Methods in quantum computing

Mária Kieferová

October 10, 2023

University of Technology Sydney

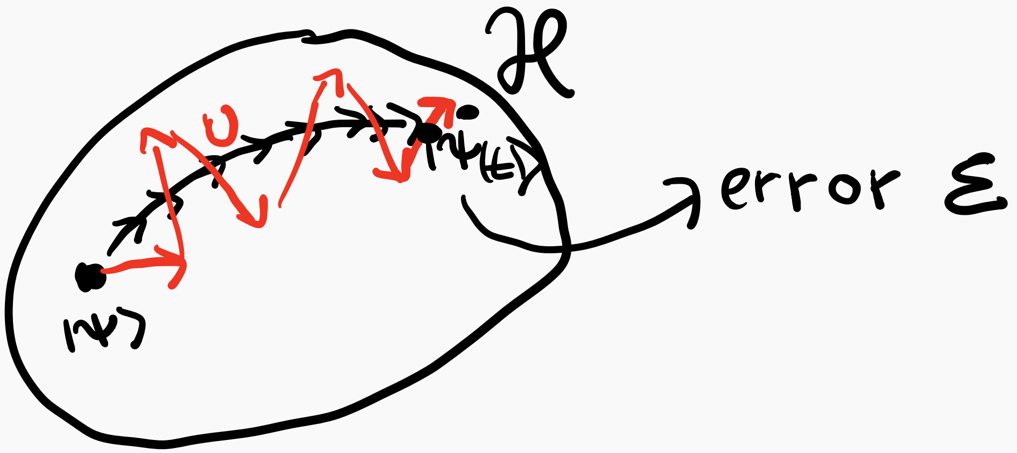
Assignments

- problem set 3 is due today
- please let me know if you didn't get an email from me about giving feedback on other videos
- pick your final project topics!

Today

- Hamiltonian simulations - finish
- Ground state preparation
- Grover search and its generalizations
- Quantum complexity
- BQP and QMA complete problems

$$|\psi(0)\rangle \rightarrow |\psi(\Delta)\rangle \rightarrow |\psi(2\Delta)\rangle \longrightarrow |\psi(t)\rangle$$



$\langle O \rangle_\psi$

$\cancel{\langle O(t) \rangle_\psi}$

The first simulation algorithm

Trotterization

For a finite t/r: $\left\| e^{(A+B)t} - \left(e^{At/r}e^{Bt/r}\right)^r \right\| \in \mathcal{O}\left(\frac{t^2}{r}\right)$

For a Hamiltonian $H = \sum_{j=1}^{m} H_j$, one can decompose the evolution with respect to H into the evolution with respect to each H_j as

error

$$\widetilde{U} = \left(e^{-iH_1 t/r} e^{-iH_2 t/r} \dots e^{-iH_m t/r}\right)^r + \mathcal{O}(\|H\|\, t^2/r).$$

$$e^{-iH_1 t} e^{-iH_2 t} \longrightarrow e^{-i(H_1+H_2)t} \; ?$$

$$[H_1, H_2] \neq 0$$

$$H = \sum_\ell \alpha_\ell H_\ell$$

$$\boxed{H_\ell = \mathbb{I} \otimes Z \otimes Z \otimes X \otimes \dots \otimes I}$$

How to simulate each term

Are there any Hamiltonians that are very easy to simulate?

$H = Z$

$$e^{-itZ} = e^{-it}|0\rangle\langle 0| + e^{it}|1\rangle\langle 1| = \begin{pmatrix} e^{-it} & \\ & e^{it} \end{pmatrix}$$

Evolution with respect to Pauli-Z is a simple single qubit gate.

$\uparrow$

easy

Exercise

Solved last tim

a) • Prove that $U^\dagger e^{-iHt} U = e^{-iU^\dagger HUt}$.

b) • What operation does $CNOT\,(\mathbb{I} \otimes Z)\,CNOT$ perform?

$$\overset{\Downarrow}{Z \otimes Z}$$

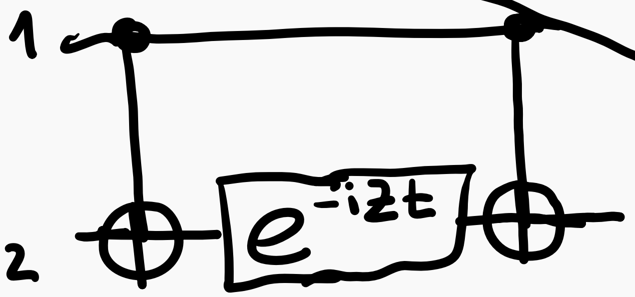
$H = Z \otimes Z$

$e^{i(Z\otimes Z)t}$

$$CNOT\, e^{-i\overset{\mathbb{1}\otimes Z}{\overset{\downarrow}{Z_2}}t}\, CNOT \overset{(a)}{=}$$

$$= e^{-i\, CNOT(\mathbb{1}\otimes Z)CNOT}$$

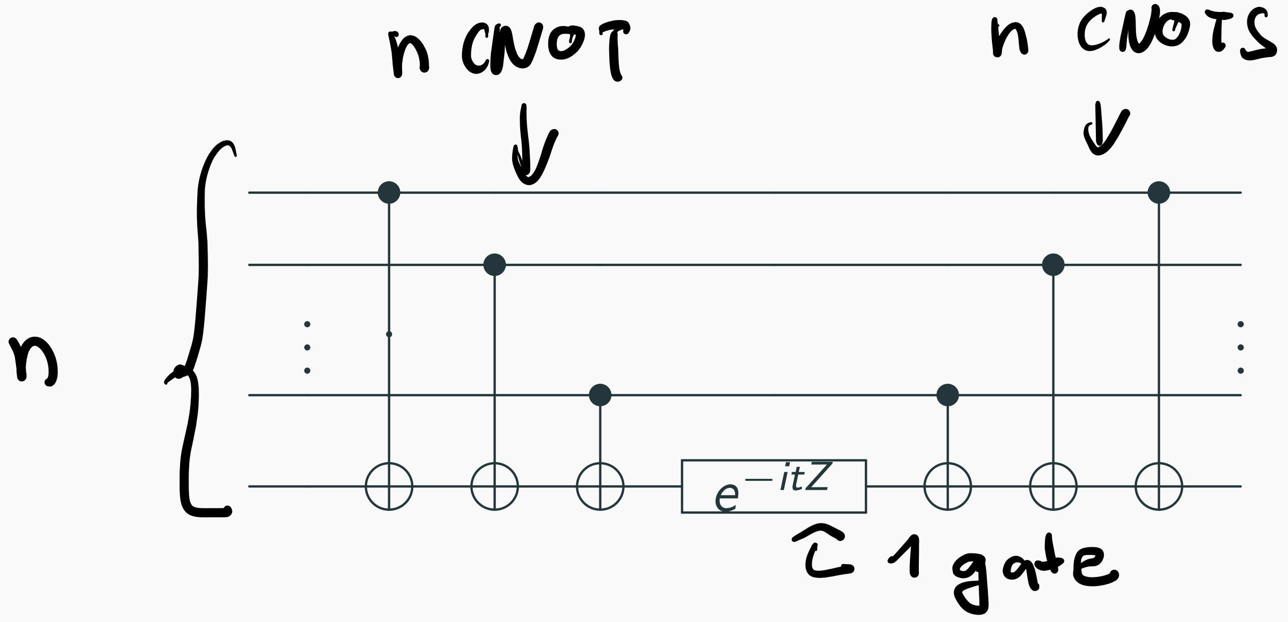
$$\overset{(b)}{=} e^{-i\, Z\otimes Z\, t}$$



$= e^{-it\, Z\otimes Z}$

Tensor product of Zs

$$H = Z_1 \otimes Z_2 \otimes \cdots \otimes Z_n.$$



How many gates do we need to simulate $e^{-it(Z_1 \otimes Z_2 \otimes \cdots \otimes Z_n)}$?

2n+1 gates

What about other Paulis?

e^{-iZt} but want e^{-iXt}

$X = Had\ Z\ Had$

$Y = S\ Had\ Z\ Had\ S^\dagger$

Complexity of simulating an evolution under a Pauli string on n qubits:

e^{-iZt} : 1

CNOTs: 2n

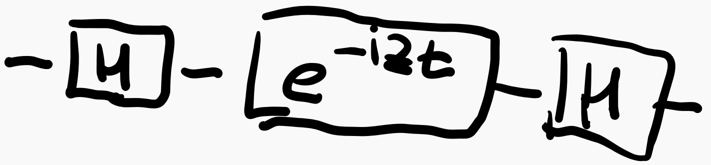
Had / S: 4n

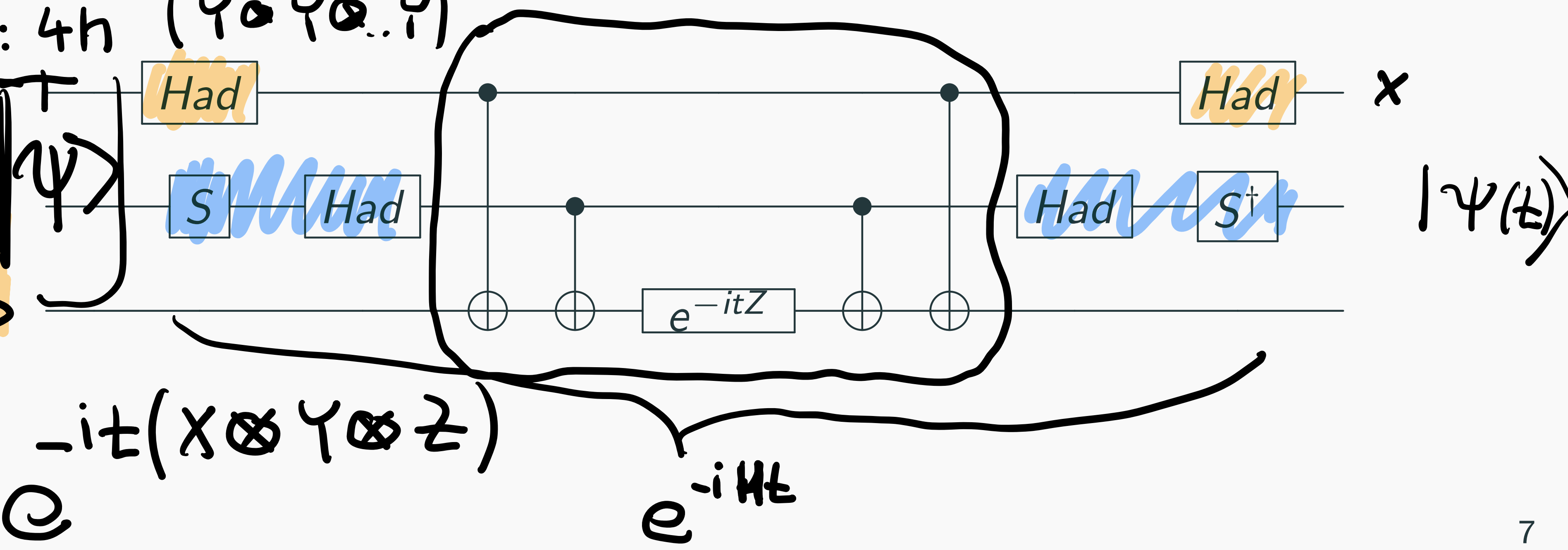
6n+1

O(n) gates

$(Y \otimes Y \otimes \dots Y)$

$e^{-it(Z \otimes Z \otimes Z)}$





Putting it all together

Hamiltonian $H = \sum_{l=1}^{L} \alpha_l P_l$ where P_l are Paulis on at most n qubits

Each $e^{-i\alpha_l P_l t}$ can be simulated with $\mathcal{O}(n)$ gates for any α_l and t.

