Quantum Algorithms

Ashley Montanaro

Department of Computer Science,
University of Bristol

2 February 2015

University of BRISTOL

EPSRC
Engineering and Physical Sciences Research Council

Introduction

What can we do with our quantum computers?

Introduction

What can we do with our quantum computers?

This talk:

1. Classic applications
2. More recent applications
3. Applications with no quantum computer required

Introduction

What can we do with our quantum computers?

This talk:

1. Classic applications
2. More recent applications
3. Applications with no quantum computer required

The Quantum Algorithm Zoo (http://math.nist.gov/quantum/zoo/) cites 245 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 that underlies Internet security 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 that underlies Internet security 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 $O(n^3)$.

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)

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 $f(x) = 1$.

0	0	1	0	0	0	1	0

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 failure probability).

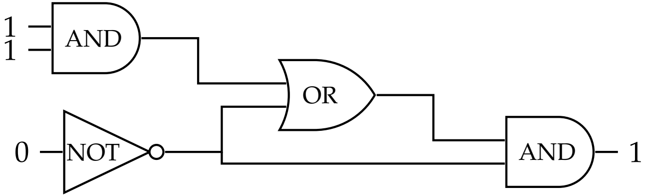
Applications of Grover's algorithm

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

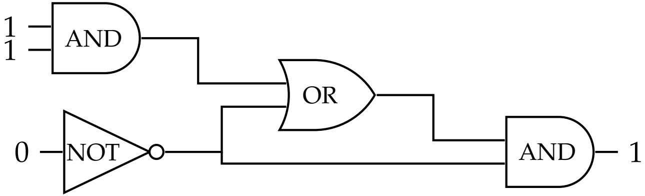
- For example, in the Circuit SAT problem we would like to find an input to a circuit on n bits such that the output is 1:



Applications of Grover's algorithm

Grover's algorithm gives a speedup over naïve algorithms for any decision problem in the complexity class 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, ...

Quadratic speedup

Is a quadratic speedup significant?

Quadratic speedup

Is a quadratic speedup significant?

- A concrete example: An instance of CIRCUIT SAT on 50 input bits.
- Assume evaluation of the circuit needs 1μs on a classical computer, and 1ms on a quantum computer.

Quadratic speedup

Is a quadratic speedup significant?

- A concrete example: An instance of CIRCUIT SAT on 50 input bits.
- Assume evaluation of the circuit needs 1μs on a classical computer, and 1ms on a quantum computer.
- Worst-case classical runtime: $\approx 2^{50} \times 10^{-6}$ seconds, or $\geqslant 35$ years.

Quadratic speedup

Is a quadratic speedup significant?

- A concrete example: An instance of CIRCUIT SAT on 50 input bits.
- Assume evaluation of the circuit needs $1\mu s$ on a classical computer, and $1ms$ on a quantum computer.
- Worst-case classical runtime: $\approx 2^{50} \times 10^{-6}$ seconds, or $\geqslant 35$ years.
- Quantum runtime: $\approx 2^{25} \times 10^{-3}$ seconds, or $\leqslant 10$ hours.

Applications of Grover's algorithm

An important generalisation of Grover's algorithm is known as 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 which are based on heuristics.

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al '04]

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al '04]
- Finding a collision in a 2-1 function $f : [n] \to [n]$ in $O(n^{1/3})$ time [Brassard et al '98]

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al '04]
- Finding a collision in a 2-1 function $f : [n] \to [n]$ in $O(n^{1/3})$ time [Brassard et al '98]
- Finding a maximal matching in a bipartite graph with V vertices and E edges in $O(V\sqrt{E}\log V)$ time [Ambainis and Špalek '05]

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al '04]
- Finding a collision in a 2-1 function $f : [n] \to [n]$ in $O(n^{1/3})$ time [Brassard et al '98]
- Finding a maximal matching in a bipartite graph with V vertices and E edges in $O(V\sqrt{E}\log V)$ time [Ambainis and Špalek '05]
- Approximating the ℓ_1 distance between probability distributions on n elements in $O(\sqrt{n})$ time [Bravyi et al '09]
- ...

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation.

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation.

