# A brief overview of some recent quantum algorithms

Ashley Montanaro

Centre for Quantum Information and Foundations,
Department of Applied Mathematics and Theoretical Physics,
University of Cambridge

16 May 2013

UNIVERSITY OF
CAMBRIDGE

# Quantum algorithms

Is there life beyond Shor and Grover?

# Quantum algorithms

Is there life beyond Shor and Grover?

- The Quantum Algorithm Zoo (http://math.nist.gov/quantum/zoo/) cites 199 papers on quantum algorithms.

- Further, in recent years a number of conceptually different underlying techniques for quantum algorithm design have been developed.

# This tutorial

In this talk, I'll focus on the techniques behind a number of quantum algorithms, and how they can be applied to various problems.

# This tutorial

In this talk, I'll focus on the techniques behind a number of quantum algorithms, and how they can be applied to various problems.

In particular:

| **Techniques** | **Applications** |
| --- | --- |
| • Phase estimation<br>• Hamiltonian simulation<br>• Quantum walks<br>• Learning graphs | • Linear equations<br>• Escaping from mazes<br>• Element distinctness<br>• Searching for subgraphs |

# This tutorial

In this talk, I'll focus on the techniques behind a number of quantum algorithms, and how they can be applied to various problems.

In particular:

| **Techniques** | **Applications** |
| --- | --- |
| • Phase estimation<br>• Hamiltonian simulation<br>• Quantum walks<br>• Learning graphs | • Linear equations<br>• Escaping from mazes<br>• Element distinctness<br>• Searching for subgraphs |

Disclaimer: This is a broad overview which will omit most technical details.

# Not this tutorial

Some other quantum algorithms which I won't mention in this talk:

- Hidden subgroups and optimal measurement (e.g. [Bacon et al 0504083])
- Number-theoretic problems (e.g. [Fontein and Wocjan 1111.1348], ...)
- Formula evaluation (e.g. [Reichardt and Špalek 0710.2630])
- Tensor contraction (e.g. [Arad and Landau 0805.0040])
- Hidden shift problems (e.g. [Gavinsky et al 1103.3017])
- Adiabatic optimisation (e.g. [Farhi et al 0001106])
- ...

(all citations are to arXiv identifiers)

# Primitive: Phase estimation

**Phase estimation** [Cleve et al 9708016] [Kitaev 9511026]

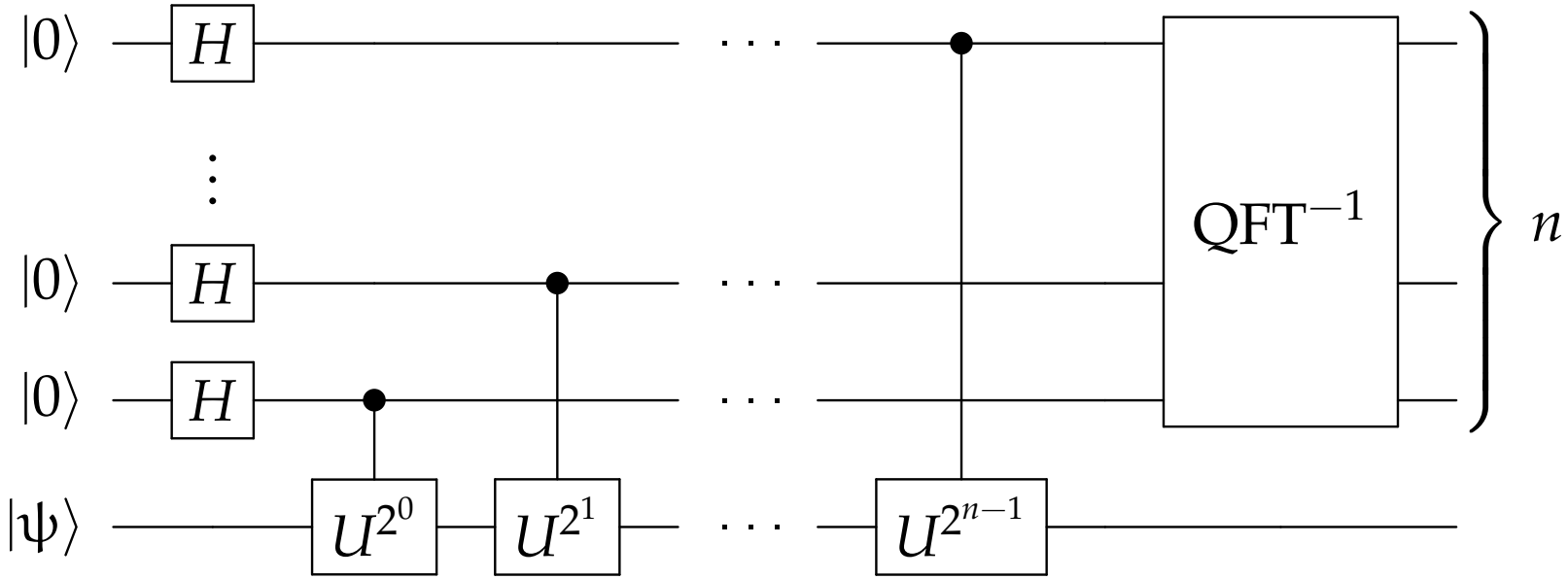
Given access to a unitary $U$ and an eigenvector $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$, we can estimate $\phi$ up to additive error $\epsilon$, with 99% probability of success, by using $U$ $O(1/\epsilon)$ times.

# Primitive: Phase estimation

**Phase estimation** [Cleve et al 9708016] [Kitaev 9511026]

Given access to a unitary $U$ and an eigenvector $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$, we can estimate $\phi$ up to additive error $\epsilon$, with 99% probability of success, by using $U$ $O(1/\epsilon)$ times.

We apply the following circuit with $n = O(\log 1/\epsilon)$:




and then measure the first $n$ qubits.

# Phase estimation

Let the measurement result be $x$ and output as our guess for $\phi$

$$0.x_1x_2\ldots x_n = \frac{x_1}{2} + \frac{x_2}{4} + \cdots + \frac{x_n}{2^n}.$$

# Phase estimation

Let the measurement result be $x$ and output as our guess for $\phi$

$$0.x_1x_2\ldots x_n = \frac{x_1}{2} + \frac{x_2}{4} + \cdots + \frac{x_n}{2^n}.$$

Why does this work?

- The state before we apply the inverse QFT is

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i \phi y} |y\rangle |\psi\rangle.$$

# Phase estimation

Let the measurement result be $x$ and output as our guess for $\phi$

$$0.x_1x_2\ldots x_n = \frac{x_1}{2} + \frac{x_2}{4} + \cdots + \frac{x_n}{2^n}.$$

Why does this work?

- The state before we apply the inverse QFT is

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i \phi y} |y\rangle |\psi\rangle.$$

- So, if $\phi = z/2^n$ for some integer $z$, the probability of getting outcome $x$ is

$$\frac{1}{2^{2n}} \left| \sum_{y=0}^{2^n-1} e^{2\pi i y(\phi - x/2^n)} \right|^2 = \frac{1}{2^{2n}} \left| \sum_{y=0}^{2^n-1} e^{\pi i y(z-x)/2^{n-1}} \right|^2 = \delta_{xz}.$$

# Phase estimation

Let the measurement result be $x$ and output as our guess for $\phi$

$$0.x_1x_2\ldots x_n = \frac{x_1}{2} + \frac{x_2}{4} + \cdots + \frac{x_n}{2^n}.$$

Why does this work?

- The state before we apply the inverse QFT is

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i \phi y} |y\rangle |\psi\rangle.$$

- So, if $\phi = z/2^n$ for some integer $z$, the probability of getting outcome $x$ is

$$\frac{1}{2^{2n}} \left| \sum_{y=0}^{2^n-1} e^{2\pi i y(\phi - x/2^n)} \right|^2 = \frac{1}{2^{2n}} \left| \sum_{y=0}^{2^n-1} e^{\pi i y(z-x)/2^{n-1}} \right|^2 = \delta_{xz}.$$

- If $\phi \approx z/2^n$, we still output $\widetilde{z} \approx z$ with high probability.

# Primitive: Hamiltonian simulation

**Hamiltonian simulation**

Given a description of a Hamiltonian $H$ on $n$ qubits and a time $t$, implement $U$ such that $\|U - e^{-iHt}\| \leqslant \epsilon$.

# Primitive: Hamiltonian simulation

## Hamiltonian simulation

Given a description of a Hamiltonian $H$ on $n$ qubits and a time $t$, implement $U$ such that $\|U - e^{-iHt}\| \leqslant \epsilon$.

Classic example:

## Simulation of $k$-local Hamiltonians

Let $H$ be a $k$-local Hamiltonian, i.e.

$$H = \sum_{j=1}^{m} H_j$$

where each $H_j$ acts only on $k = O(1)$ qubits and satisfies $\|H_j\| \leqslant L$. Then $H$ can be simulated for time $t$ in $\mathrm{poly}(n, L, t, 1/\epsilon)$ time [Lloyd, Science 273, 1073-1078 (1996)].

# Simulation of sparse Hamiltonians

A generalisation of this problem:

**Sparse Hamiltonian simulation**

Let $H$ be a Hamiltonian on $n$ qubits such that each row of $H$ has at most $d$ non-zero entries. We are given black-box access to $H$ via a function $f$ such that $f(i, j)$ returns the $j$'th non-zero element of row $i$.

We say that $H$ is $d$-sparse. (NB: a $k$-local Hamiltonian is $2^k$-sparse.)

# Simulation of sparse Hamiltonians

A generalisation of this problem:

**Sparse Hamiltonian simulation**

Let $H$ be a Hamiltonian on $n$ qubits such that each row of $H$ has at most $d$ non-zero entries. We are given black-box access to $H$ via a function $f$ such that $f(i,j)$ returns the $j$'th non-zero element of row $i$.

We say that $H$ is $d$-sparse. (NB: a $k$-local Hamiltonian is $2^k$-sparse.)

**Theorem** [Aharonov and Ta-Shma 0301023, ...]

$H$ can be simulated for time $t$ up to error $\epsilon$ with $\mathrm{poly}(n, \|H\|, t, d, 1/\epsilon)$ uses of $f$.

# Simulation of sparse Hamiltonians (sketch)

We can simulate $H$ by decomposing it as a sum of efficiently simulable Hamiltonians and recombining using the Lie-Trotter formula

$$e^{-iAt}e^{-iBt} = e^{-i(A+B)t} + O(t^2 \max\{\|A\|, \|B\|\}^2).$$

# Simulation of sparse Hamiltonians (sketch)

We can simulate $H$ by decomposing it as a sum of efficiently simulable Hamiltonians and recombining using the Lie-Trotter formula

$$e^{-iAt}e^{-iBt} = e^{-i(A+B)t} + O(t^2 \max\{\|A\|, \|B\|\}^2).$$

First imagine that $H$ is 1-sparse, i.e. each row of $H$ has at most one non-zero entry. Then $H$ looks like this:

$$\begin{pmatrix} & & * & & \\ & * & & & \\ & & & * & \\ * & & & & \\ & & * & & \\ & & & & * \\ & & & * & \end{pmatrix}$$

where only starred entries are non-zero.

# Simulation of sparse Hamiltonians (sketch)

- Thus $H$ is equivalent to a direct sum of 1 and 2-dimensional Hamiltonians $H_k$, each of which can be simulated efficiently.
- e.g. a 2D part $H_k$ on rows $i$ and $f(i,1)$ can be simulated using the Solovay-Kitaev theorem to implement $e^{-iH_k t}$ on the space spanned by $|i\rangle$, $|f(i,1)\rangle$.

# Simulation of sparse Hamiltonians (sketch)

- Thus $H$ is equivalent to a direct sum of 1 and 2-dimensional Hamiltonians $H_k$, each of which can be simulated efficiently.
- e.g. a 2D part $H_k$ on rows $i$ and $f(i,1)$ can be simulated using the Solovay-Kitaev theorem to implement $e^{-iH_k t}$ on the space spanned by $|i\rangle$, $|f(i,1)\rangle$.
- Claim: Any $d$-sparse Hamiltonian can be decomposed as a sum of $\mathrm{poly}(d,n)$ 1-sparse Hamiltonians. This decomposition can be found efficiently.

