

III. Quantum computing & algorithms

3.1. Some elements of classical computing

key elements of modern computers go back to
Charles Babbage (1791-1871)

the "founding father" of the formal theory of computation is

Alan Turing (1912-1954)

(A) What is information?

information is notion of a priori ignorance

If a dice player tells you that he has thrown a "5"
this is only information for you if you have not
watched him throwing!

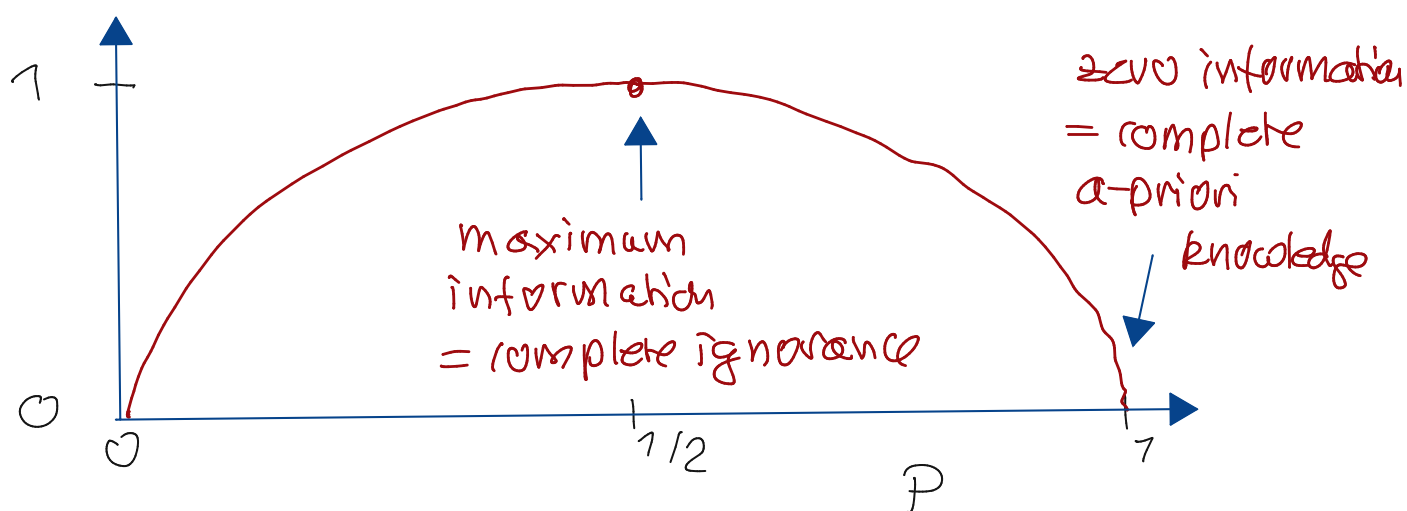
elementary unit of information is a
binary decision

(72) c bit or just bit 0 1

a quantitative measure of information is Shannon entropy (see later chapters)

a-priori probability of "0" : p
 - " - of "1" : $1-p$

$$(73) \quad H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$



(B) Computation & Turing machine

computation: systematic generation of a finite set of (binary) symbols with abstract meaning from a different set of (binary) symbols

this is equivalent to evaluating

(74) $f: \{0,1\}^n \longrightarrow \{0,1\}^m$

which can in general be reduced to a decision problem 0 = NO 1 = YES

$$(75) \quad f: \{0,1\}^n \longrightarrow \{0,1\}$$

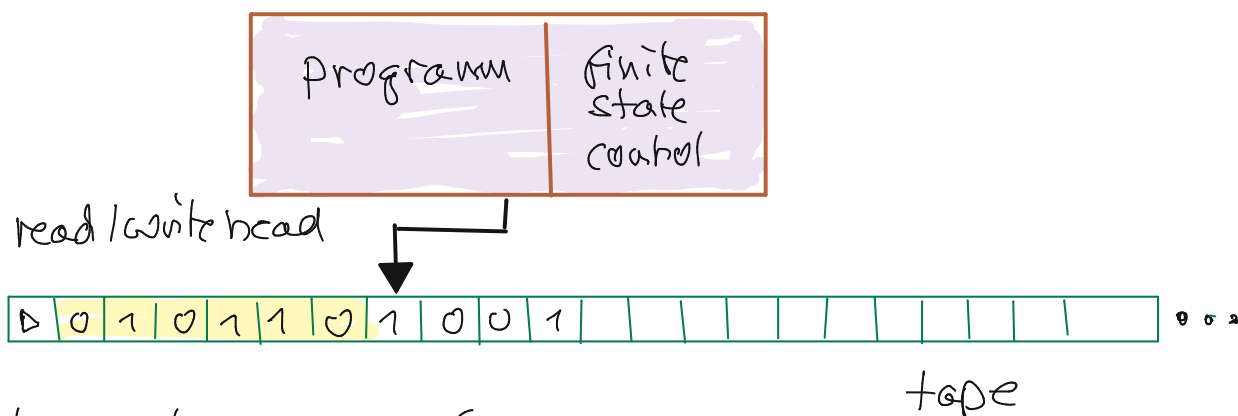
early 20th century: Hilbert's Entscheidungsproblem

Does an algorithm exist which in principle can solve all mathematical problems (decision problems)?

Hilbert conjecture: YES

Church & Turing: NO!

Turing machine (invented to solve Hilbert problem)



elements:

(A) program

(B) finite state control

(C) tape

(D) read/write head

- finite state control

finite set of internal states $q_1, q_2, \dots, q_m$

q_s starting state

q_h halting state (where machine ends up in, iff computation finishes)

- tape:

half infinite one-dimensional object with infinite sequence of tape squares labelled $0, 1, 2, 3, 4, \dots$ containing each one symbol of a finite alphabet Γ

for simplicity $\Gamma : \{0, 1, b\}$ b : "blank"
and " $\triangleright$ " to mark start at left end

- read-write head

points to one tape square

- program

finite ordered list of program lines

(q, x, q', x', Δ)

q - internal state $\longrightarrow q'$ - new internal state
 x - symbol at tape $\longrightarrow x'$ - x overwritten by x'

tape head is moving left if $S = -1$
 right if $S = +1$
 stand still if $S = 0$

(only if head is at left-most position then it stops also if $S = -1$)

- if machine does not find a program line for given q and x :

$q \longrightarrow q_h$ machine halts

Church-Turing thesis:

The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm

i.e. Turing machine is a universal computer

there are decision problems (functions) that cannot be solved on a Turing machine

"halting problem" (Turing 1936)

problem of determining from description of any computer program and arbitrary input whether the computation will stop or run forever

- remark: a machine with a finite memory will either halt or fall into periodic pattern

(C) Network model of computation

total number of decision-problem functions, eg (75)
2 outputs ; 2^n inputs $\Rightarrow 2^{2^n}$

each of these functions can be decomposed into
a sequence of elementary logic operations

input: string of bits

$$(76) \quad X = X_1 X_2 X_3 \cdots X_n \quad X_i \in \{0, 1\}$$

Since the number of decision functions is finite
we can write ("V" logic OR)

$$(77) \quad f(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee f^{(3)}(x) \cdots f^{(m)}(x)$$

where

$$(78) \quad f^{(a)}(x) = \begin{cases} 1 & \text{if } x \in X^{(a)} \\ 0 & \text{else} \end{cases}$$

i.e. $\bar{X}^{(a)}$ is the set of inputs x for which $f^{(a)}$ gives 1.

- assume $x = 1111 \dots 1 \in \bar{X}^{(a)}$ then we can write

$$(79) \quad f^{(a)}(x) = x_1 \wedge x_2 \wedge x_3 \dots \wedge x_m$$

where " $\wedge$ " logic AND

- if $x \in \bar{X}^{(a)}$ contains a "0" instead of "1" at some position then we have to write

$$(80) \quad f^{(a)}(x) = x_1 \wedge x_2 \wedge \dots \wedge (\neg x_j) \wedge \dots$$

where " $\neg$ " logic NOT

In order to evaluate eq. (77) we need an m -fold copy of the input, thus we need the operation

$$(81) \quad \text{COPY } x \rightarrow xx$$

$\Rightarrow$ to evaluate an arbitrary function

$f: \{0,1\}^n \rightarrow \{0,1\}$ we need a finite number of four elementary gates

COPY, NOT

1 bit

AND, OR

2 bit

If one is able to prepare a cbit in a certain state ($x=0$ or $x=1$) it is sufficient to have one 2-bit operation U and NOT as 1-bit operation

$$(82) \quad (x, y) \xrightarrow{U} (1-x, 1-xy)$$

is equivalent to NAND (NOT AND) if we ignore the first output. Now for $y=1$ U is equivalent to COPY if we perform additionally a bit-wise NOT. i.e.:

$$(83) \quad (x, 1) \xrightarrow{U} (1-x, 1-x) \xrightarrow[\text{NOT}]{\text{bitwise}} (x, x) \hat{=} \text{COPY}$$

U is called a universal 2-bit gate

network model of classical computation

- contains a few elementary elements
(1-bit, 2-bit gates)
- requires preparation of bits in certain states

(D) reversible computing

Many of the considered elementary gates are not reversible. Their operation is thus fundamentally

connected to entropy production and heat!

