2. Oracle-based algorithms

a) The Deutsch algorithm

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

Let f be "very hard to compute" – e.g. long circuit

Want to know: Is $f(0) = f(1)$?

(e.g.: will a specific chess move affect result?)

How often do we have to run the circuit for f

(= "evaluate f")? – We think of f as a "black box"

or "oracle": How many oracle queries are needed?

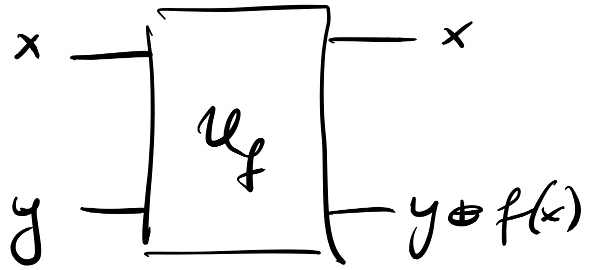
Classically, we clearly need 2 queries:

compute $f(0)$ and $f(1)$.

Can quantum physics help?

Consider reversible implementation of f:

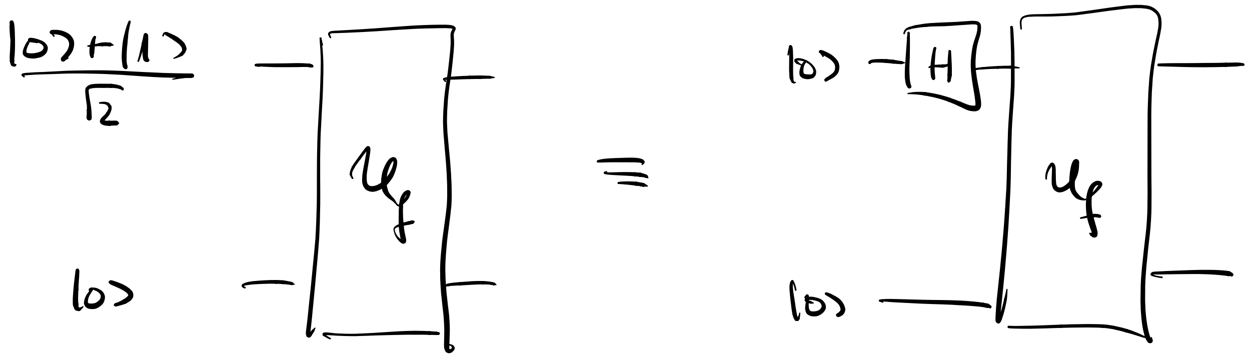
$$f^R: (x,y) \longmapsto (x, y \oplus f(x))$$



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

Try to use superpositions as inputs?

First attempt:



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

→ Have evaluated f on both outputs!

But how can we extract the relevant information (i.e. do a measurement)?

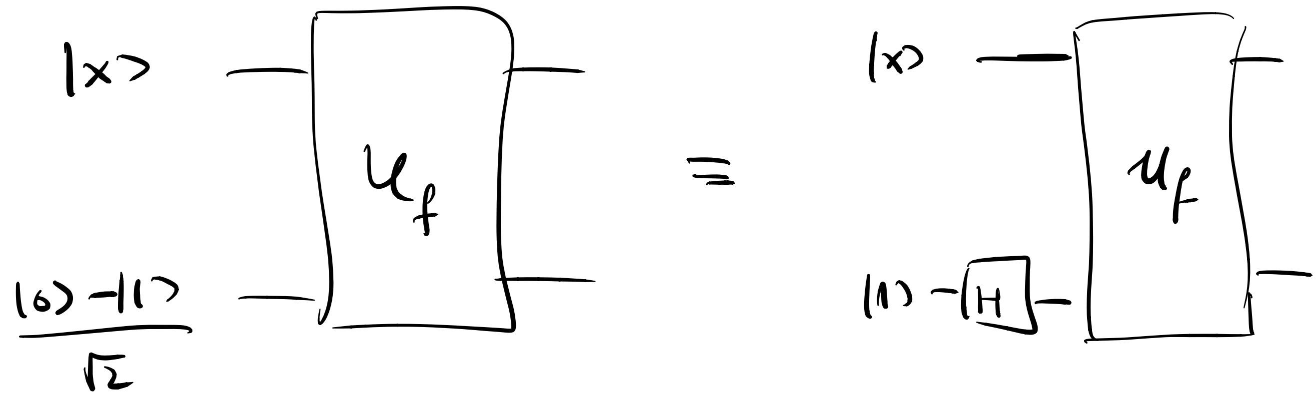
- Meas. in comp. basis: collapse superpos. to one case!
- Generally: $f(0) \neq f(1)$: outputs $\frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle + |1\rangle|1\rangle\right)$,

  $\frac{1}{\sqrt{2}}\left(|0\rangle|1\rangle + |1\rangle|0\rangle\right)$,

$\underline{f(0) = f(1)}$: outputs $|+\rangle|0\rangle$,
$|+\rangle|1\rangle$.