Using the lowest order product formula we get algorithm with complexity

$\mathcal{O}\left(\alpha t^2 n \epsilon^{-1}\right)$ where $\alpha = \sum_l |\alpha_l|$. $\|P_\ell\| = 1$

trotterization + Pauli-string evolution

→ Ising (+ ~~t~~ransverse)

→ other spin model

$\boxed{|\psi\rangle \to e^{-iHt}|\psi\rangle \quad \text{we want classical answer!}}$

→ quantum chemistry in 2nd quantization

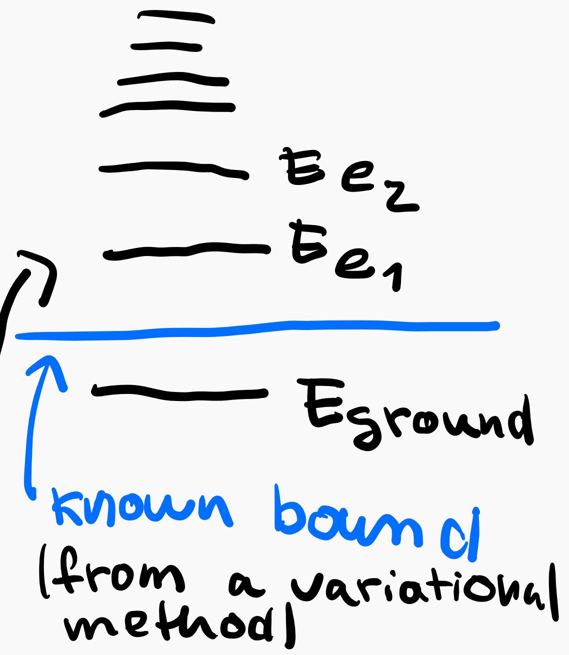
Complexity can be better with other q. algorithms

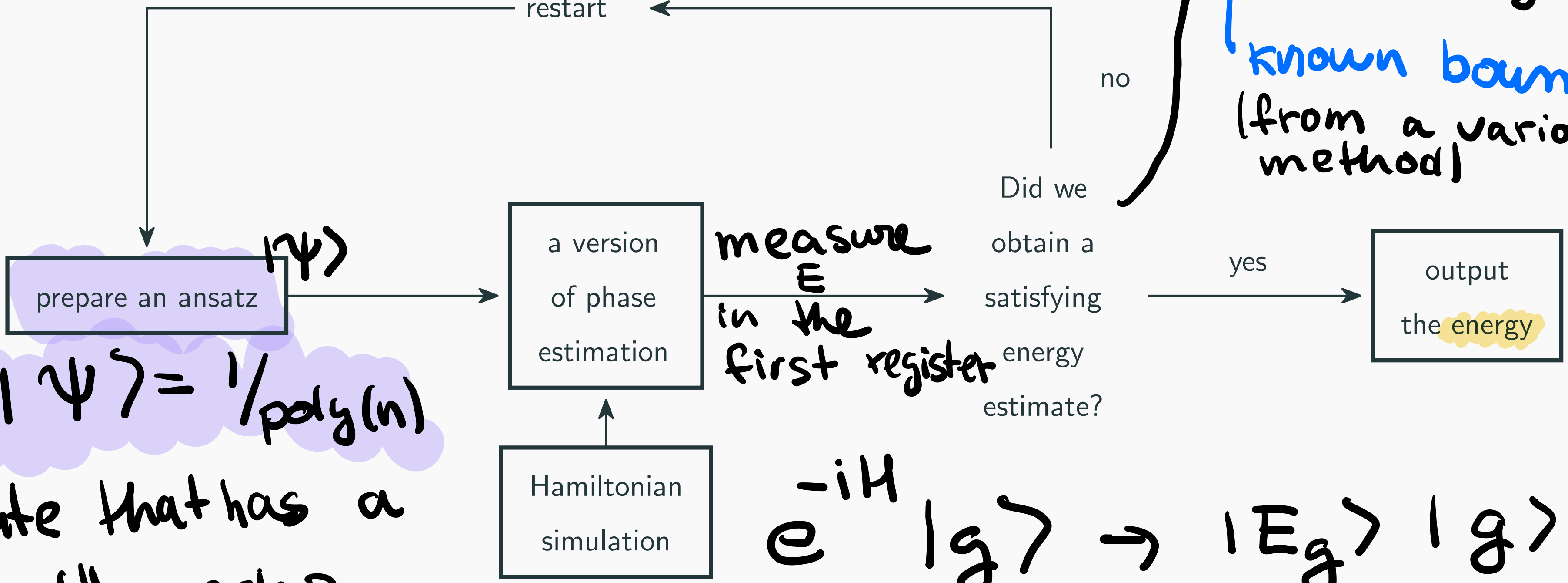
Eigenenergy Estimation

$$H = \sum_k \underset{\text{energies}}{E_k} |e_k\rangle\langle e_k| \leftarrow \text{eigenstates}$$

Goal: compute the ground state energy of a molecule

prob of success is $|\langle g|\psi\rangle|^2$





restart
no
prepare an ansatz
$|\psi\rangle$
a version of phase estimation
measure E in the first register
Hamiltonian simulation
Did we obtain a satisfying energy estimate?
yes
output the energy

$\langle g|\psi\rangle = 1/\text{poly}(n)$

state that has a "good" overlap with the ground state $|g\rangle$

$$e^{-iH}|g\rangle \to |E_g\rangle|g\rangle$$

$$e^{-iH}|\psi\rangle \to \alpha|E_g\rangle|g\rangle + \sqrt{1-|\alpha|^2}\sum_{\text{exit}}|E_{\text{exit}}\rangle|e_{\text{exit}}\rangle$$

Computing ground states of physical systems

can't be classically simulated

Consider a molecule that is too complex to compute the ground state exactly (as in assignment 2) and chemists have an approximation that gives us an upper bound on the ground state energy. This approximation allows us to construct a state with low energy (but not eigenstate), that has at least $1/poly(n)$ overlap with the ground state. Also, approximation techniques do not allow us to compute the ground state precisely enough (often up to chemical precision).

we can decide is the state is a ground state

Then, eigenstate estimation + Hamiltonian simulation can give us an "exponential" speedup for computing the ground state energy.

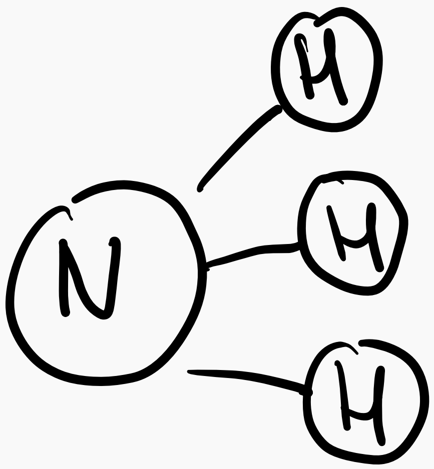
- hard classically
- we can run QPE + Ham. sim. efficiently

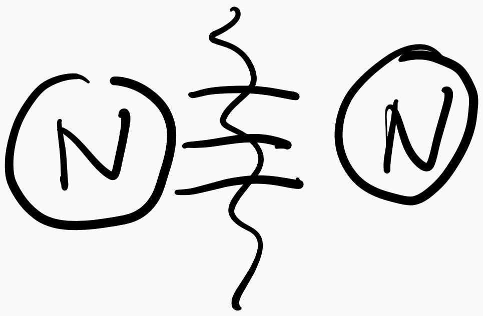
Applications

The assumption of a good ansatz is not strictly scalable but there are a lot of molecules where these assumptions are believed to be hard. More advanced techniques (Hamiltonian simulation + chemistry) allow us to compute the ground states with a few thousand logical qubits for a few molecules/materials that are not known classically up to the required precision.

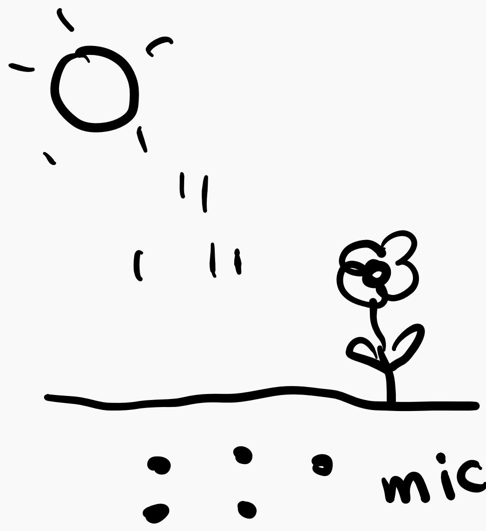
Examples

nitrogen fixation: converting N_2 into NH_3





breaking this requires A LOT of energy



biological nitrogen fixation - can't reproduce in a lab :(

FeMoco plays part
→ we should be able to compute its ground state on a quantum computer

The search problem

For $N = 2^n$, we are given a marked item $\boldsymbol{w} \in \mathbb{Z}_2^N$, and the goal is to locate $\boldsymbol{w}$.

