Note: 2nd qubit never measured (& contains no info!)

Main ideas/points:

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

6) The Deutsch-Jozsa algorithm

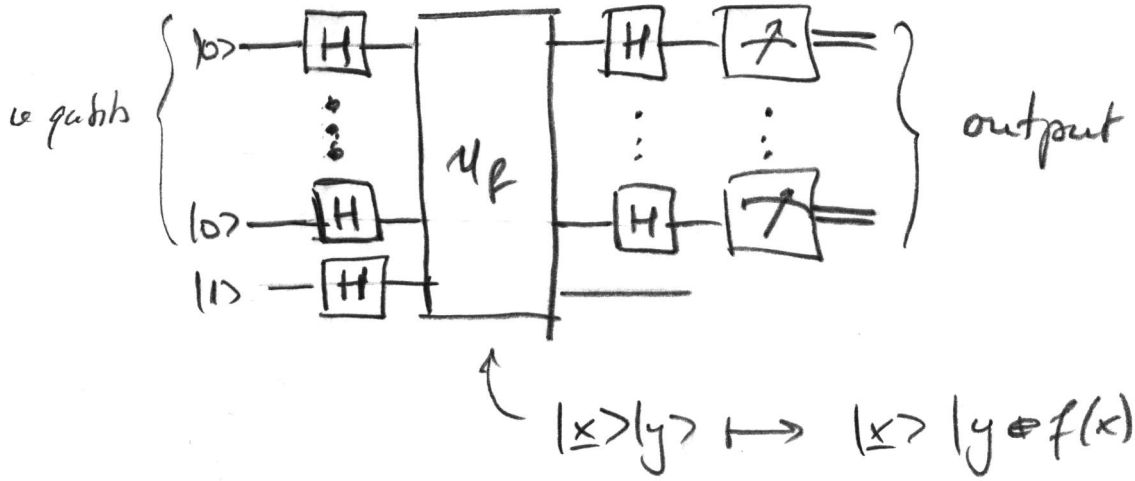
Consider $f: \{0,1\}^n \to \{0,1\}$ w/ promise that

either $f(\underline{x}) = c \quad \forall \underline{x}$ ("f constant")

or $\left|\{\underline{x} \mid f(\underline{x})=0\}\right| = \left|\{\underline{x} \mid f(\underline{x})=1\}\right|$ ("f balanced")

Want to know: Is f constant or balanced?

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



Before analyzing circuit: What is action of $H^{\otimes n}$?

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

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

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

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

($\oplus$: addition mod 2)

i.e. scalar prod. mod 2.

Analyze circuit:

$$|\underline{0}\rangle |1\rangle \xrightarrow{H^{\otimes n} \otimes H} \left(\sum |\underline{x}\rangle\right)(|0\rangle - |1\rangle)$$ (we omit normalization!)

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

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

const!

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

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

Thus: output is $y = \underline{0} \Rightarrow f$ constant

output is $y \neq \underline{0} \Rightarrow f$ balanced

$\Rightarrow$ Unambiguous discrimination w/ one evaluation of f!

What is speed-up w.r.t. classical?

- Quantum: 1 use of f.
- Classical: Worst case, we need to test $2^n/2 + 1$ values of $f \Rightarrow$ const. vs. exponential!

But: If we only want right answer w/ high probability $p = 1 - \varepsilon$, then for k queries to f

$$P_{error} \approx \underbrace{2 \cdot \left(\frac{1}{2}\right)^k}_{\text{approx. prob. for } k \text{ eq. outcomes for balanced } f,\ k \ll 2^n} \doteq \varepsilon$$

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

$\Rightarrow$ Much smaller speed-up vs. probabilistic classical algorithm (even for exp. small error, $k \sim n$).

c) Simon's algorithm

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

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

("hidden periodicity")

Problem: Find $\underline{a}$.

