The Power of Quantum Computation

Ashley Montanaro

Department of Computer Science,
University of Bristol

8 May 2014

University of BRISTOL

Introduction

What can we do with our quantum computers?

Introduction

What can we do with our quantum computers?

This talk:

1. Applications to cryptography
2. Limitations of quantum computers
3. More recent developments in quantum algorithms

Introduction

What can we do with our quantum computers?

This talk:

1. Applications to cryptography
2. Limitations of quantum computers
3. More recent developments in quantum algorithms

The Quantum Algorithm Zoo (http://math.nist.gov/quantum/zoo/) cites 214 papers on quantum algorithms alone, so this is necessarily a partial view…

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

- The best classical algorithm we have for factorisation (the number field sieve) runs in time

$$\exp(O(n^{1/3}(\log n)^{2/3})).$$

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

- The best classical algorithm we have for factorisation (the number field sieve) runs in time

$$\exp(O(n^{1/3}(\log n)^{2/3})).$$

- The RSA cryptosystem is based around the hardness of this task.
- That is, if we can factorise large integers efficiently, we can break RSA.

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

- The best classical algorithm we have for factorisation (the number field sieve) runs in time

$$\exp(O(n^{1/3}(\log n)^{2/3})).$$

- The RSA cryptosystem is based around the hardness of this task.
- That is, if we can factorise large integers efficiently, we can break RSA.

Theorem [Shor '97]

There is a quantum algorithm which finds the prime factors of an n-digit integer in time $O(n^3)$.

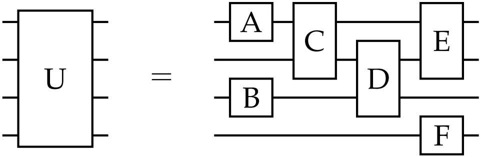
Quantum time complexity

How do we measure the complexity of algorithms which run on a quantum computer?

Quantum time complexity

How do we measure the complexity of algorithms which run on a quantum computer?

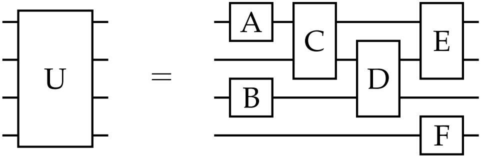
- We usually use the quantum circuit model: we imagine a quantum computation as built from a sequence of elementary operations ("quantum gates"), each acting on a small number of qubits.



Quantum time complexity

How do we measure the complexity of algorithms which run on a quantum computer?

- We usually use the quantum circuit model: we imagine a quantum computation as built from a sequence of elementary operations ("quantum gates"), each acting on a small number of qubits.

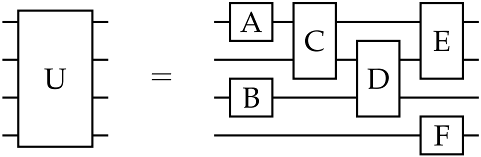


- Then the time complexity of the algorithm is (roughly) modelled by the number of quantum gates used.

Quantum time complexity

How do we measure the complexity of algorithms which run on a quantum computer?

- We usually use the quantum circuit model: we imagine a quantum computation as built from a sequence of elementary operations ("quantum gates"), each acting on a small number of qubits.



- Then the time complexity of the algorithm is (roughly) modelled by the number of quantum gates used.
- Sometimes it is reasonable to measure the complexity of the algorithms by the number of queries to the input used.

Shor's algorithm: complexity comparison

Very roughly (ignoring constant factors!):

Number of digits	Timesteps (quantum)	Timesteps (classical)
100	10^6	$\sim 4 \times 10^5$
1,000	10^9	$\sim 5 \times 10^{15}$
10,000	10^{12}	$\sim 1 \times 10^{41}$

Shor's algorithm: complexity comparison

Very roughly (ignoring constant factors!):

Number of digits	Timesteps (quantum)	Timesteps (classical)
100	10^6	$\sim 4 \times 10^5$
1,000	10^9	$\sim 5 \times 10^{15}$
10,000	10^{12}	$\sim 1 \times 10^{41}$

Based on these figures, a 10,000-digit number could be factorised by:

- A quantum computer executing 10^9 instructions per second (comparable to today's desktop PCs) in 16 minutes.