Problem

Given a Hamiltonian H describing a physical system, and an initial state $|\psi_0\rangle$ of that system, produce the state

$$|\psi_t\rangle = e^{-iHt}|\psi_0\rangle.$$

Given such an output state, measurements can be performed to determine quantities of interest about the state.

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation.

Problem

Given a Hamiltonian H describing a physical system, and an initial state $|\psi_0\rangle$ of that system, produce the state

$$|\psi_t\rangle = e^{-iHt}|\psi_0\rangle.$$

Given such an output state, measurements can be performed to determine quantities of interest about the state.

- No efficient classical algorithm is known for this task (in full generality), but efficient quantum algorithms exist for many physically reasonable cases.

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation.

Problem

Given a Hamiltonian H describing a physical system, and an initial state $|\psi_0\rangle$ of that system, produce the state

$$|\psi_t\rangle = e^{-iHt}|\psi_0\rangle.$$

Given such an output state, measurements can be performed to determine quantities of interest about the state.

- No efficient classical algorithm is known for this task (in full generality), but efficient quantum algorithms exist for many physically reasonable cases.
- Applications: quantum chemistry, superconductivity, metamaterials, high-energy physics, ... [Georgescu et al '13]

“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:

- Computing electromagnetic scattering cross-sections [Clader et al '13]
- "Solving" differential equations [Leyton and Osborne '08] [Berry '14]
- Data fitting [Wiebe et al '12]
- Space-efficient matrix inversion [Ta-Shma '13]

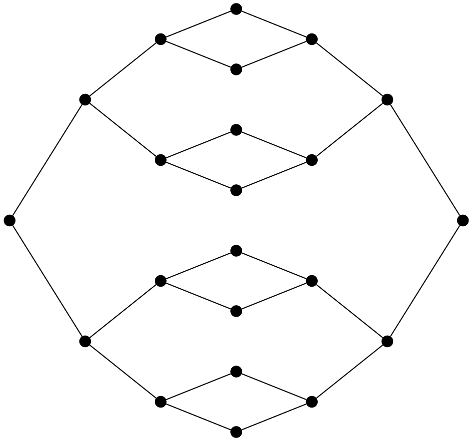
Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk.

- Two variants: continuous-time and discrete-time.
- A continuous-time quantum walk for time t on a graph with adjacency matrix A is the application of the unitary operator e^{-iAt}.
- Continuous-time quantum walks can be efficiently implemented as quantum circuits using Hamiltonian simulation.

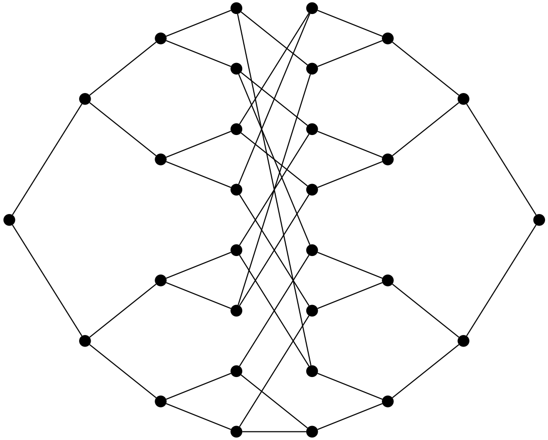
Quantum walks

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



Quantum walks

Now add a random cycle in the middle:



Quantum walk on the glued trees graph

Theorem **[Childs et al '02]**

- 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 '02]

- 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 $O(N^{1/6})$ queries to find the exit.

Quantum walk on the glued trees graph

Theorem [Childs et al '02]

- 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 $O(N^{1/6})$ queries to find the exit.

Other applications of continuous-time quantum walks:

- Spatial search [Childs and Goldstone '03]
- Evaluation of boolean formulae [Farhi et al '07] [Childs et al '07]

Element distinctness

Problem

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

Element distinctness

Problem

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

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

Element distinctness

Problem

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

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

Element distinctness

Problem

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

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

Element distinctness

Problem

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

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

Theorem [Ambainis '03]

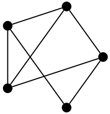
Element Distinctness can be solved using $O(n^{2/3})$ queries.