Landauer's principle

If in a computation a single bit of information is erased there is a minimum amount of energy dissipated into the environment

$$\Delta Q_{\min} = k_B T \ln 2$$

where T is the environment temperature.
Equivalently the entropy of the environment is increased by at least

$$\Delta S_{\min} = k_B \ln 2$$

Any computation has to increase the entropy of the environment, but can we make individual operations reversible and create entropy only by resetting the computer at the end of the computation?

reversible computation requires evaluation of invertible functions

$$(84) f: \{0,1\}^n \longrightarrow \{0,1\}^n \quad \swarrow$$

every function can be made reversible by addition of ancillas

$$f(x): \{0,1\}^n \rightarrow \{0,1\}^m \quad m \neq n$$

$$\tilde{f}: \{0,1\}^{n+m} \rightarrow \{0,1\}^{m+n}$$

where

$$(85) \quad \tilde{f}(\underbrace{x}_{n}, \underbrace{0,0,\dots,0}_m) = (\underbrace{x}_{n}, \underbrace{f(x)}_m)$$

unchanged: "control bit" ↑

example:

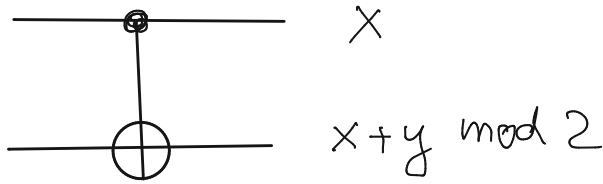
Controlled-NOT

CNOT

x ——— x

control bit

y



$x+y \bmod 2$

target bit

rem: (classical) reversible 1 or 2-bit gates are not universal since their operation can be expressed as linear matrix operation

$$(86) \quad \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}$$

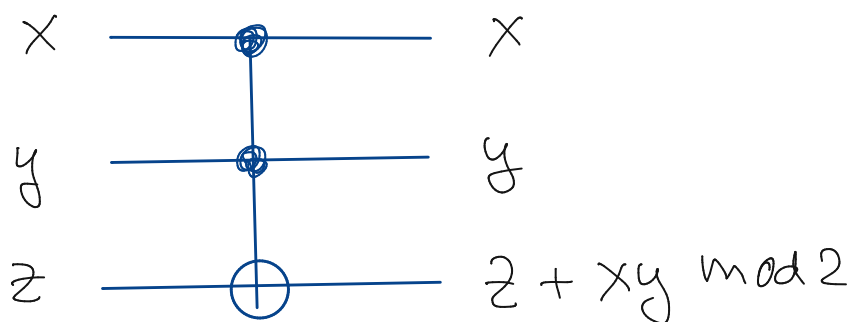
with $\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and

$$(†) \underline{M} \in \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \right\}$$

nonlinear functions $\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix}$ cannot

be represented! However, reversible n -bit gates with $n \geq 3$ are universal.

example: Toffoli gate controlled-controlled NOT



3.2 Computational complexity*

* neither complete nor fully rigorous

Question: How many elementary operations are needed to evaluate $f(x)$ if the size n (# of bits) of the input is increased?

We consider the evaluation of a function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ (algorithmically) simple if the number of elementary gates needed is a polynomial in n . These functions (problems) are said to belong to the complexity class

P = decision problems with n bit's input that require $\text{poly}(n)$ elementary gates to evaluate

All problems not belonging to this class are called (algorithmically) difficult or hard

The class of problems for which the verification of the result $f(x)=1$ is simple in the above sense

NP = decision problems with n bit's of input where the "yes" or "1" output can easily be verified by a witness

The verification of "1" output of a decision problem is equivalent of the verification of the result of a mathematical problem. E.g.

finding the prime factors of an integer is believed to be a hard problem. The verification is however simple as multiplication is simple.

example: travelling salesman problem

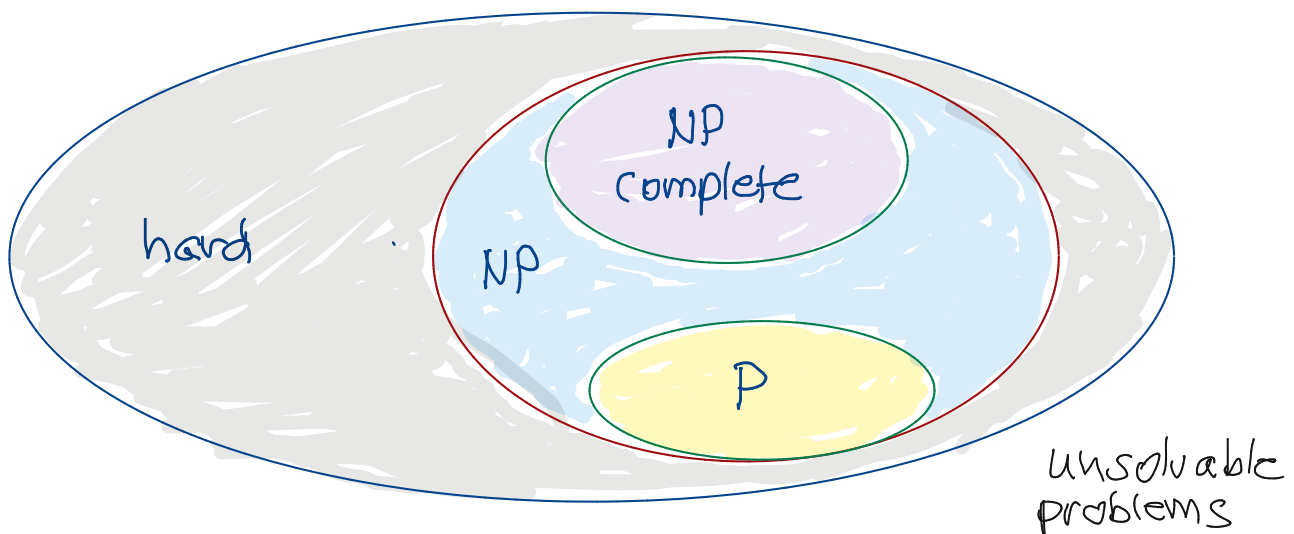
There is a special subset of NP problems, called

NP complete: problems in NP to each of (NPC) which any other NP problem can be reduced to with polynomial effort

obviously $P \subseteq NP$

it is however still an open problem if

$P \neq NP$?



example 1 for NPC: CIRCUIT-SAT

$$(881) \quad C[f(x)] = \begin{cases} 1 & \text{if } x \text{ exists such that } f(x)=1 \\ 0 & \text{if no such } x \text{ exists} \end{cases}$$

Theorem of Cook (1971): (NP completeness)

every problem in NP can be reduced to CIRCUIT-SAT with polynomial effort

so if CIRCUIT-SAT $\in P \Rightarrow NP = P$
(the same holds for any NP complete problem)

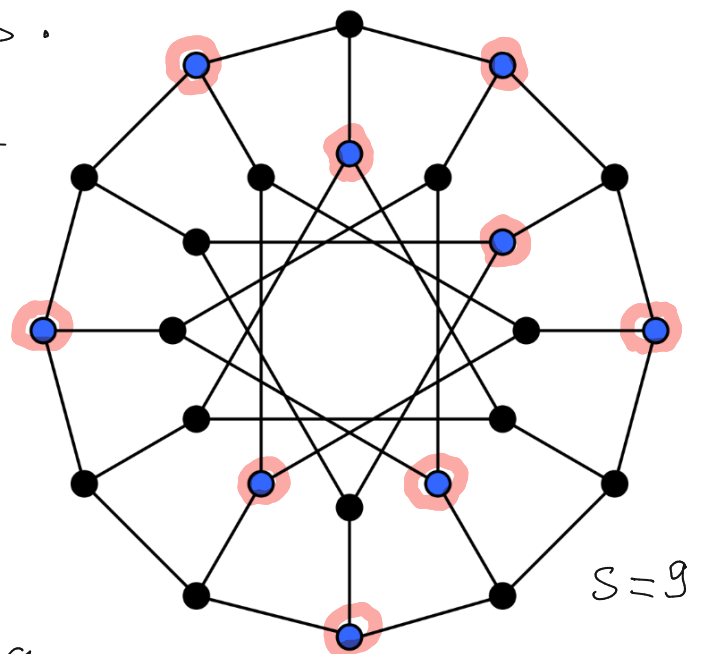
example 2 for NPC: Maximum independent set

Problem in graph theory: independent set is a set of vertices in a graph that are not pairwise directly connected. Size of an independent set = number of vertices.

maximum independent set
= maximum size

Strongly believed to
be NP but not P

from Wikipedia



- factorization of integers into prime factors is believed to be not NP complete.

So far we have considered deterministic computation. It turns out however that a computation may be more efficient if it uses elements of randomness. Such a probabilistic evaluation of a decision problem gives the correct result only with finite probability

Chererkov bound

Suppose we have an algorithm for a decision problem which gives the correct answer with probability $1/2 + \epsilon$ and the wrong answer with probability $1/2 - \epsilon$. Then n -fold repetition yields an exponential improvement

$X_1, \dots, X_n \in [0, 1]$ independent random variables

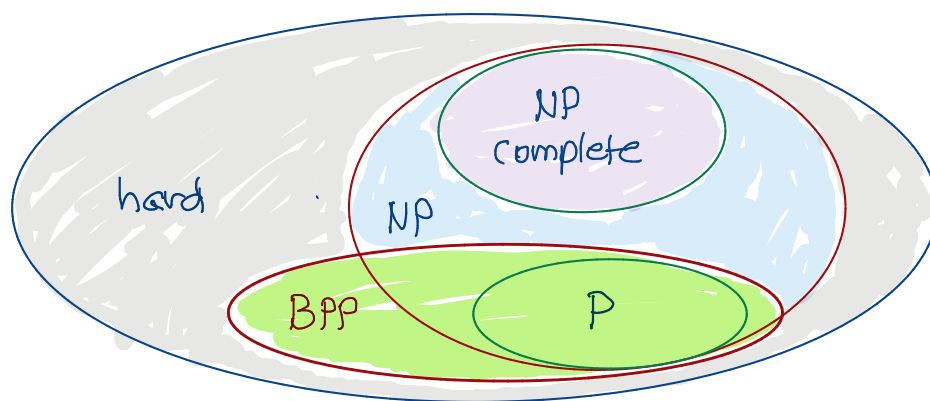
$$P(X_i = 0) = \frac{1}{2} - \epsilon$$

$$(89) \Rightarrow P\left(\frac{1}{n} \sum_{i=1}^n X_i \leq \frac{1}{2}\right) \leq e^{-2\epsilon^2 n}$$

The class of problems that can be solved with probabilistic calculations with polynomial effort and bounded error probability is called

BPP = bounded error, probabilistic with polynomial effort

The general relation between the classes is not fully known



3.3 Quantum circuit computation

(A) Qubits

quantum system with smallest non-trivial Hilbert space $d=2$

- spin $\frac{1}{2}$
- two-level atom
- polarization states of a photon

computational basis $\{ |0\rangle, |1\rangle \}$

$\langle 0|1\rangle = 0$ e.g. eigenstates of $\hat{\sigma}_z$

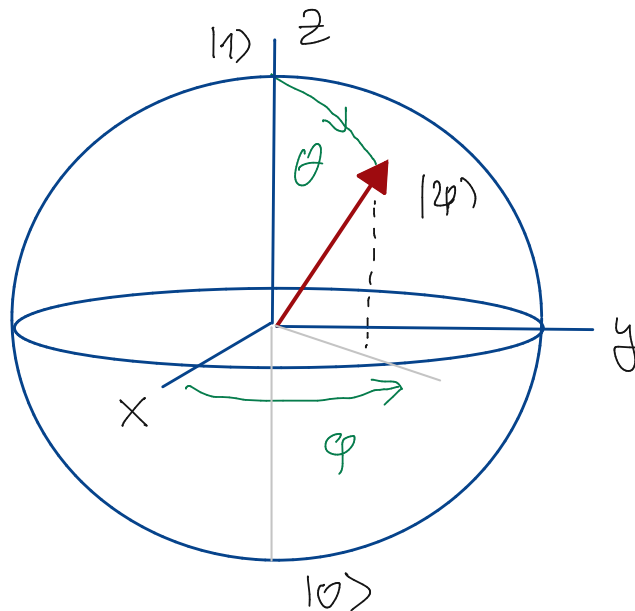
general state

$$(90) \quad |\psi\rangle = e^{i\varphi} \left(\cos\frac{\theta}{2} |1\rangle + e^{i\varphi} \sin\frac{\theta}{2} |0\rangle \right)$$

irrelevant

(θ, φ) characterize qubit

Bloch sphere



peculiarities of a qubit vs. bit:

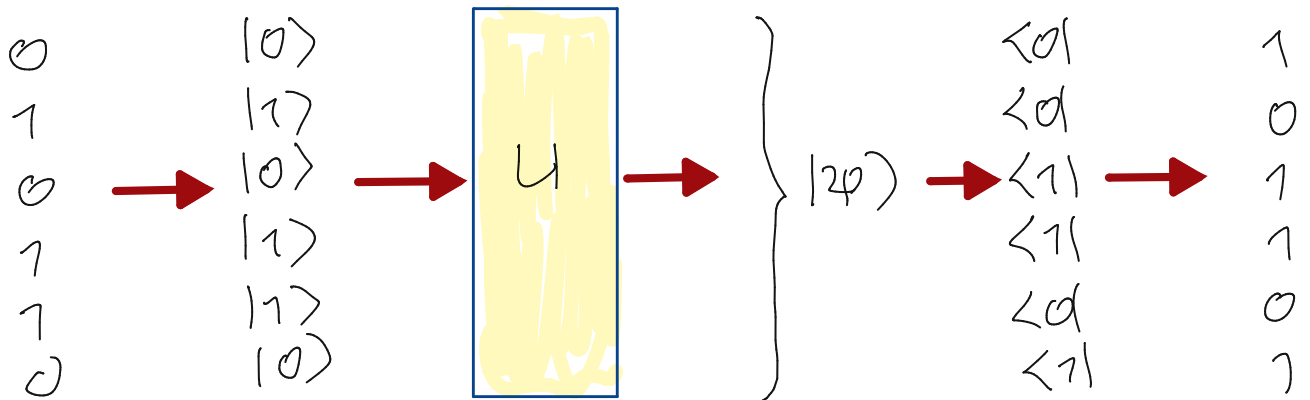
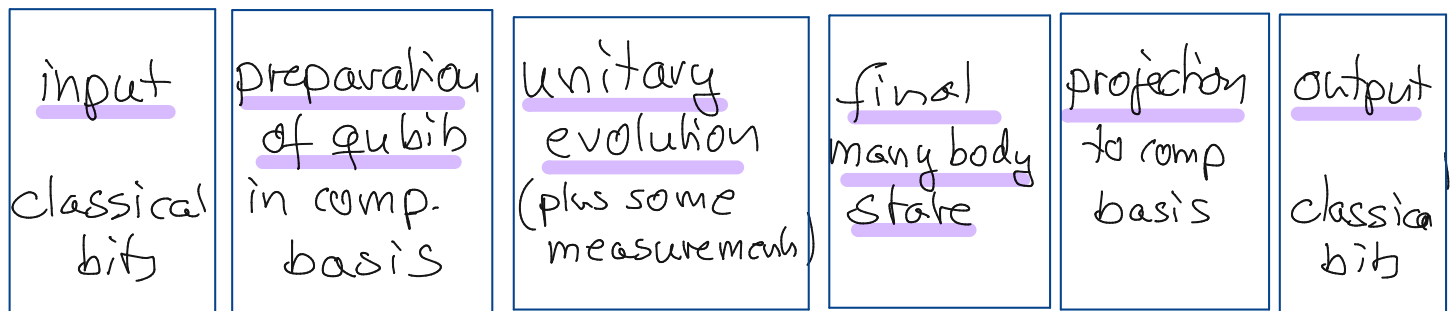
superpositions !

$\Rightarrow$ many more possibilities

However: information that can be encoded and/or read out from a single qubit is only one bit of information !

Still in intermediate operations superpositions and in particular many qubit superpositions i.e. entanglement can improve computations.

(B) Quantum circuit model

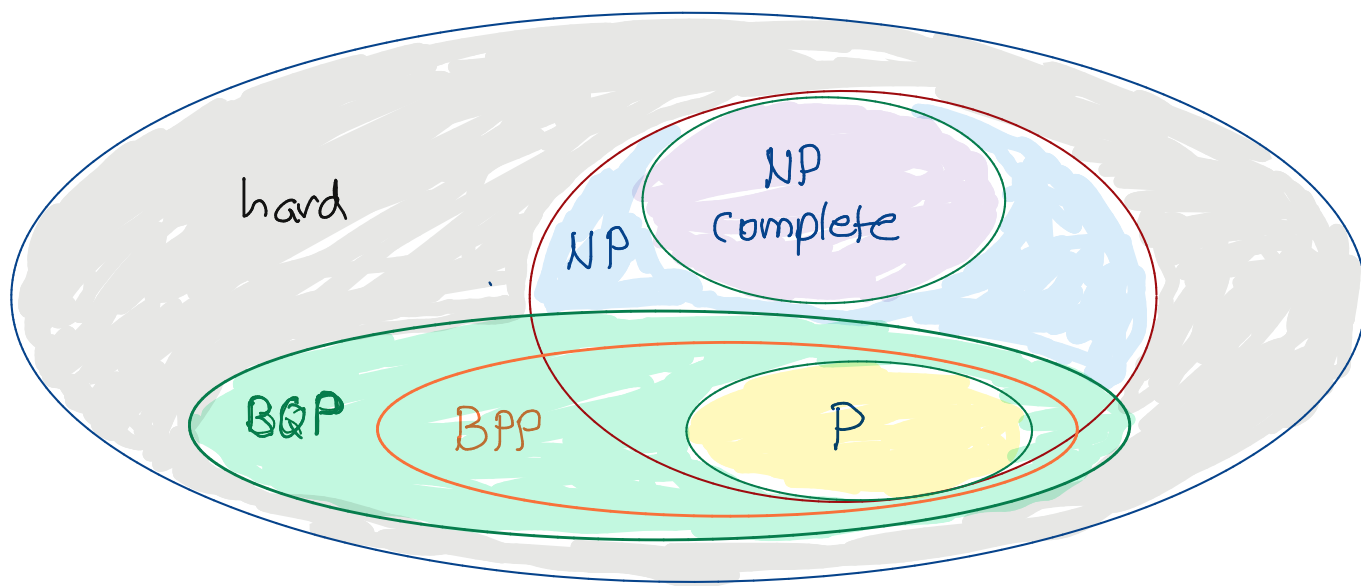


the output is in general not deterministic but probabilistic

It is believed that the class of problems that can be solved probabilistically with bounded error on a quantum computer called

BQP bounded error probabilistic, quantum with polynomial effort

is larger than BPP! -53-



decomposition of unitary operation into elementary units

Deutsch (1985)

Any d -dimensional unitary matrix can be decomposed into at most $d(d-1)/2$ unitary 2×2 matrices

unitary 2×2 matrices: (also hermitian)

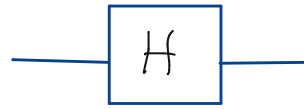
$$(91) \quad \mathbb{I}, \hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z$$

Since the 2×2 matrices can also act in the space spanned by states of two different qubits we need

1-qubit and 2-qubit gates

important 1-qubit gates

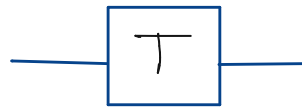
Hadamard gate



$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix}$$

$$(92) \{ |0\rangle, |1\rangle \} \xrightarrow{H} \left\{ \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \right\}$$

$\pi/8$ gate



$$e^{i\frac{\pi}{8}} \begin{pmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{pmatrix}$$

The combination of Hadamard and $\frac{\pi}{8}$ gates allows to approximate any single qubit gate with arbitrary precision

other single qubit gates

$$X \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Pauli X

$$Y \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Pauli Y

$$Z \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

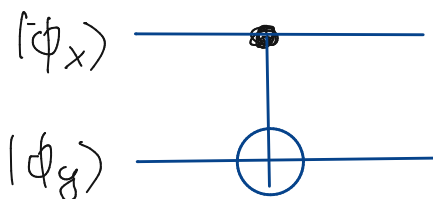
Pauli Z

$$S \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

Phase

important 2-qubit gate

CNOT gate



$$\begin{pmatrix} \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \end{pmatrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix}$$

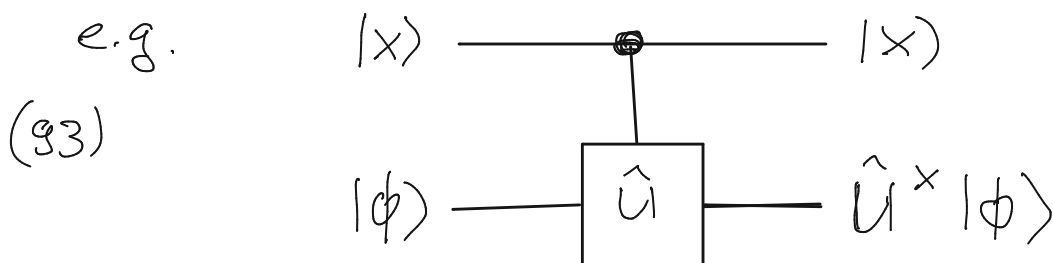
arbitrary single qubit plus CNOT gate are universal

note that in the classical gate we needed 3 bit gates (e.g. Toffoli)

How to evaluate the action of a 2-qubit gate on a general quantum state? Map is linear in Hilbert space $\Rightarrow$ sufficient to know action on basis states, i.e.

$$|\phi_x\rangle \in \{|x=0\rangle, |x=1\rangle\}$$

$$|\phi_y\rangle \in \{|y=0\rangle, |y=1\rangle\}$$



(C) Deutsch and Deutsch-Josza algorithm

We now consider an instructive example for the potential power of quantum computation! A problem that can be solved using a deterministic quantum algorithm more efficiently than with a deterministic classical one is the following

given a function $f: \{0,1\} \rightarrow \{0,1\}$
 every such function belongs to one of the
 two classes

$$(94) \quad \begin{array}{ll} f(0)=0 & \text{or} \quad f(0)=1 \\ f(1)=0 & f(1)=1 \end{array} \quad \underline{f: \text{constant}}$$

$$\begin{array}{ll} f(0)=0 & \text{or} \quad f(0)=1 \\ f(1)=1 & f(1)=0 \end{array} \quad \underline{f: \text{balanced}}$$

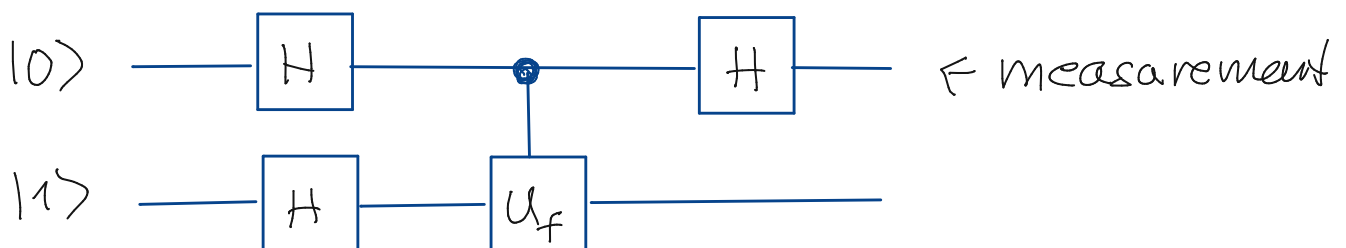
Although the classification of f into "constant" or "balanced" is only one bit of information one has to evaluate f twice to make such a classification classically.