$\Longrightarrow$ not orthogonal, i.e. not (determ.) distinguishable!

<u>Second attempt:</u>



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

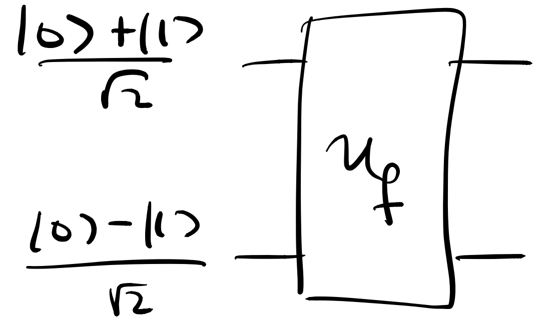
$$= \begin{cases} f(x) = 0: & |x\rangle \dfrac{|0\rangle - |1\rangle}{\sqrt{2}} \\[2ex] f(x) = 1: & |x\rangle \dfrac{|1\rangle - |0\rangle}{\sqrt{2}} \end{cases}$$

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

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

Not useful by itself: $f(x)$ only encoded in <u>global phase</u> for each <u>classical input</u> <u>$|x\rangle$</u>.

<u>Combine attempts:</u>



:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \left(|0\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} + |1\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

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

$$= \frac{(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

<u>Observations:</u>

$\longrightarrow$ No entanglement created (!)

$\to$ 2nd qubit – the one where U_f outputs the function value – is unchanged (!!)

$\to$ 1st qubit gets a phase $(-1)^{f(x)}$

("phase kick-back technique")

State of 1st qubit:

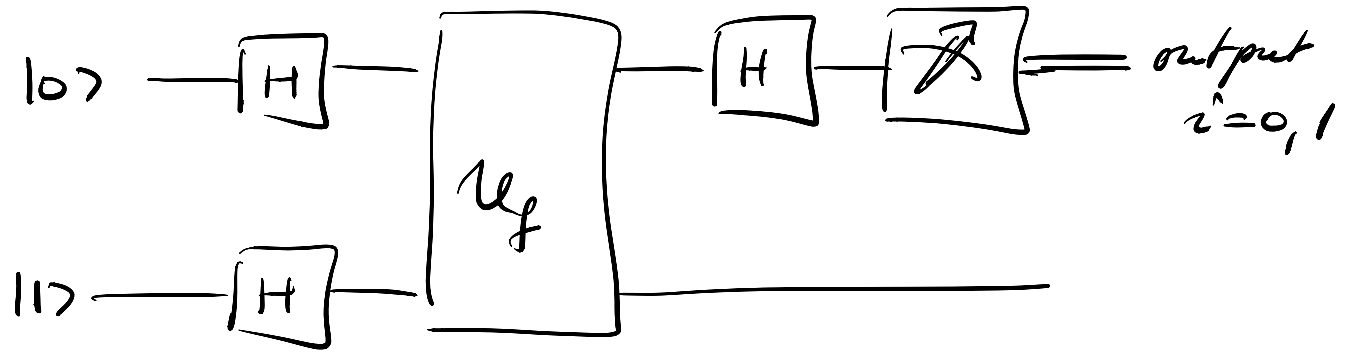
$$f(0) = f(1) \Longleftrightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

(up to irrelevant global phase)

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

Orthogonal states! $\Longrightarrow$ measurement of 1st qubit in basis $\{|+\rangle, |-\rangle\}$ (or apply $-\boxed{H}-$ & measure in $\{|0\rangle, |1\rangle\}$) allows to decide if $f(0) \overset{?}{=} f(1)$!

Deutsch algorithm:



output $i=0$: $\Rightarrow$ $f(0) = f(1)$

$i=1$: $\Rightarrow$ $f(0) \neq f(1)$

<u>One application</u> of U_f has been sufficient!