- The algorithm is based on discrete-time quantum walks.
- Time complexity is the same up to polylogarithmic factors.
- 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:

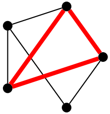
- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



Some examples

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

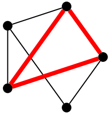
- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



Some examples

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

- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



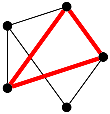
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

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

Some examples

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

- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



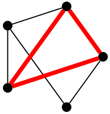
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

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

Some examples

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

- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



- 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]

Yet more algorithms

There are a number of other quantum algorithms which I don't have time to go into:

- Hidden subgroup problems (e.g. [Bacon et al '05])
- Number-theoretic problems (e.g. [Fontein and Wocjan '11], . . .)
- Formula evaluation (e.g. [Reichardt and Špalek '07])
- Tensor contraction (e.g. [Arad and Landau '08])
- Hidden shift problems (e.g. [Gavinsky et al '11])
- Adiabatic optimisation (e.g. [Farhi et al '00])
- . . .

. . . as well as the entire field of quantum communication complexity.

Quantum computing without a quantum computer

Although we don't have a large-scale quantum computer yet, ideas from quantum computation and quantum information theory have already paid dividends:

Quantum computing without a quantum computer

Although we don't have a large-scale quantum computer yet, ideas from quantum computation and quantum information theory have already paid dividends:

- The burgeoning field of Hamiltonian complexity and QMA-completeness has characterised the hardness of ground-state energy estimation problems for a variety of physical systems (e.g. [Kitaev, Shen and Vyalyi '02] [Schuch and Verstraete '09] [Cubitt and AM '13])

Quantum computing without a quantum computer

Although we don't have a large-scale quantum computer yet, ideas from quantum computation and quantum information theory have already paid dividends:

- The burgeoning field of Hamiltonian complexity and QMA-completeness has characterised the hardness of ground-state energy estimation problems for a variety of physical systems (e.g. [Kitaev, Shen and Vyalyi '02] [Schuch and Verstraete '09] [Cubitt and AM '13])
- Understanding multiple-prover quantum Merlin-Arthur proof systems has given new lower bounds on the classical complexity of computing tensor and matrix norms [Harrow and AM '10]

Quantum computing without a quantum computer

Although we don't have a large-scale quantum computer yet, ideas from quantum computation and quantum information theory have already paid dividends:

- The burgeoning field of Hamiltonian complexity and QMA-completeness has characterised the hardness of ground-state energy estimation problems for a variety of physical systems (e.g. [Kitaev, Shen and Vyalyi '02] [Schuch and Verstraete '09] [Cubitt and AM '13])
- Understanding multiple-prover quantum Merlin-Arthur proof systems has given new lower bounds on the classical complexity of computing tensor and matrix norms [Harrow and AM '10]
- New limitations on classical data structures, codes and formulas (see e.g. [Drucker and de Wolf '09])

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.

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.

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!

Primitive: Phase estimation

Phase estimation [Cleve et al '97] [Kitaev '95]

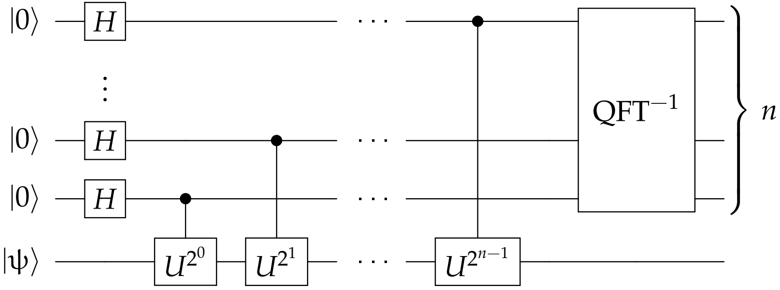
Given access to a unitary U and an eigenvector $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i \phi}|\psi\rangle$, we can estimate ϕ up to additive error ϵ with $O(1/\epsilon)$ uses of U.

Primitive: Phase estimation

Phase estimation [Cleve et al '97] [Kitaev '95]

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 ϕ up to additive error ϵ with $O(1/\epsilon)$ uses of U.

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



and then measure the first n qubits.