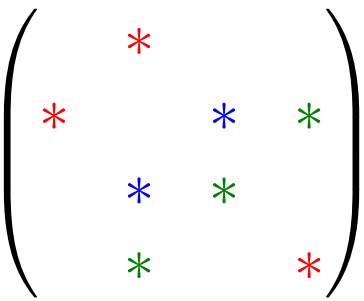
# Simulation of sparse Hamiltonians (sketch)

- Thus $H$ is equivalent to a direct sum of 1 and 2-dimensional Hamiltonians $H_k$, each of which can be simulated efficiently.
- e.g. a 2D part $H_k$ on rows $i$ and $f(i,1)$ can be simulated using the Solovay-Kitaev theorem to implement $e^{-iH_k t}$ on the space spanned by $|i\rangle$, $|f(i,1)\rangle$.
- Claim: Any $d$-sparse Hamiltonian can be decomposed as a sum of poly$(d,n)$ 1-sparse Hamiltonians. This decomposition can be found efficiently.

$$\begin{pmatrix} & * & & \\ * & & * & * \\ & * & * & \\ & * & & * \end{pmatrix}$$

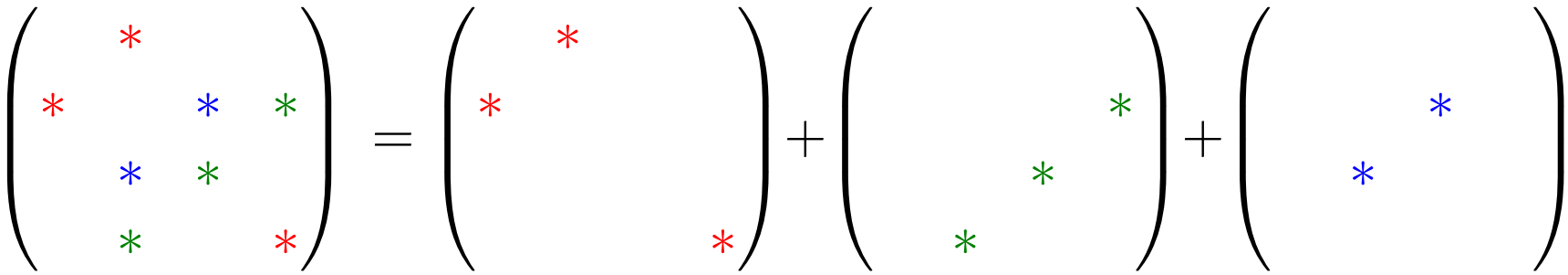
# Simulation of sparse Hamiltonians (sketch)

- Thus $H$ is equivalent to a direct sum of 1 and 2-dimensional Hamiltonians $H_k$, each of which can be simulated efficiently.
- e.g. a 2D part $H_k$ on rows $i$ and $f(i,1)$ can be simulated using the Solovay-Kitaev theorem to implement $e^{-iH_k t}$ on the space spanned by $|i\rangle$, $|f(i,1)\rangle$.
- Claim: Any $d$-sparse Hamiltonian can be decomposed as a sum of poly$(d,n)$ 1-sparse Hamiltonians. This decomposition can be found efficiently.



# Simulation of sparse Hamiltonians (sketch)

- Thus $H$ is equivalent to a direct sum of 1 and 2-dimensional Hamiltonians $H_k$, each of which can be simulated efficiently.
- e.g. a 2D part $H_k$ on rows $i$ and $f(i,1)$ can be simulated using the Solovay-Kitaev theorem to implement $e^{-iH_k t}$ on the space spanned by $|i\rangle$, $|f(i,1)\rangle$.
- Claim: Any $d$-sparse Hamiltonian can be decomposed as a sum of poly$(d,n)$ 1-sparse Hamiltonians. This decomposition can be found efficiently.

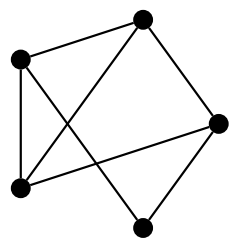


# Application: Simulating continuous-time quantum walks

- A quantum walk is the quantum analogue of a classical random walk.

# Application: Simulating continuous-time quantum walks

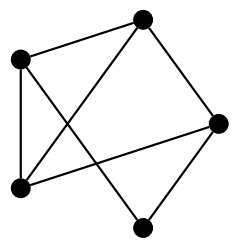
- A quantum walk is the quantum analogue of a classical random walk.
- A continuous-time quantum walk for time $t$ on a graph $G$ with adjacency matrix $A$ is simply the application of the unitary operator $e^{-iAt}$.

$G =$ 

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

# Application: Simulating continuous-time quantum walks

- A quantum walk is the quantum analogue of a classical random walk.
- A continuous-time quantum walk for time $t$ on a graph $G$ with adjacency matrix $A$ is simply the application of the unitary operator $e^{-iAt}$.

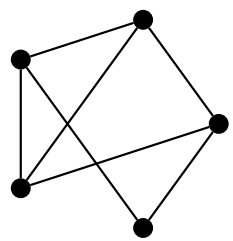
$G =$ 

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- We consider quantum walks on graphs $G$ in a black-box model where, given a vertex $i$, we can query an oracle $f(i,j)$ to learn the $j$'th neighbour of $i$.

# Application: Simulating continuous-time quantum walks

- A quantum walk is the quantum analogue of a classical random walk.
- A continuous-time quantum walk for time $t$ on a graph $G$ with adjacency matrix $A$ is simply the application of the unitary operator $e^{-iAt}$.

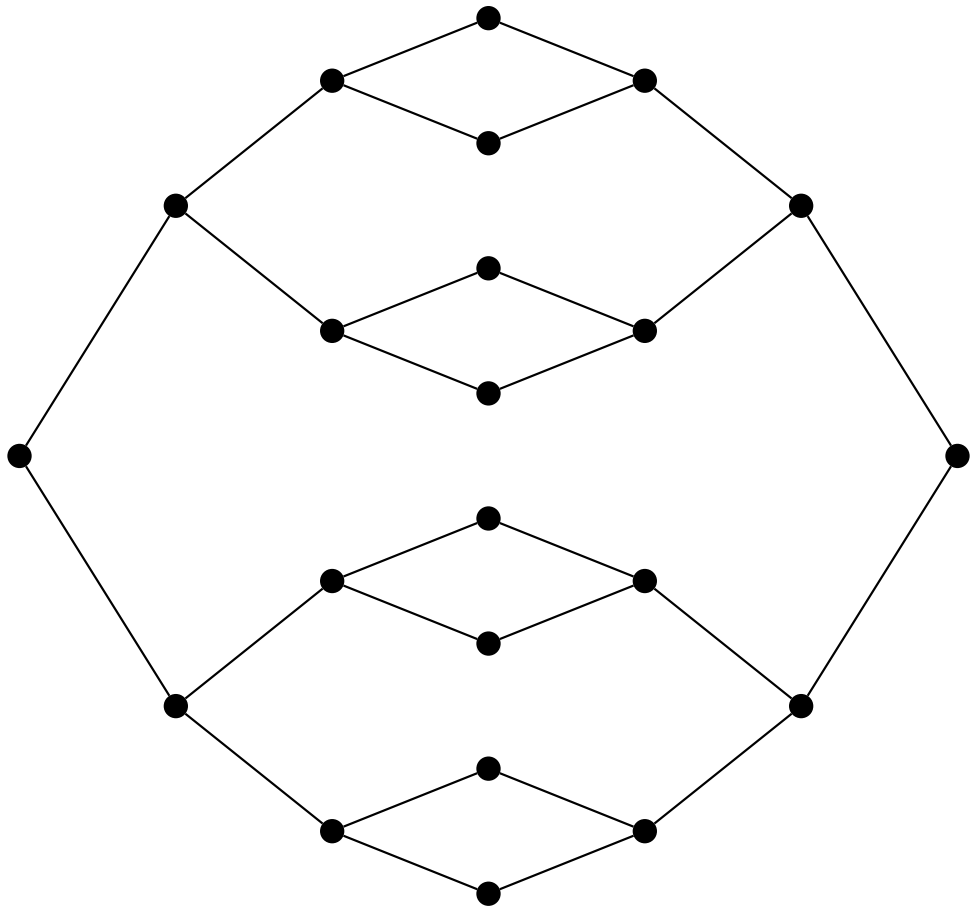
$G =$ 

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- We consider quantum walks on graphs $G$ in a black-box model where, given a vertex $i$, we can query an oracle $f(i,j)$ to learn the $j$'th neighbour of $i$.
- If $G$ is sparse (has maximum degree $O(\log N)$) then the above algorithm gives an efficient (i.e. $\mathrm{poly}(\log N)$-time) simulation of a quantum walk on $G$.

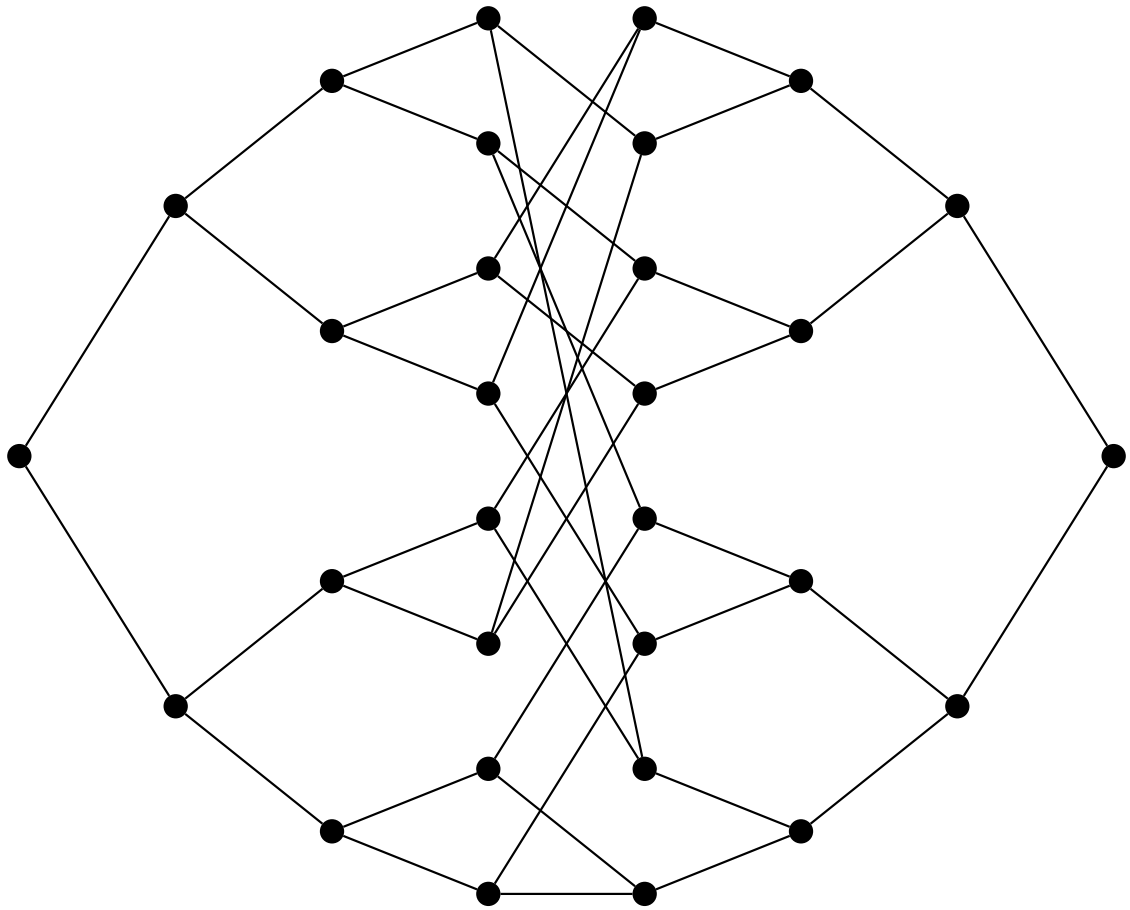
# Quantum walk on the glued trees graph

Consider the graph formed by gluing two binary trees with $N$ vertices together, e.g.:



# Quantum walk on the glued trees graph

Now add a random cycle in the middle:



# Quantum walk on the glued trees graph

**Theorem** [Childs et al 0209131]

- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.

# Quantum walk on the glued trees graph

**Theorem** [Childs et al 0209131]

- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.
- Any classical algorithm given black-box access to the graph requires $\Omega(N^{1/6})$ queries to find the exit.

# Quantum walk on the glued trees graph

**Theorem** [Childs et al 0209131]

- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.
- Any classical algorithm given black-box access to the graph requires $\Omega(N^{1/6})$ queries to find the exit.

Basic idea behind proof:

- By symmetry, the quantum walk is restricted to the subspace spanned by uniform superpositions of vertices in each column.

# Quantum walk on the glued trees graph

**Theorem** [Childs et al 0209131]

- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.
- Any classical algorithm given black-box access to the graph requires $\Omega(N^{1/6})$ queries to find the exit.

Basic idea behind proof:

- By symmetry, the quantum walk is restricted to the subspace spanned by uniform superpositions of vertices in each column.
- On this subspace, the graph looks like a line with a defect in the middle. The walk moves from one end to the other in $O(\log N)$ time.

# Quantum walk on the glued trees graph

**Theorem** [Childs et al 0209131]

- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.
- Any classical algorithm given black-box access to the graph requires $\Omega(N^{1/6})$ queries to find the exit.

Basic idea behind proof:

- By symmetry, the quantum walk is restricted to the subspace spanned by uniform superpositions of vertices in each column.
- On this subspace, the graph looks like a line with a defect in the middle. The walk moves from one end to the other in $O(\log N)$ time.
- But any classical algorithm gets stuck in the middle.

# Quantum walk on the glued trees graph

Using Hamiltonian simulation allows the above algorithm to be translated efficiently into the standard quantum circuit model and implies an exponential oracle separation between quantum and classical computation.