Shor's algorithm: complexity comparison

Very roughly (ignoring constant factors!):

Number of digits	Timesteps (quantum)	Timesteps (classical)
100	10^6	$\sim 4 \times 10^5$
1,000	10^9	$\sim 5 \times 10^{15}$
10,000	10^{12}	$\sim 1 \times 10^{41}$

Based on these figures, a 10,000-digit number could be factorised by:

- A quantum computer executing 10^9 instructions per second (comparable to today's desktop PCs) in 16 minutes.
- The fastest computer on the Top500 supercomputer list ($\sim 3.4 \times 10^{16}$ operations per second) in $\sim 1.2 \times 10^{17}$ years.

(see e.g. [Van Meter et al '05] for a more detailed comparison)

The abelian hidden subgroup problem

The underlying mathematical problem which Shor's algorithm solves is:

Hidden subgroup problem (e.g. [Boneh and Lipton '95])

Let G be a group. Given oracle access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

The abelian hidden subgroup problem

The underlying mathematical problem which Shor's algorithm solves is:

Hidden subgroup problem (e.g. [Boneh and Lipton '95])

Let G be a group. Given oracle access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

- On a quantum computer, this problem can be solved using $O(\log |G|)$ queries to f. The algorithm is also time-efficient for all abelian groups G.
- Integer factorisation reduces to the case $G = \mathbb{Z}_M$ for some integer M.

The discrete log problem

Other important special cases of the abelian hidden subgroup problem:

Discrete log problem [Shor '97]

Given $g, x \in \mathbb{Z}_p^\times$ for some prime p, find y such that $g^y = x$.

- Can be reduced to the hidden subgroup problem on $\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$.
- Breaks Diffie-Hellman, ElGamal, DSA, ...

The discrete log problem

Other important special cases of the abelian hidden subgroup problem:

Discrete log problem [Shor '97]

Given $g, x \in \mathbb{Z}_p^\times$ for some prime p, find y such that $g^y = x$.

- Can be reduced to the hidden subgroup problem on $\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$.
- Breaks Diffie-Hellman, ElGamal, DSA, ...

Elliptic curves (e.g. [Proos and Zalka '03])

There is a polynomial-time quantum algorithm for the discrete log problem in the additive group of points on an elliptic curve over a finite field.

- Breaks ECDH, ECDSA, ECxxx, ...

The Shifted Legendre Symbol problem

Shifted Legendre Symbol problem [van Dam et al '00-'06]

Given access to the function $f : \mathbb{F}_p \to \mathbb{F}_p$ such that $f(x) = \left(\frac{x+s}{p}\right)$, where $\left(\frac{x}{p}\right)$ is the Legendre symbol $x^{(p-1)/2} \pmod{p}$, find s.

The Shifted Legendre Symbol problem

Shifted Legendre Symbol problem [van Dam et al '00-'06]

Given access to the function $f : \mathbb{F}_p \to \mathbb{F}_p$ such that $f(x) = \left(\frac{x+s}{p}\right)$, where $\left(\frac{x}{p}\right)$ is the Legendre symbol $x^{(p-1)/2} \pmod{p}$, find s.

- There is a quantum algorithm which solves this problem in time $\text{poly}(\log p)$, breaking a proposed secure pseudorandom number generator [Damgård '88].

The Shifted Legendre Symbol problem

Shifted Legendre Symbol problem **[van Dam et al '00-'06]**

Given access to the function $f : \mathbb{F}_p \to \mathbb{F}_p$ such that $f(x) = \left(\frac{x+s}{p}\right)$, where $\left(\frac{x}{p}\right)$ is the Legendre symbol $x^{(p-1)/2} \pmod{p}$, find s.