Classical: Need to query $f(\underline{x}_i)$ until $f(\underline{x}_i) = f(\underline{x}_j)$ found.

k queries $\to \sim k^2$ pairs; $P\left(f(\underline{x}_i) = f(\underline{x}_j)\right) \approx 2^{-n}$

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

$\to$ need $k \sim \exp(n)$ queries!

Quantum:

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

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

Now measure B $\Rightarrow$ collapse onto random $f(\underline{x}_0)$.

Register A is collapsed to

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

How can we <u>extract a</u>? (Meas. in comp. basis collapses to $|x_0\rangle$ or $|x_0 \oplus a\rangle$)

<u>Apply $H^{\otimes n}$ again:</u>

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

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

Measure $|y\rangle \Rightarrow$ find random y s.th. $a \cdot y = 0$.

$(n-1)$ lin. indep. y w/ $a \cdot y = 0$ allow to determine a.

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

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

$\Rightarrow$ <u>Exponential speed-up</u> w.r.t. classical algorithm!

Notes:

- We don't even need to measure B (outcome is never used again!)
- $H^{\otimes n}$ can be understood as Fourier trafo over $\mathbb{Z}_2^{\times n}$

  $\Rightarrow$ period finding via Fourier trafo (cf. later!)

IV. 3. Grover's algorithm

For many hard computational problems, it is possible to check solution efficiently, but we don't know how to find it. — So-called "NP problems".

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

Reformulation:

We can compute $f(x) \in \{0,1\}$; $x \in \{0,1,\dots,N-1\}$

— $f(x)$ is a "verifier" for a solution x;

where $f(x) = 1$ means "solution correct" —

and we want to find some x_0 s.t. $f(x_0) = 1$.

(Can be interpreted 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)

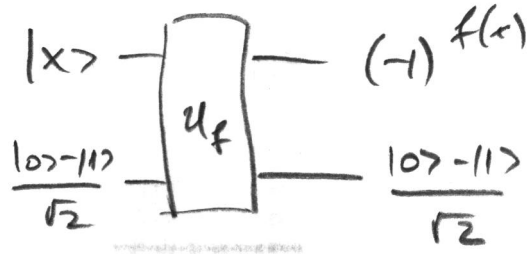
Classically: Need $O(N)$ queries to f for an unstructured search (i.e., w/out using properties of f).

Quantum computer: Will show that $O(\sqrt{N})$ queries enough.
(Note: Only quadratic speedup, but for a very large class of relevant problems)

Ingredient 1:

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

Build via



i.e., O_f flips amplitude of "marked" element.

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

(Can be interpreted 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)

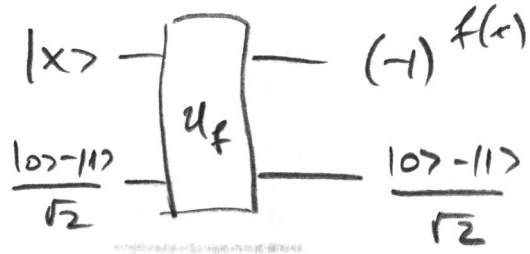
Classically: Need $O(N)$ queries to f for an unstructured search (i.e., w/out using properties of f).

Quantum computer: Will show that $O(\sqrt{N})$ queries enough.
(Note: Only quadratic speedup, but for a very large class of relevant problems)

Ingredient 1:

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

Build via



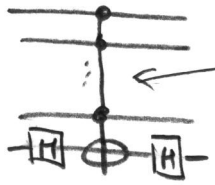
i.e., O_f flips amplitude of "marked" element.

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

Ingredient 2:

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

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



n-qubit Toffoli

→ can be realized efficiently

Again, $O_0 = I - 2|0\rangle\langle 0|$

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

(Remark: We assumed here $N = 2^n$, but not necessary. Also, every search problem can be inv. embedded s.th. $N = 2^n$.)

Algorithm:

Start from $|\psi_0\rangle = |\omega\rangle = H^{\otimes n}|0\rangle$.

