IV.2. Oracle-based algorithms

a) The Deutsch algorithm

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

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

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

How often do we have to evaluate f (= run circuit)?

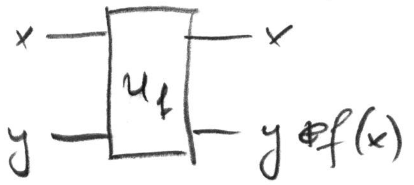
(We regard f as "black box" = "oracle":
how many queries to oracle?)

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

Can quantum mechanics do better?

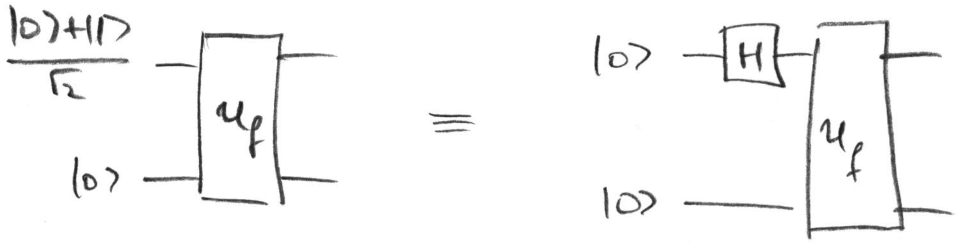
Consider reversible implementation of f:

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



$$|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$$

Try to input superpositions?



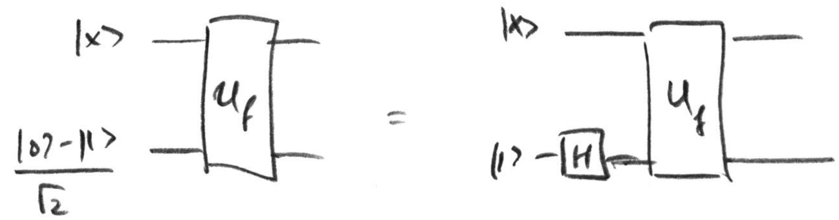
$$\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle+|1\rangle|0\rangle\right) \xmapsto{U_f} \frac{1}{\sqrt{2}}\left(|0\rangle|f(0)\rangle+|1\rangle|f(1)\rangle\right)$$

$\longrightarrow$ Have evaluated f on <u>both inputs</u>!

But: how can we <u>extract</u> relevant <u>information</u>?

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

Consider instead

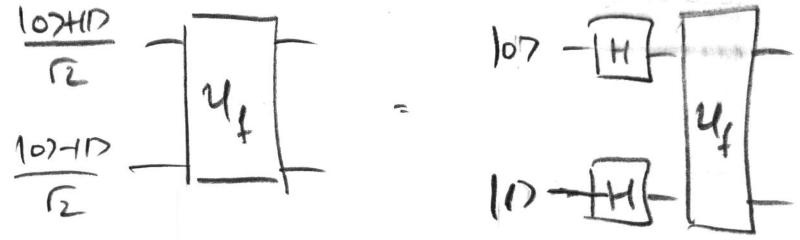


$$|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \xmapsto{U_f} |x\rangle\left(\frac{|f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}\right) =$$

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

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

Combine:



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

→ no entanglement created (!)

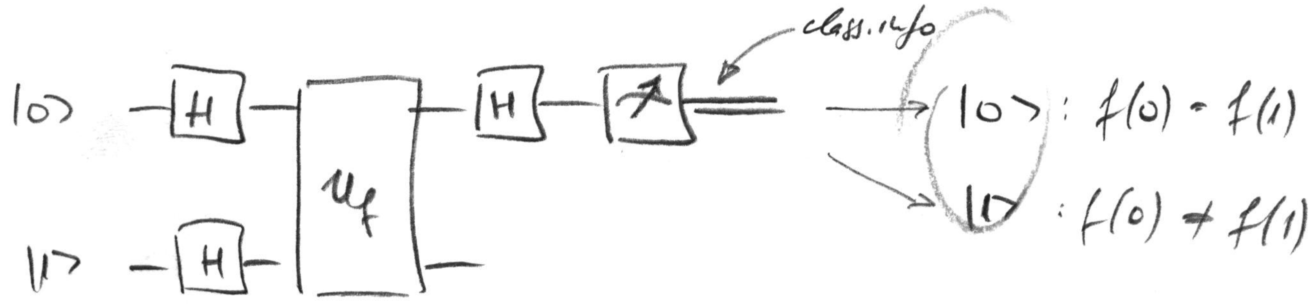
→ 2nd qubit unchanged (!!)

→ 1st qubit gets phase $(-1)^{f(x)}$

⇒ "phase kick-back" technique.