# Quantum walk on the glued trees graph

Using Hamiltonian simulation allows the above algorithm to be translated efficiently into the standard quantum circuit model and implies an exponential oracle separation between quantum and classical computation.

Other applications of continuous-time quantum walks include:

- Spatial search [Childs and Goldstone 0306054]
- Quadratic speedup for evaluation of AND-OR (game) trees [Farhi et al 0702144] [Childs et al 0702160]

# Simulation of sparse Hamiltonians

More recently, it has been shown that $d$-sparse Hamiltonians can be solved up to error $\epsilon$ with...

- $(d^2(d + \log^* n)\|H\|t)^{1+o(1)}$ uses of $f$, via decomposing $H$ in terms of galaxies [Childs and Kothari 1003.3683].

- $O(\|H\|t/\sqrt{\epsilon} + d\|H\|_{\max})$ uses of $f$, where $\|H\|_{\max} = \max_{i,j} |H_{ij}|$, via quantum walks [Berry and Childs 0910.4157].

# Application: "Solving" linear equations

A basic task in mathematics and engineering:

**Solving linear equations**

Given access to a $d$-sparse $N \times N$ matrix $A$, and $b \in \mathbb{R}^N$, output $x$ such that $Ax = b$.

# Application: "Solving" linear equations

A basic task in mathematics and engineering:

**Solving linear equations**

Given access to a $d$-sparse $N \times N$ matrix $A$, and $b \in \mathbb{R}^N$, output $x$ such that $Ax = b$.

One "quantum" way of thinking about the problem:

**"Solving" linear equations**

Given the ability to produce the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$, and access to $A$ as above, produce the state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

# Application: "Solving" linear equations

A basic task in mathematics and engineering:

## Solving linear equations

Given access to a $d$-sparse $N \times N$ matrix $A$, and $b \in \mathbb{R}^N$, output $x$ such that $Ax = b$.

One "quantum" way of thinking about the problem:

## "Solving" linear equations

Given the ability to produce the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$, and access to $A$ as above, produce the state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

Theorem: If $A$ has condition number $\kappa$ $(= \|A^{-1}\| \|A\|)$, $|x\rangle$ can be approximately produced in time poly$(\log N, d, \kappa)$ [Harrow et al 0811.3171].

# Application: "Solving" linear equations

A basic task in mathematics and engineering:

## Solving linear equations

Given access to a $d$-sparse $N \times N$ matrix $A$, and $b \in \mathbb{R}^N$, output $x$ such that $Ax = b$.

One "quantum" way of thinking about the problem:

## "Solving" linear equations

Given the ability to produce the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$, and access to $A$ as above, produce the state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

Theorem: If $A$ has condition number $\kappa$ $(= \|A^{-1}\| \|A\|)$, $|x\rangle$ can be approximately produced in time $\text{poly}(\log N, d, \kappa)$ [Harrow et al 0811.3171].

- Later improved to time $O(\kappa \log^3 \kappa \, \text{poly}(d) \log N)$ [Ambainis 1010.4458].

# The algorithm (sketch)

Assume that $A$ is Hermitian and has all eigenvalues $\lambda_i$ in the range $1/\kappa \leqslant \lambda_i \leqslant 1$. Some initial observations:

1. We can write $|b\rangle$ in the eigenbasis of $A$: $|b\rangle = \sum_i c_i |v_i\rangle$.

# The algorithm (sketch)

Assume that $A$ is Hermitian and has all eigenvalues $\lambda_i$ in the range $1/\kappa \leqslant \lambda_i \leqslant 1$. Some initial observations:

1. We can write $|b\rangle$ in the eigenbasis of $A$: $|b\rangle = \sum_i c_i |v_i\rangle$.

2. Observe that $|x\rangle = \sum_i c_i \lambda_i^{-1} |v_i\rangle$.

# The algorithm (sketch)

Assume that $A$ is Hermitian and has all eigenvalues $\lambda_i$ in the range $1/\kappa \leqslant \lambda_i \leqslant 1$. Some initial observations:

1. We can write $|b\rangle$ in the eigenbasis of $A$: $|b\rangle = \sum_i c_i |v_i\rangle$.
2. Observe that $|x\rangle = \sum_i c_i \lambda_i^{-1} |v_i\rangle$.
3. Also observe that, using Hamiltonian simulation, we can approximate the operator $e^{-iAt}$ for arbitrary $t$ in time $\mathrm{poly}(\log N, d, t)$.

# The algorithm (sketch)

1. Prepare the state $|b\rangle = \sum_i c_i |v_i\rangle$.

# The algorithm (sketch)

1. Prepare the state $|b\rangle = \sum_i c_i |v_i\rangle$.
2. Apply phase estimation to the operator $e^{-iAt}$ to produce

$$\left|b'\right\rangle = \sum_i c_i |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle.$$

# The algorithm (sketch)

1. Prepare the state $|b\rangle = \sum_i c_i |v_i\rangle$.
2. Apply phase estimation to the operator $e^{-iAt}$ to produce

$$|b'\rangle = \sum_i c_i |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle.$$

3. Add an ancilla qubit and produce

$$|b''\rangle = \sum_i c_i |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle \left( \frac{1}{\kappa \widetilde{\lambda_i}} |1\rangle + \sqrt{1 - \frac{1}{\kappa^2 \widetilde{\lambda_i}^2}} |0\rangle \right).$$

# The algorithm (sketch)

1. Prepare the state $|b\rangle = \sum_i c_i |v_i\rangle$.
2. Apply phase estimation to the operator $e^{-iAt}$ to produce

$$|b'\rangle = \sum_i c_i |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle.$$

3. Add an ancilla qubit and produce

$$|b''\rangle = \sum_i c_i |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle \left( \frac{1}{\kappa \widetilde{\lambda_i}} |1\rangle + \sqrt{1 - \frac{1}{\kappa^2 \widetilde{\lambda_i}^2}} |0\rangle \right).$$

4. Measure the ancilla; if we get 1, we have produced a state proportional to

$$\sum_i c_i \widetilde{\lambda_i}^{-1} |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle.$$

# The algorithm (sketch)

1. Prepare the state $|b\rangle = \sum_i c_i |v_i\rangle$.
2. Apply phase estimation to the operator $e^{-iAt}$ to produce

$$|b'\rangle = \sum_i c_i |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle.$$

3. Add an ancilla qubit and produce

$$|b''\rangle = \sum_i c_i |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle \left( \frac{1}{\kappa \widetilde{\lambda_i}} |1\rangle + \sqrt{1 - \frac{1}{\kappa^2 \widetilde{\lambda_i}^2}} |0\rangle \right).$$

4. Measure the ancilla; if we get 1, we have produced a state proportional to

$$\sum_i c_i \widetilde{\lambda_i}^{-1} |v_i\rangle \left|\widetilde{\lambda_i}\right\rangle.$$

5. Perform phase estimation backwards to uncompute $\widetilde{\lambda_i}$ and thus produce a state close to $|x\rangle$.

# Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

# Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.

# Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that $\mathsf{BPP} = \mathsf{BQP}$!

# Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that $\mathsf{BPP} = \mathsf{BQP}$!

Further notes:

- The dependence on $\kappa$ can be improved using amplitude amplification, but...

# Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that $\mathsf{BPP} = \mathsf{BQP}$!

Further notes:

- The dependence on $\kappa$ can be improved using amplitude amplification, but...
- Any quantum algorithm needs time $\Omega(\kappa^{1-o(1)})$ unless $\mathsf{BQP} = \mathsf{PSPACE}$.

# Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that BPP = BQP!

Further notes:

- The dependence on $\kappa$ can be improved using amplitude amplification, but...
- Any quantum algorithm needs time $\Omega(\kappa^{1-o(1)})$ unless BQP = PSPACE.
- More recent applications of this algorithm include:
  - "Solving" differential equations [Leyton and Osborne 0812.4423] [Berry 1010.2745]
  - Data fitting [Wiebe et al 1204.5242]

# Grover's algorithm

**Problem**

Given access to a function $f : [n] \to \{0, 1\}$ such that $f(i) = 1$ if and only if $i \in M$, where $|M| = \epsilon n$, output $j \in M$.

# Grover's algorithm

**Problem**

Given access to a function $f : [n] \to \{0, 1\}$ such that $f(i) = 1$ if and only if $i \in M$, where $|M| = \epsilon n$, output $j \in M$.

- Write $S_M = \text{span}\{|i\rangle : i \in M\}$, $|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle$.

# Grover's algorithm

**Problem**

Given access to a function $f : [n] \to \{0,1\}$ such that $f(i) = 1$ if and only if $i \in M$, where $|M| = \epsilon n$, output $j \in M$.

- Write $S_M = \text{span}\{|i\rangle : i \in M\}$, $|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle$.
- Grover's algorithm: starting with $|\psi\rangle$, alternately reflect about $S_M^\perp$ and $|\psi\rangle$:

$$\text{ref}(|\psi\rangle)\,\text{ref}(S_M^\perp)\,\text{ref}(|\psi\rangle)\,\text{ref}(S_M^\perp)\ldots\text{ref}(|\psi\rangle)\,\text{ref}(S_M^\perp),$$

where for a subspace $S$ (and $P$ the projector onto $S$)

$$\text{ref}(S) = 2P - I.$$

# Grover's algorithm

**Problem**

Given access to a function $f : [n] \to \{0,1\}$ such that $f(i) = 1$ if and only if $i \in M$, where $|M| = \epsilon n$, output $j \in M$.

- Write $S_M = \text{span}\{|i\rangle : i \in M\}$, $|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle$.
- Grover's algorithm: starting with $|\psi\rangle$, alternately reflect about $S_M^\perp$ and $|\psi\rangle$:

$$\text{ref}(|\psi\rangle)\,\text{ref}(S_M^\perp)\,\text{ref}(|\psi\rangle)\,\text{ref}(S_M^\perp)\ldots\text{ref}(|\psi\rangle)\,\text{ref}(S_M^\perp),$$

  where for a subspace $S$ (and $P$ the projector onto $S$)

$$\text{ref}(S) = 2P - I.$$

- $\text{ref}(S_M^\perp)$ is the operator mapping $|i\rangle \mapsto (-1)^{f(i)}|i\rangle$, which can be implemented with one query to $f$.

# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.

# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.

# Grover's algorithm

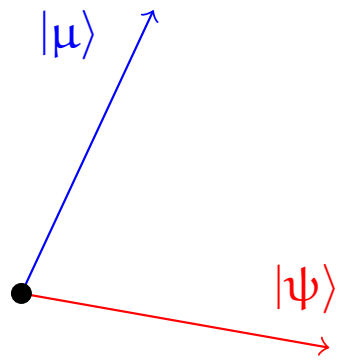
- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.
- Claim 3: the composition of two reflections is a rotation: $\mathrm{ref}(|\psi\rangle)\,\mathrm{ref}(|\mu^{\perp}\rangle)$ rotates by angle $2\phi$ from $|\psi\rangle$ to $|\mu\rangle$, where

$$\sin\phi = \langle\psi|\mu\rangle = \sqrt{\epsilon}.$$

# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.
- Claim 3: the composition of two reflections is a rotation: $\mathrm{ref}(|\psi\rangle)\,\mathrm{ref}(|\mu^{\perp}\rangle)$ rotates by angle $2\phi$ from $|\psi\rangle$ to $|\mu\rangle$, where

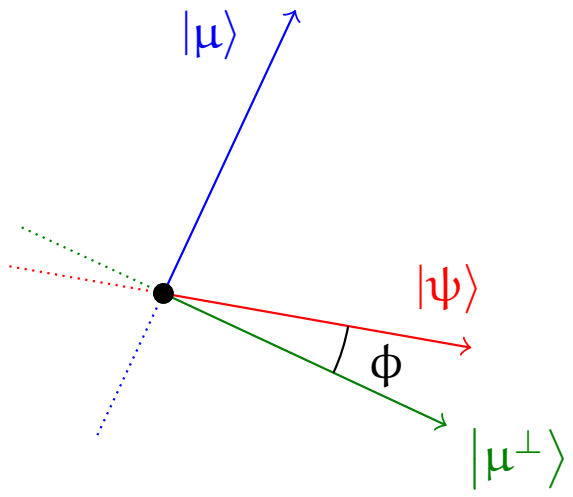
$$\sin\phi = \langle\psi|\mu\rangle = \sqrt{\epsilon}.$$



# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.
- Claim 3: the composition of two reflections is a rotation: $\mathrm{ref}(|\psi\rangle)\,\mathrm{ref}(|\mu^{\perp}\rangle)$ rotates by angle $2\phi$ from $|\psi\rangle$ to $|\mu\rangle$, where