Quantum mechanically we can make use of the possibility of superposition to reduce this to one evaluation

f is encoded in $U_f \quad |x\rangle, |y\rangle$ in comp. basis

$$(95) \quad U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

use circuit (Deutsch)



$$(96) \quad |0\rangle|1\rangle \xrightarrow{\text{1st H gate}} \frac{1}{2} (|0\rangle + |1\rangle) (|0\rangle - |1\rangle) = \frac{1}{2} \sum_{x=0}^1 |x\rangle (|0\rangle - |1\rangle)$$

$$(97) \quad U_f |x\rangle (|0\rangle - |1\rangle) = \begin{cases} |x\rangle (|0\rangle - |1\rangle) & \text{iff } f(x)=0 \\ -|x\rangle (|0\rangle - |1\rangle) & \text{iff } f(x)=1 \end{cases}$$

$$= (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle)$$

i.e.

$$\frac{1}{2} \sum_{x=0}^1 |x\rangle (|0\rangle - |1\rangle) \xrightarrow{U_f} \frac{1}{2} \sum_{x=0}^1 (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle)$$

$$(98) \quad = \pm \frac{1}{2} \begin{cases} (|0\rangle + |1\rangle) (|0\rangle - |1\rangle) & f: \text{constant} \\ (|0\rangle - |1\rangle) (|0\rangle - |1\rangle) & f: \text{balanced} \end{cases}$$

$$(99) \quad \xrightarrow{\text{2nd H}} = \pm \frac{1}{\sqrt{2}} \begin{cases} |0\rangle (|0\rangle - |1\rangle) & f: \text{constant} \\ |1\rangle (|0\rangle - |1\rangle) & f: \text{balanced} \end{cases}$$

one measurement of 1st qubit sufficient

Speedup by superposition: quantum parallelism

first experiment: Gold et al Nature 421, 48 (2003)

The effect is even more dramatic if we consider

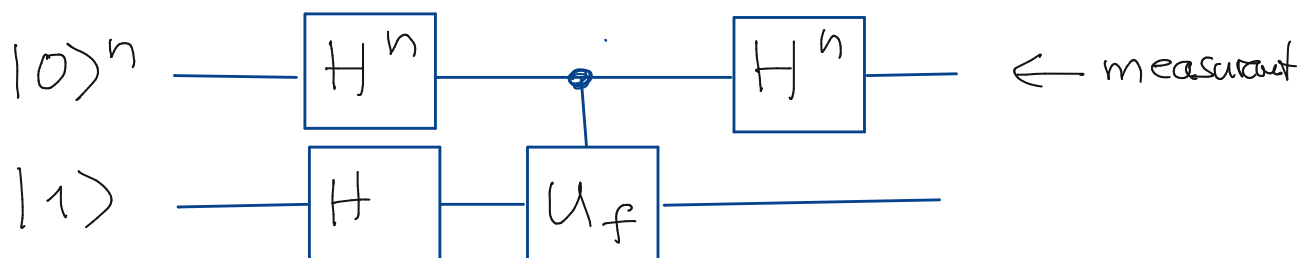
$$(100) \quad f: \{0,1\}^n \longrightarrow \{0,1\} \quad (\text{Deutsch-Jozsa})$$

- SP -

Let f be either constant, i.e. have the same output (0 or 1) for all inputs or be balanced, i.e. have the same number of outputs "0" and "1" in the output. Here again the classification of f has only one bit of information, (of course very many possible functions of type (100) can be neither constant or balanced but we here assume that f is of either the one or the other type)

Classically we have to evaluate f $2^{n-1} - 1$ times !

Quantum mechanically a single evaluation is sufficient !!



H^n : $\underbrace{H \otimes H \otimes H \otimes \dots \otimes H}_n$ bitwise Hadamard

$$\begin{aligned}
 H^n |x\rangle &= \prod_{j=1}^n \frac{1}{\sqrt{2}} \sum_{y_j=0}^1 (-1)^{x_j y_j} |y_j\rangle \\
 (101) \quad &= \frac{1}{2^{n/2}} \sum_{\underline{y}=0}^{2^n-1} (-1)^{\underline{x} \cdot \underline{y}} |\underline{y}\rangle \\
 &\quad \text{--- 59 ---}
 \end{aligned}$$

if $|\underline{x}\rangle = |x_1\rangle|x_2\rangle\cdots|x_n\rangle \quad x_i \in \{0,1\}$

and $|\underline{y}\rangle = \underbrace{|y_1\rangle|y_2\rangle\cdots|y_n\rangle}_{2^n \text{ possibilities}} \quad y_j \in \{0,1\}$

Here $\underline{x} \cdot \underline{y} = x_1 \cdot y_1 + x_2 \cdot y_2 + \cdots$

is a bitwise addition modulo 2

gate operation:

$$\underline{|0\rangle^n |1\rangle} \xrightarrow{H^n, H} \frac{1}{2^{n/2}} \sum_{\underline{x}=0}^{2^n-1} \underline{|\underline{x}\rangle} \underline{\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)}$$

$$(102) \quad \xrightarrow{U_f} \frac{1}{2^{n/2}} \sum_{\underline{x}=0}^{2^n-1} (-1)^{f(\underline{x})} \underline{|\underline{x}\rangle} \underline{\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)}$$

(see (97))

$$\xrightarrow{H^n} \frac{1}{2^n} \sum_{\underline{y}=0}^{2^n-1} \sum_{\underline{x}=0}^{2^n-1} \underbrace{(-1)^{f(\underline{x})} (-1)^{\underline{x} \cdot \underline{y}}}_{(-1)^{(\underline{x} \cdot \underline{y} + f(\underline{x}))}} \underline{|\underline{y}\rangle} \underline{\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)}$$

Now the first register consisting of n qubits is projected to the state

$$(103) \quad \underbrace{|0, 0, 0, \dots, 0\rangle}_n$$

yes $\rightarrow$ f : constant

no $\rightarrow$ f : balanced

If f is balanced the amplitude of $|y=0,0,\dots,0\rangle$ is zero since then

$$(104) \quad \sum_{\underline{x}=0}^{2^n-1} (-1)^{(\underline{x} \cdot \underline{y} + f(\underline{x}))} = \sum_{\underline{x}=0}^{2^n-1} (-1)^{f(\underline{x})} = 0$$

If f is constant $f(\underline{x}) = f \in \{0,1\}$

$$(105) \quad \sum_{\underline{x}=0}^{2^n-1} (-1)^{(\underline{x} \cdot \underline{y} + f(\underline{x}))} = (-1)^f \sum_{\underline{x}=0}^{2^n-1} (-1)^{\underline{x} \cdot \underline{y}}$$

$$= 0 \quad \text{for all } \underline{y} \text{ except } \underline{y} = (0,0,0,\dots,0)$$

□

The Deutsch-Jozsa problem is however a problem within BPP, i.e. it can be efficiently solved probabilistically with a classical computer.

3.4 BQP $\neq$ BPP : the Simon problem

We will now discuss a problem which is believed to be outside of BPP can be solved with polynomial effort using a quantum algorithm:

Simon problem

If f is balanced the amplitude of $|y=0,0,\dots,0\rangle$ is zero since then

$$(104) \quad \sum_{\underline{x}=0}^{2^n-1} (-1)^{(\underline{x} \cdot \underline{y} + f(\underline{x}))} = \sum_{\underline{x}=0}^{2^n-1} (-1)^{f(\underline{x})} = 0$$

If f is constant $f(\underline{x}) = f \in \{0,1\}$

$$(105) \quad \sum_{\underline{x}=0}^{2^n-1} (-1)^{(\underline{x} \cdot \underline{y} + f(\underline{x}))} = (-1)^f \sum_{\underline{x}=0}^{2^n-1} (-1)^{\underline{x} \cdot \underline{y}}$$

$= 0$ for all $\underline{y}$ except $\underline{y} = (0,0,0,\dots,0)$

□

The Deutsch-Jozsa problem is however a problem within BPP, i.e. it can be efficiently solved probabilistically with a classical computer.

3.4 BQP $\neq$ BPP : the Simon problem

We will now discuss a problem which is believed to be outside of BPP but can be solved with polynomial effort using a quantum algorithm:

Simon problem

given $f: \{0,1\}^n \rightarrow \{0,1\}^n$

with a binary period a , i.e.