Apply Grover iteration

$$G = -H^{\otimes n} O_0 H^{\otimes n} O_f = -O_\omega O_f :$$

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

Observation: Only 2 "special" vectors in O_f, O_ω:

$|x_0\rangle$ and $|\omega\rangle$ $\Rightarrow$ can analyze everything in two-dim. space spanned by $|x_0\rangle$ and $|\omega\rangle$!

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

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

We can always rewrite

$$a|\alpha\rangle + b|\beta\rangle = x|\omega\rangle + y|\omega^\perp\rangle, \quad \text{with } |\omega^\perp\rangle \perp |\omega\rangle$$

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

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

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

$\Rightarrow$ 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 $|\omega\rangle$!

I.e.: Grover iteration = 1) reflect about $|\beta\rangle$

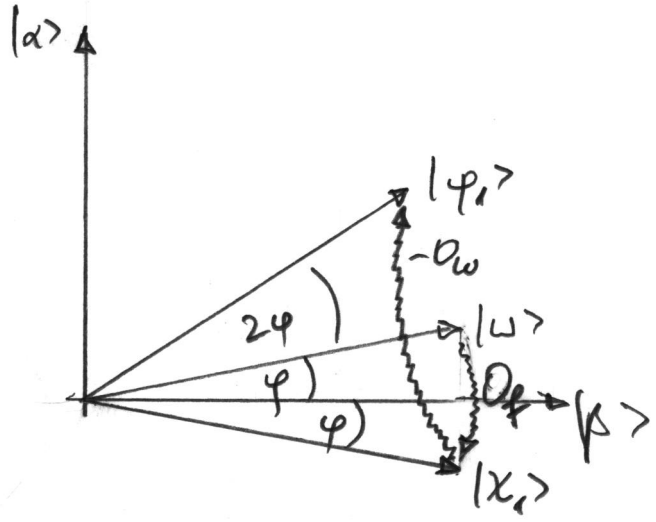
2) reflect about $|\omega\rangle$

So what happens in one iteration, if we start with $|\psi_0\rangle = |\omega\rangle$?

$$|\omega\rangle = \sin\varphi\,|\alpha\rangle + \cos\varphi\,|\beta\rangle$$

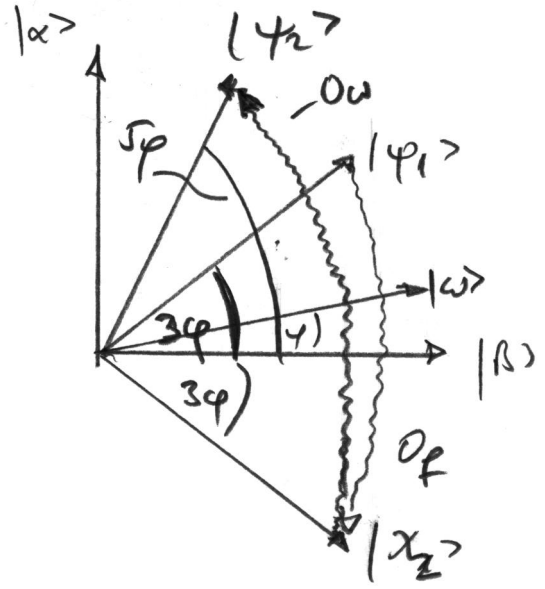
$$|\chi_1\rangle := O_f\,|\omega\rangle$$

$$|\psi_1\rangle = -O_\omega\,|\chi_1\rangle = -O_\omega O_f\,|\omega\rangle$$



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

Next iteration: $|\psi_2\rangle = -O_\omega \underbrace{O_f\,|\psi_1\rangle}_{=:|\chi_2\rangle}$



$$\Rightarrow |\psi_2\rangle = \sin(3\varphi)\,|\alpha\rangle + \cos(3\varphi)\,|\beta\rangle$$

$$\longrightarrow |\psi_k\rangle = \sin\left((2k+1)\varphi\right)|\alpha\rangle + \cos\left((2k+1)\varphi\right)|\beta\rangle .$$

Want that $\varphi_k = (2k+1)\varphi \approx \frac{\pi}{2}$. Then, meas. will w/ high prob. yield $|\alpha\rangle = |k_0\rangle$!

