1. Quantum Computation

1. The circuit model

q) Classical computation

Task of classical computer:

Solve problems ≡ compute functions:

\[ f : \{0,1\}^n \rightarrow \{0,1\}^n \]

\[ x = (x_1, \ldots, x_n) \Rightarrow f(x_1, \ldots, x_n) \]

\( f \) depends on problem, \( x \) encodes instance of problem.

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

\[ x = (x_1, x_2) \Rightarrow f(x) = \frac{x_1 \cdot x_2}{\text{binary, input}} \]

Factorization:

\[ x : \text{integer} ; f(x) : \text{list of prime factors} \]

(1/2) simple extraction
Each problem is encoded by a family of functions $f: \{0,1\}^n \rightarrow \{0,1\}^m$, $m = \text{poly}(n)$, $n \in \mathbb{N}$.

It should be possible to "construct $f$ systematically & efficiently." (to later).

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

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

\[ f(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \]

\( f_k: \{0,1\}^n \rightarrow \{0,1\} \)

\( \Rightarrow \) can reduced to boolean functions $f: \{0,1\}^n \rightarrow \{0,1\}$.

(ii) Let $L = \{ x \mid f(y) = 1 \} = \{ y_1, y_2, \ldots, y_k \}$

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

\[ f(x) = g_{y_1}(x) \lor g_{y_2}(x) \lor \ldots \lor g_{y_k}(x) \]

"\lor" logical "or": $0 \lor 0 = 0$

\[ 0 \lor 1 = 1 \]

\[ 1 \lor 0 = 1 \]

\[ 1 \lor 1 = 1 \]

\[ \Rightarrow \text{av, svc = av(svc), ok!} \]
(iii) \( \forall x \in (y_1 = x_i) \land (y_2 = x_i) \land \ldots \)

"\( \land \)" : logical "and" : \( \land 1 = 1 \)

e.g. \( \land xy = 0 \)

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

"\( \neg \)" : logical "not" : \( \neg 1 = 0 \)

\( \neg 0 = 1 \)

\[ \Rightarrow f(x) \text{ can be built from } 4 \text{ in-} \]
"and", "or", "not" gates, + copy gate \( x \mapsto \langle x, x \rangle \).

"Universal gate set."

(Note: In fact, either \( \neg(xy) \) "and" or \( \neg(xy) \) "nor",

\( \text{together w/ copy, are already universal.} \)

Circuit model of computer:

\[ \begin{cases} \text{built from universal gate set (without "logic}\right.

\text{ gate \( x \mapsto \langle x, x \rangle \)} \right). \)
Which problems can we solve efficiently in this model?

⇒ Problems where \( \# faks (\equiv \# \text{"true steps"}) \) 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?

\[ f : \{0,1\}^n \rightarrow \{0,1\} \]

\# different \( f \): \( 2^{\left(\frac{2^n}{\text{# inputs}}\right)} \)

Output: 0 or 1

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

⇒ 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 \( u \) in time \( \text{poly}(n) \) for any \( u \).
Does comp. power dep. on falk set?

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

What about other models of computation?

- CPU
- parallel computers
- Turing machines (tape based)
- Cellular automata

(\textit{other crude models...})

⇒ All known "reasonable" models can simulate each other \(\text{w/ poly}(n)\) overhead ⇒ same comp. power!

\underline{Church-Turing Thesis:} All reasonable models of computers have the same computational power.

(Note: different variants: randomness? quantums ("Shor" - C-T Thesis))
Will use circuit model to go to quantum systems:

Gates $\rightarrow$ Unitaries

But: Classical gates irreversible $\rightarrow$ Unitaries reversible.

Can we fit even classical computation in this picture?

Classical computation cannot be linear reversible:

Toffoli gate:

$x \quad y \quad z$

$x \quad y \quad$ "and"

$z \quad \oplus x \cdot y$

$\iff$ reversible!

$\rightarrow$ can simulate and/or/-ins/copy

using ancillas, e.g.:

$x \quad x$

$\quad 1$

$\quad 0$

$\oplus$ x, y = $\neg(x \cdot y)$: $\text{NAND}$

$\rightarrow$ reversible mimic gate (using ancillas)
Any $f(x)$ can be computed reversibly:

$$f^R(x,y) \rightarrow (x, f(x) \oplus y)$$

- possibly w/ ancillas - but changing anything else.
- Can be optimized to use few ancillas (Preshall)
- Everything can be computed reversibly.
- But: 3-bit gate needed ($\rightarrow$ HU)

---

5) Quantum Circuits

Model of quantum computation:

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

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

$\Rightarrow$ simulatable (poly-time) circuits.
Input: Classical input \( |\psi\rangle |x\rangle \ldots |x\rangle \) in computational basis.