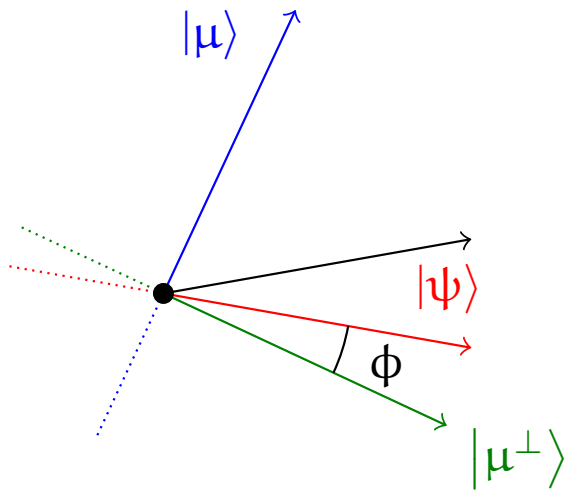
$$\sin\phi = \langle\psi|\mu\rangle = \sqrt{\epsilon}.$$



# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.
- Claim 3: the composition of two reflections is a rotation: $\mathrm{ref}(|\psi\rangle)\,\mathrm{ref}(|\mu^{\perp}\rangle)$ rotates by angle $2\phi$ from $|\psi\rangle$ to $|\mu\rangle$, where

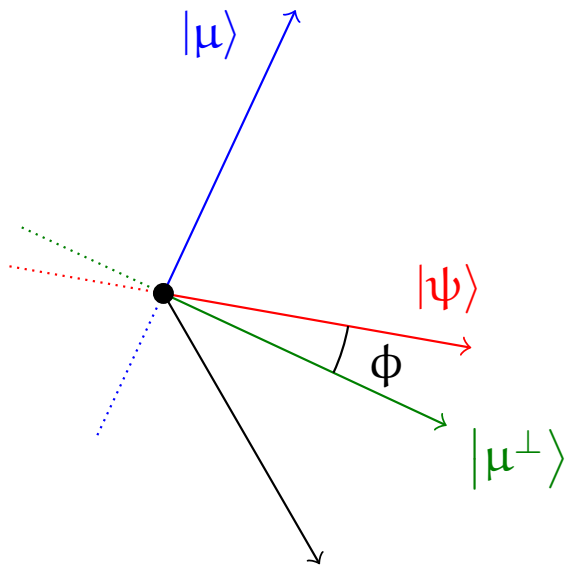
$$\sin\phi = \langle\psi|\mu\rangle = \sqrt{\epsilon}.$$



# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.
- Claim 3: the composition of two reflections is a rotation: $\mathrm{ref}(|\psi\rangle)\,\mathrm{ref}(|\mu^{\perp}\rangle)$ rotates by angle $2\phi$ from $|\psi\rangle$ to $|\mu\rangle$, where

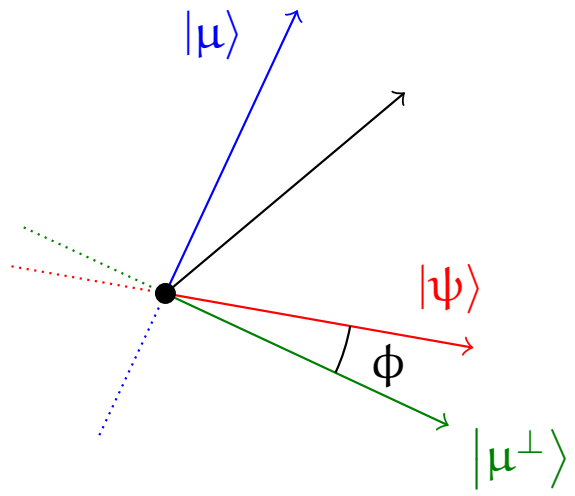
$$\sin\phi = \langle\psi|\mu\rangle = \sqrt{\epsilon}.$$



# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.
- Claim 3: the composition of two reflections is a rotation: $\mathrm{ref}(|\psi\rangle)\,\mathrm{ref}(|\mu^{\perp}\rangle)$ rotates by angle $2\phi$ from $|\psi\rangle$ to $|\mu\rangle$, where

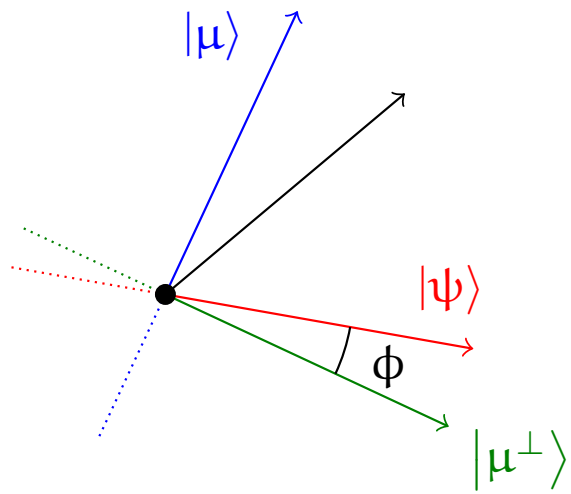
$$\sin\phi = \langle\psi|\mu\rangle = \sqrt{\epsilon}.$$



# Grover's algorithm

- Claim 1: the state of the system remains in the 2D subspace spanned by $|\psi\rangle$ and $|\mu\rangle = \sum_{i \in M} |i\rangle$.
- Claim 2: within this subspace, $\mathrm{ref}(S_M^{\perp}) = \mathrm{ref}(|\mu^{\perp}\rangle)$.
- Claim 3: the composition of two reflections is a rotation: $\mathrm{ref}(|\psi\rangle)\,\mathrm{ref}(|\mu^{\perp}\rangle)$ rotates by angle $2\phi$ from $|\psi\rangle$ to $|\mu\rangle$, where

$$\sin\phi = \langle\psi|\mu\rangle = \sqrt{\epsilon}.$$




- Thus the algorithm uses $O(1/\sqrt{\epsilon})$ queries to reach $|\mu\rangle$.

# Element distinctness

**Problem**

Given a list of $n$ integers, are they all distinct?

# Element distinctness

**Problem**

Given a list of $n$ integers, are they all distinct?

- Classical complexity: $\Theta(n)$ queries.

# Element distinctness

**Problem**

Given a list of $n$ integers, are they all distinct?

- Classical complexity: $\Theta(n)$ queries.
- Try using Grover's algorithm on the set of all pairs: $O(\sqrt{n^2}) = O(n)$.

# Element distinctness

**Problem**

Given a list of $n$ integers, are they all distinct?

- Classical complexity: $\Theta(n)$ queries.
- Try using Grover's algorithm on the set of all pairs: $O(\sqrt{n^2}) = O(n)$.

**Theorem [Ambainis 0311001]**

Element Distinctness can be solved using $O(n^{2/3})$ queries.

# Element distinctness

**Problem**

Given a list of $n$ integers, are they all distinct?

- Classical complexity: $\Theta(n)$ queries.
- Try using Grover's algorithm on the set of all pairs: $O(\sqrt{n^2}) = O(n)$.

**Theorem** [Ambainis 0311001]

Element Distinctness can be solved using $O(n^{2/3})$ queries.

- Time complexity is the same up to polylogarithmic factors.
- Generalisation to finding a $k$-subset of $[m]^n$ satisfying any property: uses $O(n^{k/(k+1)})$ queries.
- This bound is tight [Aaronson and Shi 0111102, 0112086] [Belovs and Špalek 1204.5074, 1206.6528]

# Element distinctness by local search

Idea: try to solve this problem by search on the Johnson graph.

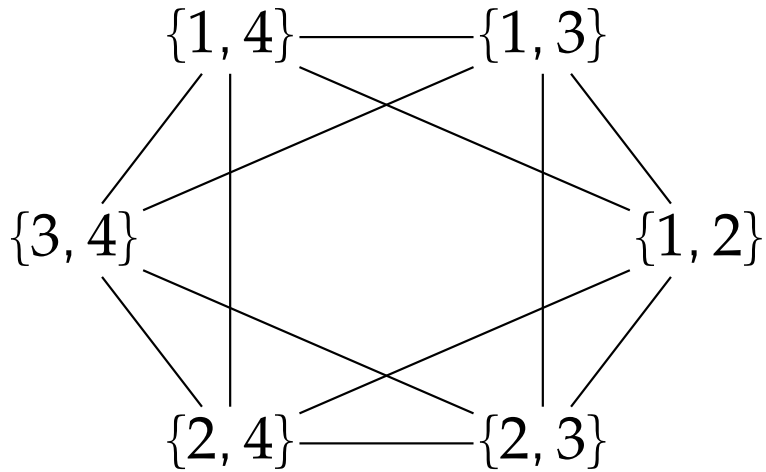
- This is the graph whose vertices are all $r$-subsets of $[n]$.
- Vertices $S$, $T$ are connected if $|S \cup T| = r - 1$.

# Element distinctness by local search

Idea: try to solve this problem by search on the Johnson graph.

- This is the graph whose vertices are all $r$-subsets of $[n]$.
- Vertices $S$, $T$ are connected if $|S \cup T| = r - 1$.

e.g. $n = 4$, $r = 2$:

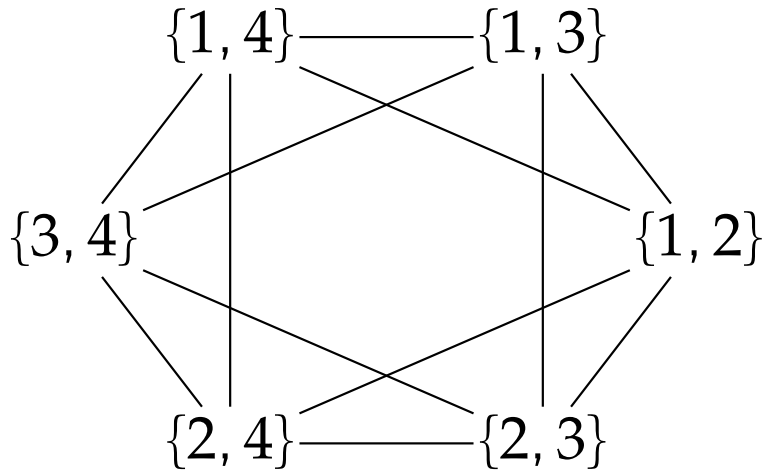


# Element distinctness by local search

Idea: try to solve this problem by search on the Johnson graph.

- This is the graph whose vertices are all $r$-subsets of $[n]$.
- Vertices $S$, $T$ are connected if $|S \cup T| = r - 1$.

e.g. $n = 4$, $r = 2$:



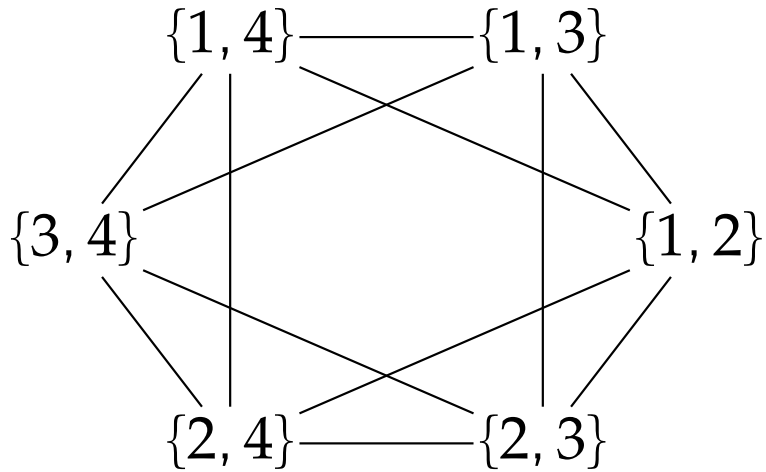

- Fraction of marked vertices: at least $\binom{n-2}{r-2}/\binom{n}{r} = \Theta(r^2/n^2)$.

# Element distinctness by local search

Idea: try to solve this problem by search on the Johnson graph.

- This is the graph whose vertices are all $r$-subsets of $[n]$.
- Vertices $S$, $T$ are connected if $|S \cup T| = r - 1$.

e.g. $n = 4$, $r = 2$:




- Fraction of marked vertices: at least $\binom{n-2}{r-2}/\binom{n}{r} = \Theta(r^2/n^2)$.
- Try Grover again: $O(\sqrt{n^2/r^2} \times r) = O(n)$.

# Search by quantum walk

Write

$$|\psi\rangle = \sum_{S,|S|=r} |S\rangle, \quad |\mu\rangle = \sum_{S,|S|=r,S \text{ good}} |S\rangle,$$

where $S$ is good if it contains duplicate elements.

# Search by quantum walk

Write

$$|\psi\rangle = \sum_{S,|S|=r} |S\rangle, \quad |\mu\rangle = \sum_{S,|S|=r,S \text{ good}} |S\rangle,$$

where $S$ is good if it contains duplicate elements.

Grover's algorithm alternates between:

- Reflections about $|\psi\rangle$ (free)
- Reflections about $|\mu^{\perp}\rangle$ (costly).

Let's think about this complexity differently.

# Search by quantum walk

Write

$$|\psi\rangle = \sum_{S,|S|=r} |S\rangle, \quad |\mu\rangle = \sum_{S,|S|=r,S \text{ good}} |S\rangle,$$

where $S$ is good if it contains duplicate elements.

Grover's algorithm alternates between:

- Reflections about $|\psi\rangle$ (free)
- Reflections about $|\mu^{\perp}\rangle$ (costly).

Let's think about this complexity differently.