$\Rightarrow$ Speed-up compared to <u>class. algorithm</u> (1 vs. 2 oracle queries).

<u>Interesting to note:</u> 2nd qubit never needs to be measured – and it contains <u>no information</u>.

<u>Two main insights:</u>

- Use input $\sum |x\rangle$ to <u>evaluate f on all inputs simultaneously</u>.
- This parallelism alone is not enough – <u>need</u> a <u>smart</u> way to <u>read out the relevant information</u>.

However, a constant speed-up is not that impressive – in particular, it is highly architecture-dependent!

<u>Thus:</u>

b) The Deutsch-Jozsa algorithm

Consider $f: \{0,1\}^n \to \{0,1\}$ with promise (i.e., a condition we know is met by f) that

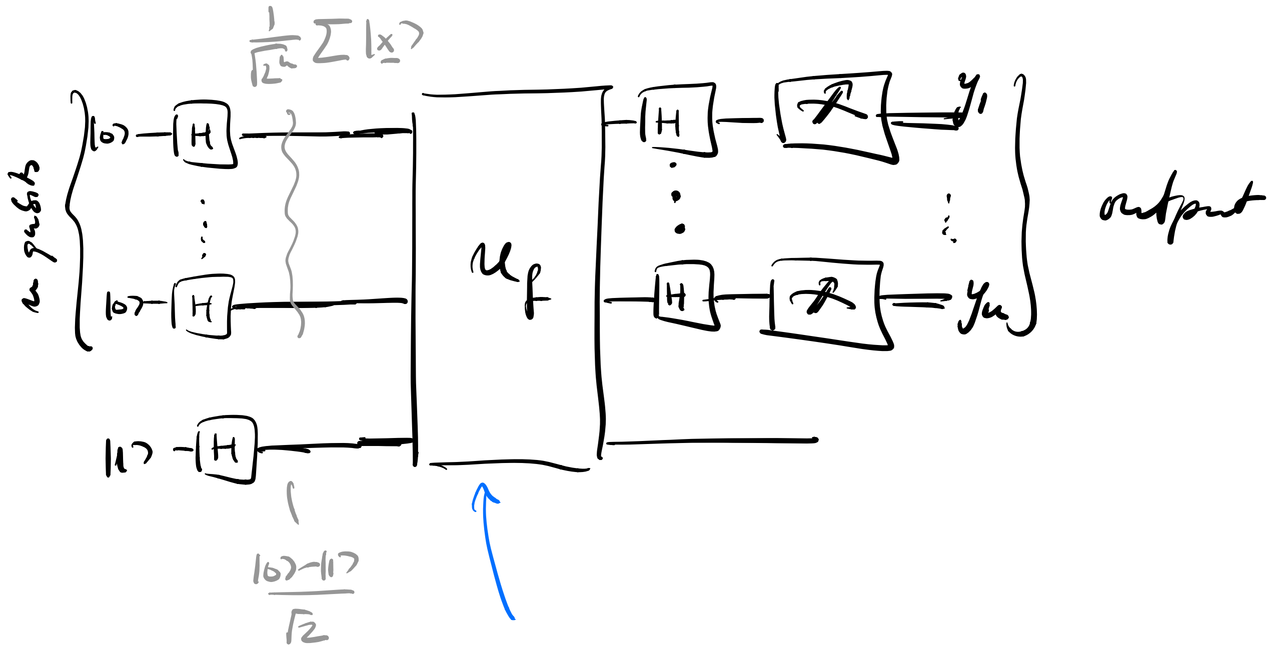
either $f(\underline{x}) = c \quad \forall \underline{x}$ ("f constant")

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

Want to know: Is f constant or balanced?

How many queries needed?

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



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

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

$$H: \; |x\rangle \longmapsto \frac{1}{\sqrt{2}} \sum_{y=0,1} (-1)^{x \cdot y} |y\rangle$$

$$H^{\otimes n}: \; |x_1, \dots, x_n\rangle \longmapsto \frac{1}{\sqrt{2^n}} \sum_{\underline{y}} (-1)^{x_1 y_1} \cdots (-1)^{x_n y_n} |y_1, \dots, y_n\rangle$$

or:

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

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

("scalar product" mod 2).

is NOT a scalar product!

Analysis of circuit:

we omit normalization!

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

phase kick-back

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

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

$p_{\underline{y}} := |a_{\underline{y}}|^2$ is the probability to measure $\underline{y} = (y_1, \dots, y_n)$.