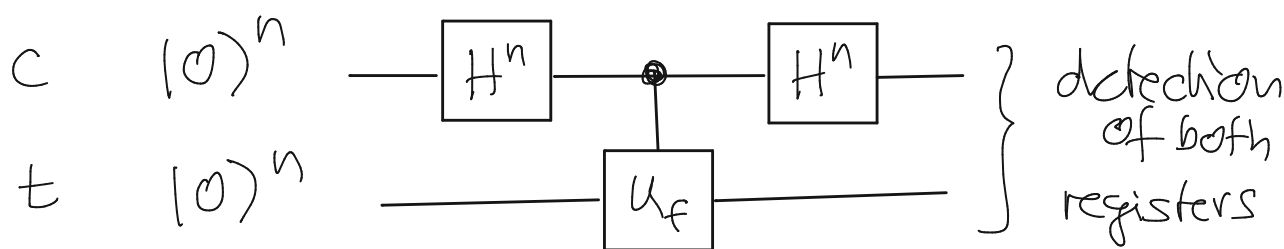
$$(106) \quad f(\underline{x}) = f(\underline{y}) \quad \text{if} \quad \underline{y} = \underline{x} \oplus \underline{a}$$

Here $\oplus$ is bitwise addition modulo 2. The task is to find the period a (note $\underline{x} \oplus \underline{a} \oplus \underline{a} = \underline{x}$)

- classically this is a NP hard problem since we have to test pairs $\underline{x}, \underline{y}$ such that

$$(107) \quad \underline{x} \oplus \underline{y} = \underline{a} \quad \text{note:} \quad (\underline{x} \oplus \underline{x} = \underline{0})$$

- quantum mechanically



where $f(\underline{x})$ is encoded in U_f

short notation $|0\rangle \equiv |0\rangle^n = |0\rangle|0\rangle \dots |0\rangle$

$|\underline{x}\rangle, |\underline{y}\rangle$ in n -dim. computational basis
e.g. $|\underline{x}\rangle = |001101011\dots\rangle$

$$(108) \quad U_f |\underline{x}\rangle |\underline{y}\rangle = |\underline{x}, \underline{y} \oplus f(\underline{x})\rangle$$

then

$$\begin{aligned}
 |0\rangle|0\rangle &\xrightarrow{H^n} \frac{1}{2^{n/2}} \sum_{\underline{x}=0}^{2^n-1} |\underline{x}\rangle|0\rangle \\
 (109) \quad &\xrightarrow{U_f} \frac{1}{2^{n/2}} \sum_{\underline{x}=0}^{2^n-1} |\underline{x}\rangle|f(\underline{x})\rangle
 \end{aligned}$$

- measure 2nd register (target) $\Rightarrow$ this yields randomly one of the 2^n possible outputs of f :

$$\begin{array}{l}
 1 \quad |f(0,0,0,\dots)\rangle \\
 n \quad |f(1,0,0,\dots)\rangle, |f(0,1,0,\dots)\rangle \dots |f(0,0,\dots,0,1)\rangle \\
 n(n-1) \quad |f(1,1,0,\dots)\rangle, |f(1,0,1,\dots)\rangle \dots |f(0,0,\dots,1,1)\rangle \\
 \vdots \quad \quad \quad \vdots \\
 1 \quad |f(1,1,1,\dots)\rangle
 \end{array}$$

Suppose measurement yielded $|f(\underline{x}_0)\rangle$
 e.g. $|f(0,1,0,1,0,\dots,0)\rangle$

Since both $\underline{x} = \underline{x}_0$ and $\underline{x} = \underline{x}_0 \oplus \underline{a}$ give the same result, the first register is after the measurement in the state

$$(110) \quad \frac{1}{\sqrt{2}} (|\underline{x}_0\rangle + |\underline{x}_0 \oplus \underline{a}\rangle)$$

Now how can we determine a ?

- bitwise Hadamard of 1st register

$$\begin{aligned}
 & H^n \frac{1}{\sqrt{2}} \left(|x_0\rangle + |x_0 \oplus a\rangle \right) = \\
 (111) \quad & = \frac{1}{2^{(n+1)/2}} \sum_{\underline{y}=0}^{2^n-1} \left[(-1)^{\underline{x}_0 \cdot \underline{y}} + (-1)^{(\underline{x}_0 + \underline{a}) \cdot \underline{y}} \right] |\underline{y}\rangle \\
 & = \frac{1}{2^{(n+1)/2}} \sum_{\underline{y}=0}^{2^n-1} (-1)^{\underline{x}_0 \cdot \underline{y}} \left[1 + (-1)^{\underline{a} \cdot \underline{y}} \right] |\underline{y}\rangle
 \end{aligned}$$

Now since

$$(112) \quad \underline{a} \cdot \underline{y} = a_1 y_1 + a_2 y_2 + \dots \pmod{2} \in \{0, 1\}$$

$\Rightarrow$

$$\begin{aligned}
 & H^n \frac{1}{\sqrt{2}} \left(|x_0\rangle + |x_0 \oplus a\rangle \right) \\
 (113) \quad & = \frac{1}{2^{(n+1)/2}} \sum_{\underline{a} \cdot \underline{y} = 0} (-1)^{\underline{x}_0 \cdot \underline{y}} |\underline{y}\rangle
 \end{aligned}$$

- measurement of $|\underline{y}\rangle$ register projects to one of the 2^{n-1} states $|\underline{y}\rangle$ with $\underline{a} \cdot \underline{y} = 0$

this gives one value of $\underline{y}$, say $\underline{y}_0$

- repeat the protocol until we have n linear independent binary vectors $\underline{y}_j$

with

$$(114) \quad \underline{y_j} \cdot \underline{a} = 0 \quad j=1, 2, \dots, n$$

This yields n linear equations for the n components of $\underline{a}$

$$(115) \quad y_{j1}a_1 + y_{j2}a_2 + \dots + y_{jn}a_n \stackrel{\text{mod } 2}{=} 0 \quad j=1..n$$

$\Rightarrow$ determine $\underline{a}$

In order to find n linear independent vectors $\underline{y_j}$ the algorithm has to be performed on average $O(n)$ times i.e. with polynomial effort.

The Simon problem shows

$$BQP > BPP$$

(i.e. $BQP \neq BPP$)

3.5 Grover's search algorithm

Problem: given an unsorted list with N entries (reverse telephone book)
How to find an entry (e.g. a given telephone number)?

- classically $N/2$ inquiries are needed!

remark: a search in an ordered list requires only $O(\log_2 N)$ inquiries.
(successive bi-section)

$$0 \leq x \leq N$$

$$0 \leq x < N/2 \quad N/2 \leq x \leq N$$

$$0 \leq x < N/4 \quad N/4 \leq x < N/2 \quad N/2 \leq x < \frac{3}{4}N \quad \frac{3}{4}N \leq x \leq N$$

$\vdots$ etc

lev Grover found an algorithm that finds an entry in

$$O(\sqrt{N})$$

steps quantum mechanically. This is practically very relevant but does not change the complexity class as

$$(116) \quad N = 2^{\log_2 N} \quad \sqrt{N} = 2^{\frac{1}{2} \log_2 N}$$

both scale exponentially with number of bits $(\log_2 N)$ needed to encode N

lets assume that the unsorted list (inverse telephone book) is encoded in an oracle

For a given telephone number $\underline{\omega}$ corresponding to the persons name $\underline{x}_0$ we have

$$(117) \quad \begin{aligned} f_{\underline{\omega}}(\underline{x}) &= 1 & \text{for } \underline{x} = \underline{x}_0 \\ f_{\underline{\omega}}(\underline{x}) &= 0 & \text{else} \end{aligned}$$

quantity we look for!

$f_{\underline{\omega}}$ is encoded in a unitary $U_{\underline{\omega}}$

$$(118) \quad U_{\underline{\omega}} |\underline{x}\rangle |y\rangle = |\underline{x}\rangle |y \oplus f_{\underline{\omega}}(\underline{x})\rangle$$

Note that for given $\underline{x}$ (name) it is easy to find $\underline{\omega}_0$ (telephone number) since in this direction the list is ordered, i.e. $\underline{x} \rightarrow f_{\underline{\omega}}(\underline{x})$ is easy: if $\underline{\omega} = \underline{\omega}_0$ then $f_{\underline{\omega}} = 1$ else $f_{\underline{\omega}} = 0$.

As seen before

$$(119) \quad U_{\underline{\omega}} |\underline{x}\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = (-1)^{f_{\underline{\omega}}(\underline{x})} |\underline{x}\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

I.e. ignoring the second register which stays unchanged, the action of $U_{\underline{\omega}}$ is

$$(120) \quad U_{\underline{\omega}} |\underline{x}\rangle \longrightarrow (-1)^{f_{\underline{\omega}}(\underline{x})} |\underline{x}\rangle$$

This means $U_{\underline{\omega}}$ changes the sign of all states orthogonal to the state $|\underline{x}_0\rangle$ corresponding to $\underline{\omega}$ (input telephone number).

(130)

$$U_w \hat{=} \mathbb{1} - 2 |\underline{x}_0\rangle \langle \underline{x}_0|$$

Now we use as input a superposition of computational basis states as input

$$(131) \quad |\underline{s}\rangle = \frac{1}{\sqrt{N}} \sum_{\underline{x}=0}^{N-1} |\underline{x}\rangle$$

$|\underline{s}\rangle$ contains the state $|\underline{x}_0\rangle$ which we are looking for with probability $P_{x_0} = 1/N$.

Grover's Algorithm aims at an iterative amplification of this probability. To this end we apply

$$(132) \quad R = \bar{U}_s U_w$$

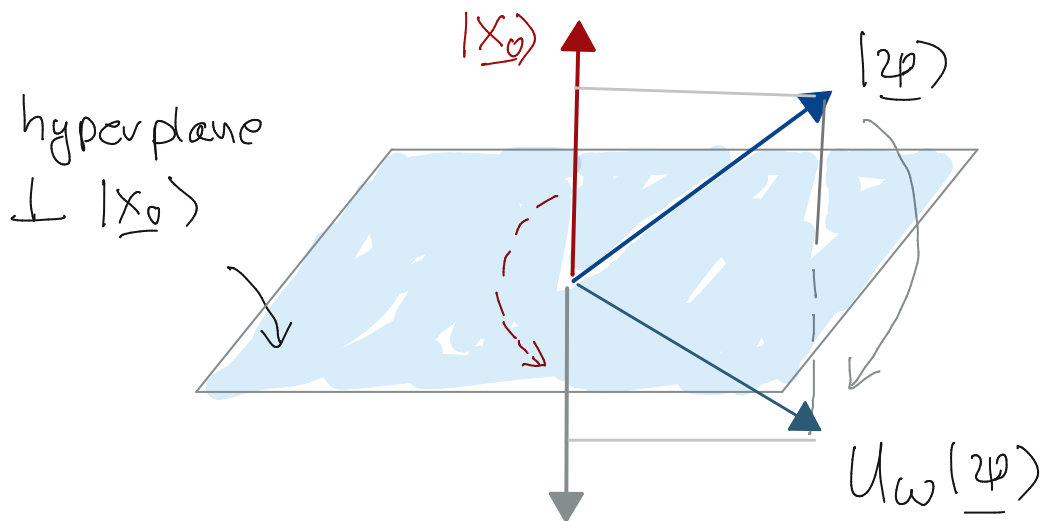
$$(133) \quad \text{where } \bar{U}_s = 2|\underline{s}\rangle \langle \underline{s}| - \mathbb{1}$$