- There is a quantum algorithm which solves this problem in time $\mathrm{poly}(\log p)$, breaking a proposed secure pseudorandom number generator [Damgård '88].
- Allows certain algebraically homomorphic cryptosystems to be broken.
- Assume that we have access to a deterministic encryption function $E : \mathbb{F}_p \to X$ such that, given the encryptions $E(x)$, $E(y)$ of $x, y \in \mathbb{F}_p$, we can construct $E(x + y)$ and $E(xy)$ efficiently.
- Then (modulo some technicalities) using this algorithm we can find s efficiently given $E(s)$.

Grover's algorithm

One of the most basic problems in computer science is unstructured search.

Grover's algorithm

One of the most basic problems in computer science is unstructured search.

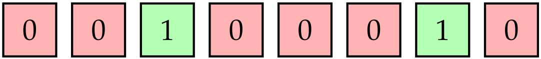
- Imagine we have access to a function $f : \{0, 1\}^n \to \{0, 1\}$ which we treat as a black box.

Grover's algorithm

One of the most basic problems in computer science is unstructured search.

- Imagine we have access to a function $f : \{0,1\}^n \to \{0,1\}$ which we treat as a black box.

- We want to find an x such that $f(x) = 1$.



Grover's algorithm

One of the most basic problems in computer science is unstructured search.

- Imagine we have access to a function $f : \{0,1\}^n \to \{0,1\}$ which we treat as a black box.
- We want to find an x such that $f(x) = 1$.

0	0	1	0	0	0	1	0

- On a classical computer, this task could require 2^n queries to f in the worst case. But on a quantum computer, Grover's algorithm [Grover '97] can solve the problem with $O(\sqrt{2^n})$ queries to f (and bounded error).

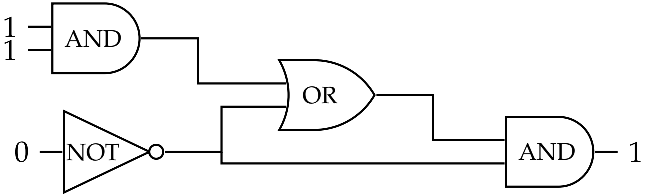
Applications of Grover's algorithm

Grover's algorithm gives a speedup over naïve algorithms for any decision problem in NP, i.e. where we can verify the solution efficiently.

Applications of Grover's algorithm

Grover's algorithm gives a speedup over naïve algorithms for any decision problem in NP, i.e. where we can verify the solution efficiently.

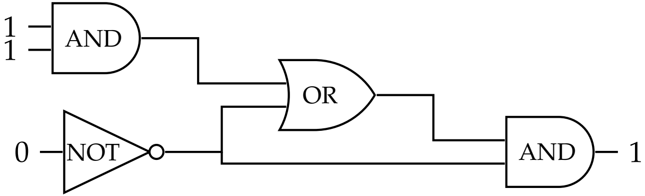
- For example, in the CIRCUIT SAT problem we would like to find an input to a circuit on n bits such that the output is 1:



Applications of Grover's algorithm

Grover's algorithm gives a speedup over naïve algorithms for any decision problem in NP, i.e. where we can verify the solution efficiently.

- For example, in the CIRCUIT SAT problem we would like to find an input to a circuit on n bits such that the output is 1:



- Grover's algorithm improves the runtime from $O(2^n)$ to $O(2^{n/2})$: applications to design automation, circuit equivalence, model checking, ...

Applications of Grover's algorithm

An important generalisation: amplitude amplification.

Amplitude amplification [Brassard et al '00]

Assume we are given access to a "checking" function f, and a probabilistic algorithm $\mathcal{A}$ such that

$$\Pr[\mathcal{A} \text{ outputs } w \text{ such that } f(w) = 1] = \epsilon.$$

Then we can find w such that $f(w) = 1$ with $O(1/\sqrt{\epsilon})$ uses of f.

- Gives a quadratic speed-up over classical algorithms based on the use of f as a black box.

Applications of Grover's algorithm

An important generalisation: amplitude amplification.

Amplitude amplification [Brassard et al '00]

Assume we are given access to a "checking" function f, and a probabilistic algorithm $\mathcal{A}$ such that

$$\Pr[\mathcal{A} \text{ outputs } w \text{ such that } f(w) = 1] = \epsilon.$$

Then we can find w such that $f(w) = 1$ with $O(1/\sqrt{\epsilon})$ uses of f.

