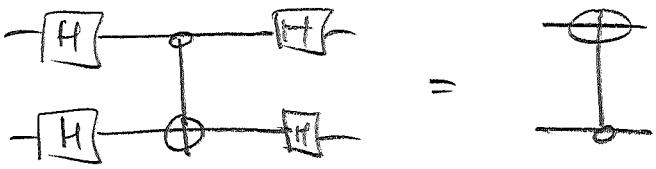
$$H R_z(\phi) H = R_x(\phi)$$

Graphical notation:

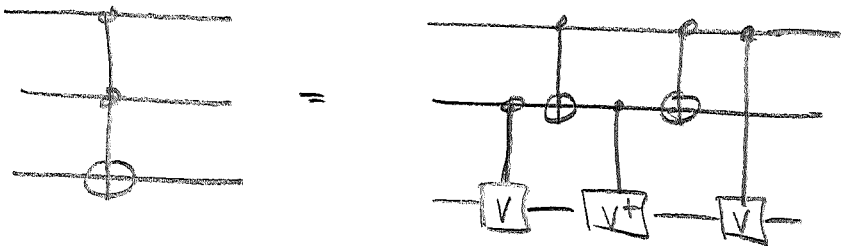


"Controlled-Z"

"Controlled-Phase", CPHASE



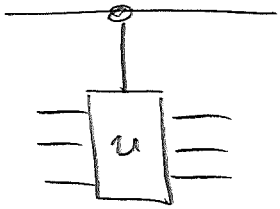
Toffoli:



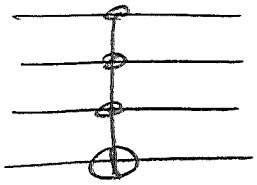
with $V = \frac{1-i}{2}(I + iX)$, and  = $\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & V \end{array}\right)$

"controlled -V"

Moreover: If we know how to build any classical U, we can also build "controlled - U":



→ just replace every Toffoli (class. univ.!) by Toffoli w/ 3 controls:



: can be built from normal Toffoli (→ HW)

What are other univ. gate sets? → Many!

Note: Different notions of universality:

- exact univ.: any n-qubit U can be realized exactly
- approx. univ.: any n-qubit U can be approx. by gate set (w/ reasonable conv.?)

Examples of approx. univ. gate sets:

- CNOT + 2 random 1-qubit gates
- CNOT + H + $T = R_z(\pi/4)$ ("$\pi/8$-gate")
- most 2-qubit gates alone

Solovay-Kitaev-Thm: A universal gate set for $SU(2^n)$ can approximate any $U \in SU(2^n)$ up to ε with $O(\text{poly}(\log(1/\varepsilon)))$ gates.

2. Oracle-based algorithms

a) The Deutsch algorithm

Consider $f: \{0,1\} \to \{0,1\}$.

Let f be "very hard" to compute (e.g. by circuit).

Want to know: Is $f(0) \stackrel{?}{=} f(1)$?

How often do we have to evaluate f?

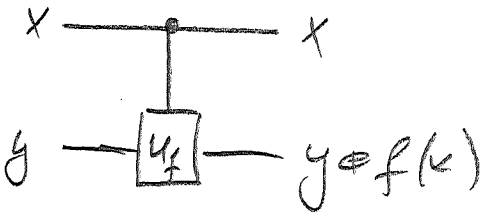
(Treat f as "black box" = "oracle": how many queries to oracle?)

→ Classically: 2 queries: $f(0)$, $f(1)$.

Can Q.M. help?

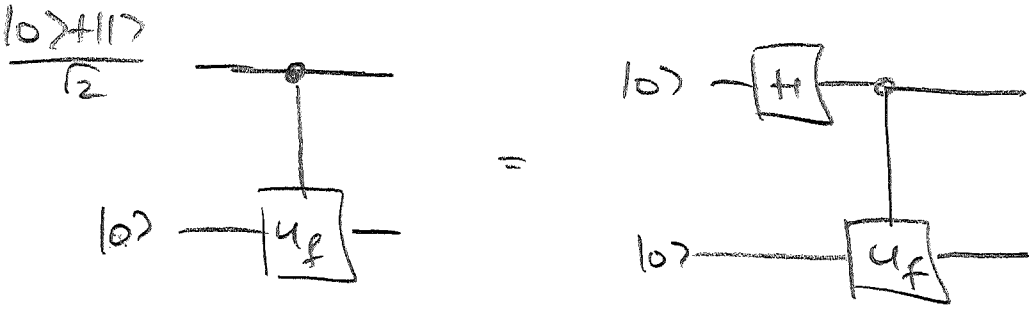
Consider reversible implementation of f:

$$f^R : (x,y) \longmapsto (x, y \oplus f(x))$$



$: |x\rangle|y\rangle \longmapsto |x\rangle|y \oplus f(x)\rangle$

→ try to use superpositions?