Output: Measure all qubits at the end in the "computational basis" \( \{0\}, \{1\}\).

(Note: Other measurements do not help: can use last gate set to do any PDA/1.

- Not area, qubits = tracing out = meas. and summing result.
- Rees, at earlier time: can be postponed until end (no locality.)

Which gate set should we choose?

- Continuum of gates: much more nice!
- Turns out: 1+2 qubit gate sufficient
  (unlike classical!)

- Almost all 2-qubit gates are performed by themselves
  (in an approx. sense, if involved phases incommensurate)
Our choice:

(i) 1-qubit rotations about $X$ & $Z$ axes:

\[
R_x(\phi) = e^{-iX\phi/2} \quad \text{or} \quad x = (0, 1) \quad ; \quad x^2 = 1
\]

\[
R_z(\phi) = e^{-iZ\phi/2} \quad \text{or} \quad z = (1, 0) \quad ; \quad z^2 = 1
\]

For $\pi^2 = 1$:

\[
e^{i\pi\phi/2} = \cos\phi/2 I + i\sin\phi/2 \sigma_i
\]

\[
\Rightarrow R_x(\phi) = \begin{pmatrix}
\cos\phi/2 & -i\sin\phi/2 \\
i\sin\phi/2 & \cos\phi/2
\end{pmatrix}
\]

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

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

(\text{\textcopyright} Euler angles!)

\[
U = R_x(\alpha) R_z(\beta) R_x(\gamma)
\]

(ii) one 2-qubit gate (almost always used)

Typically we choose "controlled NOT"

\[
\text{CNOT} = \begin{pmatrix}
x & x \\
y & x \otimes y
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]
The Pauli y iff \( x = 1 \): Classical gate!

Turns out: Unitary gate set can be used to exactly create any \( n \)-qubit gate. (Of course, not iff: there are "even more" \( n \)-qubit circuits than \( n \)-bit functions.)

**Some important gate sets + their definitions:** \((\rightarrow \text{HW})\):

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

\[ H R_x(\phi) H = R_x(\phi) \]

\[ H R_z(\phi) H = R_z(\phi) \]

**Graphical notation:**

\[
\begin{array}{c}
\begin{array}{c}
\text{Hadamard} \\
\text{operation}
\end{array}
\begin{array}{c}
\text{X} \\
\text{operation}
\end{array}
\begin{array}{c}
\text{Hadamard} \\
\text{operation}
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\text{Hadamard} \\
\text{operation}
\end{array}
\begin{array}{c}
\text{X} \\
\text{operation}
\end{array}
\begin{array}{c}
\text{Hadamard} \\
\text{operation}
\end{array}
\end{array}
\]

"Controlled-\( Z \)"

"Controlled-Phase", CPHASE
\[
T_{\text{offli}}:
\begin{align*}
\begin{array}{c}
\text{with } V = \frac{1-i}{2} (I + iX), \text{ and } \\
\frac{1}{V} = \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix}
\end{array}
\end{align*}
\]

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

\[
\begin{array}{c}
\text{just replace every Toffoli (class. case!)} \text{ by} \\
\text{Toffoli w/ 3 controls:}
\end{array}
\]

\[
\begin{array}{c}
\text{can be built from normal} \\
\text{Toffoli (→ HW)}
\end{array}
\]
What are the units, jake eki? — Many!

Note: Different notions of universality:

- exact universal: any n-qubit U can be realized exactly
- approximate universal: any n-qubit U can be approx. by gate set (w/ reasonable cond.)

Examples of approximate jake universal:

- CNOT + 2 random 1-qubit gates
- CNOT + H + T = R_{z}^{\theta_{\pi}} (\approx \pi) (^{\pi}_{\frac{\pi}{4}} - - jake)
- most 2-qubit gates alone

Slovay-Kitaev-Thou: A universal gate set for SU(2^n) can approximate any U \in SU(2^n) up to \epsilon with

O(\text{poly}(\log(1/\epsilon))) gates.
a) The Deutsch algorithm

Consider \( f : \{0,1\} \rightarrow \{0,1\} \).