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

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



<u>One application</u> of U_f sufficient ⇒ speed-up w.r.t. classical algorithm!

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

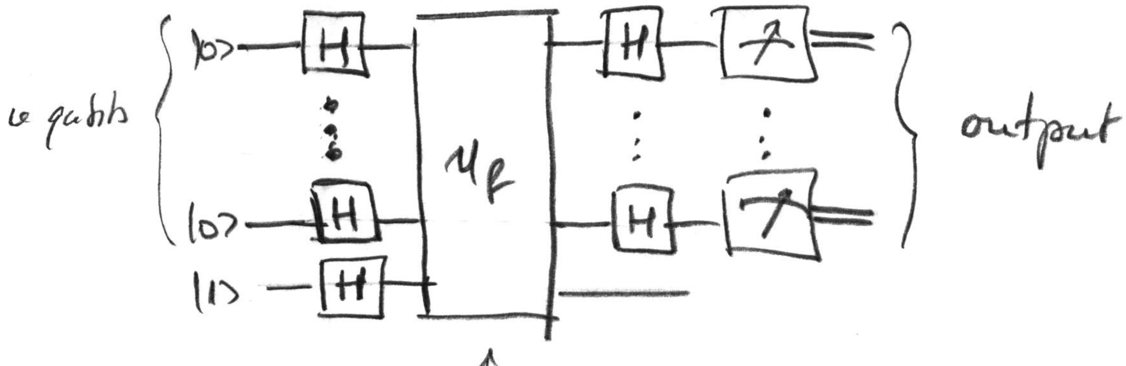
Consider $f: \{0,1\}^n \rightarrow \{0,1\}$ w/ promise that

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 $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$:



$|\underline{x}\rangle|y\rangle \longmapsto |\underline{x}\rangle|y \oplus f(x)\rangle$

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

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

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

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

where $\underline{x} \cdot \underline{y} := x_1 y_1 \oplus x_2 y_2 \oplus \dots \oplus x_n y_n$, addition mod 2

i.e. scalar prod. mod 2.

Analyze circuit: we omit normalization!

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

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

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

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

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

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

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

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

c) Simon's algorithm

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

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

$\Longrightarrow p_{success} \leq k^2 2^{-n}$

$\Longrightarrow$ 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 $\Longrightarrow$ collapse onto random $f(\underline{x}_0)$.

Register A is collapsed to

$$c \cdot \sum_{x : f(\underline{x}) = f(\underline{x}_0)} |\underline{x}\rangle = \frac{1}{\sqrt{2}} \left(|\underline{x}_0\rangle + |\underline{x}_0 \oplus \underline{a}\rangle \right)$$

How can we <u>extract a</u>? (Meas. in comp. basis collapses to $|\underline{x}_0\rangle$ or $|\underline{x}_0 \oplus \underline{a}\rangle$)

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

$$H^{\otimes n} \left(\frac{1}{\sqrt{2}} \left(|\underline{x}_0\rangle + |\underline{x}_0 \oplus \underline{a}\rangle \right) \right) = \frac{1}{\sqrt{2^{n+1}}} \sum_{y} \underbrace{\left[(-1)^{\underline{x}_0 \cdot y} + (-1)^{(\underline{x}_0 \oplus \underline{a}) \cdot y} \right]} |y\rangle$$

$$= 2 \cdot (-1)^{\underline{x}_0 \cdot y} \quad \text{if } \underline{a} \cdot y = 0$$

$$= 0 \quad \text{if } \underline{a} \cdot y = 1$$

$$= \frac{1}{\sqrt{2^{n-1}}} \sum_{y : \underline{a} \cdot y = 0} (-1)^{\underline{x}_0 \cdot y} |y\rangle$$

Measure $|y\rangle \Longrightarrow$ find random y s.th. $\underline{a} \cdot y = 0$.

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

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

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

$\Longrightarrow$ <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.th. $f(x_0)=1$.

(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 <u>unique</u>.
(Generalization: later / homework)

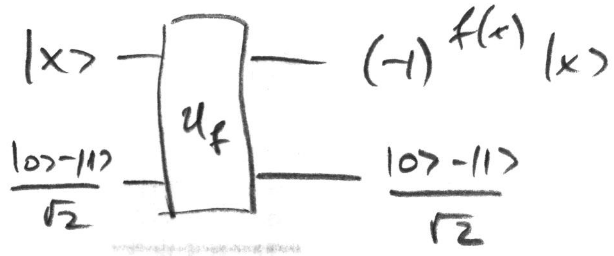
<u>Classically</u>: Need $O(N)$ queries to f for an <u>unstructured search</u> (i.e., w/out using properties of f).

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

<u>Ingredient 1</u>:

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