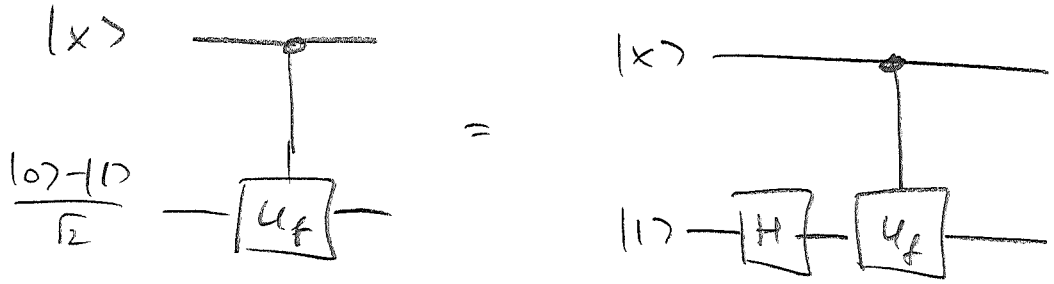
$$\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle + |1\rangle|0\rangle\right) \xrightarrow{U_f} \frac{1}{\sqrt{2}}\left(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle\right)$$

$\Rightarrow$ Have evaluated f on both inputs!

But: how can we extract (relevant) information?

- Meas. of qubit 1: collapse superposition!
- Meas. of qubit 2: ?

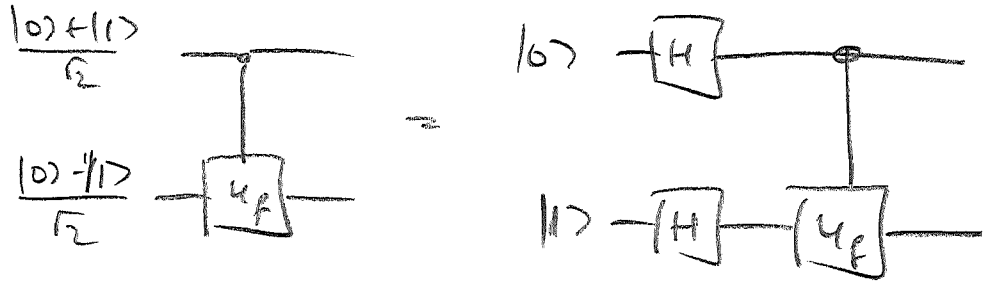
Consider



$$|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \longmapsto |x\rangle\left(\frac{|f(x)\rangle - |1\oplus f(x)\rangle}{\sqrt{2}}\right) =$$

$$= \left\{\begin{array}{ll} \underline{f(x)=0}: & |x\rangle\,\dfrac{|0\rangle-|1\rangle}{\sqrt{2}} \\ \underline{f(x)=1}: & |x\rangle\,\dfrac{|1\rangle-|0\rangle}{\sqrt{2}} \end{array}\right\} = |x\rangle\cdot\left[(-1)^{f(x)}\,\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right].$$

Combine:



$$|0\rangle|1\rangle \xrightarrow{H\otimes H} \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \xrightarrow{U_f} \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

→ no entanglement created (!)

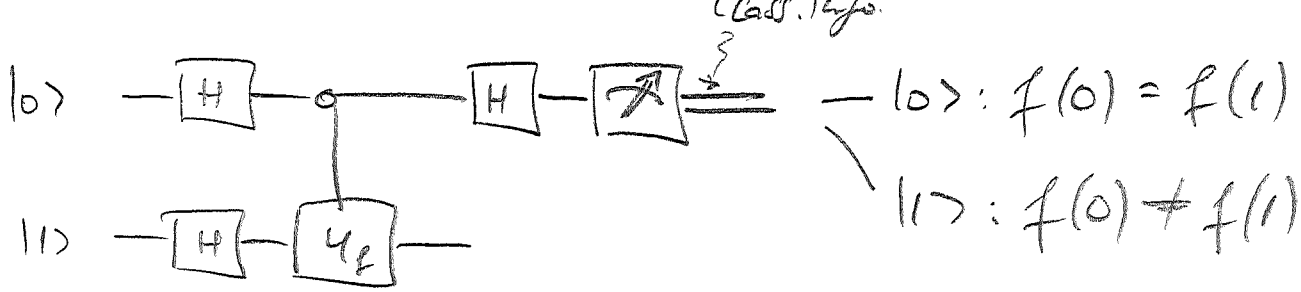
→ 2nd qubit unchanged (!!)

→ 1st qubit gets phase $(-1)^{f(x)}$

⇒ "phase kick-back" technique

⇒ 1st qubit = $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$: $f(0)=f(1)$

$= \frac{|0\rangle-|1\rangle}{\sqrt{2}}$: $f(0)\neq f(1)$.



⇒ One application of U_f sufficient!

→ factor 2 faster than classically!

Note: 2nd qubit need not be measured (and contains no information!)