We have: $|\omega\rangle = \frac{1}{\sqrt{N}}|\alpha\rangle + \sqrt{\frac{N-1}{N}}\,|\beta\rangle$

$$= \sin\varphi\,|\alpha\rangle + \cos\varphi\,|\beta\rangle$$

$$\Rightarrow \frac{\sin\varphi}{\cos\varphi} = \frac{\sqrt{\frac{1}{N}}}{\sqrt{\frac{N-1}{N}}} = \frac{1}{\sqrt{N-1}}$$

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

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

$\Rightarrow$ $O(\sqrt{N})$ calls to f sufficient!

Quadratic speed-up w.r.t. classical algorithms for general search problems!

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

- Can be adapted to case where K is unknown.

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

Recall: Simon's algorithm $\rightarrow$ use $H^{\otimes n} \hat{=}$ Fourier trafo over $\mathbb{Z}_2^{\otimes n}$ to find period in $\mathbb{Z}_2^{\otimes n}$.

$\rightarrow$ Can we construct a general Q. Fourier Trafo?

$\rightarrow$ Can it be implemented efficiently?

$\rightarrow$ Applications?

a) The Quantum Fourier Transform (QFT)

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

Define QFT:

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

(Note: QFT: $\sum x_j |j\rangle \longmapsto \sum x_j e^{2\pi i jk/N} |k\rangle = \sum y_k |k\rangle$

$\Rightarrow$ QFT applies FT to amplitudes)

Computational cost of FT: $O(N^2)$ operations.

With $N = 2^n$ $\rightarrow$ exp. cost in # of bits n.

Fast FT (FFT): $O(N \log N)$, but still $N = \exp(n)$.

Will show: QFT can be implemented in $O(n^2)$ steps

$\longrightarrow$ exponential speed-up!

Rewrite QFT in binary:

* $N = 2^n$
* Write j in binary: $j = j_1 j_2 \dots j_n = j_1 2^{n-1} + j_2 2^{n-2} + \dots + j_n 2^0$
* Decimal part: $0.j_\ell j_{\ell+1} \dots j_n = \frac{1}{2} j_\ell + \frac{1}{4} j_{\ell+1} + \dots$

Then:

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

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

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

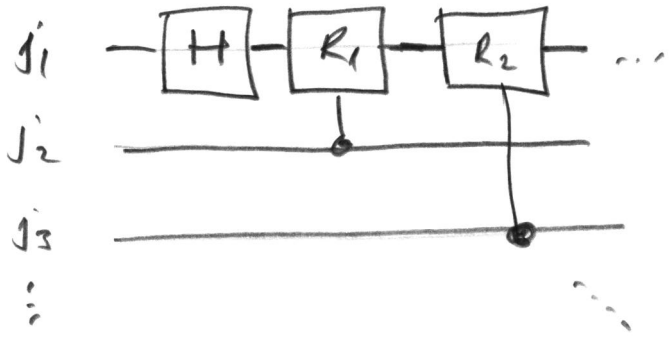
$$= \bigotimes_{\ell=1}^{n} \frac{1}{\sqrt{2}} \left[|0\rangle + e^{2\pi i j 2^{-\ell}} |1\rangle \right]$$

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

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

How to implement this map?