<u>f constant:</u> $f(\underline{x}) = c$

$$a_{\underline{y}} = (-1)^c \underbrace{\sum_{\underline{x}} (-1)^{\underline{x}\cdot\underline{y}}}_{\propto\, \delta_{\underline{y},\underline{0}}} = (-1)^c \delta_{\underline{y},\underline{0}}$$

<u>f balanced:</u>

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

$$= \sum_{x} (-1)^{f(\underline{x})} \overset{\uparrow}{=} 0$$

<u>f balanced!</u>

<u>Thus:</u>

Output $\underline{y} = \underline{0} \Longrightarrow$ f constant

Output $\underline{y} \neq \underline{0} \Longrightarrow$ f balanced

$\Longrightarrow$ We can <u>unambiguously</u> <u>distinguish</u> the 2 cases with <u>one query to the oracle for f!</u>

What is the speed-up vs. classical methods?

Quantum: 1 use of f.

Classical: Worst case, we have to determine

$2^{n-1}+1$ values of f to be sure!

$\Rightarrow$ exponential vs. constant!

But: If we are ok to get right answer with very high probability $p = 1 - p_{error}$, then for k queries to f,

$$p_{error} \approx 2 \cdot \underbrace{\left(\frac{1}{2}\right)^k}$$

$\approx$ prob. to get $k\times$ same outcome for balanced f, if $k \ll 2^n$.

i.e.: $k \sim \log(1/p_{error})$.

Randomized classical: Much smaller speed-up vs. randomized classical algorithm (even for exp. small error, $k \sim n$ oracle calls are sufficient.)

c) Simon's algorithm

... will give us a true exponential speedup (also rel. to <u>randomized</u> class. algorithms) in terms of oracle queries!

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

with <u>promise</u>:

$$\exists \, \underline{a} \neq \underline{0} \text{ s.th. } f(\underline{x}) = f(\underline{y}) \text{ exactly if } \underline{y} = \underline{x} \oplus \underline{a}.$$

("hidden periodicity")

Task: Find $\underline{a}$ by querying f.

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

Roughly: k queries $x_1, \dots, x_k \longrightarrow \sim k^2$ pairs,

for each pair: prob $\left(f(\underline{x}_i) = f(\underline{x}_j)\right) \approx 2^{-n}$

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

$\Rightarrow$ need $k \sim 2^{cn}$ queries!

Quantum algorithm (Simon's algorithm):

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

ii) Apply $U_f : |\underline{x}\rangle|\underline{y}\rangle \longmapsto |\underline{x}\rangle|\underline{y} \oplus f(\underline{x})\rangle$

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

iii) Measure B. $\Rightarrow$ Collapse onto random $f(\underline{x}_0)$ (and thus random $\underline{x}_0$).

$\Rightarrow$ Register A collapses onto

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

— How can we extract a? —

(Meas. in comp. basis → collapse on rand. x_0: useless.)

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

$$\boxed{H^{\otimes n} |\underline{x}\rangle \propto \sum_{\underline{y}} (-1)^{\underline{x} \cdot \underline{y}} |\underline{y}\rangle}$$

$$H^{\otimes n} \left(\frac{1}{\sqrt{2}} \left(|\underline{x}_0\rangle \oplus |\underline{x}_0 \oplus a\rangle \right) \right)$$

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

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

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

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

v) Measure in comp. basis:

$\implies$ obtain random y s.th. $\underline{a} \cdot y = 0$.

$(n-1)$ lin. indep. vectors y_i (over $\mathbb{Z}_2$) s.th. $\underline{a} \cdot y_i = 0$ allow to determine $\underline{a}$ (solve lin. eq. – e.g. Gaussian elimination).

Space of lin. dep. vectors of k vectors grows as 2^k

$\implies$ $O(1)$ chance to find randomly a lin. indep. vector

$\implies$ $O(n)$ random y are enough

$\Rightarrow$ $O(n)$ oracle queries are enough (on average)

Classical: 2^{cn} queries

Quantum: $c' \cdot n$ queries

$\Big\}$ exponential speed-up!

(in terms of oracle queries)

Notes: • We don't have to measure B — we never use the outcome! (But: Derivation easier this way!)

• $H^{\otimes n} \hat{=}$ (discrete) Fourier transform over $\mathbb{Z}_2^{\times n}$

$\longrightarrow$ period finding via Fourier transform