Associate each vertex $S$ of the graph with some data $d(S)$, such that given the state $|S\rangle|d(S)\rangle$, the checking step is free (i.e. requires no further queries).

- For element distinctness, $d(S)$ is the subset of the input elements corresponding to $S$.

# Search by quantum walk

Write

$$|\psi_d\rangle = \sum_{S,|S|=r} |S\rangle|d(S)\rangle, \quad |\mu_d\rangle = \sum_{S,|S|=r,S \text{ good}} |S\rangle|d(S)\rangle.$$

We would like to rotate between $|\psi_d\rangle$ and $|\mu_d\rangle$.

# Search by quantum walk

Write

$$|\psi_d\rangle = \sum_{S,|S|=r} |S\rangle|d(S)\rangle, \quad |\mu_d\rangle = \sum_{S,|S|=r,S \text{ good}} |S\rangle|d(S)\rangle.$$

We would like to rotate between $|\psi_d\rangle$ and $|\mu_d\rangle$.

We can construct $|\psi_d\rangle$ using $r$ queries.

- So we can reflect about $|\psi_d\rangle$ using $2r$ queries.

# Search by quantum walk

Write

$$|\psi_d\rangle = \sum_{S,|S|=r} |S\rangle|d(S)\rangle, \quad |\mu_d\rangle = \sum_{S,|S|=r,S \text{ good}} |S\rangle|d(S)\rangle.$$

We would like to rotate between $|\psi_d\rangle$ and $|\mu_d\rangle$.

We can construct $|\psi_d\rangle$ using $r$ queries.

- So we can reflect about $|\psi_d\rangle$ using $2r$ queries.

As in Grover's algorithm, reflecting about $\left|\mu_d^{\perp}\right\rangle$ can be done by checking whether $d(S)$ contains any duplicates.

- So this step does not require any queries.

# Search by quantum walk

Write

$$|\psi_d\rangle = \sum_{S,|S|=r} |S\rangle|d(S)\rangle, \quad |\mu_d\rangle = \sum_{S,|S|=r,S \text{ good}} |S\rangle|d(S)\rangle.$$

We would like to rotate between $|\psi_d\rangle$ and $|\mu_d\rangle$.

We can construct $|\psi_d\rangle$ using $r$ queries.

- So we can reflect about $|\psi_d\rangle$ using $2r$ queries.

As in Grover's algorithm, reflecting about $\left|\mu_d^{\perp}\right\rangle$ can be done by checking whether $d(S)$ contains any duplicates.

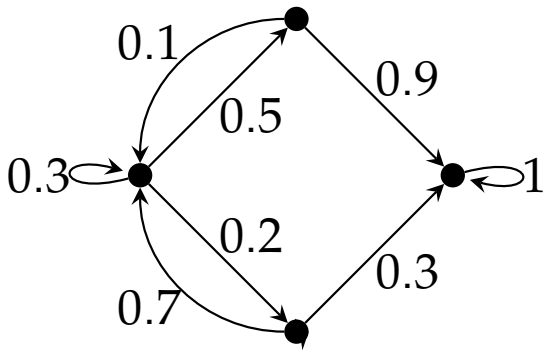
- So this step does not require any queries.

Can we speed up the first step?

# Markov chains

- A Markov chain $M = (p_{ij})$ is a stochastic linear map $\mathbb{R}^n \to \mathbb{R}^n$.
- Equivalently: a random walk on a directed graph.

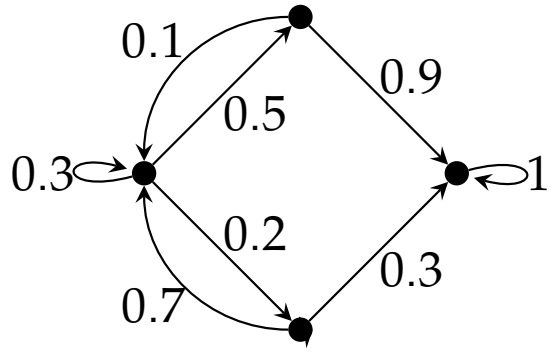
$$\begin{pmatrix} 0.3 & 0.5 & 0.2 & 0 \\ 0.1 & 0 & 0 & 0.9 \\ 0.7 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



# Markov chains

- A Markov chain $M = (p_{ij})$ is a stochastic linear map $\mathbb{R}^n \to \mathbb{R}^n$.
- Equivalently: a random walk on a directed graph.

$$\begin{pmatrix} 0.3 & 0.5 & 0.2 & 0 \\ 0.1 & 0 & 0 & 0.9 \\ 0.7 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$




$M$ is said to be:

- irreducible if any vertex can be reached from any other vertex;
- ergodic if it is irreducible and aperiodic;
- symmetric if $p_{ij} = p_{ji}$.

# Markov chains

Let $M$ have eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \lambda_n$.

- If $M$ is ergodic, it has a unique stationary distribution $\pi$, i.e. an eigenvector with eigenvalue 1.
- If $M$ is symmetric, $\pi$ is the uniform distribution.
- The eigenvalue gap is $\delta = 1 - \max_{i>1} |\lambda_i|$.

# Markov chains

Let $M$ have eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \lambda_n$.

- If $M$ is ergodic, it has a unique stationary distribution $\pi$, i.e. an eigenvector with eigenvalue 1.
- If $M$ is symmetric, $\pi$ is the uniform distribution.
- The eigenvalue gap is $\delta = 1 - \max_{i>1} |\lambda_i|$.

**Classical mixing**

Applying $O(1/\delta)$ steps of $M$ to an arbitrary initial distribution is sufficient to approximately produce the distribution $\pi$.

# Bipartite quantum walks

Given an (ergodic, symmetric) Markov chain $M = (p_{xy})$, define

$$|p_x\rangle = \sum_y \sqrt{p_{xy}}|y\rangle$$

and set

$$X = \mathrm{span}\{|x\rangle|p_x\rangle\}, \;\; Y = \mathrm{span}\{|p_y\rangle|y\rangle\}.$$

# Bipartite quantum walks

Given an (ergodic, symmetric) Markov chain $M = (p_{xy})$, define

$$|p_x\rangle = \sum_y \sqrt{p_{xy}}|y\rangle$$

and set

$$X = \text{span}\{|x\rangle|p_x\rangle\}, \;\; Y = \text{span}\{|p_y\rangle|y\rangle\}.$$

$W = \text{ref}(Y)\,\text{ref}(X)$ is "the" quantum walk corresponding to $M$.

# Bipartite quantum walks

Given an (ergodic, symmetric) Markov chain $M = (p_{xy})$, define

$$|p_x\rangle = \sum_y \sqrt{p_{xy}}|y\rangle$$

and set

$$X = \mathrm{span}\{|x\rangle|p_x\rangle\},\ Y = \mathrm{span}\{|p_y\rangle|y\rangle\}.$$

$W = \mathrm{ref}(Y)\,\mathrm{ref}(X)$ is "the" quantum walk corresponding to $M$.

**Crucial fact [Szegedy 0401053]**

The unique eigenvector of $W$ with eigenvalue 1 is

$$|\pi\rangle = \sum_x \sqrt{\pi_x}|x\rangle|p_x\rangle.$$

For each singular value $\cos(\theta)$ of $M$, $\theta \in [0, \pi/2]$, $W$ has corresponding eigenvalues $e^{\pm 2i\theta}$; all other eigenvalues are $-1$.

# Bipartite quantum walks

- Now $|1 - e^{\pm 2i\theta}| = \sqrt{2(1-\cos(2\theta))} \geqslant 2\sqrt{1-\cos\theta} \geqslant 2\sqrt{\delta}$.

# Bipartite quantum walks

- Now $|1 - e^{\pm 2i\theta}| = \sqrt{2(1-\cos(2\theta))} \geqslant 2\sqrt{1-\cos\theta} \geqslant 2\sqrt{\delta}$.
- So, using phase estimation, we can distinguish between the stationary state $|\pi\rangle$ and any state orthogonal to $|\pi\rangle$ using $O(1/\sqrt{\delta})$ steps of $W$.

# Bipartite quantum walks

- Now $|1 - e^{\pm 2i\theta}| = \sqrt{2(1-\cos(2\theta))} \geqslant 2\sqrt{1-\cos\theta} \geqslant 2\sqrt{\delta}$.
- So, using phase estimation, we can distinguish between the stationary state $|\pi\rangle$ and any state orthogonal to $|\pi\rangle$ using $O(1/\sqrt{\delta})$ steps of $W$.
- This implies that we can approximately implement the operation $\mathrm{ref}(|\pi_d\rangle)$ using $O(1/\sqrt{\delta})$ queries!

# Bipartite quantum walks

- Now $|1 - e^{\pm 2i\theta}| = \sqrt{2(1-\cos(2\theta))} \geqslant 2\sqrt{1-\cos\theta} \geqslant 2\sqrt{\delta}$.
- So, using phase estimation, we can distinguish between the stationary state $|\pi\rangle$ and any state orthogonal to $|\pi\rangle$ using $O(1/\sqrt{\delta})$ steps of $W$.
- This implies that we can approximately implement the operation $\mathrm{ref}(|\pi_d\rangle)$ using $O(1/\sqrt{\delta})$ queries!

Back to element distinctness:

- For the Johnson graph, eigenvalue gap $\delta = \Theta(1/r)$.

# Bipartite quantum walks

- Now $|1 - e^{\pm 2i\theta}| = \sqrt{2(1-\cos(2\theta))} \geqslant 2\sqrt{1-\cos\theta} \geqslant 2\sqrt{\delta}$.
- So, using phase estimation, we can distinguish between the stationary state $|\pi\rangle$ and any state orthogonal to $|\pi\rangle$ using $O(1/\sqrt{\delta})$ steps of $W$.
- This implies that we can approximately implement the operation $\mathrm{ref}(|\pi_d\rangle)$ using $O(1/\sqrt{\delta})$ queries!

Back to element distinctness:

- For the Johnson graph, eigenvalue gap $\delta = \Theta(1/r)$.
- Thus the overall complexity of the quantum walk algorithm for element distinctness is

$$O(r + \sqrt{n^2/r^2} \times \sqrt{r}) = O(r + n/\sqrt{r}).$$

# Bipartite quantum walks

- Now $|1 - e^{\pm 2i\theta}| = \sqrt{2(1-\cos(2\theta))} \geqslant 2\sqrt{1-\cos\theta} \geqslant 2\sqrt{\delta}$.
- So, using phase estimation, we can distinguish between the stationary state $|\pi\rangle$ and any state orthogonal to $|\pi\rangle$ using $O(1/\sqrt{\delta})$ steps of $W$.
- This implies that we can approximately implement the operation $\mathrm{ref}(|\pi_d\rangle)$ using $O(1/\sqrt{\delta})$ queries!

Back to element distinctness:

- For the Johnson graph, eigenvalue gap $\delta = \Theta(1/r)$.
- Thus the overall complexity of the quantum walk algorithm for element distinctness is

$$O(r + \sqrt{n^2/r^2} \times \sqrt{r}) = O(r + n/\sqrt{r}).$$

- Taking $r = O(n^{2/3})$ we get a complexity of $O(n^{2/3})$.

# General search problems

More generally, for any search problem of this form we have:

**Theorem** [Magniez et al 0608026]

A marked element can be found with cost $O(S+\frac{1}{\sqrt{\epsilon}}(\frac{1}{\sqrt{\delta}}U+C))$.

where

- $S$ is the setup cost to construct $\sum_x \sqrt{\pi_x}|x\rangle|d(x)\rangle$;
- $U$ is the update cost to perform one step of the walk $W$;
- $C$ is the checking cost to determine if $x$ is marked.

# General search problems

More generally, for any search problem of this form we have:

**Theorem** [Magniez et al 0608026]

A marked element can be found with cost $O(S+\frac{1}{\sqrt{\epsilon}}(\frac{1}{\sqrt{\delta}}U+C))$.

where

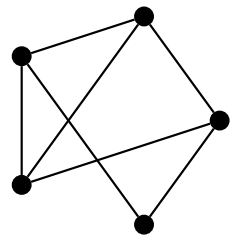
- $S$ is the setup cost to construct $\sum_x \sqrt{\pi_x}|x\rangle|d(x)\rangle$;
- $U$ is the update cost to perform one step of the walk $W$;
- $C$ is the checking cost to determine if $x$ is marked.

NB: prior important quantum walk algorithms by [Szegedy 0401053], [Ambainis 0311001] are subtly different...