Let \( f \) be "very hard" to compute (e.g., by circuit). Want to know: is \( f(0) = f(1) \)?

How often do we have to evaluate \( f \)?

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

\( \rightarrow \) Classically: 2 queries: \( f(0) \), \( f(1) \).

Can Q. H. help?

Consider reversible implementation of \( f \):

\[ f^R : (x,y) \rightarrow (x, y \oplus f(x)) \]

\[ y \xrightarrow{y_{\text{out}}} y \oplus f(x) \]

\( \rightarrow \) try to use superpositions?
\[ \frac{10 + 11 \langle \frac{11}{12} \rangle}{\sqrt{2}} = |0\rangle - \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \]

\[ \frac{10 - 11 \langle \frac{11}{12} \rangle}{\sqrt{2}} \]

\[ \frac{10 - 11 \langle \frac{11}{12} \rangle}{\sqrt{2}} \to \frac{1}{\sqrt{2}} (|0\rangle f(x) + |1\rangle f(y) \rangle) \]

\[ \Rightarrow \text{ have evaluated } f \text{ on both inputs!} \]

\[ \text{But: how can we extract (relevant) information?} \]

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

Consider:

\[ |x\rangle \quad \frac{10 - 11 \langle \frac{11}{12} \rangle}{\sqrt{2}} = \quad \frac{10 - 11 \langle \frac{11}{12} \rangle}{\sqrt{2}} \]

\[ |y\rangle \to \quad |x\rangle \left( \frac{|f(x)\rangle - |\Phi f(x)\rangle}{\sqrt{2}} \right) = \]

\[ \left\{ \begin{array}{l}
\text{if } f(x) = 0: \\
\text{if } f(x) = 1:
\end{array} \right. 
\]

\[ \left| x \right\rangle \begin{bmatrix} \frac{10 - 11 \langle \frac{11}{12} \rangle}{\sqrt{2}} \end{bmatrix} \left( -1 \right)^{f(x)} \frac{10 - 11 \langle \frac{11}{12} \rangle}{\sqrt{2}} \right] . \]
Combine:

\[ \frac{10\rangle + 11\rangle}{\sqrt{2}} \quad \text{and} \quad \frac{10\rangle - 11\rangle}{\sqrt{2}} \]

\[ 10\rangle \quad \text{H} \quad 11\rangle \quad \text{H} \quad \text{H} \]

\[ 10\rangle \quad \text{H} \quad \text{H} \]

\[ \frac{10\rangle + 11\rangle}{\sqrt{2}} \left( \frac{10\rangle - 11\rangle}{\sqrt{2}} \right) \xrightarrow{U_f} \frac{1}{\sqrt{2}} \left( (-1) f(0) 0\rangle + (-1) f(1) 1\rangle \right) \left( \frac{10\rangle - 11\rangle}{\sqrt{2}} \right) \]

\[ \Rightarrow \text{no entanglement created} \]

\[ \Rightarrow 2 \text{nd qubit unchanged} \]

\[ \Rightarrow 1 \text{st qubit gets phase } (-1)^f(x) \]

\[ \Rightarrow \text{"phase kick-back" technique} \]

\[ \Rightarrow \text{1st qubit} = \frac{10\rangle + 11\rangle}{\sqrt{2}} : f(0) = f(1) \]

\[ \Rightarrow \text{1st qubit} = \frac{10\rangle - 11\rangle}{\sqrt{2}} : f(0) \neq f(1) \]

\[ \xrightarrow{\text{classical simulation}} \]

\[ 10\rangle \quad \text{H} \quad \text{H} \quad \text{H} \quad \text{H} \quad \text{H} \]

\[ 10\rangle : f(0) = f(1) \]

\[ 11\rangle : f(0) \neq f(1) \]

\[ \Rightarrow \text{One application of } U_f \text{ sufficient!} \]

\[ \Rightarrow \text{factor 2 faster than classically!} \]
Two core ideas:

- Use input $\Sigma(x)$ to evaluate $f$ at input $x$.
- Need way to read out relevant info!