Two core ideas:

- Use input $\sum |x\rangle$ to evaluate f on all inputs simultaneously
- Need way to read out relevant info!

b) The Deutsch-Jozsa algorithm

Consider $f: \{0,1\}^n \to \{0,1\}$ w/ promise that

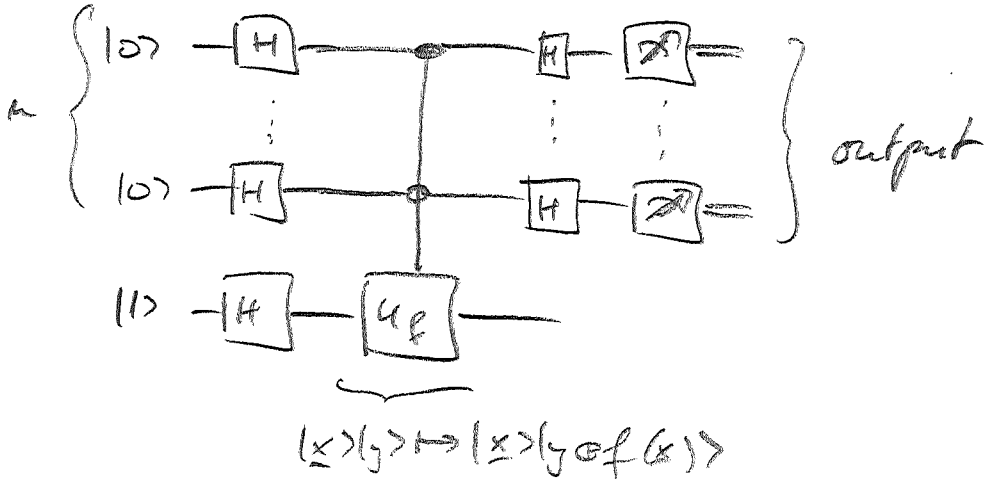
either $f(x) = c \ \forall x$ "constant"

or $\#\{x \mid f(x)=0\} = \#\{x \mid f(x)=1\}$ "balanced"

Want to know: Is f constant or balanced?

→ How many queries to f do we need?

Use same idea: Input $\sum |x\rangle$ and $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$:



What is action of $H^{\otimes n}$?

$$H: |x\rangle \longmapsto \frac{1}{\sqrt{2}} \sum (-1)^{x \cdot y} |y\rangle$$

$$H^{\otimes n}: |x_1, \dots, x_n\rangle \longmapsto \frac{1}{\sqrt{2^n}} \sum (-1)^{x_1 y_1} (-1)^{x_2 y_2} \cdots |y_1, \dots, y_n\rangle$$

$$|\underline{x}\rangle \longmapsto \frac{1}{\sqrt{2^n}} \sum (-1)^{\underline{x} \cdot \underline{y}} |\underline{y}\rangle$$

$$(\underline{x} \cdot \underline{y} := x_1 y_1 \oplus x_2 y_2 \oplus \dots \oplus x_n y_n)$$

$$\Rightarrow |\underline{0}\rangle|1\rangle \xmapsto{H^{\otimes n} \otimes H} \text{(omit prefactors!)} \left(\sum |\underline{x}\rangle\right)(|0\rangle - |1\rangle)$$

$$\xmapsto{U_f} \left(\sum_{\underline{x}} (-1)^{f(\underline{x})} |\underline{x}\rangle\right)(|0\rangle - |1\rangle)$$

$$\xmapsto{H^{\otimes n} \otimes I} \left(\sum_{\underline{y}} \underbrace{\sum_{\underline{x}} (-1)^{f(\underline{x}) + \underline{x} \cdot \underline{y}}}_{\circledast} |\underline{y}\rangle\right)(|0\rangle - |1\rangle)$$

f constant: $\circledast = (-1)^{f(\underline{x})} \cdot \underbrace{\sum_{\underline{x}} (-1)^{\underline{x} \cdot \underline{y}}}_{\delta_{\underline{y},\underline{0}}} = (-1)^{f(\underline{x})} \delta_{\underline{y},\underline{0}}$ (const.)

f balanced: For $\underline{y} = \underline{0}$, $\circledast = \sum_{\underline{x}} (-1)^{f(\underline{x}) + \underline{x} \cdot \underline{0}} = \sum_{\underline{x}} (-1)^{f(\underline{x})} = \underline{0}$

$\Rightarrow$ output is $y = 0 \Longrightarrow f$ constant

— " — $y \neq 0 \Longrightarrow f$ balanced.

$\Rightarrow$ Deterministic discrimination w/ one eval. of f!

What is speed-up?

- Quantum: 1 use of f.
- Class: Worst case, we need to test $2^{n/2} - 1$ values of f to be sure $\Rightarrow$ const. vs. exp.!
- Bad: If we are happy w/ correct answer w/ high prob. ($\leftrightarrow$ q. comp. will also have errors), e.g. $p = 1 - \varepsilon$, then for k tests

$$P_{\text{error}} \approx \underbrace{2 \cdot \left(\frac{1}{2}\right)^k}_{\substack{\text{prob. of } k \text{ eq. outcomes} \\ \text{if } f \text{ is balanced}}} \stackrel{!}{=} \varepsilon$$

$\Rightarrow k \sim \log 1/\varepsilon.$

$\Rightarrow$ Much smaller speed-up vs. probabilistic class. algorithm!

c) Simon's algorithm

$$f: \{0,1\}^n \to \{0,1\}^n$$

Promise: $\exists a$ s.th. $f(x) = f(y)$ iff $x \oplus a = y$.

("hidden periodicity")

Problem: Find a.

Classical: Need to query $f(x_i)$ until $f(x_i) = f(x_j)$ is found.

