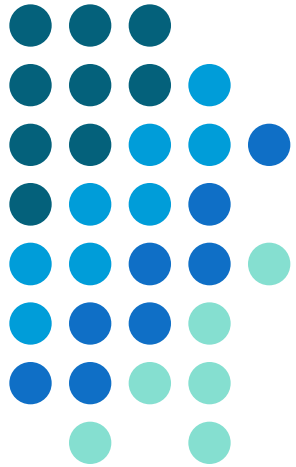
Quantum Algorithms for Testing Graph Expansion and Bipartiteness

Yi-Kai Liu

National Institute of Standards and Technology

& University of Maryland



Joint work with:
Andris Ambainis (University of Latvia)
Andrew M. Childs (University of Maryland)

Quantum algorithms

- What is a quantum computer?

LETTER

doi:10.1038/nature18648

Demonstration of a small programmable quantum computer with atomic qubits

S. Debnath[1], N. M. Linke[1], C. Figgatt[1], K. A. Landsman[1], K. Wright[1] & C. Monroe[1,2,3]

Quantum algorithms

- What is a quantum computer?
  - State of n qubits is described by a unit vector $|\psi\rangle \in (C^2)^{\times n} \cong C^{2^n}$
  - Logical operations are unitary matrices acting on 1 or 2 qubits, e.g., $(I^{\times k}) \times U \times (I^{\times(n-k-2)})$
  - Measurement returns a binary string $z \in \{0,1\}^n$ with probability $|\psi_z|^2$

LETTER

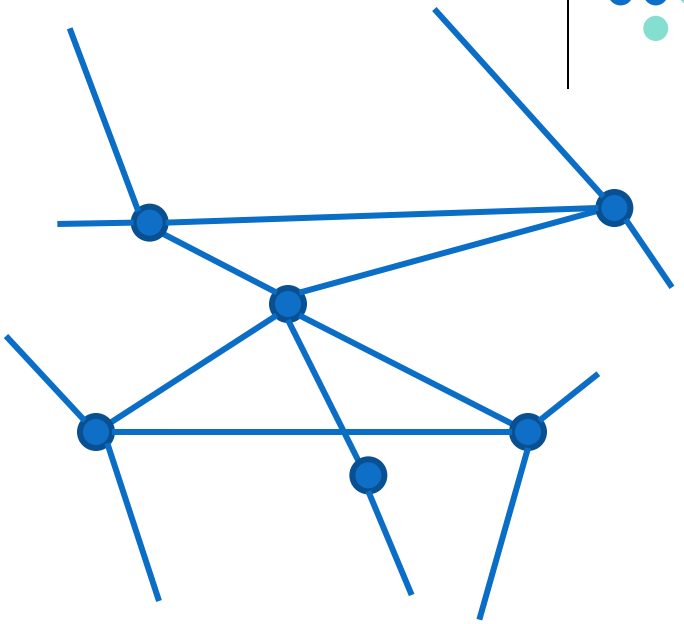
doi:10.1038/nature18648

Demonstration of a small programmable quantum computer with atomic qubits

S. Debnath[1], N. M. Linke[1], C. Figgatt[1], K. A. Landsman[1], K. Wright[1] & C. Monroe[1,2,3]

Property testing

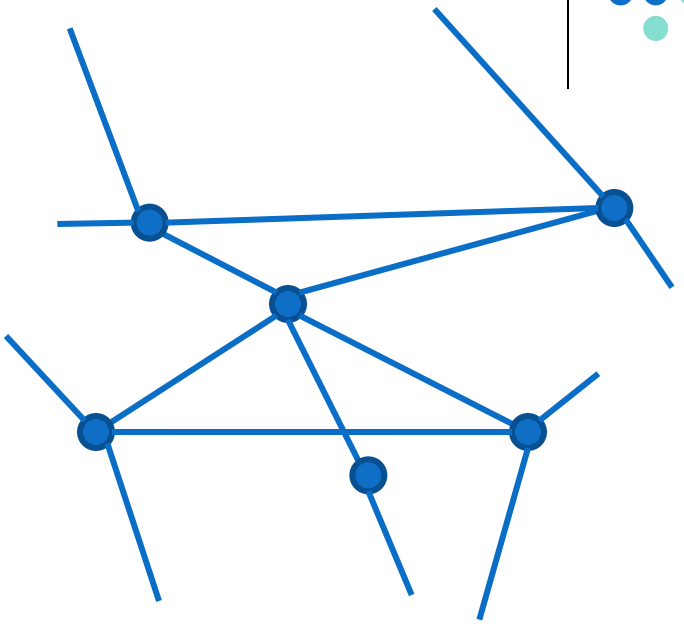
- Graph G, with N vertices
- Constant degree d