6) The Deutsch-Jozsa algorithm

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

- $f(x) = c \quad \forall x$ "constant"
- or $\#\{x | f(x) = 0\} = \#\{x | f(x) = 1\}$ "balanced"

Want to know: Is $f$ constant or balanced?

- How many queries to $f$ do we need?

Use same idea: Input $\{\frac{2\sqrt{2}}{2}, \frac{2\sqrt{2}}{2}\}$ and $10\over\sqrt{2}$:

\[
\begin{array}{c}
\left|0\right> \quad \text{H} \quad \left|+\right> \\
\left|1\right> \quad \text{H} \quad \left|+\right> \\
\left|0\right> \quad \text{H} \quad \left|+\right> \\
\left|1\right> \quad \text{H} \quad \left|+\right> \\
\end{array}
\]

Output

\[
\left|\psi\right> = \left|\psi\right>_{0\oplus f(x)}
\]

Note: 2nd query need not be measured (and contains no information...)
What is action of $H^{ou}$?

$H: \left| x \right> \rightarrow \frac{1}{\sqrt{2^n}} \sum (-1)^{x \cdot y} \left| y \right>$

$H^{ou}: \left| x_1, \ldots, x_n \right> \rightarrow \frac{1}{\sqrt{2^n}} \sum (-1)^{x \cdot y_1} (-1)^{x_2 y_2} \ldots \left| y_1, \ldots, y_n \right>$

$\left| x \right> \rightarrow \frac{1}{\sqrt{2^n}} \sum (-1)^{x \cdot y} \left| y \right>$

$(x \cdot y := x_1 y_1 \otimes x_2 y_2 \otimes \ldots \otimes x_n y_n)$

$\Rightarrow \left| 0 \right> \left< 1 \right| \xrightarrow{H^{ou} \otimes H} \left( \sum \left| x \right> \right) \left( \left| 0 \right> - \left| 1 \right> \right)$

$\xrightarrow{U_x f} \left( \sum_{x} (-1)^{f(x)} \left| x \right> \right) \left( \left| 0 \right> - \left| 1 \right> \right)$

$\xrightarrow{H^{ou} \otimes I} \left( \sum_{y} \sum_{x} (-1)^{f(x)} + x \cdot y \right) \left| y > \right> \left( \left| 0 \right> - \left| 1 \right> \right)$

If constant: $\otimes = (-1)^{f(x)} \sum_{x} (-1)^{x \cdot y} = (-1)^{f(x)} \sum_{y} \delta_{y, 0}$

If balanced: $\text{For } y = 0, \otimes = \sum_{x} (-1)^{f(x)} + x \cdot 0 = \sum_{x} (-1)^{f(x)} = 0$
⇒ output is $y = 0 \Rightarrow f$ constant

⇒ $y \neq 0 \Rightarrow f$ balanced.

⇒ Unambiguous discrimination w/ one call of $f$!

What is speed-up?

* Quantum: 1 run of $f$.

* Class: Worst case, we need to test $2^{\frac{n}{2}} - 1$ values of $f$ to be sure $\Rightarrow$ const. vs. exp.!!

* But: If we are happy with correct answer w/ high prob. (⇒ q. comp. will also have errors), e.g.

  \[ p = 1 - \varepsilon, \text{ then for } k \text{ tests} \]

  \[ \text{error} \approx 2 \cdot \left(\frac{1}{2}\right)^k = \varepsilon \]

  \[ \text{prob. of } k \text{ eq. outcomes} \]

  \[ \text{if } f \text{ is balanced} \]

⇒ $k \approx \log \frac{1}{\varepsilon}$. 

⇒ Much smaller speed-up vs. probabilistic class. algorithm!
c) Simon's algorithm

\[ f : \{0,1\}^n \to \{0,1\}^n \]

**Proof:** For s.t. \( f(x) = f(y) \) 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: \( n k^2 \) pairs; \( P(f(x_i) = f(x_j)) = 2^{-n} \).

\[ \Rightarrow P_{success} = k^2 2^{-n} \]

So need \( k \approx \exp(cA) \) queries!

**Quantum:**

Start with \[ \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = H^n |0...0\rangle \]

\[ M_f : \left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \right)_A |0\rangle_\mathcal{B} \mapsto \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle_A |f(x)\rangle_\mathcal{B}. \]

Now meas. \( \mathcal{B} \) \( \Rightarrow \) collapse onto random \( f(x_0) \).