For k queries: $\sim k^2$ pairs; $P(f(x_i) = f(x_j)) = 2^{-n}$.

$\Rightarrow$ $P_{\text{succes}} \leq k^2 2^{-n}$

$\Rightarrow$ need $k \sim \exp(cn)$ queries!

Quantum:

Start with $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = H^{\otimes n} |0 \dots 0\rangle$

$$U_f : \left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle_A\right) |0\rangle_B \longmapsto \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle_A |f(x)\rangle_B .$$

Now meas. B $\Rightarrow$ collapse onto <u>random</u> $f(x_0)$.

A is collapsed to

$$\frac{1}{N} \sum_{x : f(x) = f(x_0)} |x\rangle = \frac{1}{\sqrt{2}} \left(|x_0\rangle + |x_0 \oplus a\rangle\right)$$

How can we extract a? (Meas. in comp. basis gives x_0 or $x_0 \oplus a$!)

Apply $H^{\otimes n}$ again:

$$H^{\otimes n}\left(\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle)\right) = \frac{1}{\sqrt{2^{n+1}}} \sum_y \underbrace{\left[(-1)^{x_0 \cdot y} + (-1)^{(x_0+a)\cdot y}\right]}_{= 2\cdot(-1)^{x_0\cdot y} \text{ if } a\cdot y = 0,\ = 0 \text{ if } a\cdot y = 1} |y\rangle$$

$$= \frac{1}{\sqrt{2^{n-1}}} \sum_{y : a\cdot y = 0} (-1)^{x_0 \cdot y} |y\rangle$$

Measure $|y\rangle \Rightarrow$ find random vector s.th. $ay = 0$.

$(n-1)$ lin. indep. y allow to compute a.

Need $O(n)$ random y to have $(n-1)$ lin. indep. ones.

$\Rightarrow$ a is found in $O(n)$ steps!

$\Rightarrow$ Exponential speed-up!

Notes: • Need not measure B register! (Outcome indep. of B meas.!)

• $H^{\otimes n}$ can be seen as Fourier transform over $(\mathbb{Z}_2)^{\otimes n}$

$\Rightarrow$ period finding via Fourier trafo (cf. Lect. 1)

3. Grover's algorithm

Common hard computational problem:

We know how to check solution efficiently, but we want to find a solution:

Many problems: Graph coloring, factoring, 3-SAT, Hamiltonian path, ...

Class of "NP problems".

Re-formulation:

We know how to compute $f(x) \in \{0,1\}$ ("verifier" for solution x, $1 \equiv$ "good solution"), and want to find x_0 s.th. $f(x_0) = 1$.

(Can be seen as "database search": Want to find "marked element" x_0 in an unstructured database.)

Assume for now that $x_0 : f(x_0) = 1$ is unique.
(Generalization: Later / homework)

Classically: Will need $O(N)$ queries to f for unstructured search (i.e. don't use any properties of f).

Will see: Quantumly, $O(\sqrt{N})$ queries enough.

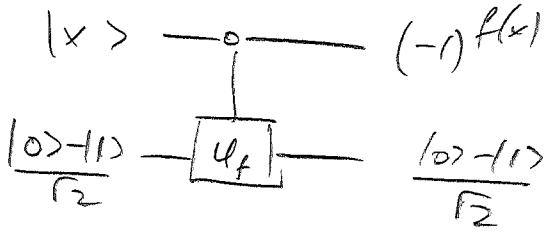
(Note: Quadr. speedup – worse than Simon – but very relevant problem!)

Consider $f: \{0,\dots,N-1\} \to \{0,1\}$

Ingredient 1:

Oracle $O_f : |x\rangle \longmapsto (-1)^{f(x)}|x\rangle = (-1)^{\delta_{x,x_0}}|x\rangle$

Build via



(Note: $x = 0,\dots,N-1$; $|x\rangle \in \mathbb{C}^N$)

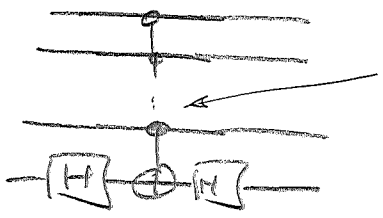
Note that $O_f = I - 2\cdot|x_0\rangle\langle x_0|$:

Flips amplitude of "marked" element.

Ingredient 2:

Unitary $O_0 : |x\rangle \longmapsto (-1)^{\delta_{x,0}}|x\rangle$

Corresponds to $C^{n-1}Z =$



We have $O_0 = I - 2|0\rangle\langle 0|$

$$\Rightarrow O_\omega := H^{\otimes n} O_0 H^{\otimes n} = I - 2|\omega\rangle\langle\omega|; \quad |\omega\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle.$$

Algorithm:

Start from $|\psi_0\rangle = |\omega\rangle$ and apply

Grover iteration $G = -H^{\otimes n} O_0 H^{\otimes n} O_f = (-O_\omega) \cdot O_f$:

$$|\psi_k\rangle \longmapsto |\psi_{k+1}\rangle = G|\psi_k\rangle = -O_\omega \cdot O_f |\psi_k\rangle.$$

Note: Only 2 "special" vectors in O_f & O_ω: $|x_0\rangle$ and $|\omega\rangle$

$\Rightarrow$ can analyze everything in 2-dim space spanned by $|x_0\rangle$ & $|\omega\rangle$!