The classical solution is easy to see. In the worse case, the algorithm has to check all N items in order to find .w

optimal $O(N)$

"oracle" – a procedure that when presented with an item tells you if it is marked or not

Quantum algorithm

of U_G call

Oracle U_G so that

item

$$U_\mathrm{G}\,|\boldsymbol{x}\rangle \overset{\downarrow}{=} \begin{cases} -|\boldsymbol{x}\rangle\,, & \text{if } \boldsymbol{x} = \boldsymbol{w} \\ |\boldsymbol{x}\rangle\,, & \text{otherwise} \end{cases}.$$

The oracle can be expressed as

$$U_\mathrm{G} = \mathbb{1} - 2\,|\boldsymbol{w}\rangle\langle\boldsymbol{w}|\,. \tag{1}$$

using oracle calls + gates

Operation with gates only form a diffusion operator:

zero oracle calls for U_d

$$\boxed{U_\mathrm{d} = 2\,|\boldsymbol{s}\rangle\langle\boldsymbol{s}| - \mathbb{1},}$$

where

$$|\boldsymbol{s}\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle\,.$$ uniform superposition

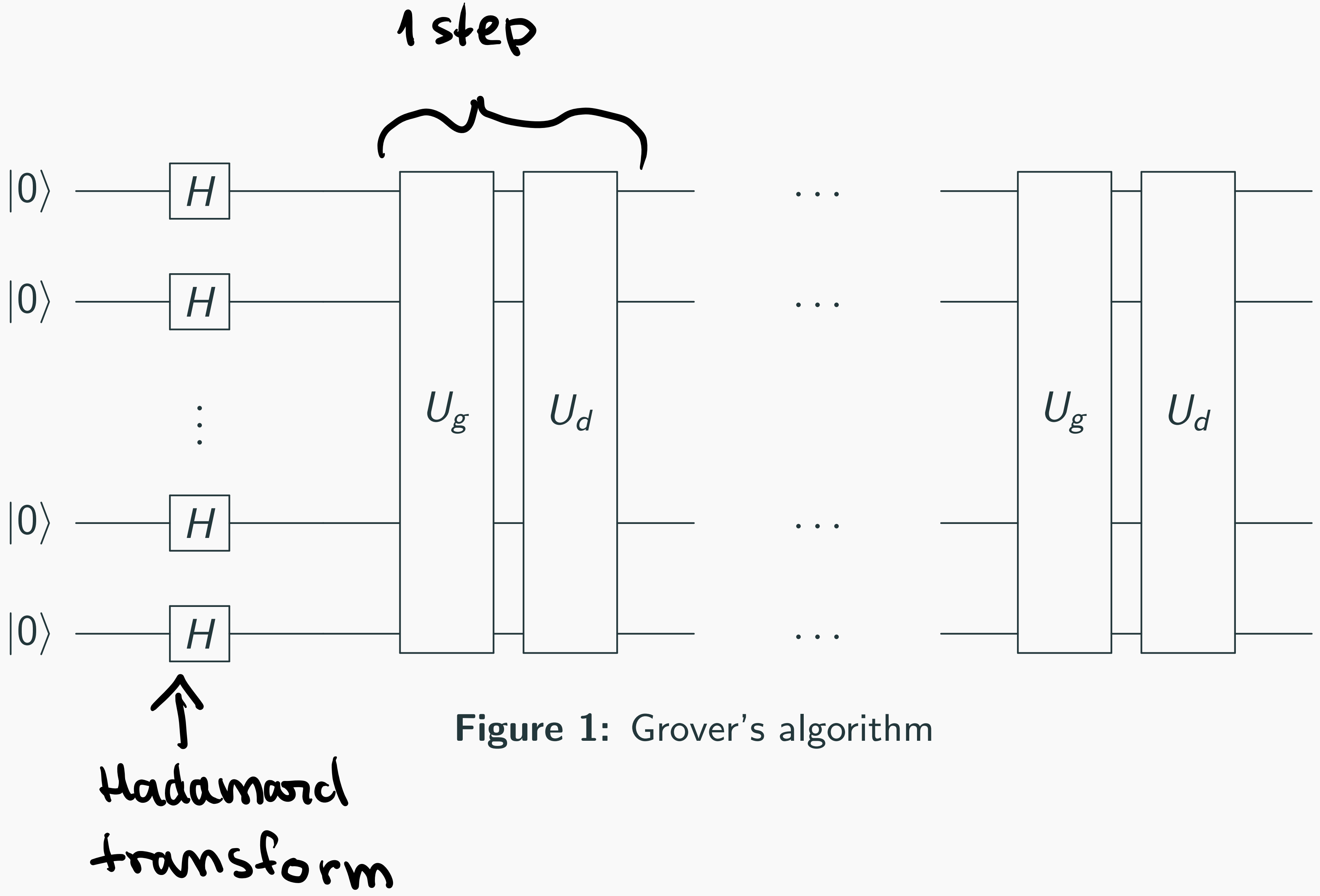


Figure 1: Grover's algorithm

First step

$\langle w^\perp | w \rangle = 0$

$|w\rangle$ - marked state

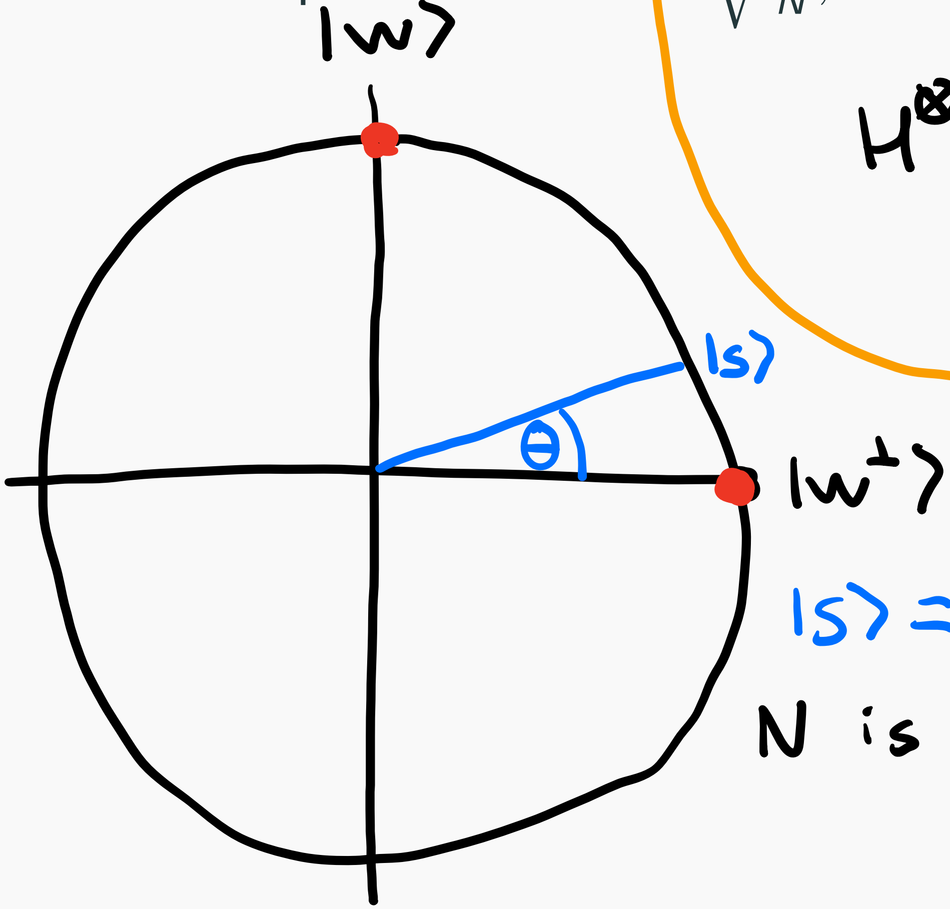
there are many state orthogonal to $|w\rangle$ but we only care about this one.

Denote $|\boldsymbol{w}^\perp\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq w} |\boldsymbol{x}\rangle$.

One can see that the uniform superposition state $|\boldsymbol{s}\rangle$ at the step t_1 can be decomposed into $|\boldsymbol{s}\rangle = \sqrt{\frac{1}{N}}\,|\boldsymbol{w}\rangle + \sqrt{\frac{N-1}{N}}\,|\boldsymbol{w}^\perp\rangle$, and the angle θ in corresponds to $\sin\theta = \sqrt{\frac{1}{N}}$, $\qquad \cos\theta = \sqrt{\frac{N-1}{N}}$

only 1 marked item

$$H^{\otimes n}|00\ldots0\rangle = |s\rangle = \frac{1}{\sqrt{N}}\sum |x\rangle$$
$$= \frac{1}{\sqrt{N}}|w\rangle + \frac{1}{\sqrt{N}}\sum_{x \neq w} |x\rangle$$



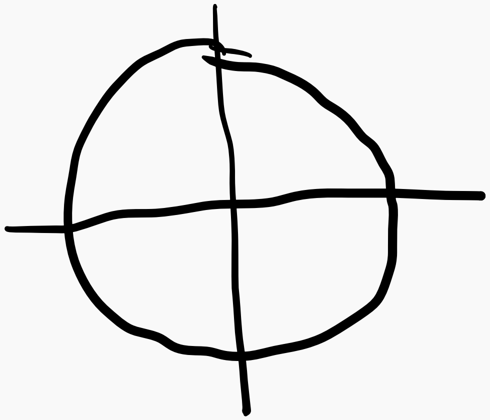
$|s\rangle = \sin\theta\,|w\rangle + \cos\theta\,|w^\perp\rangle$

N is large $\rightarrow$ θ is very small

2 dimensional subspace

$\mathcal{H} = \text{span}\{|x\rangle\}$

Our Hilbert space is N-dimensional, HOWEVER, following Grover's steps, we will stay in a subspace spanned by $|w\rangle, |w^{\perp}\rangle$. Furthermore, we will only need to consider a real, linear combination of $|w\rangle$ and $|w^{\perp}\rangle$.



all relevant states can we written as

$|\psi\rangle = \sin\varphi|w\rangle + \cos\varphi|w\rangle$

for some $\varphi \in [0, 2\pi)$

Optional: you can show that span$\{|w\rangle, |w^{\perp}\rangle\}$ is closed under U_g, U_d.