\( A \) 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 θ? (Uses n camp, basic trig gives θ₀ or 2ζθ₀!)

Apply \( H^\text{sn} \) again:

\[
H^\text{sn}(\frac{1}{2}(|x⟩ + |θ\theta⟩)) = \frac{1}{2^u+1} \sum_{y} \left[ (-1)^{y_0} y + (-1)^{y_0+|y|} y \right] |y⟩
\]

\[
= 2(-1)^{y_0} y \quad \text{if} \quad x\cdot y = 0
\]

\[
= 0 \quad \text{if} \quad x\cdot y = 1
\]

\[
= \frac{1}{2^u-1} \sum_{y: x\cdot y = 0} (-1)^{y_0} y |y⟩
\]

Note: \( |y⟩ \) → find random vector s. t. \( x\cdot y = 0 \).

\( (u-1) \) r.v. indep. \( y \) allow to compute \( θ \).

Need \( O(u) \) random \( y \) to have \( (u-1) \) r.v. indep. ones.

\( \Rightarrow \) \( θ \) is found in \( O(u) \) steps!

\( \Rightarrow \) \underline{Exponential speed-up!}

Notes:

- Need not measure \( B \) registers. (Outcome indep. of \( B \) meas.)

- \( H^\text{sn} \) can be seen as Fourier transform over \( \mathbb{Z}_2^u \)

\( \Rightarrow \) period finding via Fourier transforms (cf. later!)
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: NP problems.

Re-formulation:
We know how to compute \( f(x) \in \{0, 1\} \) ("verify" for solution \( x \), \( 1 \) = "good solution"), and want to find \( x_0 \) s.t. \( f(x_0) = 1 \).

(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 \( \mathcal{O}(N) \) queries to \( f \) for unstructured search (i.e. what using properties of \( f \)).
\[ \text{Will see: Quantumly, } O(\sqrt{n}) \text{ queries enough.} \]

\[ \text{(Note: Quicks. speedup - worse than Grover - but very relevant problem!)} \]

Consider \( f: \{0, \ldots, N-1\} \to \{0, 1\} \)

**In Problem 1:**

Oracle \( O_f \): \( |x\rangle \mapsto (-1)^{f(x)} |x\rangle \)

\[ \text{Build via} \]

\[ |x\rangle \begin{array}{c}
\downarrow \quad \text{U}_f \\
|0\rangle \cdots |1\rangle
\end{array} \quad (-1)^{f(x)} |x\rangle \]

\[ \text{(Note: } x = 0, \ldots, N-1; \ |x\rangle \in \mathbb{C}^n) \quad |x\rangle \in \mathbb{C}^n \]

Note that \( O_f = I - 2 |x_0\rangle \langle x_0| \):

Flips amplitude of "marked" element.

**In Problem 2:**

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

Corresponds to \( C^{n+2} \)

\[ \text{We have } O_0 = I - 2 |0\rangle \langle 0| \]

\[ = D \quad O_{\omega} = H^\omega \circ O_0 \circ H^\omega = I - 2 |\omega\rangle \langle \omega| \quad |\omega\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \]
Algorithm:

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

**Grove Iteration**

$G = -H^* O_0 H O^*_F = (-O_\omega) O_F$

\[
|\psi_k\rangle \rightarrow |\psi_{k+1}\rangle = G |\psi_k\rangle = -O_\omega \cdot O_F |\psi_k\rangle.
\]

**Note:** Only 2 "special" vectors in $O_F$ & $O_\omega$: $|x_0\rangle$ and $|\omega\rangle$

$\Rightarrow$ can analyze everything in two-dim space

spanned by $|x_0\rangle$ & $|\omega\rangle$.

Define vectors

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

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

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

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

with $|\omega^\perp\rangle \perp |\omega\rangle$. 
What is effect of $O_\omega$ on $\alpha$?