R is called reflection about the mean

geometric illustration

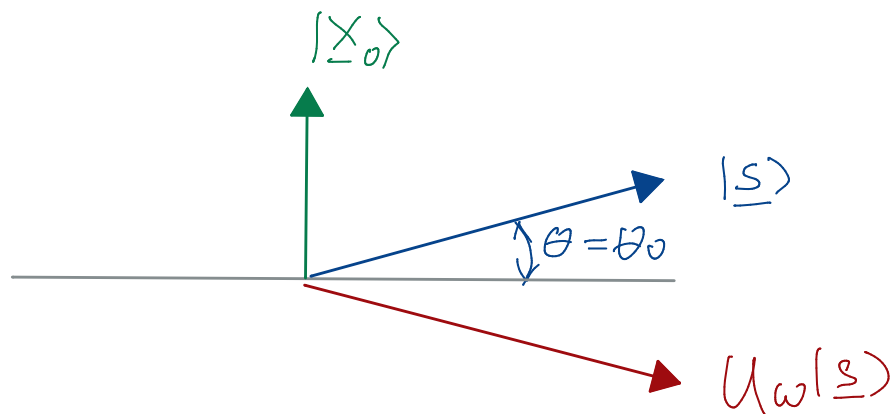
$$(134) \quad |\langle \underline{s} | \underline{x}_0 \rangle| = \frac{1}{\sqrt{N}} \doteq \sin \theta \approx \theta$$

- action of U_w : The component of a general state $|\psi\rangle$ proportional to $|\underline{x}_0\rangle$ changes sign

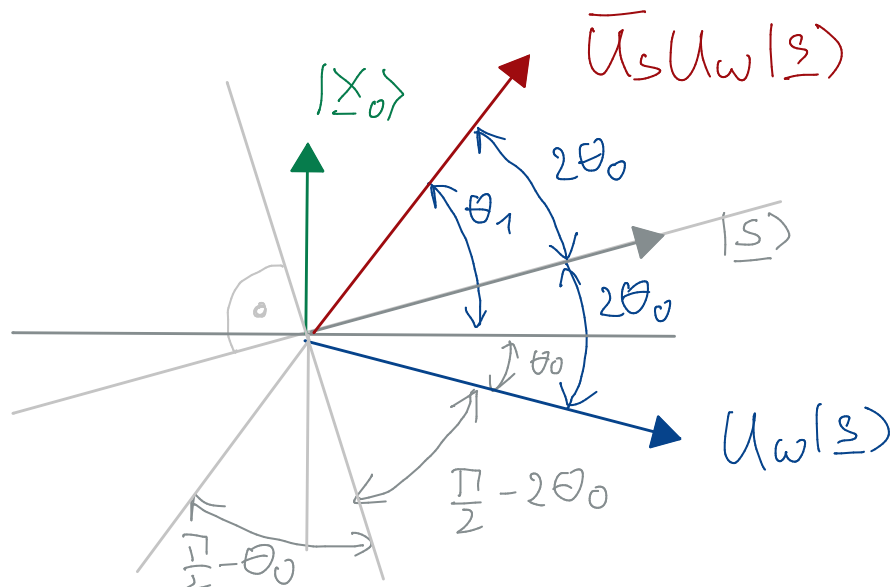


• action of $\bar{U}_S U_w$

(i)



(ii)



$$(135) \quad \theta_1 = \pi - 2\left(\frac{\pi}{2} - 2\theta_0\right) - \theta_0 = 3\theta_0$$

i.e. in every iteration step the vector is shifted by $\Delta\theta = 2\theta_0$ in the direction of $|\underline{x}_0\rangle$

So
(136) $|\psi_0\rangle = |\underline{x}\rangle$ and $|\psi_{k+1}\rangle = R |\psi_k\rangle$

(137) Thus $|\psi_k\rangle = \alpha_k \sum_{\underline{x} \neq \underline{x}_0} |\underline{x}\rangle + \beta_k |\underline{x}_0\rangle$

where

(138) $\alpha_k = \frac{1}{\sqrt{N-1}} \cos((2k+1)\theta)$ $\beta_k = \sin((2k+1)\theta)$

The amplitude of the state $|\underline{x}_0\rangle$, which we are looking for keeps increasing as long as

$$(2k+1)\theta \leq \frac{\pi}{2}$$

Optimum number $k_{\text{opt}} = T$ of steps is
 $(2T+1)\theta = \frac{\pi}{2}$. Using that for large N
 $\sin\theta \simeq \theta = \frac{1}{\sqrt{N}} \Rightarrow$

(139)
$$T = \frac{\pi}{4} \sqrt{N} + \mathcal{O}(1)$$

number
of steps
in Grover
algorithm

and $\sin((2T+1)\theta) \simeq 1$.

J.e. after $\mathcal{O}(\sqrt{N})$ steps $|\underline{x}_0\rangle$ is approximated with probability close to 1 !

Implementation of $\bar{U}_S$

$$(140) \quad |\underline{s}\rangle = H^{(n)} |\underline{0}\rangle = \frac{1}{2^{n/2}} \sum_{\underline{x}=0}^{2^n-1} |\underline{x}\rangle$$

i.e.

$$(141) \quad \bar{U}_S = 2|\underline{s}\rangle\langle\underline{s}| - \mathbb{I} = H^{(n)} (2|\underline{0}\rangle\langle\underline{0}| - \mathbb{I}) H^{(n)}$$

$H^{(n)}$ is easy, but how about $2|\underline{0}\rangle\langle\underline{0}| - \mathbb{I}$,
or equivalently $1 - 2|\underline{0}\rangle\langle\underline{0}|$?

$$1 - 2|\underline{0}\rangle\langle\underline{0}| = \underbrace{\bar{X}^{(n)}}_{\text{bitwise Pauli X}} \underbrace{(1 - 2|\underline{1}\rangle\langle\underline{1}|)}_{\text{changes sign of } |1111\dots 1\rangle \text{ only}} \bar{X}^{(n)}$$

$1 - 2|\underline{1}\rangle\langle\underline{1}|$ is unitary as

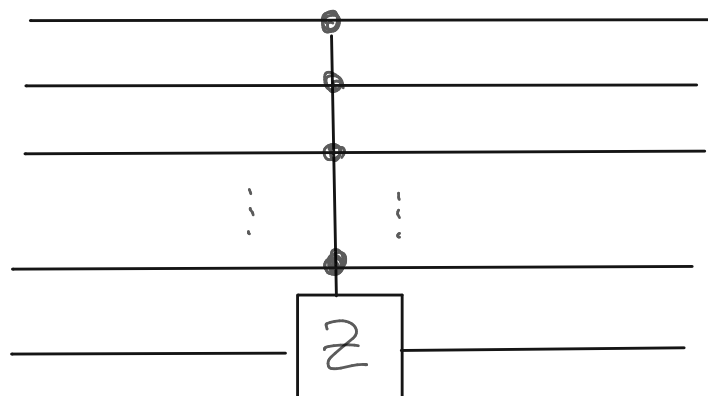
$$(1 - 2|\underline{1}\rangle\langle\underline{1}|)^{\dagger} = (1 - 2|\underline{1}\rangle\langle\underline{1}|)$$

$$\text{and } (1 - 2|\underline{1}\rangle\langle\underline{1}|)(1 - 2|\underline{1}\rangle\langle\underline{1}|) =$$

$$= 1 - 4|\underline{1}\rangle\langle\underline{1}| + 4|\underline{1}\rangle\langle\underline{1}| = 1 \quad \checkmark$$

and can be implemented by an N -fold controlled gate

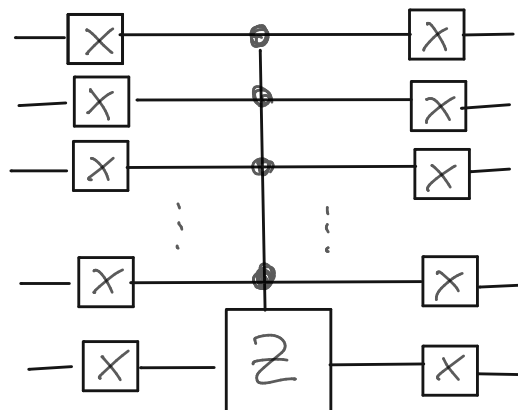
$$1 - 2|1\rangle\langle 1|$$



} all of these bits need to be in $|1\rangle$ else gate does nothing

If all but the last qubit are in $|1\rangle$ and the last in $|0\rangle$ the state is unchanged. If the last qubit is in $|1\rangle$ the sign is flipped to, i.e.

So $\bar{U}_S$ is



This gate can be implemented with $\mathcal{O}(n = \log N)$ elementary gates. I.e. Grover's search algorithm requires

$$(142) \quad \sim \sqrt{N} \text{ poly}(\log N)$$

gate operations.

One can show that Grover's algorithm is optimal for a search in an unsorted list

3.6 Quantum Fourier Transform

and period finding by Shor

Generalizing Simon's problem let us consider a function

$$(143) \quad y = f(x)$$

defined on a finite interval discretized into 2^n elements characterized by an integer

$$(144) \quad x \in \{0, 1, 2, \dots, 2^n - 1\}$$

$$(145) \quad f: \underbrace{\{0, 1\}^n}_x \longrightarrow \underbrace{\{0, 1\}^n}_y$$

Let assume $f(x)$ has an unknown period r

$$1 \ll r \ll 2^n$$

$$(146) \quad f(x + m r) = f(x) \quad \begin{array}{l} r, m \in \mathbb{N} \\ x, x + m r \in \{0, 1, 2, \dots, 2^n - 1\} \end{array}$$

$\uparrow$
"normal" addition

To find r is classically a difficult task even if the evaluation of $f(x)$ itself can be made in $\text{poly}(n)$ steps!

A key element to finding periods is the discrete Fourier transformation:

$$f(x) \longrightarrow \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i x y}{N}} f(y)$$

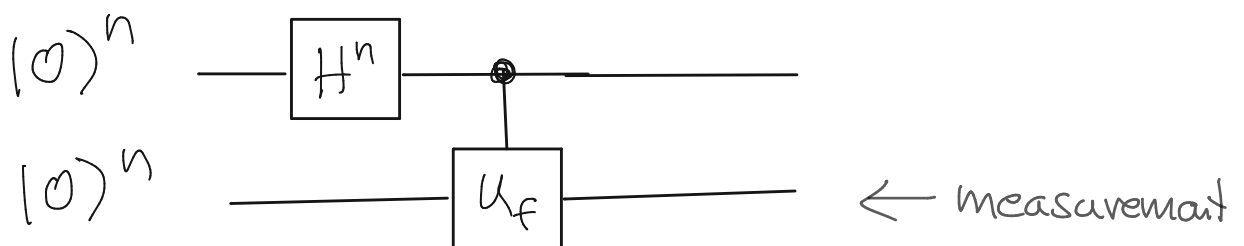
Since $N = 2^n$ the FTr also requires exponentially many operations.