Define vectors

$$|\alpha\rangle := |x_0\rangle$$

$$|\beta\rangle := \frac{1}{\sqrt{N-1}} \sum_{x \neq x_0} |x\rangle \qquad |\alpha\rangle \perp |\beta\rangle.$$

Since $|\omega\rangle = \frac{1}{\sqrt{N}}|\alpha\rangle + \sqrt{\frac{N-1}{N}}|\beta\rangle$, we can always rewrite

$$a|\alpha\rangle + b|\beta\rangle = x|\omega\rangle + y|\omega^\perp\rangle$$

with $|\omega^\perp\rangle \perp |\omega\rangle$.

What is effect of O_f & $(-O_\omega)$?

$$O_f\left(a|\alpha\rangle + b|\beta\rangle\right) \underset{A}{=} -a|\alpha\rangle + b|\beta\rangle$$

$$O_f = I - 2|\alpha\rangle\langle\alpha|$$

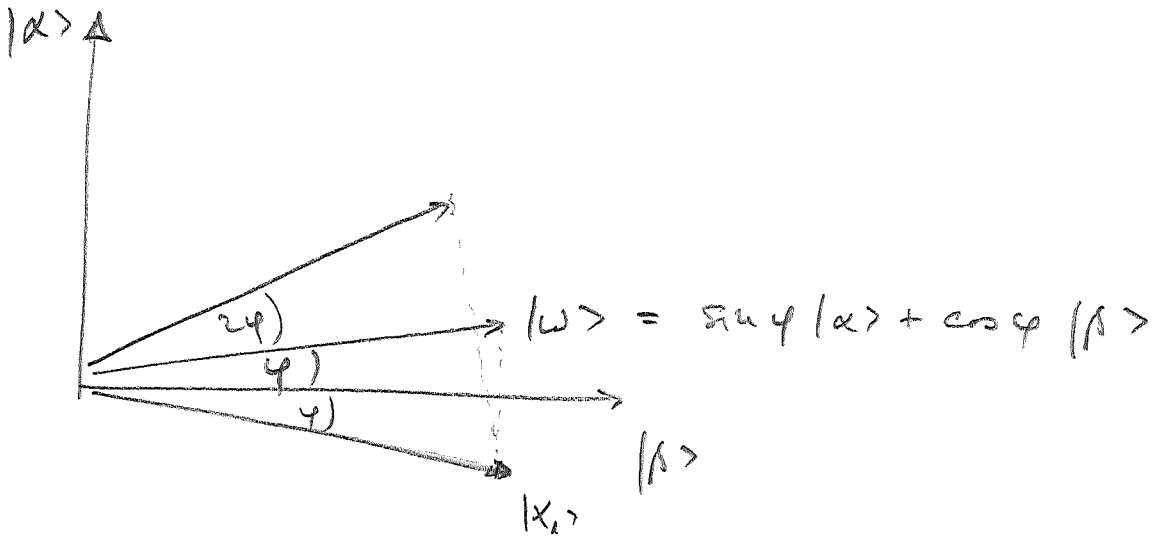
$\Rightarrow$ Reflection about $|\beta\rangle$!

$$(-O_\omega)\left(x|\omega\rangle + y|\omega^\perp\rangle\right) = x|\omega\rangle - y|\omega^\perp\rangle$$

$\Rightarrow$ Reflection about $|\omega\rangle$!

So... what happens in a Grover iteration, starting with $|\psi_0\rangle = |\omega\rangle$?

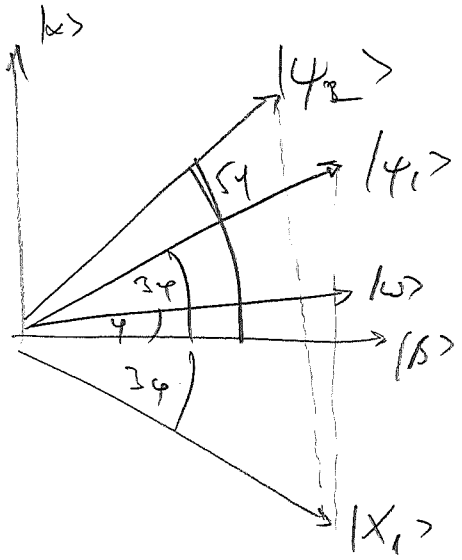
$$|\psi_1\rangle = -O_\omega \underbrace{O_f|\omega\rangle}_{|x_1\rangle} \quad ; \quad |\omega\rangle = \sin\varphi\,|\alpha\rangle + \cos\varphi\,|\beta\rangle.$$



$$\Rightarrow \quad |\psi_1\rangle = \sin(3\varphi)\,|\alpha\rangle + \cos(3\varphi)\,|\beta\rangle.$$

Next iteration:

$$|\psi_2\rangle = -O_\omega \cdot \frac{O_f |\psi_1\rangle}{|\chi_2\rangle}$$



$$\Rightarrow |\psi_2\rangle = \sin(5\varphi)|\alpha\rangle + \cos(5\varphi)|\beta\rangle.$$

$$\Rightarrow |\psi_k\rangle = \sin((2k+1)\varphi)|\alpha\rangle + \cos((2k+1)\varphi)|\beta\rangle$$

Want that $\varphi_k = (2k+1)\varphi \approx \frac{\pi}{2}$

$\Rightarrow$ measurement will with high prob. yield $|\alpha\rangle = |\omega\rangle$!