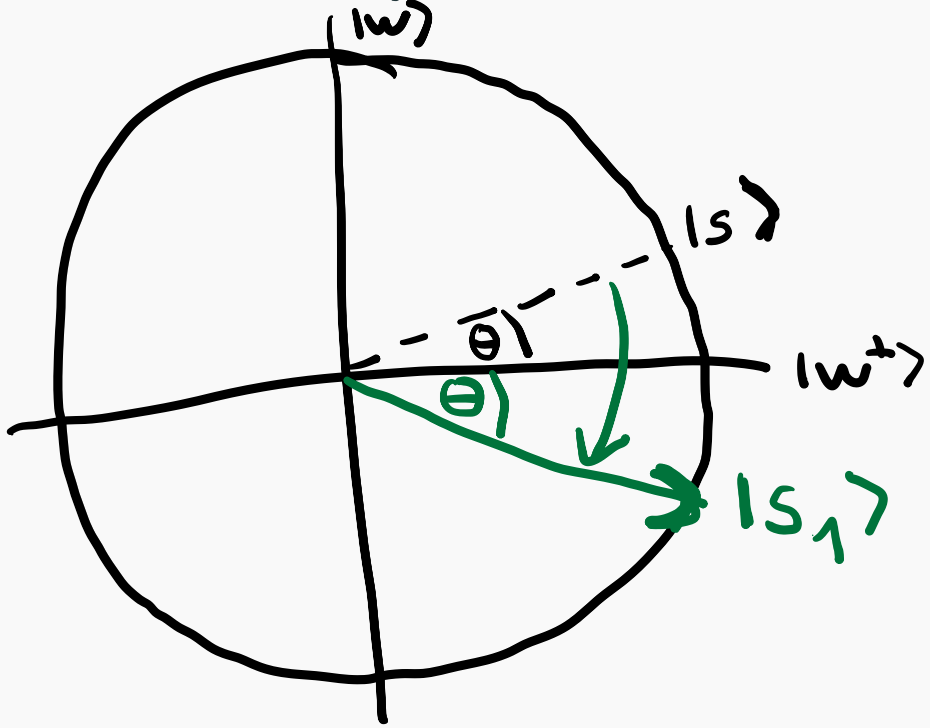
Oracle

oracle $U_G|w\rangle = -|w\rangle$ $U_G|w^\perp\rangle = |w^\perp\rangle$

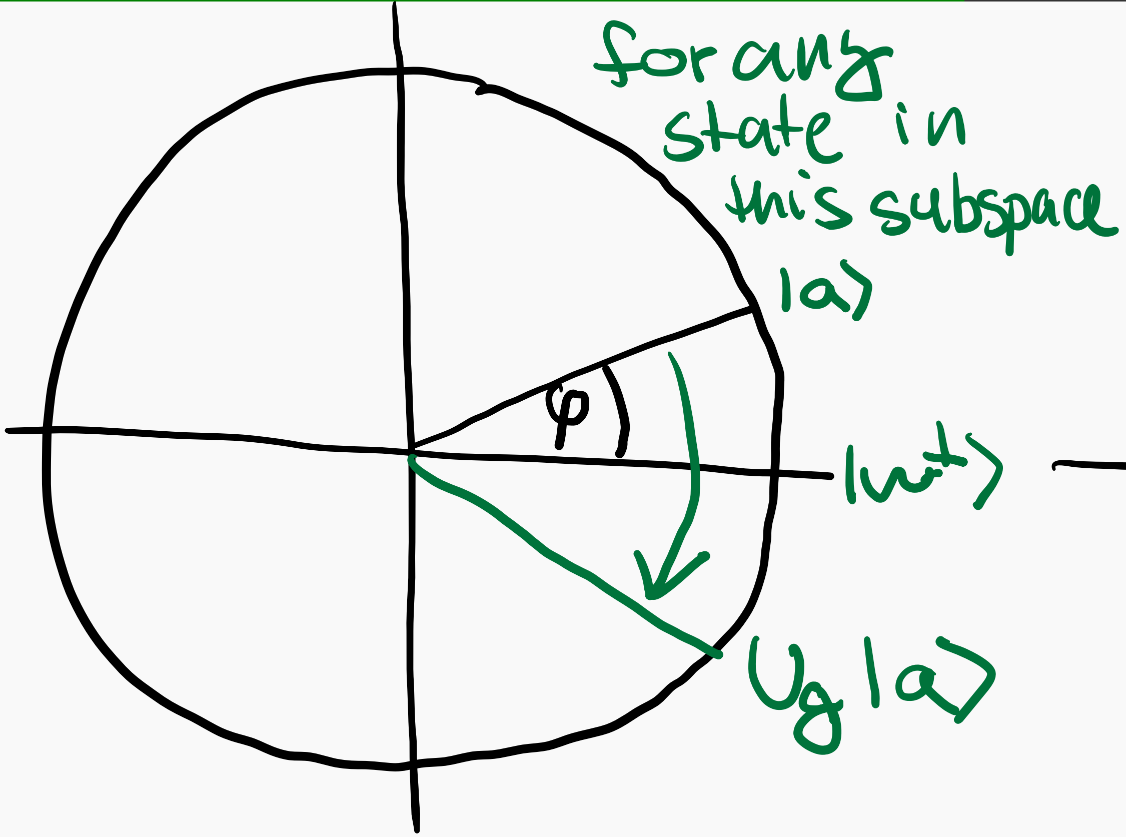
Application of U_G leads to $U_G|x(x\neq w)\rangle = |x\rangle$

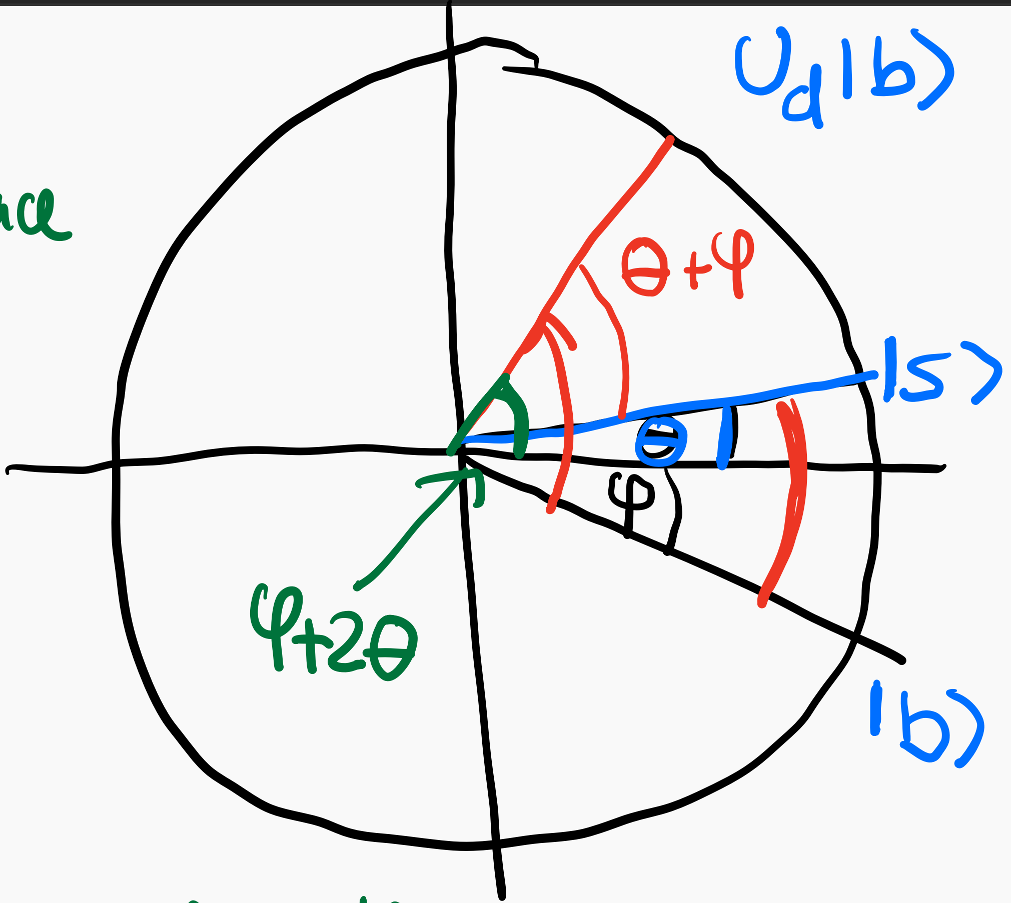
$$\begin{aligned} U_\mathrm{G}\,|\boldsymbol{s}\rangle &= -\sqrt{\frac{1}{N}}\,|\boldsymbol{w}\rangle + \sqrt{\frac{N-1}{N}}\,\left|\boldsymbol{w}^\perp\right\rangle \\ &= -\sin\theta\,|\boldsymbol{w}\rangle + \cos\theta\,\left|\boldsymbol{w}^\perp\right\rangle. \end{aligned}$$

Geometrically, the oracle U_G reflects the vector $|\boldsymbol{s}\rangle$ along the axis $\left|\boldsymbol{w}^\perp\right\rangle$.



Geometric interpretation





$$U_g|a\rangle = U_g(\sin\varphi|w\rangle + \cos\varphi|w^\perp\rangle$$

$$= -\sin\varphi|w\rangle + \cos\varphi|w^\perp\rangle$$

U_d - reflection around uniform superposition $|s\rangle$

Diffusion operator

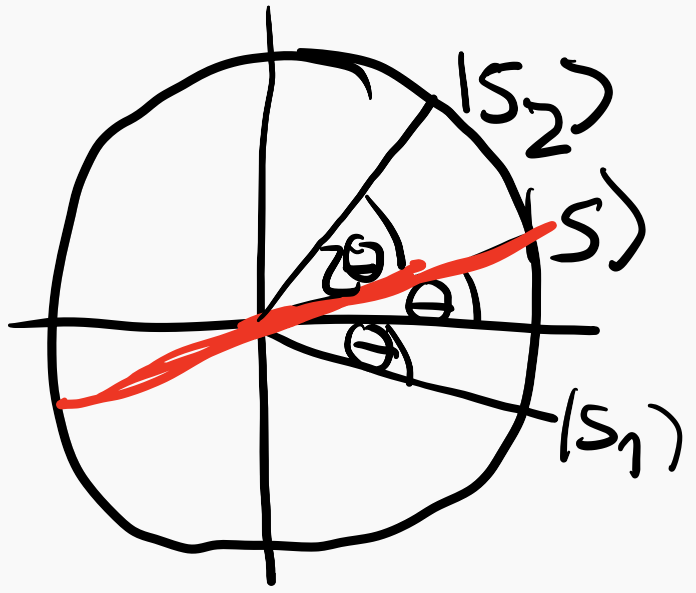
Application of U_d at the third step t_3 to Eq. (2) is equivalent to reflect the state $U_\mathrm{G}\left|\boldsymbol{s}\right\rangle$ along the axis $\left|\boldsymbol{s}\right\rangle$. Therefore,

$$U_\mathrm{d}U_\mathrm{G}\left|\boldsymbol{s}\right\rangle = \sin 3\theta\left|\boldsymbol{w}\right\rangle + \cos 3\theta\left|\boldsymbol{w}^\perp\right\rangle .$$

$$U_d\left|s_1\right\rangle = U_d\left(-\sin\theta\left|w\right\rangle + \cos\theta\left|w^\perp\right\rangle\right)$$

$$= \sin(2\theta)\left|w\right\rangle + \cos(3\theta)\left|w^\perp\right\rangle$$

$$= \left|s_2\right\rangle$$



After $U_d U_G\left|s\right\rangle$

we went from

$\sin(\theta)\left|w\right\rangle + \cos(\theta)\left|w^\perp\right\rangle$

$\rightarrow \sin(3\theta)\left|w\right\rangle + \cos(3\theta)\left|w^\perp\right\rangle$