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



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

action of circuit (up to norm.):

$H_1: |j_1\rangle \longmapsto |0\rangle + e^{2\pi i\, 0.j_1}|1\rangle$

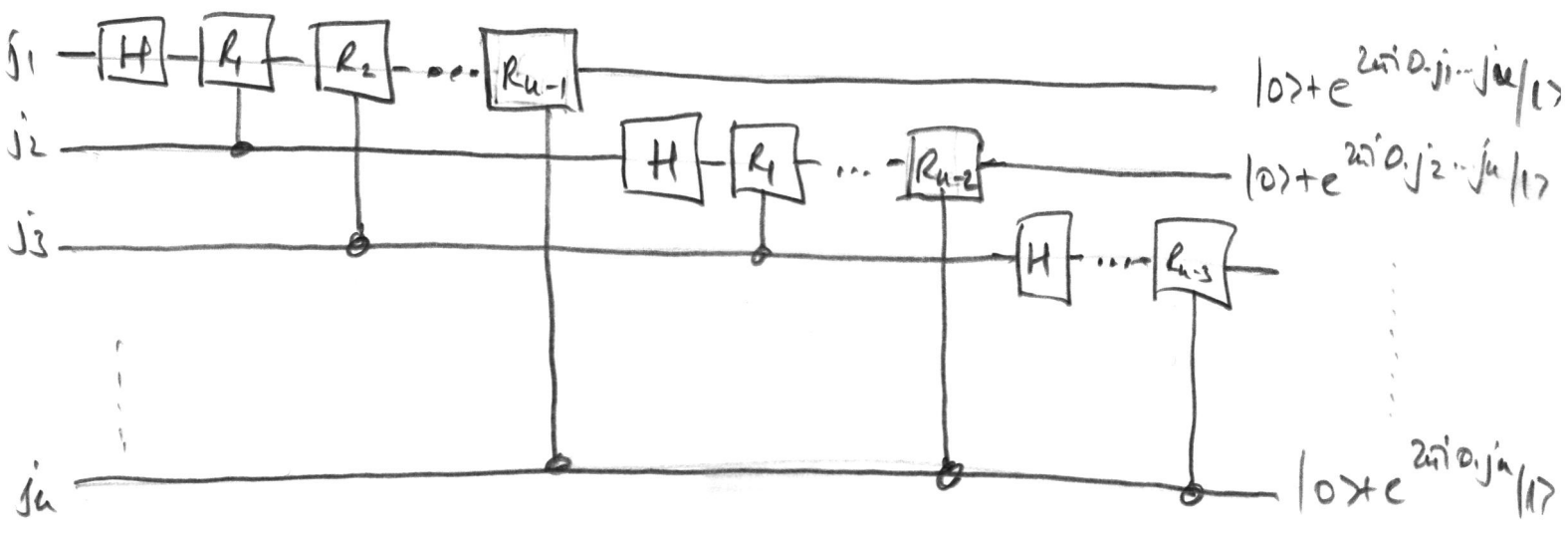
$c\text{-}R_1: (|0\rangle + e^{2\pi i\, 0.j_1}|1\rangle)|j_2\rangle \longmapsto (|0\rangle + e^{2\pi i\, 0.j_1 j_2}|1\rangle)|j_2\rangle$

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

$\vdots$ etc.

$\Rightarrow$ obtain n-th Qubit of QFT on 1st qubit.

Continue like that for $(j_2 \cdots j_n)$, $(j_3, \cdots, j_n)$, ...



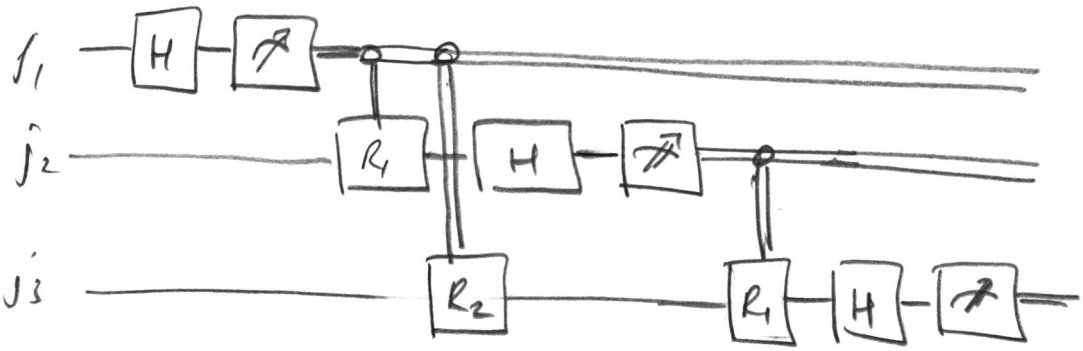
Gate count: $\frac{n(n+1)}{2} = O(n^2)$ gates !