We have: $$\frac{\sin\varphi}{\cos\varphi} = \frac{\frac{1}{\sqrt{N}}}{\sqrt{\frac{N-1}{N}}} = \frac{1}{\sqrt{N-1}}$$

$$\Rightarrow \varphi \approx \frac{1}{\sqrt{N}} \text{ for large } N!$$

$$\Rightarrow k \approx \frac{\pi}{4}\sqrt{N}$$

$\Rightarrow O(\sqrt{N})$ steps sufficient!

$\Rightarrow$ Quadratic speed-up for general search problems!

Note: $\circ K$ solutions. Same method works with $O\left(\sqrt{\frac{N}{K}}\right)$ steps. ($\rightarrow$ HW).

$\circ$ Can also be adapted to the case when K is unknown.

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

Simon's alg.: Use $H^{\otimes n} \hat{=}$ Fourier trafo in $(\mathbb{Z}_2)^n$ to find period in $(\mathbb{Z}_2)^n$.

$\rightarrow$ Can we find general quantum F.T.?

$\rightarrow$ Can it find periods?

$\rightarrow$ Applications?

a) The Quantum Fourier Transform (QFT)

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

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

y is FT of x: $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{2\pi i jk/N}$

QFT:

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

(Equiv.: $\sum_{j=0}^{N-1} x_j |j\rangle \longmapsto \sum_{j,k} x_j e^{2\pi i jk/N} |k\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$)

Classical FT: $O(N^2)$ operations.

W.l.o.g. $N = 2^n \Rightarrow$ exp. scaling in # of bits n.

better: FFT (fast FT): $O(N \log N)$, but still exp. in n!

Will see: QFT can be implemented efficiently – in $O(n^2)$ steps!

Rewrite QFT:

- Use $N = 2^n$
- Write j in binary: $j = j_1 j_2 \cdots j_N = j_1 \cdot 2^{n-1} + j_2 \cdot 2^{n-2} + \cdots + j_n 2^0$
- Decimal point: $0.j_\ell j_{\ell+1} \cdots j_n = \frac{1}{2} j_\ell + \frac{1}{4} j_{\ell+1} + \cdots + \frac{1}{2^{n-\ell+1}} j_n$.

Then

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

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

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

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

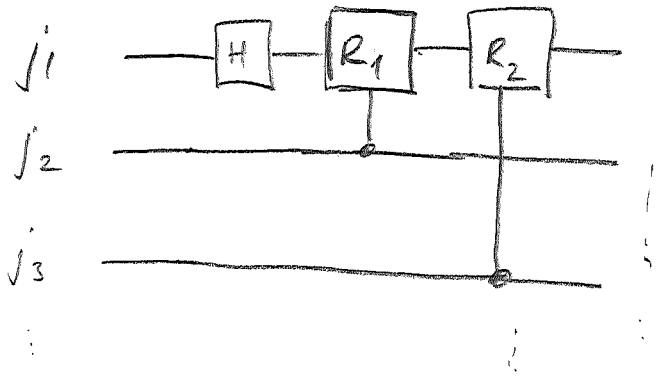
$$= \bigotimes_{\ell=1}^{n} \frac{1}{\sqrt{2}} \left[|0\rangle + \underbrace{e^{2\pi i j 2^{-\ell}}}_{\downarrow} |1\rangle \right]$$

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

$$= \frac{|0\rangle + e^{2\pi i \, 0.j_n} |1\rangle}{\sqrt{2}} \cdot \frac{|0\rangle + e^{2\pi i \, 0.j_{n-1} j_n} |1\rangle}{\sqrt{2}} \cdots \frac{|0\rangle + e^{2\pi i \, 0.j_1 \cdots j_n} |1\rangle}{\sqrt{2}}$$

How to build circuit?

Start w/ rightmost term: $\frac{|0\rangle + e^{2\pi i\, 0.j_1 j_2 \dots j_n}|1\rangle}{\sqrt{2}}$



$$R_d = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i \cdot 2^{-(d+1)}} \end{pmatrix}$$