After k steps

every $U_d U_g$ adds 2θ

By induction,

$$(U_{\mathrm{d}} U_{\mathrm{G}})^k |\boldsymbol{s}\rangle = \overset{\downarrow}{\sin[(2k+1)\theta]} |\boldsymbol{w}\rangle + \cos[(2k+1)\theta] |\boldsymbol{w}^{\perp}\rangle .$$

If we measure after k iterations, the probability of obtaining the target element $\boldsymbol{w}$ is

get close to 1

$(2k+1)\theta \approx \pi/2$

$$p_k := \Pr\{\boldsymbol{w} \text{ appears}\} = \sin^2(\underbrace{(2k+1)\theta}_{\pi/2}).$$

If we choose $k = \frac{\pi}{4\theta} - \frac{1}{2}$, then we get the state $|\boldsymbol{w}\rangle$ with certainty because $p_k = 1$. Since $\arcsin\theta \geq \theta$, then

$\sin\theta = \sqrt{\frac{1}{N}}$

$\sin\theta \approx \theta \approx \sqrt{\frac{1}{N}}$

quadratic speedup compared to classical

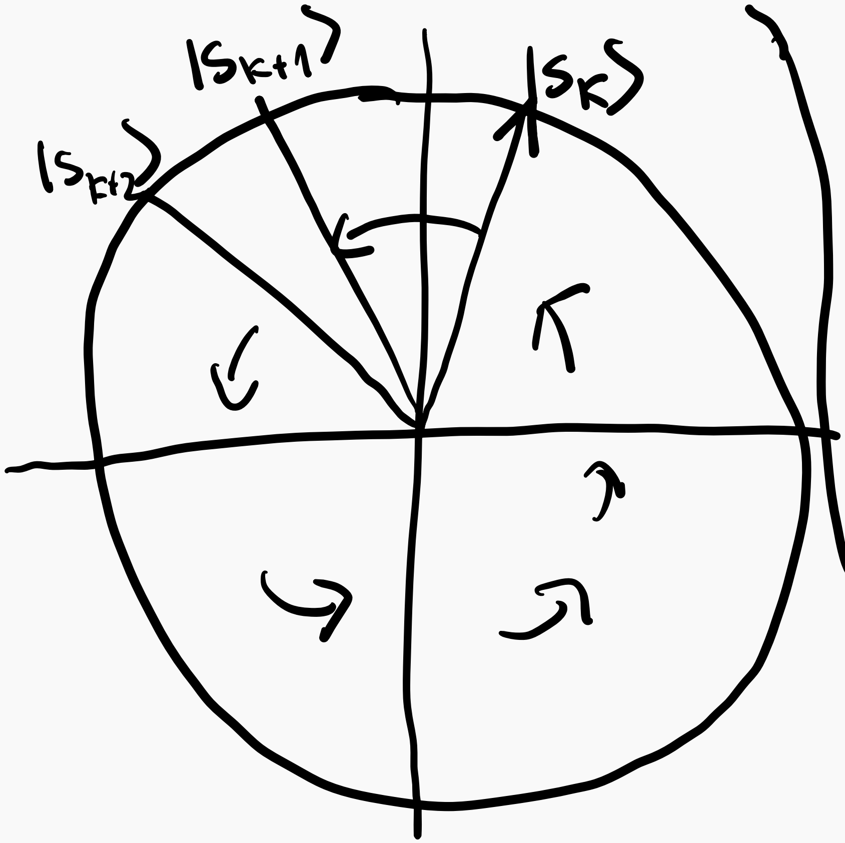
$$\tilde{k} \leq \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{N} = O(\sqrt{N}).$$

$$k \approx \frac{\pi}{4}\sqrt{N} = O(\sqrt{N})$$

Grover conclusion

If we continue rotating past $\frac{\pi}{4}\sqrt{N}$ steps, the amplitude on the good step decreases but there are modifications of the algorithm that overcome this issue.

It can be proven that Grover's algorithm is asymptotically optimal.



with ℓ marked items

→ similar derivation

if you know ℓ

$\sqrt{\frac{N}{\ell}}$ step

→ No quantum alg. can do better (asympt.)

Grover is not for database search

Grover is formulated as an oracular speedup.

When can we the oracle actually implement and still keep a speedup?

Converting an entire database into an oracle is not efficient, the algorithm would be slower than classical search.

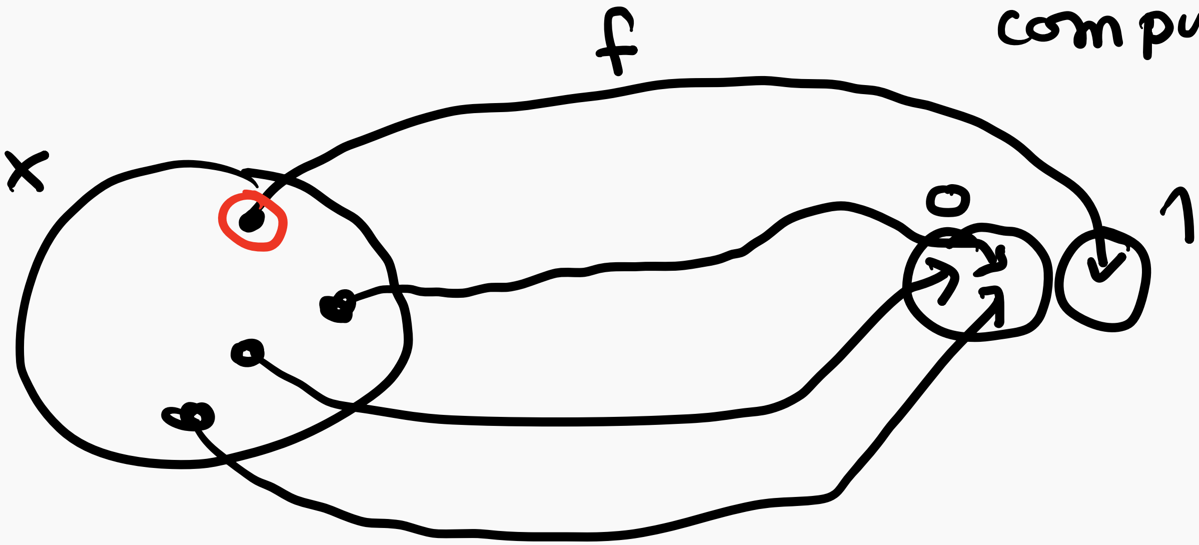
Grover as commuting preimage

Assume a function $f : x \to \{0, 1\}$. Find an x such that $f(x) = 1$.

The oracle U_G only needs to implement function f. Useful when f is easy to compute but does not have structure.

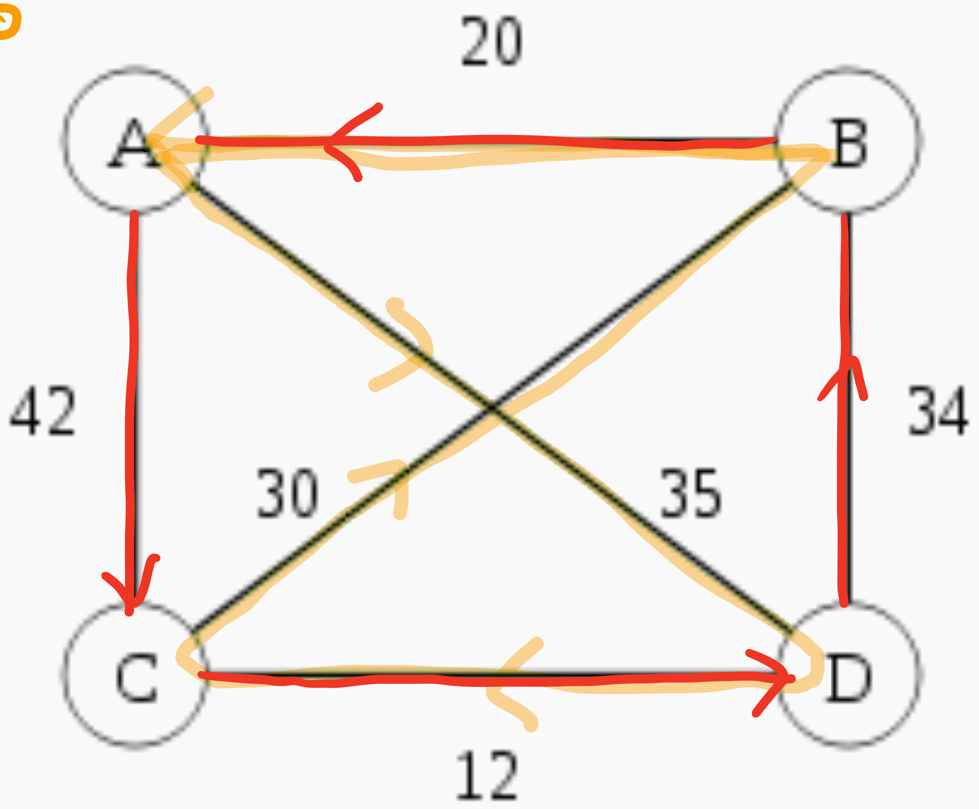
Asymptotic speedup for optimization problems, e.g. find a route shorter than m.