$O_\omega (a |\alpha> + b |\beta>) = \frac{1}{a} (-a |\alpha> + b |\beta>)$

$O_\omega = I - 2 |\alpha><\alpha|$

$\implies$ Reflection about $|\beta>!$

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

$\implies$ Reflection about $|\omega>!$

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

$|\psi_1> = -\omega O_\omega |\omega> = \sin \phi |\alpha> + \cos \phi |\beta>.$

$|\alpha> \Delta$

$\implies$ $|\psi_1> = \sin (2\phi) |\alpha> + \cos (2\phi) |\beta>.$
Next iteration:

\[ | \psi_2 \rangle = -e^{i \theta} \frac{| \psi_1 \rangle}{| x_2 \rangle} \]

\[ = | \psi_2 \rangle = \sin \left( 5\varphi \right) | x \rangle + \cos \left( 5\varphi \right) | \beta \rangle. \]

\[ = | \psi_k \rangle = \sin \left( (2k+1)\varphi \right) x + \cos \left( (2k+1)\varphi \right) | \beta \rangle. \]

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

\[ \Rightarrow \text{measurement will yield high prob. yield } | x \rangle = | k \rangle! \]

We have:

\[ \sin \varphi = \frac{\sin \varphi}{\cos \varphi} = \frac{\sqrt{n}}{\sqrt{1 - \frac{1}{N}}} = \frac{1}{\sqrt{N-1}} \]

\[ \Rightarrow \theta \approx \frac{1}{N} \text{ for large } N. \]

\[ \Rightarrow k \approx \frac{\pi}{4} \sqrt{N} \]
$\Rightarrow O(n)$ steps sufficient!

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

Note: If $K$ solutions, same method works in $O\left(\frac{K}{\sqrt{n}}\right)$ steps.

Can also be adapted to the case where $K$ is unknown.

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

Shor's alg.: Use $H^\otimes n = \text{Fourier} \otimes 2$ to find period $r$ in $(\mathbb{Z}_2)^\otimes n$.

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

$\Rightarrow$ Can it find periods?

$\Rightarrow$ Applications?
The Quantum Fourier Transform (QFT)

\[ x = (x_0, \ldots, x_{n-1}) \in \mathbb{C}^n \]
\[ y = (y_0, \ldots, y_{n-1}) \in \mathbb{C}^n \]

\[ y \text{ is } \mathcal{F}_n \text{ of } x : \quad y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / n} \]

\[ \mathcal{QFT}: \quad |j\rangle \rightarrow \frac{1}{\sqrt{n}} \sum_{k=0}^{N-1} e^{2\pi i j k / n} |k\rangle \]

(Equiv.: \[ \sum_{j=0}^{N-1} x_j |j\rangle \rightarrow \sum_{j,k} x_j e^{2\pi i j k / n} |k\rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} y_k |k\rangle \])

**Classical FFT:** \( O(N^2) \) operations.

With \( N=2^n \) \( \Rightarrow \) exp. scaling in \( \# \text{ of bits } n \).

Better: \( \mathcal{FFT} (\text{fast FFT}): O(N \log N) \), but still exp. in \( n \).

Will see: QFT can be implemented efficiently - in \( O(n^2) \) steps!
Revisit QFT:

- Let $N = 2^n$

- Work with\( j \)'s: \( j' = j_1 j_2 \ldots j_N = j_1 2^{n-1} + j_2 2^{n-2} + \ldots + j_N 2^0 \)

**Decimal point:** \( 0.j_1 j_2 \ldots j_N = \frac{1}{2} j' + \frac{1}{2} (j_1 + \ldots + \frac{1}{2^{n-1}} j_N) \).

Then,

\[
| j' \rangle \mapsto \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j' k / 2^n} | k \rangle
\]

\[
= \frac{1}{2^{n/2}} \sum_{k_{n-1}}^1 \sum_{k_{n-2}}^1 \cdots \sum_{k_1}^1 \sum_{k_0}^1 e^{2\pi i j' \left( \sum_{e=0}^n k_e 2^{-e} \right)} | k_{n-1} k_{n-2} \ldots k_0 \rangle
\]

\[
= \frac{1}{2^{n/2}} \sum_{k_{n-1}}^1 \sum_{k_{n-2}}^1 \cdots \sum_{k_1}^1 \sum_{k_0}^1 \bigotimes_{e=1}^n \left( e^{2\pi i j' k_e 2^{-e}} | k_e \rangle \right)
\]

\[
= \bigotimes_{e=1}^n \left[ \frac{1}{2^{n/2}} \sum_{k_{n-e}}^1 e^{2\pi i j' k_e 2^{-e}} | k_e \rangle \right]
\]

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

Use: \( j' 2^{-e} = j_1 j_2 \ldots j_{n-e} j_{n-e+1} \ldots j_N \) \[
 e^{2\pi i j' k_{n-e} 2^{-e}} = 1
\]

\[
= \frac{| 0 \rangle + e^{2\pi i j' j_N} | 1 \rangle}{\sqrt{2}} \ldots \frac{| 0 \rangle + e^{2\pi i j' j_1} | 1 \rangle}{\sqrt{2}} \ldots \frac{| 0 \rangle + e^{2\pi i j' j_{n-1}} | 1 \rangle}{\sqrt{2}}
\]
How to build circuit?