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

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

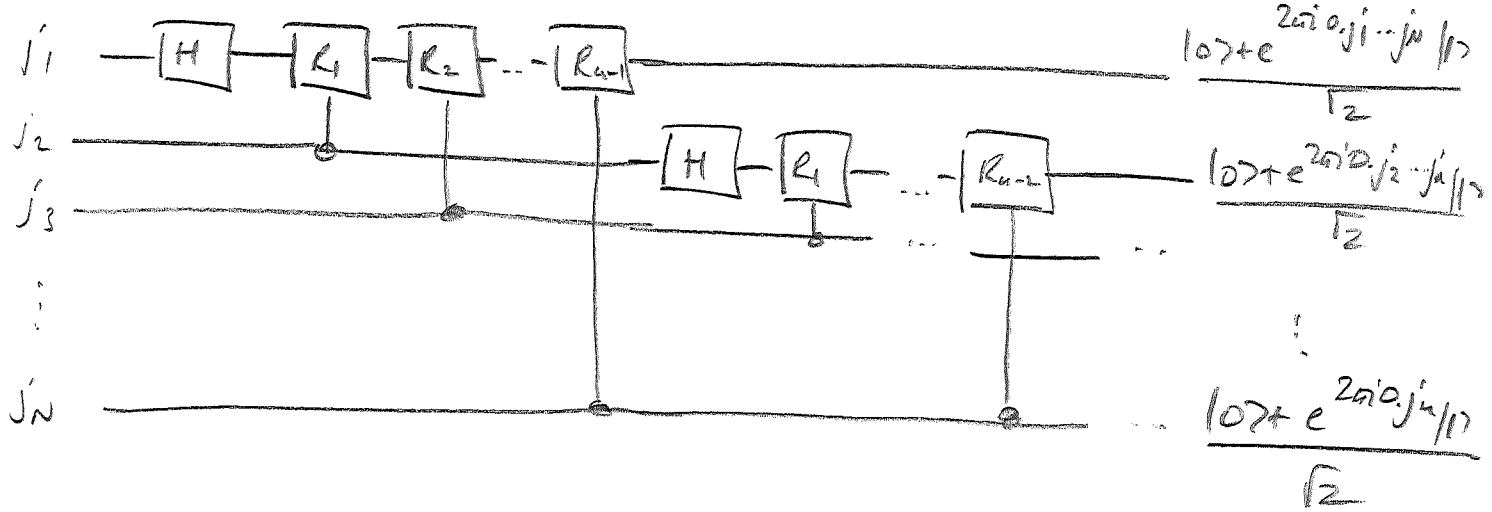
$$C\text{-}R_2 : \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2}|1\rangle\right)|j_2\rangle|j_3\rangle \longmapsto \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2 j_3}|1\rangle\right)|j_2\rangle|j_3\rangle$$

$\vdots$

etc.

$\Longrightarrow$ obtain with Result of QFT as 1st qubit!

Continue the steps:



# of gates: $\frac{n(n+1)}{2} = O(n^2) \Rightarrow$ efficient!

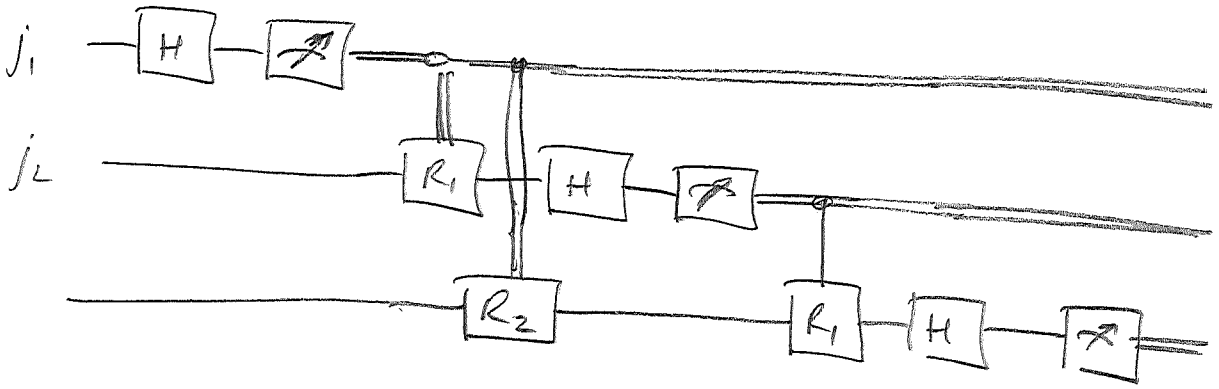
Notes: • Output qubits are in reverse order.

(Reordering: $O(n^2)$ operations.)

• Since  $\Rightarrow$ Rd only dep.

on classical value of upper qubit $\Rightarrow$

If we need to measure register a/fter QFT, we can measure a/fter H & control Rd from meas. outcome:



$\Rightarrow$ can be implemented w/ one-qubit gates only!

6) Period finding:

Use of QFT: finding periods (cf. Simon)

Consider $f: \{0,1\}^n \to \{0,1\}^n$ s.th. $\exists r > 0$ s.th.

$f(x) = f(x+r)$. (And otherwise $f(x) \neq f(y)$).

Can we find r? — Assume $r \ll 2^n$. ← "sufficiently small"

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

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

(2) Measure $|f(x)\rangle_B$ — A collapses to

$$\frac{1}{\sqrt{k_0}} \sum_{k=0}^{k_0-1} |x_0 + kr\rangle \quad \left(\frac{2^n}{r} - 1 < k_0 \leq \frac{2^n}{r}\right)$$

(Note: As in Simon, we can omit this step!)