# Some examples

The above framework lends itself to many different search problems, such as:

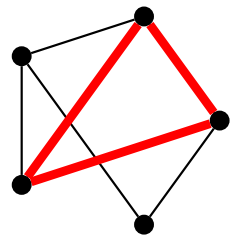
- Finding a triangle in a graph: $O(n^{1.3})$ queries, vs. classical $O(n^2)$ [Magniez et al 0310134]



# Some examples

The above framework lends itself to many different search problems, such as:

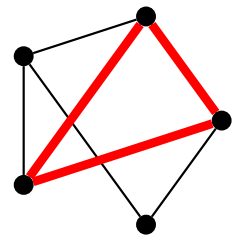
- Finding a triangle in a graph: $O(n^{1.3})$ queries, vs. classical $O(n^2)$ [Magniez et al 0310134]



# Some examples

The above framework lends itself to many different search problems, such as:

- Finding a triangle in a graph: $O(n^{1.3})$ queries, vs. classical $O(n^2)$ [Magniez et al 0310134]



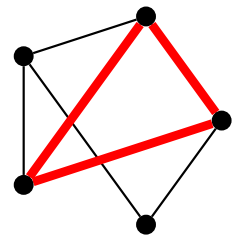
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek 0409035]

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 3 \\ -2 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 5 & -2 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \stackrel{?}{=} \begin{pmatrix} -1 & 4 & -3 \\ 1 & 5 & 4 \\ 1 & -9 & 5 \end{pmatrix}$$

# Some examples

The above framework lends itself to many different search problems, such as:

- Finding a triangle in a graph: $O(n^{1.3})$ queries, vs. classical $O(n^2)$ [Magniez et al 0310134]



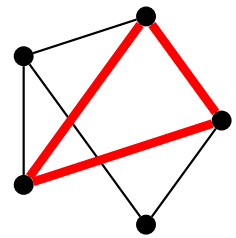
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek 0409035]

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 3 \\ -2 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 5 & -2 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \stackrel{?}{=} \begin{pmatrix} -1 & 4 & -3 \\ 1 & 5 & 4 \\ 1 & -9 & 5 \end{pmatrix}$$

# Some examples

The above framework lends itself to many different search problems, such as:

- Finding a triangle in a graph: $O(n^{1.3})$ queries, vs. classical $O(n^2)$ [Magniez et al 0310134]



- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek 0409035]

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 3 \\ -2 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 5 & -2 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \stackrel{?}{=} \begin{pmatrix} -1 & 4 & -3 \\ 1 & 5 & 4 \\ 1 & -9 & 5 \end{pmatrix}$$

- Testing group commutativity: $O(n^{2/3} \log n)$ queries, vs. classical $O(n)$ [Magniez and Nayak 0506265]

# Learning graphs

Learning graphs can be seen as a far-reaching way of generalising the element distinctness algorithm.

# Learning graphs

Learning graphs can be seen as a far-reaching way of generalising the element distinctness algorithm.

**Definition [Belovs 1105.4024]**

A learning graph $G$ is a directed acyclic graph with vertices labelled by subsets of $[n]$. It has edges of the form $S \to S \cup \{j\}$ for some $j \in [n] \backslash S$. Each edge $e$ has a weight $w(e) \geqslant 0$.

# Learning graphs

Learning graphs can be seen as a far-reaching way of generalising the element distinctness algorithm.

**Definition [Belovs 1105.4024]**

A learning graph $G$ is a directed acyclic graph with vertices labelled by subsets of $[n]$. It has edges of the form $S \to S \cup \{j\}$ for some $j \in [n] \backslash S$. Each edge $e$ has a weight $w(e) \geqslant 0$.

- Let $f : [m]^n \to \{0, 1\}$ be a function we want to compute.

# Learning graphs

Learning graphs can be seen as a far-reaching way of generalising the element distinctness algorithm.

**Definition [Belovs 1105.4024]**

A learning graph $G$ is a directed acyclic graph with vertices labelled by subsets of $[n]$. It has edges of the form $S \to S \cup \{j\}$ for some $j \in [n] \backslash S$. Each edge $e$ has a weight $w(e) \geqslant 0$.

- Let $f : [m]^n \to \{0, 1\}$ be a function we want to compute.
- Each $x$ such that $f(x) = 1$ is associated with a flow on $G$, where a flow is a function $p_x(e)$ assigning intensities to edges such that:

# Learning graphs

Learning graphs can be seen as a far-reaching way of generalising the element distinctness algorithm.

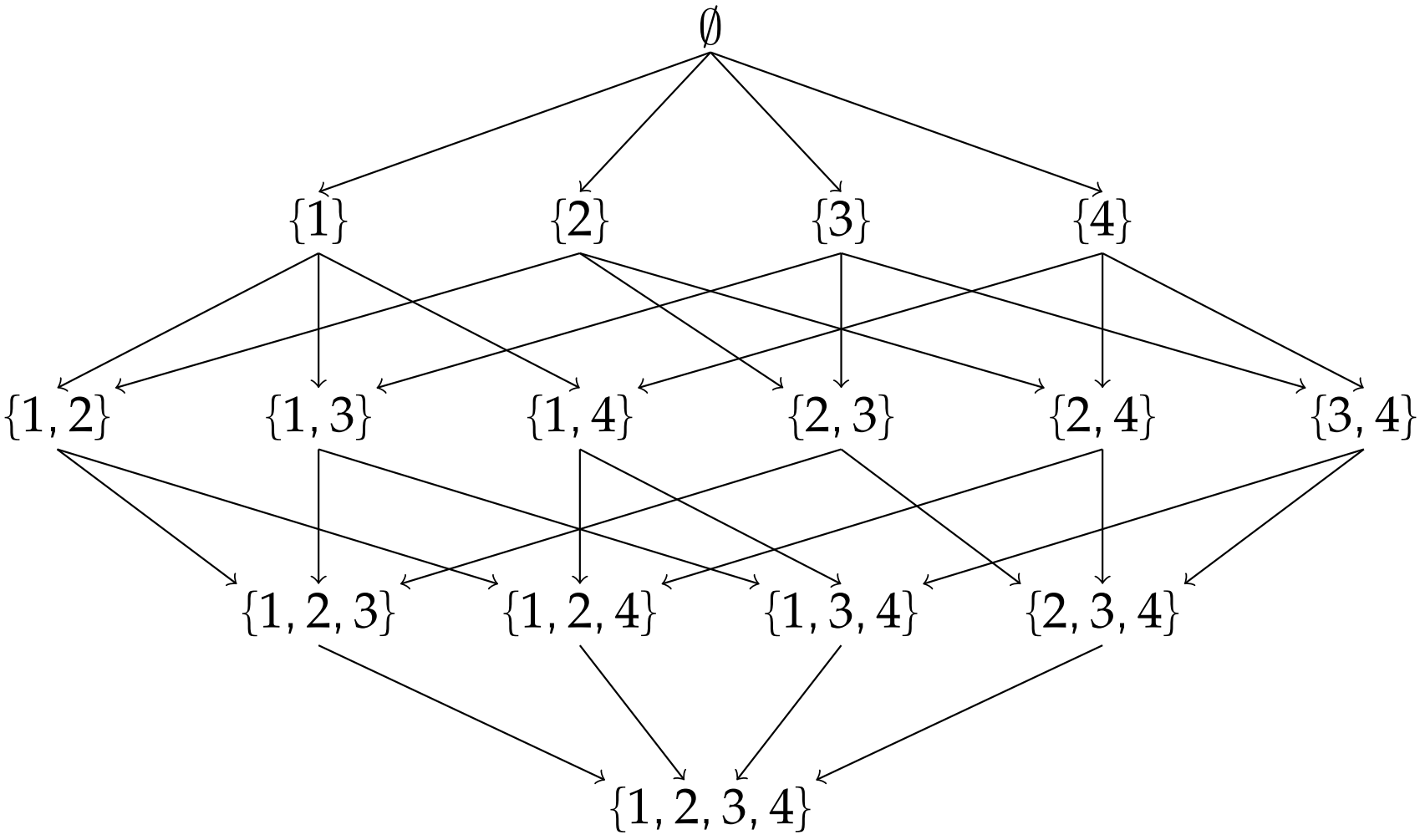
**Definition [Belovs 1105.4024]**

A learning graph $G$ is a directed acyclic graph with vertices labelled by subsets of $[n]$. It has edges of the form $S \to S \cup \{j\}$ for some $j \in [n] \backslash S$. Each edge $e$ has a weight $w(e) \geqslant 0$.

- Let $f : [m]^n \to \{0, 1\}$ be a function we want to compute.
- Each $x$ such that $f(x) = 1$ is associated with a flow on $G$, where a flow is a function $p_x(e)$ assigning intensities to edges such that:
  - The source of the flow is $\emptyset$. The sum of the intensities of its outgoing edges equals 1.
  - Every vertex that contains a 1-certificate of $f$ is a sink.
  - For all other vertices, the sum of the intensities of the outgoing edges equals the sum of the intensities of the incoming edges.

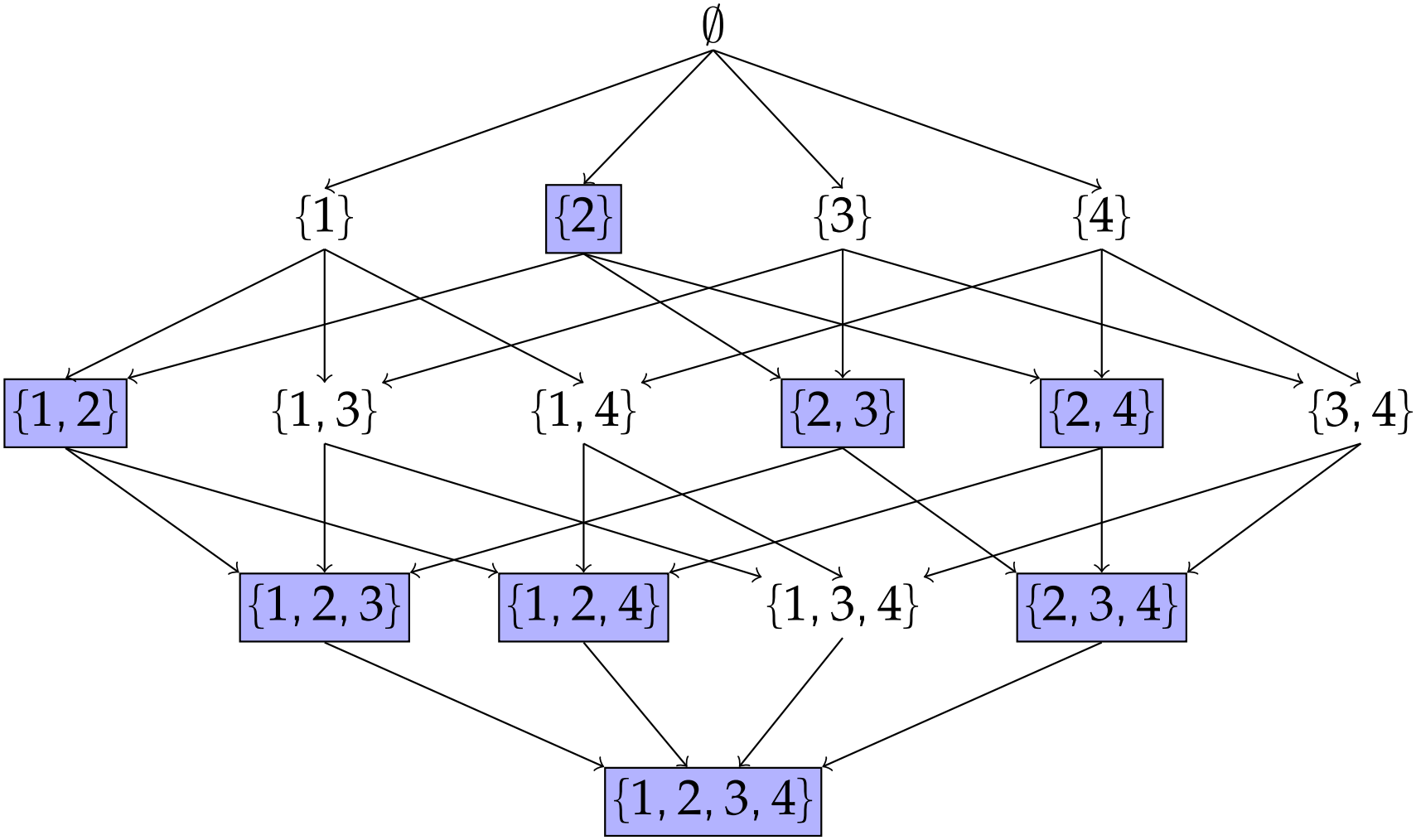
# A simple learning graph

A learning graph for any function $f : [m]^4 \to \{0, 1\}$ looks like this (weights not shown):