Start with rightmost term:

$$10\gamma + e^{2\alpha_i \sigma_i j_i} |\gamma\rangle$$

\[
\sqrt{2}
\]

$$R_d = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\alpha_i} e^{-\left(\alpha_i j_i\right)} \end{pmatrix}$$

- \text{H : } |j_1\rangle \longrightarrow |0\rangle + e^{2\alpha_1 \sigma_1 j_1} |j_1\rangle

- \text{C-R}_{1}: \left(|0\rangle + e^{2\alpha_1 \sigma_1 j_1} |1\rangle\right) |j_2\rangle \longrightarrow \left(|0\rangle + e^{2\alpha_1 \sigma_1 j_1} |1\rangle\right) |j_2\rangle

- \text{C-R}_{2}: \left(|0\rangle + e^{2\alpha_1 \sigma_1 j_1} |1\rangle\right) |j_2\rangle |j_3\rangle \longrightarrow \left(|0\rangle + e^{2\alpha_1 \sigma_1 j_2 j_3} |1\rangle\right) |j_2\rangle |j_3\rangle

\text{ etc. }

\Rightarrow \text{ obtain a black box QFT a } \text{rt } j_2 \text{ bit?}

Continue like that:

\[
\begin{array}{c}
|j_1\rangle \\
|j_2\rangle \\
|j_3\rangle \\
|\vdots\rangle \\
|j_n\rangle
\end{array}
\]
Notes:
- Output qubits are in reverse order. (Reversing: $O(n^2)$ qubits.)
- Since $\frac{\mu(n)}{2} = O(n^2)$ -> lead only dep.
- On classical value of upper qubit ->

If we need to measure registers after QFT, we can measure after $H$ & control $|1\rangle$ from meas. outcome:

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

5) Period finding:

Use of QFT: finding periods (cf. Simon)
Consider \( f: S_0, 1^u \to S_0, 1^u \) s.t. \( \exists r > 0 \) s.t., (119) 

\[ f(x) = f(x + r). \quad (\text{And otherwise } f(x) \neq f(y)). \]

Can we find \( r \)? — Assume \( r \ll 2^u \).

Use \( \| \phi \|_A \rightarrow \| \phi \|_B \) if \( \| \phi \|_B > \| \phi \|_A \).

1. \[
\frac{1}{2^u/2} \sum_{x \in A} |x \rangle \left\langle 0 \right|_B \rightarrow \frac{1}{2^u/2} \sum_{x \in A} \sum_{y \in f(x)} |x \rangle \left\langle y \right|_B.
\]

2. Measure \( |f(x)\rangle_\lambda \rightarrow A \) collapses to

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

(Not: As a lemma, we can omit this step.)

3. Apply QFT:

\[
\frac{1}{2^{u/2} \sqrt{k_0}} \sum_{k=0}^{k_0-1} \sum_{l=0}^{2^u-1} e^{2\pi i \left( k_0 + kr \right) l/2^u} |k\rangle
\]

\[
= \frac{2^u-1}{2\pi i x_0 \left\langle e^{2\pi i x_0 / 2^u} \right\rangle} \sum_{k=0}^{k_0-1} \frac{1}{2^{u/2} k_0} e^{2\pi i k r e^{2\pi i / 2^u}} |e\rangle
\]

\[= \phi_e \]
Prob. distr. $|ae|^2$ of meas. $|e|$: Should be centered around $l$ s.t.
\[ \frac{r^2}{2^n} \text{ is integer}. \]

More precisely:

Consider only $l$ s.t.
\[ l = \frac{2^n}{r} \cdot s + \frac{r}{2^n} d_s \quad |d_s| \leq \frac{1}{2}; \quad s = 0, \ldots, r-1 \]

\[ \Rightarrow \quad ae = \frac{1}{2^{n/2} k_0} \sum_{k=0}^{k_0-1} e^{2\pi i \frac{r}{2^n} d_s} \]

\[ = \frac{1}{2^{n/2} k_0} \frac{e^{2\pi i \frac{r}{2^n} d_s k_0} - 1}{e^{2\pi i \frac{r}{2^n} d_s} - 1} \]

\[ \frac{k_0}{r} - 1 \leq k_0 \leq \frac{2^n}{r} \quad r \ll 2^n \Rightarrow \frac{k_0 r}{2^n} \approx 1 \]

\[ \approx \frac{1}{2^{n/2} k_0} \frac{e^{2\pi i d_s} - 1}{e^{2\pi i \frac{r}{2^n} d_s} - 1} \]

\[ \sin x \approx \frac{x}{\pi/2} \]

\[ \Rightarrow \quad |ae|^2 = \frac{1}{2^n k_0} \left( \frac{\sin \left( \frac{\pi r}{2^n} d_s \right)}{\pi \left( \frac{\pi}{2^n} d_s \right)} \right)^2 \approx \frac{1}{2^n k_0} \]

\[ \frac{\pi^2 d_s^2}{\pi^{n/2} \frac{2^n}{2^{n/2}} \frac{d_s}{\pi/2}} \]
Since \( s = 0, \ldots, r-1 \):

\[
\Pr( | e - \frac{2^u}{r} s | < \frac{1}{2} ) \geq \frac{4}{\pi^2} \approx 0.41
\]

⇒ with high prob., we obtain an \( e \) s.t.

\[
\frac{e}{2^u} \approx \frac{s}{r}
\]

⇒ can be used to determine \( \frac{s}{r} \) w.s.l.p.

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

⇒ Quantum algorithm for period finding!
Factoring:

One use of period finding: factoring.

Problem: Given \( N \) (not prime). Find \( x \) s.t. \( x \mod N \) divides \( 1 \).

Algorithm:

1. Select random \( q \), \( 2 \leq q \leq N \).

   \[ \gcd(q,N) > 1 \implies \text{fail.} \]

   \[ \text{if } \gcd(q,N) = 1 \]

   Assume \( \gcd(q,N) = 1 \).

2. The smallest \( r \) with \( q^r \mod N = 1 \) is called order of \( q \mod N \).

Note: Existence follows since \( C = \{ a \mid a < N, \gcd(a,N) = 1 \} \) is a group: if \( ab \mod N = ab' \mod N \implies a(b-b') \mod N = 0 \implies N \mid b-b' \implies b = b' \).

Thus, \( \forall a \implies ab \mod N \) is bijective, and thus \( \exists b \text{ s.t. } ab \mod N = 1 \).
\( r \) is the period of \( f_{N, a}(x) = a^x \mod N \).

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

With \( x = x_{m-1} 2^{m-1} + x_{m-2} 2^{m-2} + \ldots + x_0 \),

\[
a^x \mod N = \underbrace{\left( \frac{a}{a} \right)^{x_{m-1}}} \cdot \underbrace{\left( \frac{a}{a^2} \right)^{x_{m-2}}} \cdots \underbrace{\left( \frac{a}{a^r} \right)^{x_0}} \mod N,
\]

eff. computed by \( a \mapsto a^2 \mapsto (a^2)^2 \mapsto \ldots \)

\( \Rightarrow \) \( r \) can be found eff. w/ a quantum computer.

2. Assume \( r \) even:

\[
a^r \mod N = 1 \iff N \mid (a^r - 1) \iff
\]

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

and \( N \nmid (a^{r/2} - 1) \) (otherwise, \( a^{r/2} \mod N = 1 \frac{r}{2} \))

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

or \( N \) has \( \lcm \)-irr. common factors with both \( a^{r/2} + 1 \)

\( \Rightarrow 1 \nmid \gcd(N, a^{r/2} + 1) \mid N \)

\( \Rightarrow \) found non-trivial factor of \( N \)!
algorithm successful as long as

(i) $n$ even and (ii) $N \times (a^{\frac{n}{2}}+1)$

can be done to happen with $p \geq \frac{1}{2}$ for
random choice of $a$.
(Unless $N = p^k$, $p$ prime $\Rightarrow$ can be checked
taking logs.)

efficient quantum algorithm for factoring!