(3) Apply QFT:

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

$$= \sum_{\ell=0}^{2^n-1} e^{2\pi i x_0 \ell/2^n} \sum_{k=0}^{k_0-1} \underbrace{\frac{1}{2^{n/2}\sqrt{k_0}} e^{2\pi i k r \ell/2^n}}_{=: a_\ell} |\ell\rangle$$

Prob. distr. $|a_\ell|^2$ of meas. $|\ell\rangle$: Should be centered around ℓ s.th. $\frac{r\ell}{2^u} \approx$ integer!

More precisely:

Consider only ℓ s.th.

$$\ell = \frac{2^u}{r} \cdot s + \delta_s \; ; \; |\delta_s| \leq \frac{1}{2} ; \quad s = 0, \ldots, r-1$$

$$\Rightarrow a_\ell = \frac{1}{2^{u/2}\sqrt{k_0}} \sum_{k=0}^{k_0-1} e^{2\pi i k \left(\cancel{s} + \frac{r}{2^u}\delta_s \right)}$$

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

$$\frac{2^u}{r} - 1 \leq k_0 \leq \frac{2^u}{r} \; ; \; r \ll 2^u \Rightarrow \frac{k_0 r}{2^u} \approx 1$$

$$\approx \frac{1}{2^{u/2}\sqrt{k_0}} \; \frac{e^{2\pi i \delta_s} - 1}{e^{2\pi i \, r/2^u \delta_s} - 1}$$

$\sin x \geq \frac{x}{\pi/2}$

$$\Rightarrow |a_\ell|^2 = \frac{1}{2^u k_0} \left(\frac{\sin(\pi \delta_s)}{\sin\left(\frac{\pi r}{2^u}\delta_s\right)} \right)^2 \geq \frac{1}{2^u k_0} \; \frac{\cancel{\pi^2 \delta_s^2}}{\cancel{\pi^2/4}\,\delta} \; \frac{\cancel{\pi^2 r^2}}{2^{2u}} \cancel{\delta_s^2}$$

($\sin\left(\frac{\pi r}{2^u}\delta_s\right) \ll 1$)

$$= \frac{4}{\pi^2 r} \cdot \frac{1}{\frac{k_0 r}{2^u}} = \frac{4}{\pi^2} \frac{1}{r}.$$

$\frac{k_0 r}{2^u} \approx 1$

Since $s = 0, \ldots, r-1$:

$$\text{Prob}\left(\left| \ell - \frac{2^u}{r} s \right| \leq \frac{1}{2} \right) \geq \frac{4}{\pi^2} \approx 0.41$$

$\Rightarrow$ with suff. high prob., we obtain an ℓ s.t.

$$\frac{\ell}{2^u} \approx \frac{s}{r}$$

$\Rightarrow$ can be used to determine $\frac{s}{r}$ w.h.p.

If s and r are coprime (i.e. $\gcd(r,s) = 1$) — happens w/ suff. high prob (can be shown e.g. using density of primes) — we can infer r!

$\Rightarrow$ Quantum algorithm for period finding!

c) Factoring:

One use of period finding: factoring.

Problem: Given N (not prime). Find non-trv. r : $r \mid N$.
(↑ divides.)

Algorithm:

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

If $\gcd(a,N) > 1 \Rightarrow$ done.

↑ cf. computable (Euclid's algorithm)

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

(2) The smallest r with $a^r \bmod N = 1$ is called order of $a \bmod N$.

Note: Existence (of r) follows since $G = \{a \mid a < N, \gcd(a,N) = 1\}$ is a group: If $ab \bmod N = ab' \bmod N \Rightarrow$
$\Rightarrow a(b-b') \bmod N = 0 \Rightarrow N \mid b-b' \Rightarrow b = b'$.
Thus, $b \mapsto ab \bmod N$ is bijective, and thus $\exists b$ s.t. $ab \bmod N = 1$.

r is the period of $f_{N,a}(x) = a^x \bmod N$.

$f_{N,a}(x)$ can be computed efficiently:

With $x = x_{m-1} 2^{m-1} + x_{m-2} 2^{m-2} + \dots + x_0$,

$$a^x \bmod N = \underbrace{\left(a^{(2^{m-1})}\right)^{x_{m-1}}}_{\uparrow} \cdot \left(a^{(2^{m-2})}\right)^{x_{m-2}} \cdots a^{x_0} \bmod N,$$

eff. computable by $a \overset{\cdot^2}{\mapsto} a^2 \overset{\cdot^2}{\mapsto} (a^2)^2 \mapsto \dots$

$\Rightarrow$ r can be found eff. w/ a quantum computer!

(3) Assume r even:

$$a^r \bmod N = 1 \iff N \mid (a^r - 1) \iff$$

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

and $N \nmid (a^{r/2} - 1)$ (otherwise, $a^{r/2} \bmod N = 1$ ↯).

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

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

$\Rightarrow 1 \neq \gcd(N, a^{r/2} \pm 1) \mid N$

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

$\Rightarrow$ algorithm successful as long as

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

$\Rightarrow$ can be shown to happen with $p \geq 1/k$ for random choice of a.

(Unless $N = p^k$, p prime $\rightarrow$ can be checked taking logs.)

$\Rightarrow$ efficient quantum algorithm for factoring!