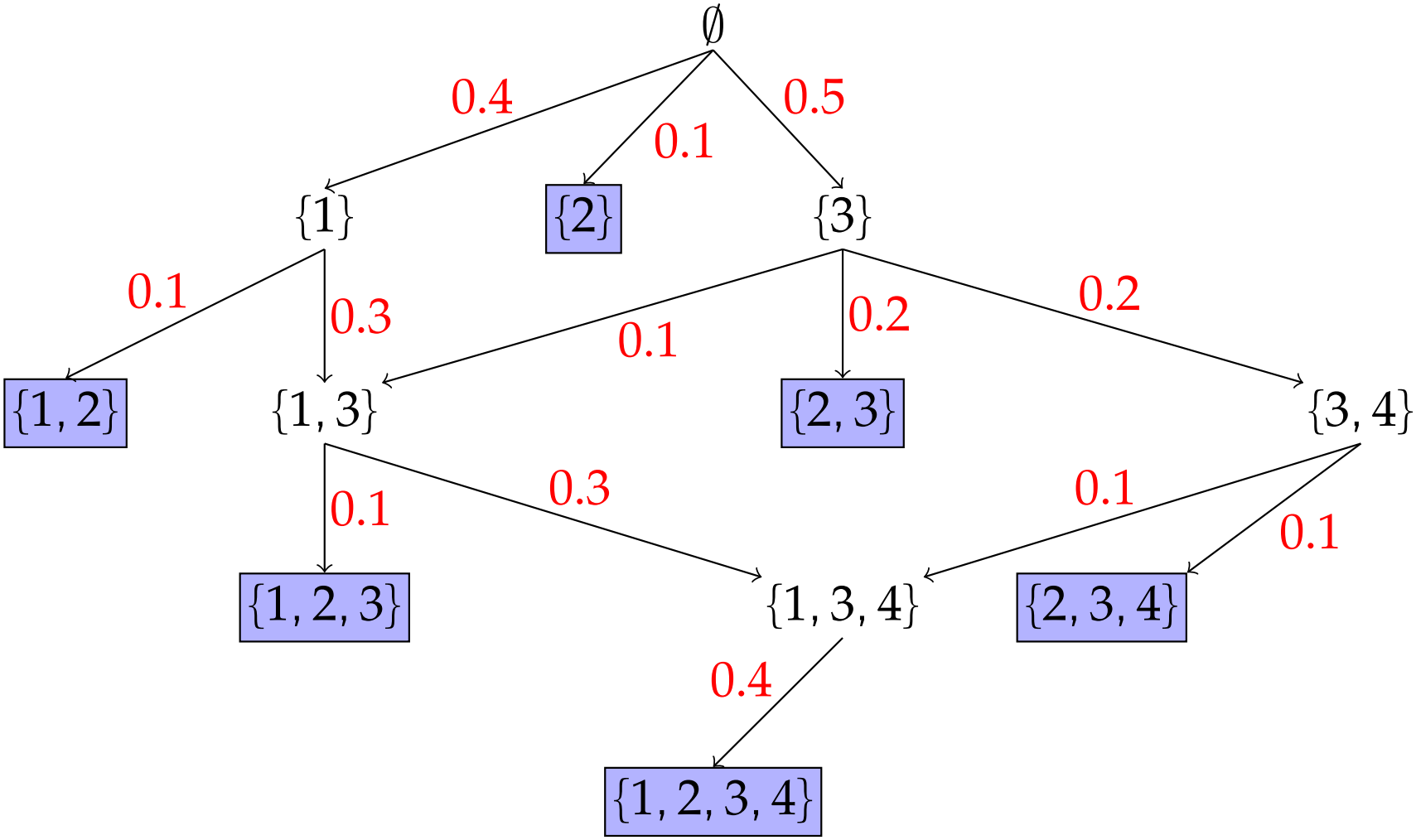
# A simple learning graph

- Let $f$ be the OR function on 4 bits, and $x = 0100$.
- The highlighted vertices contain 1-certificates for $x$.



# A simple learning graph

- Let $f$ be the OR function on 4 bits, and $x = 0100$.
- Thus the following is a valid flow:



# Learning graph complexity

Define the negative and positive complexities of $(G, x)$ as

$$C^0(G) = \sum_{e \in E} w(e), \quad C^1(G, x) = \sum_{e \in E} \frac{p_x(e)^2}{w_e}.$$

Then define the complexity of $G$ as

$$C(G) = \sqrt{C^0(G) \max_x \{C^1(G, x)\}}.$$

# Learning graph complexity

Define the negative and positive complexities of $(G, x)$ as

$$C^0(G) = \sum_{e \in E} w(e), \quad C^1(G, x) = \sum_{e \in E} \frac{p_x(e)^2}{w_e}.$$

Then define the complexity of $G$ as

$$C(G) = \sqrt{C^0(G) \max_x \{C^1(G, x)\}}.$$

**Theorem** **[Belovs 1105.4024] [Belovs and Lee 1108.3022]**

If there is a learning graph for $f$ with complexity $C$, there is a quantum query algorithm using $O(C)$ queries to compute $f$.

# Learning graph complexity

Define the negative and positive complexities of $(G, x)$ as

$$C^0(G) = \sum_{e \in E} w(e), \quad C^1(G, x) = \sum_{e \in E} \frac{p_x(e)^2}{w_e}.$$

Then define the complexity of $G$ as

$$C(G) = \sqrt{C^0(G) \max_x \{C^1(G, x)\}}.$$

**Theorem** **[Belovs 1105.4024] [Belovs and Lee 1108.3022]**

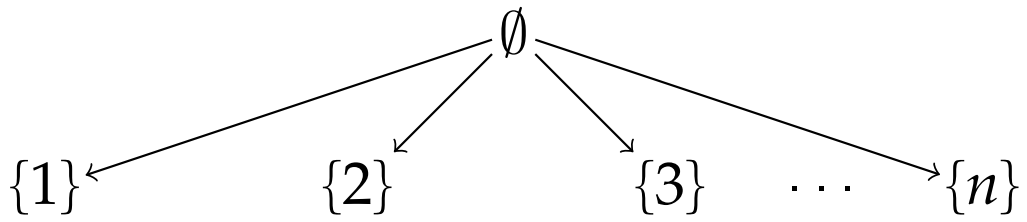
If there is a learning graph for $f$ with complexity $C$, there is a quantum query algorithm using $O(C)$ queries to compute $f$.

Various proofs known:

- Via span programs [Belovs 1105.4024] [Reichardt 1005.1601]
- Via a direct solution to the adversary bound [Belovs and Lee 1108.3022]
- Via quantum walks [Belovs 1302.3143]

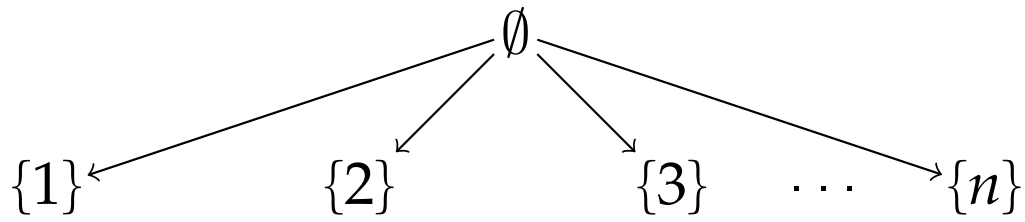
# Example: rederiving Grover's algorithm

Consider the OR function on $n$ bits, $\mathrm{OR}_n(x) = 0 \Leftrightarrow x = 0^n$. We use the following learning graph with weight 1 on each edge shown, and weight 0 on other edges:

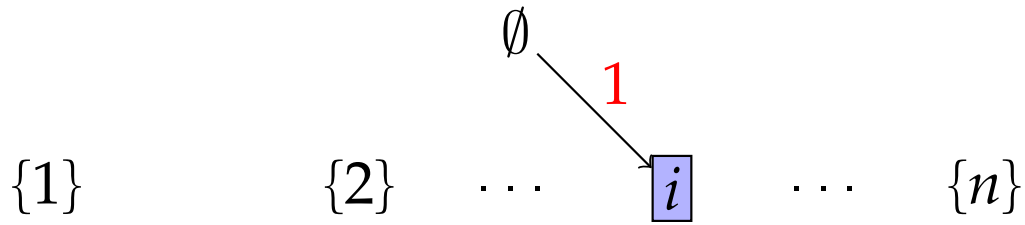


# Example: rederiving Grover's algorithm

Consider the OR function on $n$ bits, $\mathrm{OR}_n(x) = 0 \Leftrightarrow x = 0^n$. We use the following learning graph with weight 1 on each edge shown, and weight 0 on other edges:

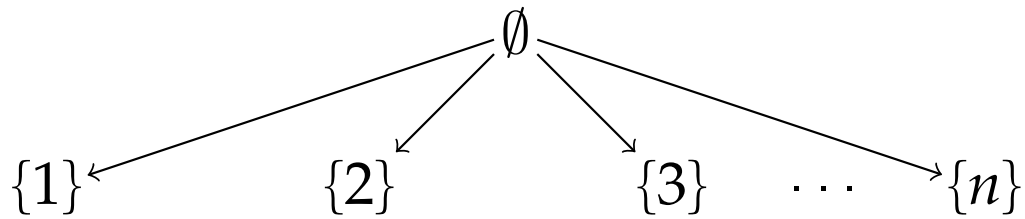



For any $x$ such that $\mathrm{OR}_n(x) = 1$, there exists $i$ such that $x_i = 1$. We use the flow

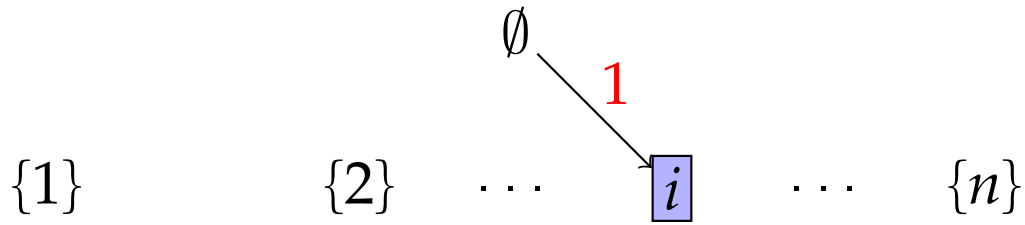


# Example: rederiving Grover's algorithm

Consider the OR function on $n$ bits, $\mathrm{OR}_n(x) = 0 \Leftrightarrow x = 0^n$. We use the following learning graph with weight 1 on each edge shown, and weight 0 on other edges:




For any $x$ such that $\mathrm{OR}_n(x) = 1$, there exists $i$ such that $x_i = 1$. We use the flow




Then $C^0(G) = n$, $C^1(G, x) = 1$, so $C(G) = \sqrt{n}$ (note $0/0 = 0$!).

# Example 2: element distinctness again

- It will be helpful to generalise learning graphs by allowing transitions between sets which differ in size by more than 1.

- For an edge $S \to S \cup T$, define the length $\ell(e) = |T \backslash S|$.

- Generalise the weighted negative and positive complexities of $(G, x)$ to

$$C^0(G) = \sum_{e \in E} \ell(e) w(e), \; C^1(G, x) = \sum_{e \in E} \frac{\ell(e) p_x(e)^2}{w_e}.$$

- The equivalent claims about quantum query complexity still hold.

# Example 2: element distinctness again

**Problem**

Given $x \in [m]^n$, do there exist $i \neq j$ such that $x_i = x_j$?

We will use the following learning graph, parametrised by an integer $r$:

- For all $S \subseteq [n]$ such that $|S| = r$, there is an edge $\emptyset \to S$ with weight $w_1$;

# Example 2: element distinctness again

**Problem**

Given $x \in [m]^n$, do there exist $i \neq j$ such that $x_i = x_j$?

We will use the following learning graph, parametrised by an integer $r$:

- For all $S \subseteq [n]$ such that $|S| = r$, there is an edge $\emptyset \to S$ with weight $w_1$;
- For all $S \subseteq [n]$ such that $|S| = r$, and all $i \notin S$, there is an edge $S \to S \cup \{i\}$ with weight $w_2$;

# Example 2: element distinctness again

**Problem**

Given $x \in [m]^n$, do there exist $i \neq j$ such that $x_i = x_j$?

We will use the following learning graph, parametrised by an integer $r$:

- For all $S \subseteq [n]$ such that $|S| = r$, there is an edge $\emptyset \to S$ with weight $w_1$;
- For all $S \subseteq [n]$ such that $|S| = r$, and all $i \notin S$, there is an edge $S \to S \cup \{i\}$ with weight $w_2$;
- For all $S \subseteq [n]$ such that $|S| = r + 1$, and all $j \notin S$, there is an edge $S \to S \cup \{j\}$ with weight $w_3$.