(A) Period finding using the quantum Fourier transformation

We will now show that given we can implement the quantum Fourier transformation ($x, y \in \mathbb{N}$)

$$(147) \quad \text{QFT} \quad |x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{y=1}^{N-1} e^{\frac{2\pi i x y}{N}} |y\rangle$$

We can determine the period r . Similar to Simon problem



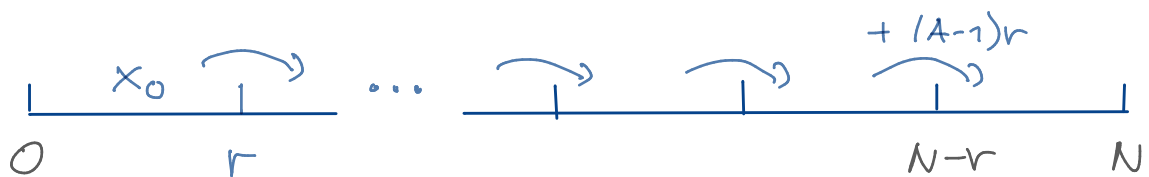
$$(148) \quad \begin{aligned} |0\rangle &= |0\rangle^n \\ |0\rangle |0\rangle &\xrightarrow{H^n} \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |0\rangle \\ &\xrightarrow{U_f} \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |f(x)\rangle \end{aligned}$$

Measurement of 2nd register with result $f(x_0)$
 (where x_0 is $0 \leq x_0 < r$ but unknown)
 projects 1st register to

$$(149) \quad \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle \quad A \in \{0, 1, 2, \dots, 2^n - 1\}$$

where

$$(150) \quad N - r \leq x_0 + (A-1)r < N$$



or

$$(151) \quad \boxed{A-1 < \frac{N}{r} < A+1} \quad \frac{x_0}{r} < 1$$

If N/r is integer $\Rightarrow A = N/r$.

Problem: How to find r from eq. (149)?
 Projection to computational basis would not give information about r .

$\Rightarrow$ QFT r also

$$(152) \quad \begin{aligned} \text{QFT} \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle &= \\ &= \frac{1}{\sqrt{NA}} \sum_{y=0}^{N-1} e^{\frac{2\pi i x_0 y}{N}} \sum_{j=0}^{A-1} e^{\frac{2\pi i r y j}{N}} |y\rangle \end{aligned}$$

The probability to find a specific $|y\rangle$ in this state is

$$(153) \quad P(y) = \frac{A}{N} \left[\frac{1}{A} \sum_{j=0}^{A-1} e^{2\pi i \frac{ry}{N} j} \right]^2$$

Here the unknown x_0 does no longer appear! 😊

$P(y)$ is large if $\frac{ry}{N}$ is an integer.

- E.g. if $\frac{N}{r} = A$ (see eq. (151)) then

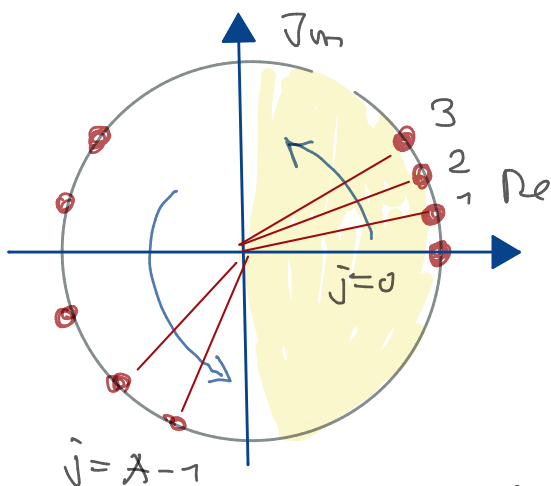
$$P(y) = \underset{\substack{\uparrow \\ \frac{N}{r} = A}}{\frac{1}{r}} \left[\frac{1}{A} \sum_{j=0}^{A-1} e^{2\pi i \frac{jy}{A}} \right]^2 = \begin{cases} \frac{1}{r} & y = mA \\ 0 & \text{else} \end{cases}$$

- In general:

$$(154) \quad \theta_y \equiv 2\pi \frac{ry(\bmod N)}{N}$$

$$(155) \quad \sum_{j=0}^{A-1} e^{ij\theta_y} = \frac{e^{iA\theta_y} - 1}{e^{i\theta_y} - 1}$$

For which values of y is this expression large?



Note $A-1 < \frac{N}{r}$

$\Rightarrow$ all terms in (155) are in the same half of the complex plane if

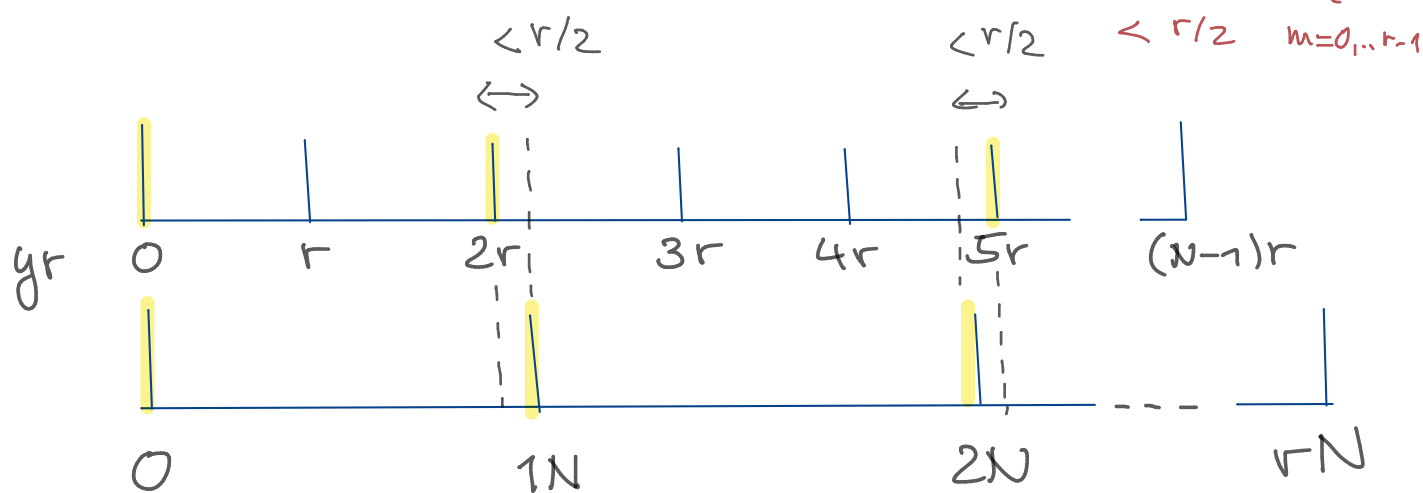
$$-\frac{\pi r}{N} \leq \theta_y \leq \pi \frac{r}{N} \quad (156)$$

In this case $p(y)$ in (153) is extremal. Eq. (156) is equivalent to

$$(157) \quad -\frac{r}{2} \leq yr \pmod{N} \leq \frac{r}{2}$$

There are exactly r values of y in $\{0, 1, \dots, N-1\}$ which fulfill this condition:

example



$$y=0$$

$$y=2$$

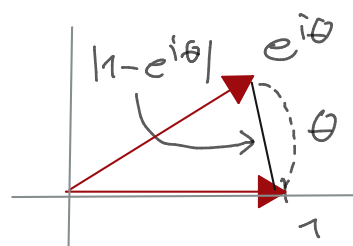
$$y=5 \dots \text{fulfill (157)}$$

Now we show that the measurement of the first register after QFT_r yields with probability larger than $4/\pi^2 \approx 0.405$ one of the r possible values of y that fulfill eq. (157)!

To see this note

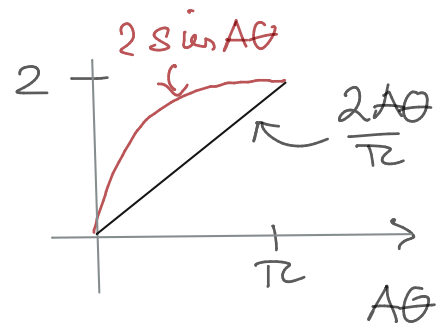
$$|(1 - e^{i\theta})| \leq |\theta|$$

- 77 -



furthermore for $A|\theta| \leq \pi$

$$|1 - e^{iA\theta}| \geq \frac{2A|\theta|}{\pi}$$



$$\text{since } |1 - e^{iA\theta}| = |e^{i\frac{A\theta}{2}}| 2 \sin \frac{A\theta}{2} = 2 \sin \frac{A\theta}{2}$$

This gives the estimate for $A|\theta| \leq \pi$

$$(158) \quad \left| \frac{e^{iA\theta} - 1}{e^{i\theta} - 1} \right| \geq \frac{2A|\theta|}{\pi} \frac{1}{|\theta|} = \frac{2A}{\pi}$$

thus

$$p(y) = \frac{A}{N} \left| \frac{1}{A} \sum_{j=0}^{A-1} e^{ij\theta_y} \right|^2 = \frac{A}{N} \left| \frac{1}{A} \frac{e^{iA\theta_y} - 1}{e^{i\theta_y} - 1} \right|^2$$

$$(159) \quad \geq \frac{A}{N} \frac{4}{\pi^2} \stackrel{\text{eq. (157)}}{\geq} \frac{4}{\pi^2} \left(\frac{1}{r} - \frac{1}{N} \right) \geq \frac{4}{\pi^2} \frac{1}{r}$$

Since there are exact r values of y that fulfill (157) the total probability is

$$(160) \quad P_{\text{tot}} = \sum_{\substack{y \text{ that} \\ \text{fulfill (157)}}} p(y) \geq \frac{4}{\pi^2} \quad \square$$

$\Rightarrow$ A measurement of QFT-transformed state projects with probability $P_{\text{tot}} \geq 4/\pi^2 \simeq 0.405$ to one of the y -values that fulfill eq. (157), i.e.

$$(161) \quad k \cdot \frac{N}{r} - \frac{1}{2} \leq y \leq k \cdot \frac{N}{r} + \frac{1}{2} \quad k \in \{0, 1, \dots, r-1\}$$

We see that knowledge of (one of these) y value

$$(162) \quad y \rightarrow \frac{k}{r}$$

With high probability k and r have no common divisor, which allows us to determine r . In the unlikely event that k and r have a common divisor the check if r is a period will give a "no" and we have to repeat the measurement to get a different y .

- Since the measurement of QFT transformed register gives a value of y that fulfills (157) only with finite probability we have to check if

$$(163) \quad f(x) \stackrel{?}{=} f(x+r)$$

by evaluating U_f . If (163) is not fulfilled the actually detected value of y could be near a "true" value and we can check neighboring values or make a new measurement.