- Gives a quadratic speed-up over classical algorithms based on the use of f as a black box.
- These primitives can be used to obtain many speedups over classical algorithms, e.g. finding a collision in a 2-1 function $f : [N] \to [N]$ with $O(N^{1/3})$ queries [Brassard et al '98] (but note controversy [Bernstein '09])

What quantum computers can't do

A number of bounds on the power of quantum computation are known.

Most results are in the query complexity model where we assume the algorithm wants to solve some problem given only access to an oracle as a black box. For example:

- Any quantum algorithm solving the unstructured search problem must use $\Omega(2^{n/2})$ queries [Bennett et al '97].
- Any quantum algorithm finding a collision in a 2-1 function $f : [N] \to [N]$ must use $\Omega(N^{1/3})$ queries to the function [Aaronson and Shi '04].

What quantum computers can't do (yet)

Hidden subgroup problem

Let G be a group. Given access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

What quantum computers can't do (yet)

Hidden subgroup problem

Let G be a group. Given access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

- Solving the HSP for the dihedral group (in a certain way) gives a quantum algorithm for the shortest vector problem (SVP) in lattices [Regev '04].

What quantum computers can't do (yet)

Hidden subgroup problem

Let G be a group. Given access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

- Solving the HSP for the dihedral group (in a certain way) gives a quantum algorithm for the shortest vector problem (SVP) in lattices [Regev '04].
- Solving the HSP for the symmetric group gives a quantum algorithm for graph isomorphism.

What quantum computers can't do (yet)

Hidden subgroup problem

Let G be a group. Given access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

- Solving the HSP for the dihedral group (in a certain way) gives a quantum algorithm for the shortest vector problem (SVP) in lattices [Regev '04].
- Solving the HSP for the symmetric group gives a quantum algorithm for graph isomorphism.

There is no known efficient quantum algorithm (i.e. running in time $\text{poly}(\log |G|)$) for all nonabelian groups G.

- In particular, the best known algorithm for the dihedral group is subexponential-time: $2^{O(\sqrt{|G|})}$ [Kuperberg '05].

McEliece cryptosystem

The McEliece cryptosystem is (roughly) based on the hardness of finding transformations between equivalent linear codes.

The McEliece cryptosystem

Let C be an (n, k) linear code which can correct t errors. Let G be the $n \times k$ generator matrix for C, let S be a random $k \times k$ invertible matrix, and let P be a random $n \times n$ permutation. Then the public key is $G' = SGP$.

McEliece cryptosystem

The McEliece cryptosystem is (roughly) based on the hardness of finding transformations between equivalent linear codes.

The McEliece cryptosystem

Let C be an (n, k) linear code which can correct t errors. Let G be the $n \times k$ generator matrix for C, let S be a random $k \times k$ invertible matrix, and let P be a random $n \times n$ permutation. Then the public key is $G' = SGP$.

- There can be no efficient attack on this cryptosystem based on Fourier sampling (the key ingredient in Shor's algorithm) [Dinh et al '10]...

McEliece cryptosystem

The McEliece cryptosystem is (roughly) based on the hardness of finding transformations between equivalent linear codes.

The McEliece cryptosystem

Let C be an (n, k) linear code which can correct t errors. Let G be the $n \times k$ generator matrix for C, let S be a random $k \times k$ invertible matrix, and let P be a random $n \times n$ permutation. Then the public key is $G' = SGP$.

- There can be no efficient attack on this cryptosystem based on Fourier sampling (the key ingredient in Shor's algorithm) [Dinh et al '10]...
- ...however, Grover's algorithm improves the runtime of the best known classical algorithms by a square root [Bernstein '10].

"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 $Ax = b$.

"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 $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

“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 κ $(= \|A^{-1}\|\|A\|)$, $|x\rangle$ can be approximately produced in time poly$(\log N, d, \kappa)$ [Harrow et al ’08].