Notes: • Output is in reverse order (but reordering $\sim O(n)$ ops).

•  ⇒ can reverse C-R_d gates.

Upper (control) line acts as control in comp. basis:

If we measure after QFT, we can meas. after H & control R_d-gates classically!



⇒ only one-qubit gates needed!

6) Period finding

Use of QFT: period finding?

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

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

Can we find r?

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

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 $B \to A$ collapses to

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

(Note: As for Simon, we could omit this step!)

③ Apply QFT & measure in comp. basis:

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

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

$|a_\ell|^2$: probability of outcome ℓ.

If $r \ll 2^n$: many values of k & almost periodic

$\Rightarrow |a_\ell|^2$ peaked around ℓ s.th. $\frac{r\ell}{2^n} \approx$ integer

Explicit analysis of a_ℓ shows: w.h.p., we obtain ℓ s.th.

↳ "with high probability"

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

If $r \ll 2^n$, this can be used to determine $\frac{s}{r}$ w.h.p.

If s, r are coprime (i.e., $\gcd(s,r) = 1$) – this happens with large enough prob., related to density of primes – we can infer r!

$\Rightarrow$ Quantum Algorithm for period finding!

c) Application: factoring

One use of of period finding: factoring.

Given N (not prime) $\rightarrow$ find non-trivial r: $r \mid N$

↑

"divides"

If $r \ll 2^n$: many values of k & almost periodic

$\Rightarrow |a_\ell|^2$ peaked around ℓ s.th. $\frac{r\ell}{2^n} \approx$ integer

Explicit analysis of a_ℓ shows: w.h.p., we obtain ℓ s.th.

↳ "with high probability"

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

If $r \ll 2^n$, this can be used to determine $\frac{s}{r}$ w.h.p.

If s, r are coprime (i.e., $\gcd(s,r) = 1$) – this happens with large enough prob., related to density of primes – we can infer r!

$\Rightarrow$ Quantum Algorithm for period finding!

c) Application: factoring

One use of of period finding: factoring.

Given N (not prime) $\rightarrow$ find non-trivial r: $r \mid N$

↑

"divides"

Algorithm:

① Select random a, $2 \leq a < N$.

If $\gcd(a,N) \geq 1 \Longrightarrow$ done!

↳ eff. computable! (Euclid's algorithm)

So assume $\gcd(a,N) = 1$.

② Let r be the smallest r s.th. $a^r \bmod N = 1$.

(Existence: (!) $\exists x,y: a^x \equiv a^y \bmod N \Rightarrow a^x(1-a^{y-x}) \equiv 0 \bmod N$

$\Rightarrow N \mid a^x(1-a^{y-x}) \overset{\gcd(a,N)=1}{\Longrightarrow} N \mid 1-a^{y-x} \Rightarrow a^{y-x} \equiv 1 \bmod N$)

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

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

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

$$a^x \bmod N = \underbrace{\left(a^{(2^{m-1})}\right)^{x_{m-1}} \cdot \left(a^{(2^{m-2})}\right)^{x_{m-2}} \cdot \dots}_{\text{eff. computable by } a \mapsto a^2 \mapsto a^4 \mapsto a^8 \mapsto \dots} \bmod N,$$

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

(3) Assume r even:

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

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-triv. factor of N!

$\Rightarrow$ Algorithm successful as long as
(i) r even & (ii) $N \nmid (a^{r/2} + 1)$

Can be shown to happen w/ $p \geq 1/2$ for random choice of a (unless $N = p^k$, p prime → can be checked by taking logs)

$\Rightarrow$ <u>efficient quantum algorithm for factoring</u>

<u>"Shor's algorithm"</u>