$\Rightarrow$ Quantum Fourier transform allows to find the period of a function $f(x)$ with polynomial effort.

(B) Implementation of Quantum Fourier transform

How to implement

$$(164) \sum_{x=0}^{N-1} f(x) |x\rangle \longrightarrow \sum_{y=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{\frac{2\pi i x y}{N}} f(x) |y\rangle$$

This operation corresponds to the multiplication with the $N \times N$ matrix

$$M_{xy} = \left(e^{\frac{2\pi i}{N}} \right)^{xy}$$

which is exponentially large in n ($N = 2^n$). Naively one would expect that this requires $\underline{O(N^2)}$ elementary operations. (:(

This is however not the case!

- On the classical level there is the Fast-Fourier-Transform (FFT)

which only needs $O(N \log N)$ steps:

binary representation of x, y :

$$x_i, y_i \in \{0, 1\}$$

$$(165) \quad \begin{aligned} x &= x_{n-1} 2^{n-1} + x_{n-2} 2^{n-2} + \dots + x_1 2 + x_0 \\ y &= y_{n-1} 2^{n-1} + y_{n-2} 2^{n-2} + \dots + y_1 2 + y_0 \end{aligned}$$

- 78 -

In order to evaluate $\left(e^{\frac{2\pi i}{N}}\right)^{xy}$ we can ignore all terms in $x \cdot y$ that are multiples of $N = 2^n$, i.e.

$$\begin{aligned}
 (166) \quad \frac{x \cdot y}{N} &= \frac{x \cdot y}{2^n} \hat{=} y_{n-1} \frac{x_0}{2} + y_{n-2} \left(\frac{x_1}{2} + \frac{x_0}{2^2} \right) \\
 &\quad + \dots + y_0 \left(\frac{x_{n-1}}{2} + \frac{x_{n-2}}{2^2} + \dots + \frac{x_0}{2^n} \right) \\
 &\quad \text{FFT} \\
 &= y_{n-1} [x_0] + y_{n-2} [x_0 x_1] + \dots \\
 &\quad + \dots + y_0 [x_0 x_1 \dots x_{n-1}]
 \end{aligned}$$

where

$$(167) \quad [x_0 x_1 \dots x_k] \equiv \frac{x_k}{2} + \frac{x_{k-1}}{2^2} + \dots + \frac{x_0}{2^{k+1}}$$

To evaluate this expression $O(n)$ steps are needed i.e.

$$\tilde{f}(y) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi i \frac{xy}{N}} f(x)$$

can be evaluated in $N \cdot O(n) = N O(\log N)$ steps. Speed-up is unfortunately only

$$O(N^2) \xrightarrow{\text{FFT}} O(N \log N) \text{ classical}$$

► Using quantum parallelism we can perform the Fourier transform much more efficiently!

$$\text{QFT}|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^N e^{2\pi i \frac{xy}{N}} |y\rangle =$$

$$(167) = \frac{1}{\sqrt{N}} \left(|0\rangle + e^{2\pi i [x_0]} |1\rangle \right) \left(|0\rangle + e^{2\pi i [x_0 x_1]} |1\rangle \right) \dots \left(|0\rangle + e^{2\pi i [x_0 x_1 x_2 \dots x_{n-1}]} |1\rangle \right)$$

where

$$(168) \quad [x_0 x_1 \dots x_k] \equiv \frac{x_k}{2} + \frac{x_{k-1}}{2^2} + \dots + \frac{x_0}{2^k}$$

are the coefficients that appear in the FFT (166).

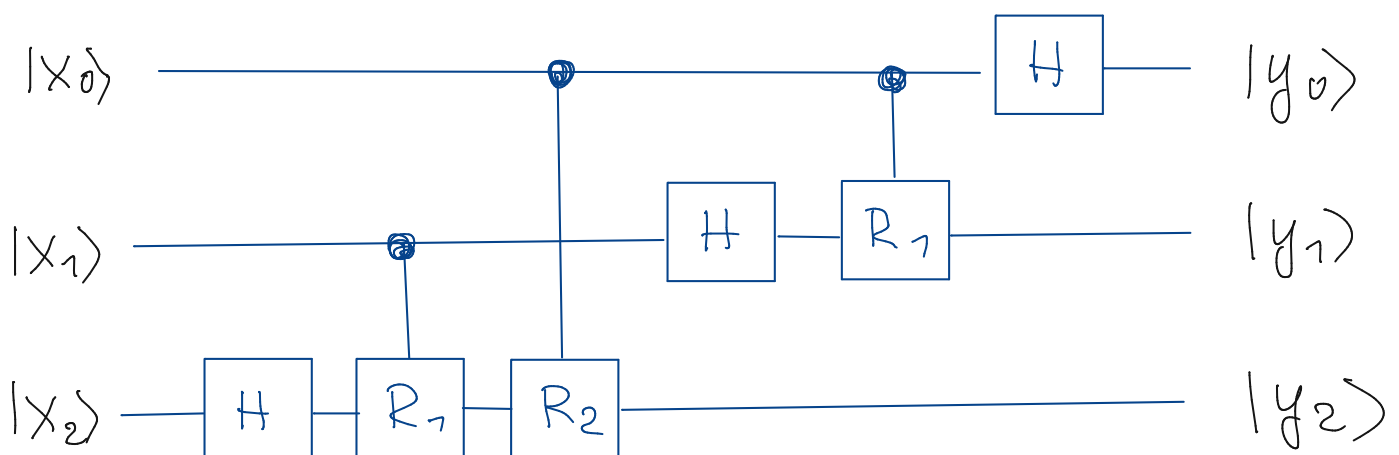
Now, eq. (167) can be efficiently implemented!

example $n=3$

$$x \rightarrow x_0, x_1, x_2 \rightarrow |x_0\rangle |x_1\rangle |x_2\rangle$$

binary representation $x_0, x_1, x_2 \in \{0,1\}$

$$x = x_0 + 2x_1 + 2^2 x_2$$



— PC —

where

$$(169) \quad H |x_j\rangle = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i [x_j]} |1\rangle)$$

and

$$(170) \quad R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2^k} \end{pmatrix}$$

i.e.

$$|x_0\rangle |x_1\rangle |x_2\rangle \xrightarrow{H^{(3)}} |x_0\rangle |x_1\rangle \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i [x_2]} |1\rangle)$$

$$\xrightarrow{R_1^{(3)}} |x_0\rangle |x_1\rangle \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i [x_2]} e^{i\pi \frac{x_1}{2}} |1\rangle)$$

$$\xrightarrow{R_2^{(3)}} |x_0\rangle |x_1\rangle \frac{1}{\sqrt{2}} (|0\rangle + \underbrace{e^{\frac{2\pi i x_2}{2}} e^{i\pi \frac{x_1}{2}} e^{i\pi \frac{x_0}{2^2}}}_{e^{2\pi i [x_0 x_1 x_2]}} |1\rangle)$$

$$(171) \xrightarrow{H^{(2)}} \frac{1}{\sqrt{4}} |x_0\rangle (|0\rangle + e^{2\pi i [x_1]} |1\rangle) (|0\rangle + e^{2\pi i [x_0 x_1 x_2]} |1\rangle)$$

$$\xrightarrow{R_1^{(2)}} \frac{1}{\sqrt{4}} |x_0\rangle (|0\rangle + e^{2\pi i [x_0 x_1]} |1\rangle) (|0\rangle + e^{2\pi i [x_0 x_1 x_2]} |1\rangle)$$

$$\xrightarrow{H^{(1)}} \frac{1}{\sqrt{8}} (|0\rangle + e^{2\pi i [x_0]} |1\rangle) (|0\rangle + e^{2\pi i [x_0 x_1]} |1\rangle) \cdot (|0\rangle + e^{2\pi i [x_0 x_1 x_2]} |1\rangle)$$

□

I.e. for the QFTrafo one Hadamard and $n(n-1)/2$ R gates are required per input qubit

QFTrafo requires $O((\log N)^2)$ gates

Corollary:

A Quantum computer can determine the period of a function $f: \mathbb{Z}_N \rightarrow \mathbb{Z}_M$ in

$$O((\log N)^2)$$

Steps, while a classical algorithm needs $O(N \log N)$ steps

(C) Period finding and factorization into prime numbers

We now discuss how an efficient period-finding algorithm can be used to determine prime factors of large integer numbers

given N : what is $k \neq 1, N$ with

$$(172) \quad N = k \cdot r \quad ?$$

Best known classical sieve

$$O\left[\exp\left((\log N)^{1/3} (\log \log N)^{2/3}\right)\right]$$

$\Rightarrow$ NP problem (not proven however)

Here we make this a BQP problem

1

choose $a < N$ randomly

2

determine largest common divisor of a and N using e.g. the Euclid method

$\Rightarrow \text{poly}(\log N)$

— If a divisor exist the search is done!

— With much much higher probability such a divisor does not exist $\Rightarrow$ step (3)

3

If a and N do not have a common divisor then there exists an r such that

$$(173) \quad a^r \equiv 1 \pmod{N} \quad (\text{theorem of Euler})$$

4

The number r from (173) is the period of the function

$$(174) \quad f_{N,a}(x) = a^x \bmod N$$

5

This period can be found efficiently using QFTrafo with effort $O((\log N)^2)$

6

If r from step 5 is even $\Rightarrow$

$$(175) \quad \underbrace{\left(a^{\frac{r}{2}} - 1\right)}_{=\alpha} \underbrace{\left(a^{\frac{r}{2}} + 1\right)}_{=\beta} = a^r - 1 = 0 \bmod N$$

i.e.

$\Rightarrow$ $\alpha - \beta$ is a multiple of N !

$$(176) \quad \alpha \beta = m N$$

Now determine largest common divisor of (α, N) and of (β, N)

e.g. with Euklid algorithm. Either (α, N) or (β, N) must have a common divisor which is a factor of N !

If r from step 5 is odd $\Rightarrow$ repeat procedure with different a

Determining the integer factors of an integer N can be done using a quantum algorithm in $\text{poly}(\log(N))$ steps

i.e. factorization $\in$ BQP

An immediate consequence is that the RSA scheme of cryptography is no longer secure $\Rightarrow$ quantum cryptography (later)

3.7 Gottesman-Knill theorem

A theorem that highlights how subtle the potential advantage of quantum circuits is, is the following

A quantum circuit that involves state preparation in the computational basis, Hadamard gates, phase gates, CNOT-gates and Pauli gates as well as measurements in the computational basis may be efficiently simulated on a classical machine

proof: later ; used in fault-tolerant computation
Shore is not of this type !