If you can easily compute f, we can implement U_G



Example: Travelling salesman problem (TSP)

Grover can give a quadratic speedup compared to brute-fore search (but often there are better methods)



x - route on the graph

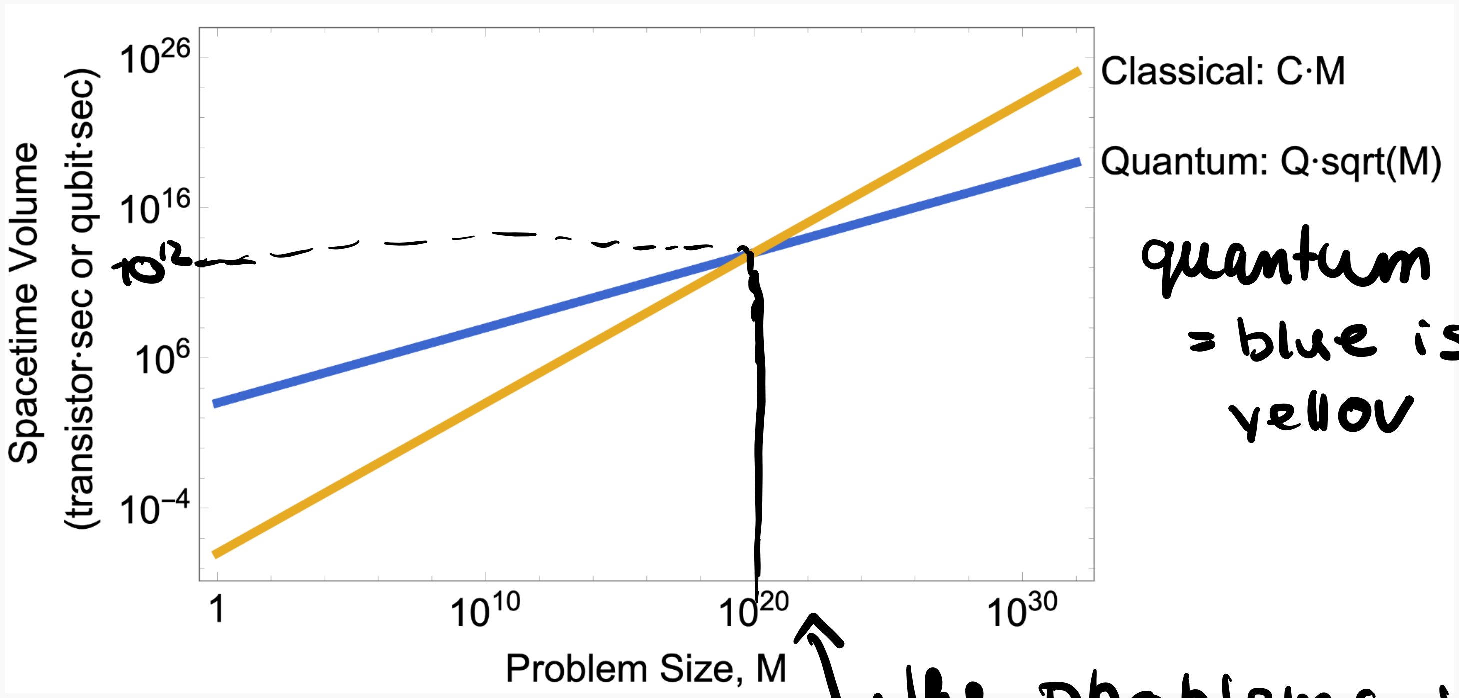
$f(x) = 1$ iff the route is shorter than $\underline{100}$

$42+12+34+20=$
108
$\rightarrow$ the red route is not marked

How useful are Grover-like speedups?

Given the overheads for compilation and error correction, quadratic speedups are not sufficient for a quantum advantage in the foreseeable future.

source: Google



quantum advantage = blue is below yellow

the problems where we get quantum advantage are way too large

Generalization of Grover - amplitude amplification

In Grover, the Hadamard transform gave us

amplify the amplitude on $|w\rangle$

$$H^{\otimes n}|0\ldots0\rangle = \sqrt{\frac{1}{N}}\,|\boldsymbol{w}\rangle + \sqrt{\frac{N-1}{N}}\,|\boldsymbol{w}^{\perp}\rangle \tag{2}$$

Assume that we have an operator V that prepares the good state $|w\rangle$ with some amplitude α

good state

$$V|0\ldots0\rangle = \alpha\,|\boldsymbol{w}\rangle + \sqrt{1-|\alpha|^2}\,|\boldsymbol{w}^{\perp}\rangle \tag{3}$$

we can repeat Grover-like like steps to prepare $|\boldsymbol{w}\rangle$ using $V, V^{\dagger}$ and U_G.

AA + other algorithm that implements V

Turing machines

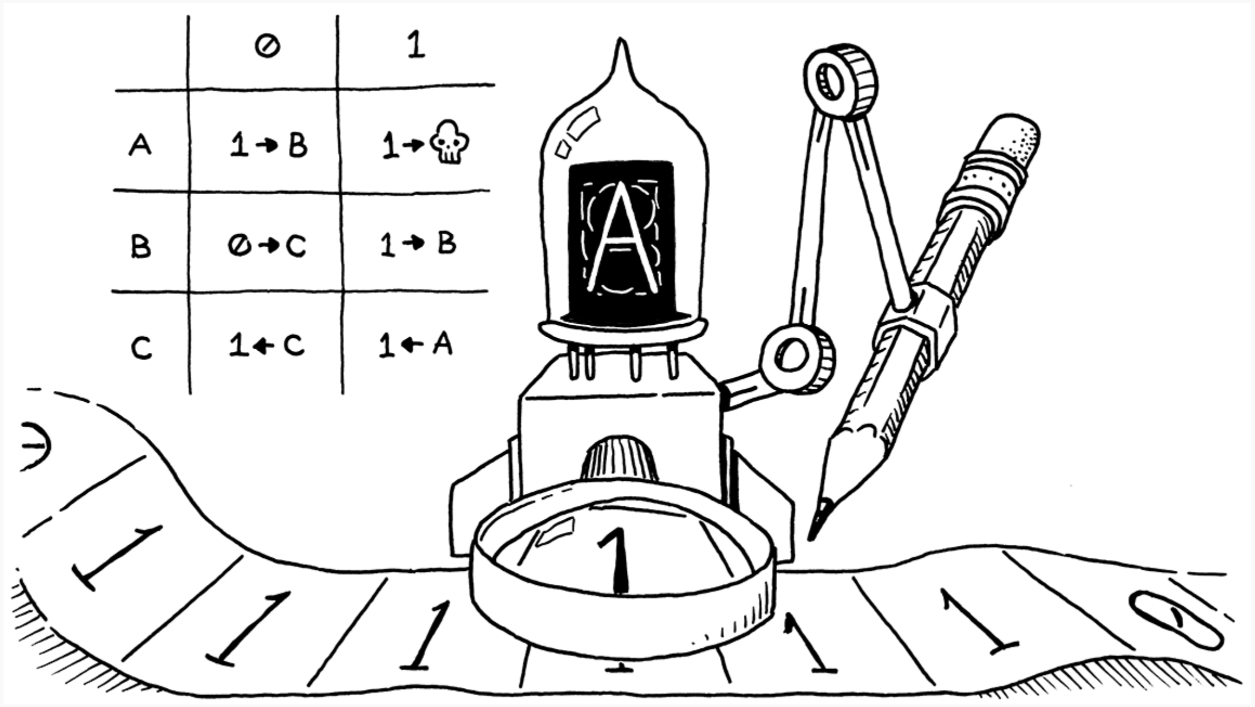


Figure 2: Turing machine

deterministic TM: there is exactly 1 allowed move

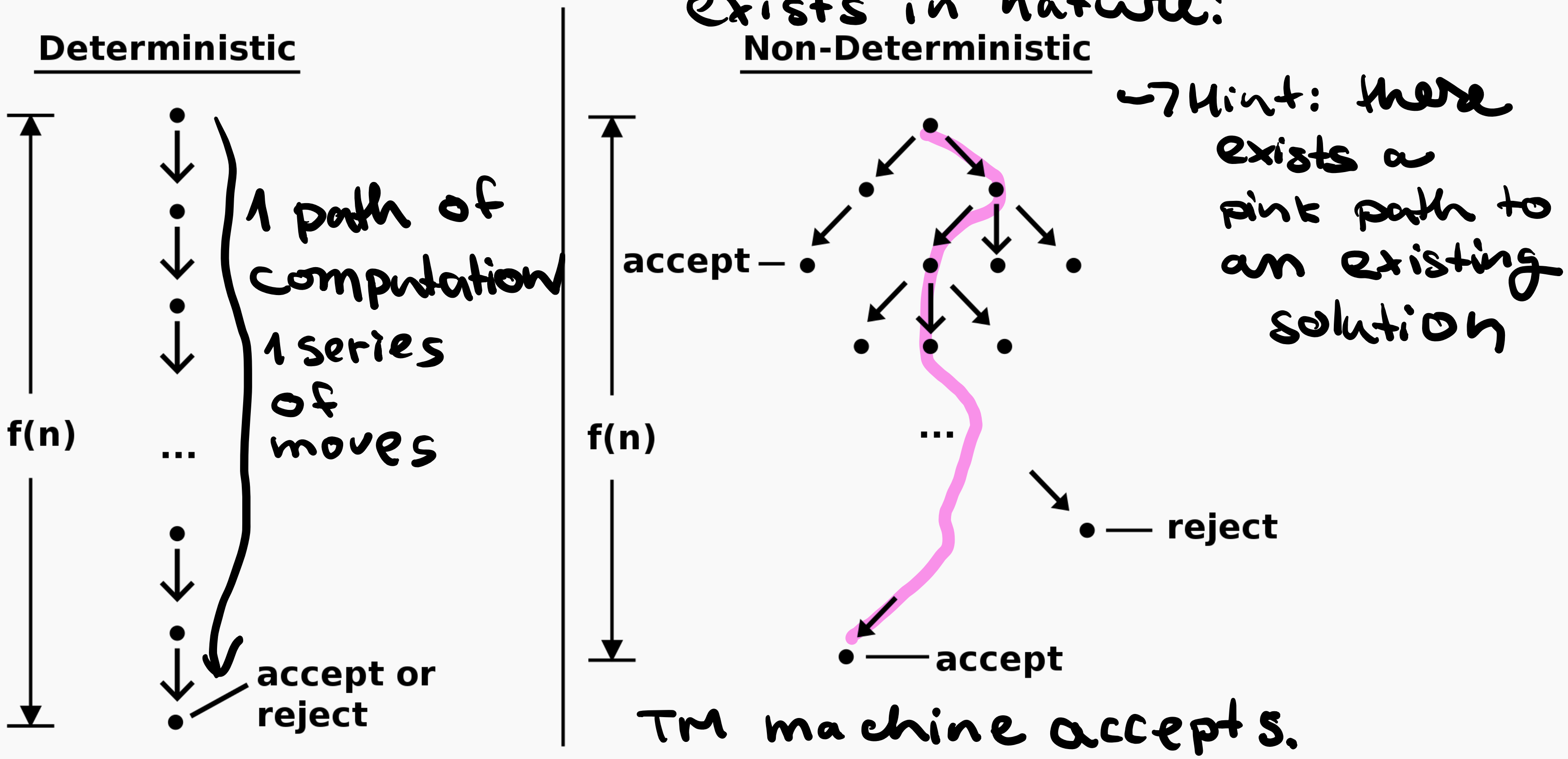
probabilistic TM: flip a coin every time head → do something
tail → do something else

non-deterministic TM: several allowed moves, TM takes all of them, accepts if there $\exists$ a series of moves that accept

Non-determinism

the transition function allows multiple possible actions for any configuration

Not a model that exists in nature!