“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 κ $(= \|A^{-1}\| \|A\|)$, $|x\rangle$ can be approximately produced in time $\mathrm{poly}(\log N, d, \kappa)$ [Harrow et al ’08].

Later improved to time $O(\kappa \log^3 \kappa \, \mathrm{poly}(d) \log N)$ [Ambainis ’10].

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 $\mathsf{BPP} = \mathsf{BQP}$!

More recent applications of this algorithm include:

- "Solving" differential equations [Leyton and Osborne '08] [Berry '10]
- Data fitting [Wiebe et al '12]
- Space-efficient matrix inversion [Ta-Shma '13]

Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk. They have many applications, such as the element distinctness problem.

Problem

Given a set of n integers, are they all distinct?

Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk. They have many applications, such as the element distinctness problem.

Problem

Given a set of n integers, are they all distinct?

- Classically, we need to look at all n integers.

Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk. They have many applications, such as the element distinctness problem.

Problem

Given a set of n integers, are they all distinct?

- Classically, we need to look at all n integers.
- Try using Grover's algorithm on the set of all pairs: $O(\sqrt{n^2}) = O(n)$.

Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk. They have many applications, such as the element distinctness problem.

Problem

Given a set of n integers, are they all distinct?

- Classically, we need to look at all n integers.
- Try using Grover's algorithm on the set of all pairs: $O(\sqrt{n^2}) = O(n)$.

Theorem [Ambainis '03]

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

Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk. They have many applications, such as the element distinctness problem.

Problem

Given a set of n integers, are they all distinct?

- Classically, we need to look at all n integers.
- Try using Grover's algorithm on the set of all pairs: $O(\sqrt{n^2}) = O(n)$.

Theorem [Ambainis '03]

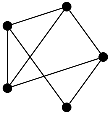
Element Distinctness can be solved using $O(n^{2/3})$ queries.

- The algorithm is based on discrete-time quantum walks.
- Generalisation to finding a k-subset of $\mathbb{Z}^n$ satisfying any property: uses $O(n^{k/(k+1)})$ queries.

Some examples

The same quantum walk framework lends itself to many different search problems, such as:

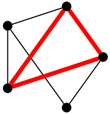
- Finding a triangle in a graph: $O(n^{1.3})$ queries, vs. classical $O(n^2)$ [Magniez et al '03] [Jeffery et al '12]



Some examples

The same quantum walk framework lends itself to many different search problems, such as:

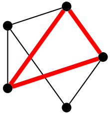
- Finding a triangle in a graph: $O(n^{1.3})$ queries, vs. classical $O(n^2)$ [Magniez et al '03] [Jeffery et al '12]



Some examples

The same quantum walk 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 '03] [Jeffery et al '12]



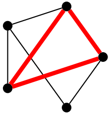
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

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

Some examples

The same quantum walk 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 '03] [Jeffery et al '12]



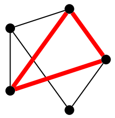
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

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

Some examples

The same quantum walk 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 '03] [Jeffery et al '12]



- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

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

- Testing group commutativity: $O(n^{2/3} \log n)$ queries, vs. classical $O(n)$ [Magniez and Nayak '05]

Summary and further reading

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

Summary and further reading

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

However, we now also have many lower bounds in the black-box model of query complexity, which map out the power of quantum computation and show its limitations.

Summary and further reading

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

However, we now also have many lower bounds in the black-box model of query complexity, which map out the power of quantum computation and show its limitations.

Some further reading:

- "Quantum algorithms for algebraic problems" [Childs and van Dam '08]
- "Quantum walk based search algorithms" [Santha '08]
- "Quantum algorithms" [Mosca '08]
- "New developments in quantum algorithms" [Ambainis '10]

Summary and further reading

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

However, we now also have many lower bounds in the black-box model of query complexity, which map out the power of quantum computation and show its limitations.

Some further reading:

- "Quantum algorithms for algebraic problems" [Childs and van Dam '08]
- "Quantum walk based search algorithms" [Santha '08]
- "Quantum algorithms" [Mosca '08]
- "New developments in quantum algorithms" [Ambainis '10]

Thanks!