# Example 2: element distinctness again

**Problem**

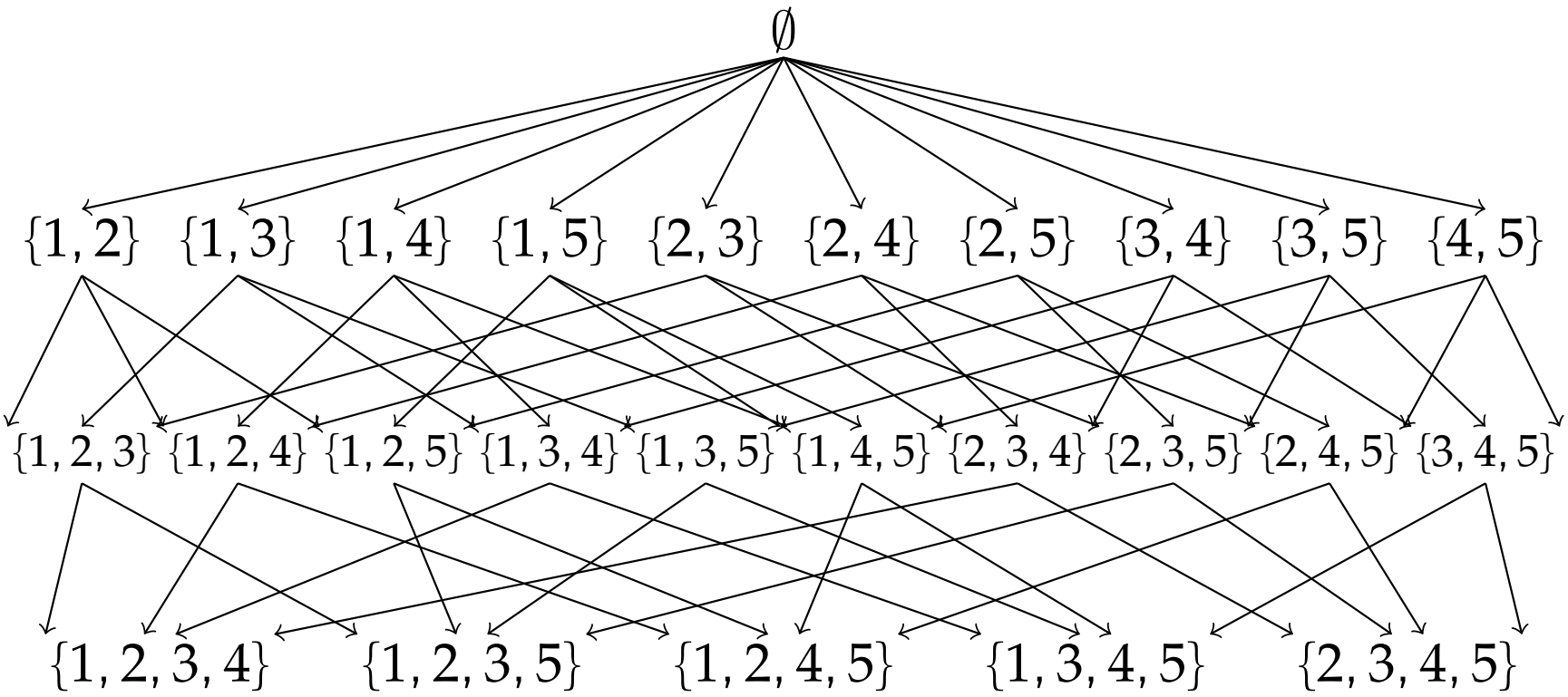
Given $x \in [m]^n$, do there exist $i \neq j$ such that $x_i = x_j$?

We will use the following learning graph, parametrised by an integer $r$:

- For all $S \subseteq [n]$ such that $|S| = r$, there is an edge $\emptyset \to S$ with weight $w_1$;
- For all $S \subseteq [n]$ such that $|S| = r$, and all $i \notin S$, there is an edge $S \to S \cup \{i\}$ with weight $w_2$;
- For all $S \subseteq [n]$ such that $|S| = r + 1$, and all $j \notin S$, there is an edge $S \to S \cup \{j\}$ with weight $w_3$.
- All other edges have weight 0.

# Example 2: element distinctness again

For example, take $n = 5$, $r = 2$:



# Example 2: element distinctness again

For $x$ such that $x_i \neq x_j$:

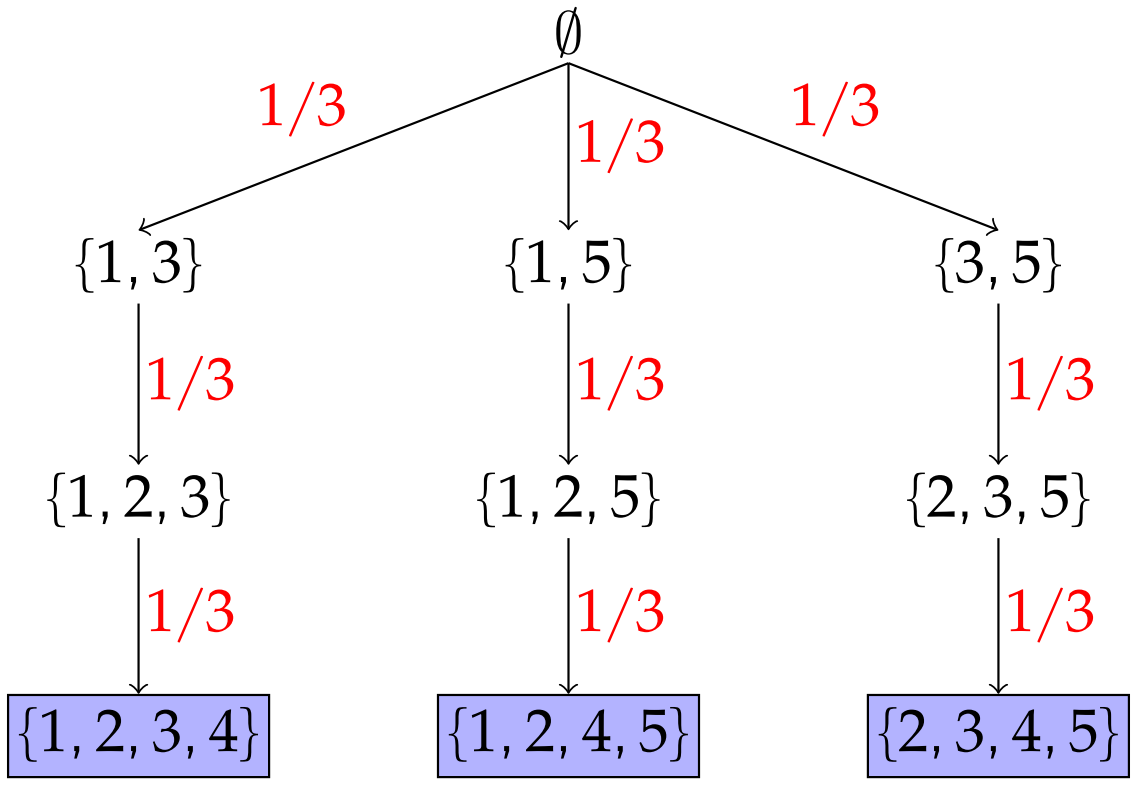
- In the first step, put uniform intensity on the edges corresponding to sets $S$ such that $i, j \notin S$.
- In the second step, put all intensity on edges $S \to S \cup \{i\}$.
- In the third step, put all intensity on edges $T \to T \cup \{j\}$.

# Example 2: element distinctness again

For $x$ such that $x_i \neq x_j$:

- In the first step, put uniform intensity on the edges corresponding to sets $S$ such that $i, j \notin S$.
- In the second step, put all intensity on edges $S \to S \cup \{i\}$.
- In the third step, put all intensity on edges $T \to T \cup \{j\}$.

For example, consider $i = 2$, $j = 4$:



# Example 2: element distinctness again

With this scheme, it turns out that

$$C^0(G) = w_1 r \binom{n}{r} + w_2 (n-r) \binom{n}{r} + w_3 (n-r-1) \binom{n}{r+1}$$

and

$$C^1(G,x) = \frac{r}{w_1 \binom{n-2}{r}} + \frac{1}{w_2 \binom{n-2}{r}} + \frac{1}{w_3 \binom{n-2}{r}}.$$

# Example 2: element distinctness again

With this scheme, it turns out that

$$C^0(G) = w_1 r \binom{n}{r} + w_2 (n-r) \binom{n}{r} + w_3 (n-r-1) \binom{n}{r+1}$$

and

$$C^1(G,x) = \frac{r}{w_1 \binom{n-2}{r}} + \frac{1}{w_2 \binom{n-2}{r}} + \frac{1}{w_3 \binom{n-2}{r}}.$$

Choosing $w_1$, $w_2$, $w_3$ such that each term in the first sum is equal to 1, we get

$$C(G) = O\left(r + \sqrt{n} + n/\sqrt{r}\right)$$

and taking $r = n^{2/3}$ gives $C(G) = O(n^{2/3})$.

# Some other learning graph algorithms

A number of other query algorithms have been discovered in the learning graph model:

- Triangle finding: $O(n^{1.296\ldots})$ [Belovs 1105.4024], $O(n^{1.286\ldots})$ [Lee et al 1210.1014].
- Associativity testing: $O(n^{1.423\ldots})$ [Lee et al 1210.1014]
- Arbitrary subgraphs $H$, $H = k$: $O(n^{2-2/k-g(H)})$ [Zhu 1109.4165], [Lee et al 1109.5135]
- $k$-distinctness: $o(n^{3/4})$ [Belovs and Lee 1108.3022], [Belovs 1205.1534].

All of these beat previously known algorithms based on amplitude amplification and quantum walk.

# Summary and open problems

- There are many quantum algorithms, solving many different problems, using many different techniques.

# Summary and open problems

- There are many quantum algorithms, solving many different problems, using many different techniques.
- The line between techniques and applications is blurry!
  - Efficient Hamiltonian simulation $\Rightarrow$ implementation of quantum walks
  - Implementation of quantum walks $\Rightarrow$ efficient Hamiltonian simulation

# Summary and open problems

- There are many quantum algorithms, solving many different problems, using many different techniques.
- The line between techniques and applications is blurry!
  - Efficient Hamiltonian simulation $\Rightarrow$ implementation of quantum walks
  - Implementation of quantum walks $\Rightarrow$ efficient Hamiltonian simulation

Some specific open problems which seem interesting:

- Find an optimal method for Hamiltonian simulation.

# Summary and open problems

- There are many quantum algorithms, solving many different problems, using many different techniques.
- The line between techniques and applications is blurry!
  - Efficient Hamiltonian simulation $\Rightarrow$ implementation of quantum walks
  - Implementation of quantum walks $\Rightarrow$ efficient Hamiltonian simulation

Some specific open problems which seem interesting:

- Find an optimal method for Hamiltonian simulation.
- Understand efficiency in the learning graph model (significant recent progress by [Belovs 1302.3143] [Jeffery et al 1210.1199]).

# Summary and open problems

- There are many quantum algorithms, solving many different problems, using many different techniques.
- The line between techniques and applications is blurry!
  - Efficient Hamiltonian simulation $\Rightarrow$ implementation of quantum walks
  - Implementation of quantum walks $\Rightarrow$ efficient Hamiltonian simulation

Some specific open problems which seem interesting:

- Find an optimal method for Hamiltonian simulation.
- Understand efficiency in the learning graph model (significant recent progress by [Belovs 1302.3143] [Jeffery et al 1210.1199]).
- Find more algorithmic applications of exponentially faster hitting of quantum walks.

# Thanks!

Some further reading:

- "Quantum algorithms for algebraic problems" [Childs and van Dam 0812.0380]
- "Quantum walk based search algorithms" [Santha 0808.0059]
- "Quantum algorithms" [Mosca 0808.0369]
- "New developments in quantum algorithms" [Ambainis 1006.4014]
- "Quantum algorithms for formula evaluation" [Ambainis 1006.3651]
- "Efficient simulation of Hamiltonians" [Kothari (master's thesis, Waterloo)]

# $k$-local Hamiltonian simulation (proof sketch)

- Using the Lie-Trotter product formula

$$e^{-iA}e^{-iB} = e^{-i(A+B)} + O(\max\{\|A\|, \|B\|\}^2),$$

we get that for $n = \Omega(m^3(Lt)^2/\epsilon)$

$$\left\| \left( e^{-iH_1 t/n} e^{-iH_2 t/n} \dots e^{-iH_m t/n} \right)^n - e^{-i(H_1+\dots+H_m)t} \right\| \leqslant \epsilon.$$

# $k$-local Hamiltonian simulation (proof sketch)

- Using the Lie-Trotter product formula

$$e^{-iA}e^{-iB} = e^{-i(A+B)} + O(\max\{\|A\|, \|B\|\}^2),$$

we get that for $n = \Omega(m^3(Lt)^2/\epsilon)$

$$\left\| \left( e^{-iH_1 t/n} e^{-iH_2 t/n} \dots e^{-iH_m t/n} \right)^n - e^{-i(H_1+\dots+H_m)t} \right\| \leqslant \epsilon.$$

- As each Hamiltonian $H_j$ is $O(1)$-local, $e^{-iH_j t/n}$ can be approximated efficiently by the Solovay-Kitaev theorem.

# $k$-local Hamiltonian simulation (proof sketch)

- Using the Lie-Trotter product formula

$$e^{-iA}e^{-iB} = e^{-i(A+B)} + O(\max\{\|A\|, \|B\|\}^2),$$

  we get that for $n = \Omega(m^3(Lt)^2/\epsilon)$

$$\left\| \left( e^{-iH_1 t/n} e^{-iH_2 t/n} \dots e^{-iH_m t/n} \right)^n - e^{-i(H_1+\dots+H_m)t} \right\| \leqslant \epsilon.$$

- As each Hamiltonian $H_j$ is $O(1)$-local, $e^{-iH_j t/n}$ can be approximated efficiently by the Solovay-Kitaev theorem.

This algorithm can be improved using higher-order product formulae.

- In particular [Berry et al 0508139], we can simulate $H$ for time $t$ with a circuit which runs in time

$$m^2 \|H\| t e^{O(\sqrt{\log m \|H\| t/\epsilon})}.$$