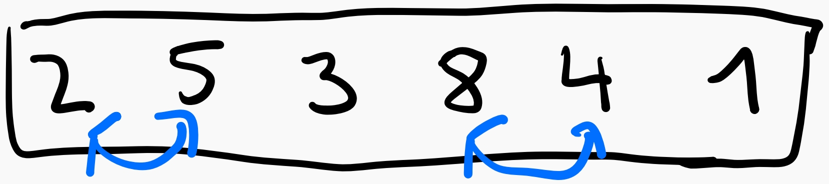


TM machine accepts.

complexity classes stranting with "N"

verify solution

Example: sort



N numbers

deterministic:
(normal)

complexity to sort is

$O(N \log N)$ - optimal

easy $O(N^2)$

nondeterministic:

What is the complexity to verify if an array is sorted? $O(N)$

Complexity classes

problems that can be solved on a deterministic Turing machine within a certain resource contain

Time-restricted: P - poly-time computation
EXP - exp-time

SPACE - PSPACE - (D) TM uses poly-amount of space on the tape
EXPSPACE etc

NP- problems where the solution can be verified in poly time

BPP - solve it in poly-time using random numbers

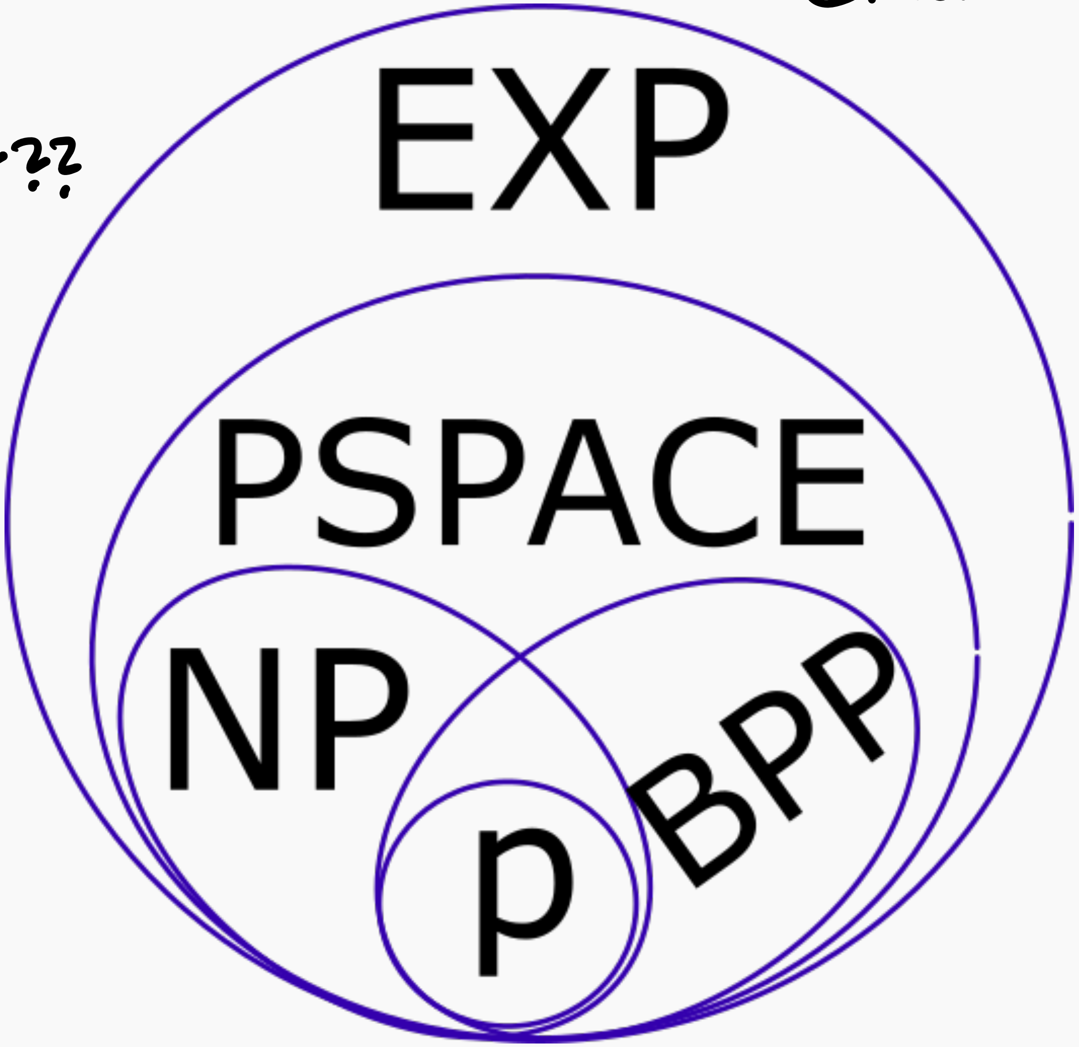
PSPACE

We can constrain the space a TM can use

conjecture:
NP≠P
maybe P= BPP???

conjecture:
NP ≠ PSPACE

every problem from P is also in NP (PSPACE, EXP, ...)



classical complexity classes

P, EXP - amount of time needed

SPACE

BPP - poly time + randomness

NP - verify in poly time

P vs NP

Figure 3: Selected complexity classes and the relationships between them.

Some subsets might not be strict.

Complete and hard problems

(NP)-hard - Imagine you have a magic box (or oracle) to solve an NP-hard problem, this oracle can be used to efficiently solve any problem in NP.

(NP)-complete → the problem is NP-hard and it is in NP.

→ "hardest" problems in NP.

also for different complexity classes PSPACE-complete...

If a problem is NP complete /hard ⇒ if don't think we can solve it on a classical (or even quantum) computer.

THE END

BQP

- class of problems that can be solved on a quantum computer in polynomial time

Let $A = (A_{yes}, A_{no})$ be a promise problem and let $c, s : \mathbb{N} \to [0,1]$ be functions. Then

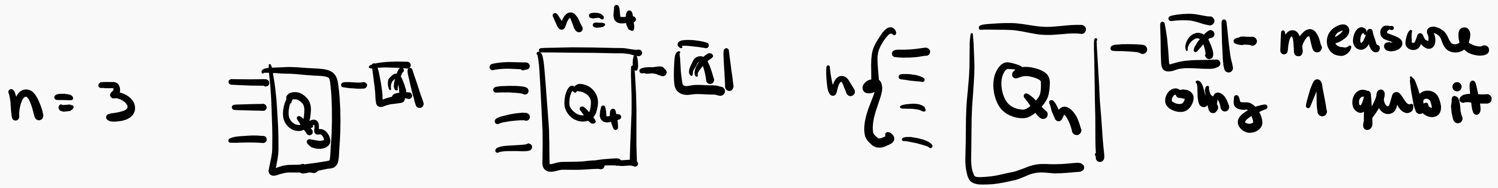
$A \in \mathbf{BQP}(c,s)$ if and only if there exists a polynomial-time uniform family of quantum circuits $Q_n : n \in N$, where Q_n takes n qubits as input and outputs 1 bit, such that

if $x \in A_{yes}$ then $\Pr[Q_{|x|}(x) = 1] \geq c(|x|)$, and

if $x \in A_{no}$ then $Pr[Q_{|x|}(x) = 1] \leq s(|x|)$.

promise

The class **BQP** is defined as $\mathbf{BQP} = \mathbf{BQP}(2/3, 1/3)$.

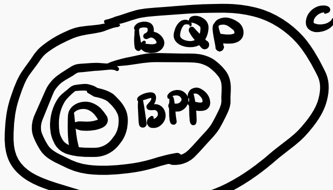


gates is polynomial in n,

- we can easily construct a circuit for each

H -Ising on n qubits,
Is $\langle\psi|H|\psi\rangle \leq E$
given a quantum circuit for $|\psi\rangle$

poly-time complexity classes



BQP

Problems solved efficiently on a quantum computer

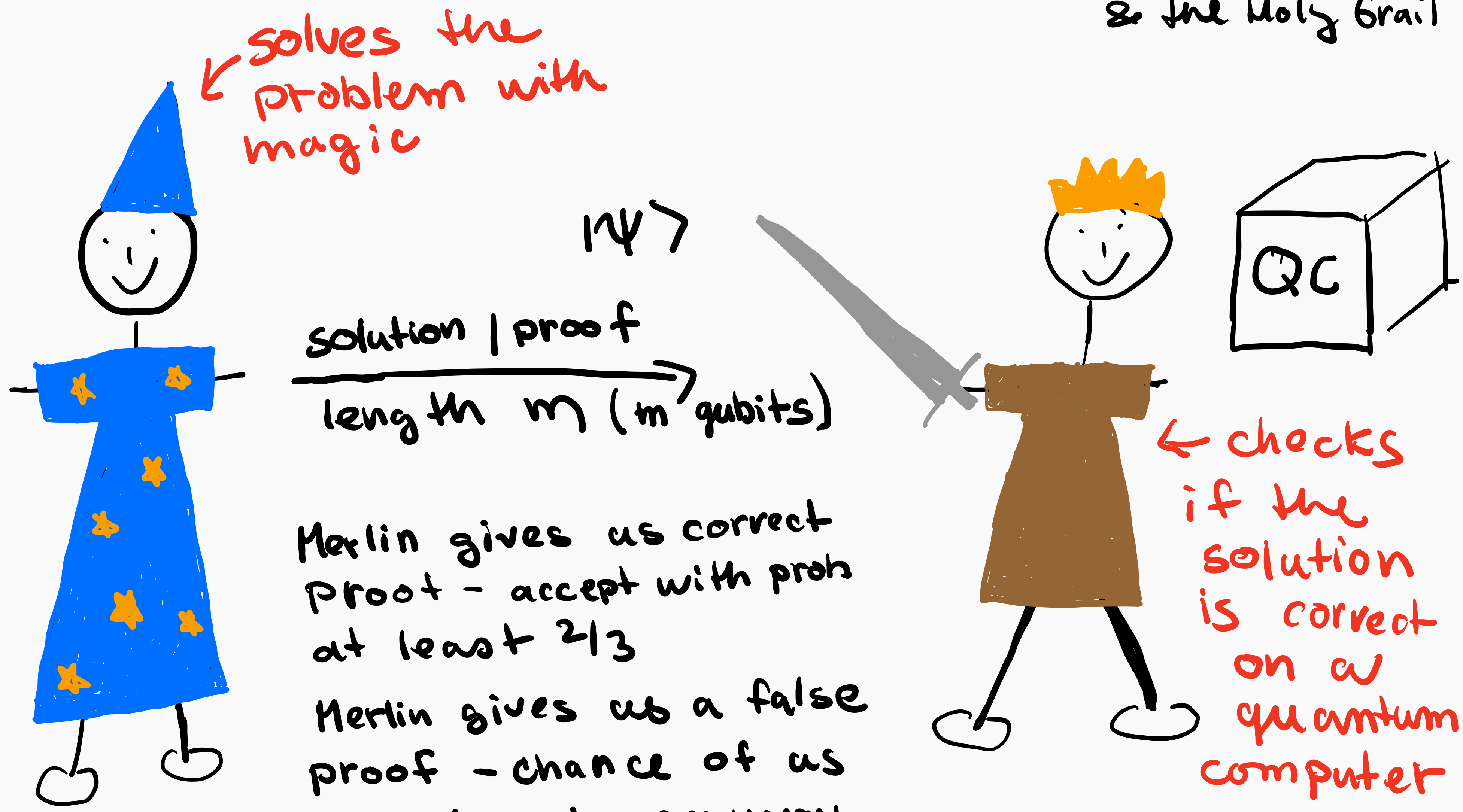
BQP-complete problems

- "hardest" problems that can be solved on a quantum computer
- we don't believe these can be solved efficiently on a classical computer (otherwsice we could simulate quantum computers efficiently)
- Hamiltonian simulation, quantum linear systems of equations (HHL)

Quantum analogue of NP

QMA - Quantum Merlin Arthur

watch Monty Python & the Holy Grail


solves the problem with magic
|ψ⟩
solution | proof
length m (m qubits)
QC
checks if the solution is correct on a quantum computer

Merlin gives us correct proof - accept with prob at least 2/3

Merlin gives us a false proof - chance of us believing him anyway is at most 1/3

QMA

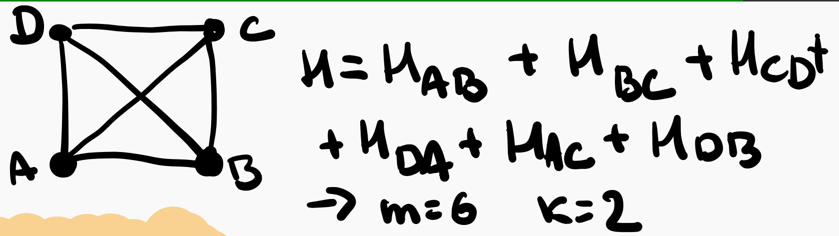
Let $A = (A_{yes}, A_{no})$ be a promise problem and let $\overset{2/3}{c}, \overset{1/3}{s} : N \to [0,1]$ be functions. Then $A \in \mathbf{QMA}(c,s)$ if and only if there exists a polynomial-time uniform family of quantum circuits $\{Q_n : n \in \mathbb{N}\}$, where Q_n takes $p(n)$ qubits as input for some polynomial p and outputs 1 bit, such that

- (Completeness) if $x \in A_{yes}$ then there exists an $p(n)$-qubit state $|\psi\rangle$ such that $Pr\big[Q_n(x, |\psi\rangle) = 1\big] \geq c(n)$), and
- (Soundness) if $x \in A_{no}$ then for all $p(n)$-qubit state $|\psi\rangle$, $Pr\big[Q_n(x, |\psi\rangle) = 1\big] \leq s(n)$.

The class QMA is defined as QMA(2/3,1/3).

QMA complete problems are unlikely to be solved efficiently on a quantum computer.

QMA complete problems



D
C
A
B
$H = H_{AB} + H_{BC} + H_{CD} + H_{DA} + H_{AC} + H_{DB}$
$\rightarrow m=6 \quad k=2$

k-local Hamiltonian problem:

$k \geq 2$

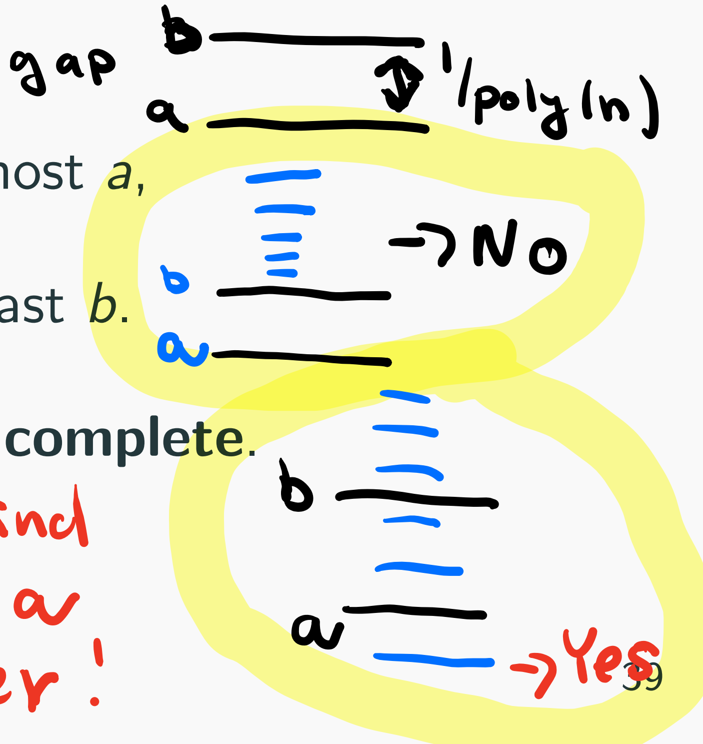
A k-local Hamiltonian H is a summation $H = \sum_{j=1}^{m} H_j$ of local terms H_j acting on at most k qubits (out of n qubits). The k-local Hamiltonian problem is the promise problem with

Input: (H, a, b) where H is a k-local Hamiltonian, a, b are real numbers such that $b - a \geq 1/poly(n)$,

Decision Problem

(1) Yes instances: The smallest eigenvalue of H is at most a,

(0) No instances: The smallest eigenvalue of H is at least b.



For $k \geq 2$, the local Hamiltonian problem is **QMA complete**.

Promise: we never see

We cannot always find ground states on a quantum computer!

Quantum classes

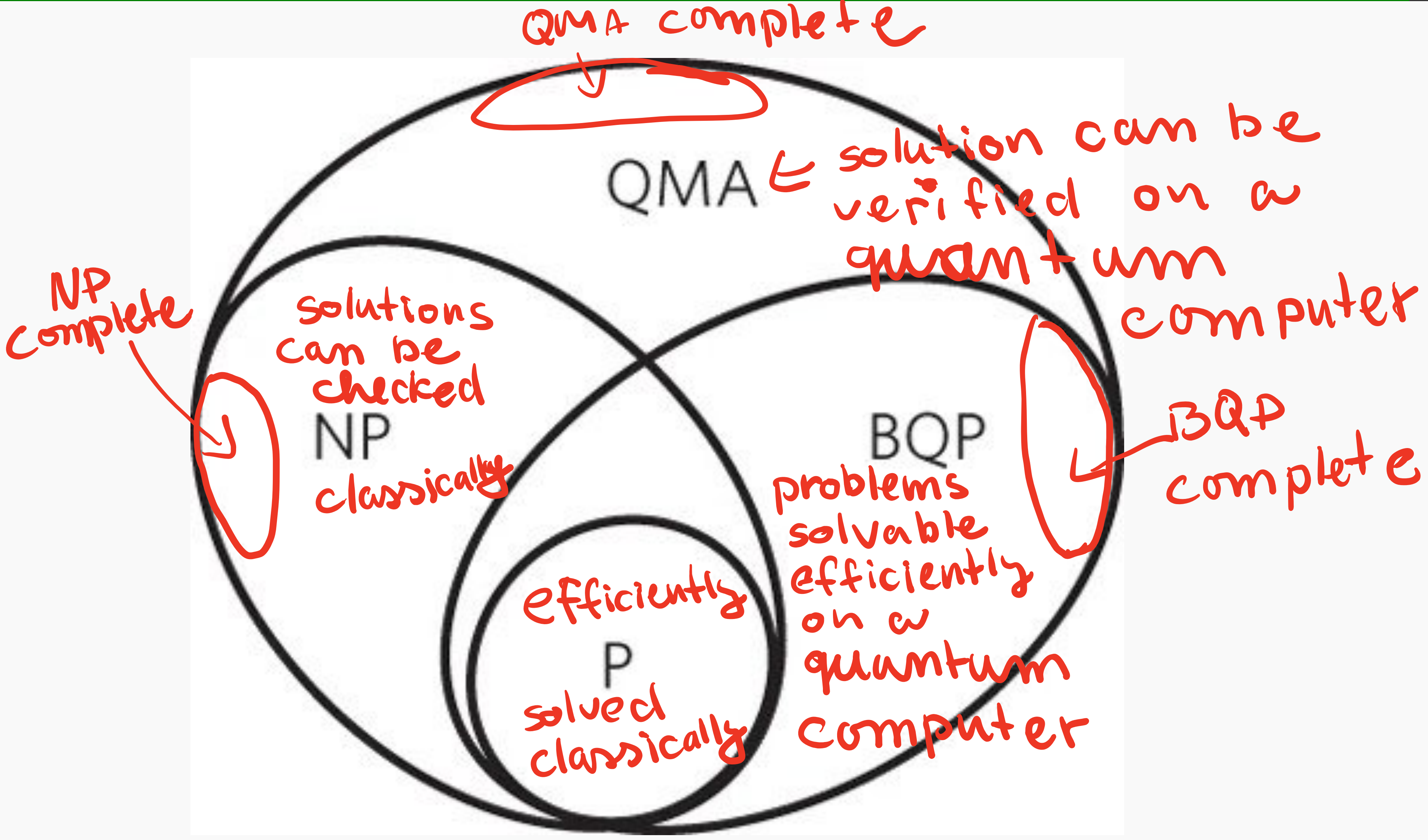


Figure 4: Quantum complexity classes in relation to P and NP. Source: Schuch and Verstraete

There is more!

UTS algorithms class

Complexity ZOO