- Problem:
  Decide whether G has property P, or is ε-far from having P

Property testing

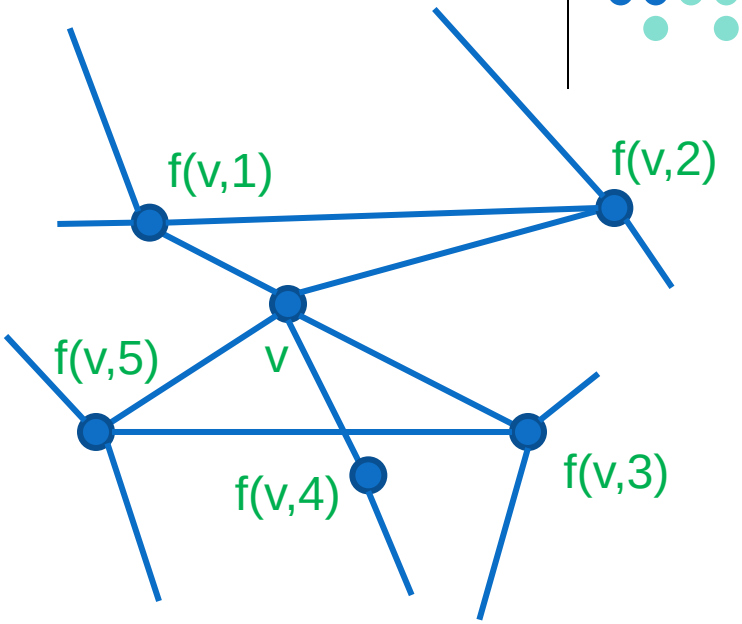
- Graph G, with N vertices
- Constant degree d



- Problem:
  Decide whether G has property P, or is ε-far from having P
  - ε-far: “differ by at least εNd edges,” where ε = constant
  - Answer is insensitive to errors in the description of G
  - Algorithms can run in sublinear time, by looking at random subsets of G

Property testing

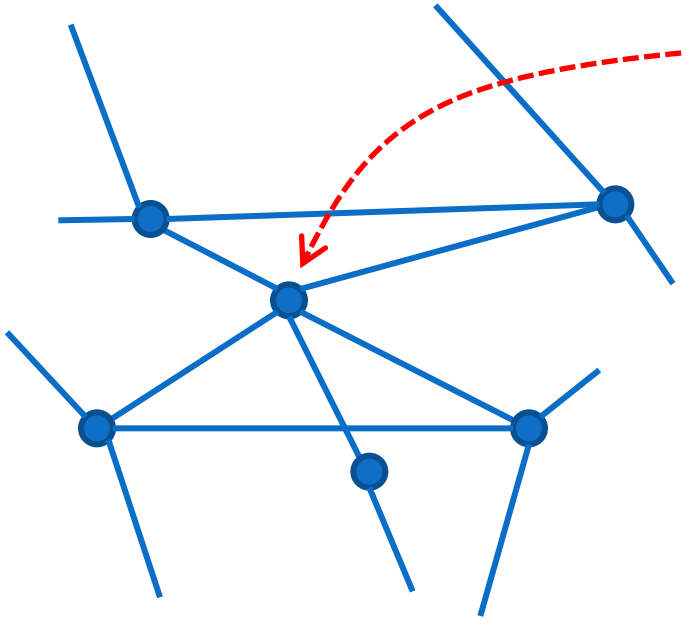
- Graph G, with N vertices
- Constant degree d
- Graph is specified by "adjacency list" oracle:
  $f: [N] \times [d] \rightarrow [N] \cup \{?\}$
  - Each query takes O(Nd) gates, but only O(log(Nd)) depth



- Problem:
  Decide whether G has property P, or is ε-far from having P
  - ε-far: "differ by at least εNd edges," where ε = constant
  - Answer is insensitive to errors in the description of G
  - Algorithms can run in sublinear time, by looking at random subsets of G

Test whether G is an expander

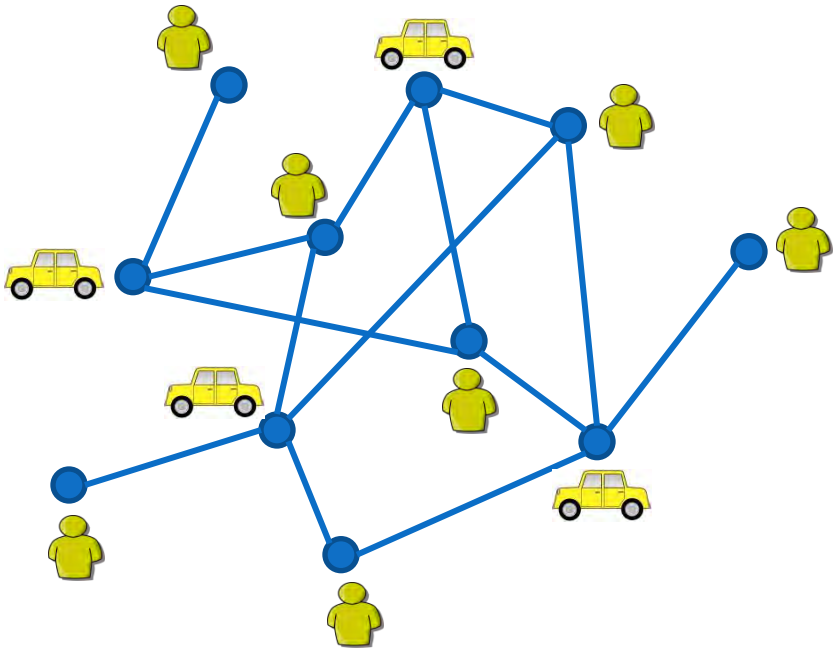
- How quickly can information spread through a network?



https://xkcd.com/1197/

Test whether G is bipartite

- Are there two types of nodes in the network?




Drivers


Passengers

This talk

- Testing *expansion* and *bipartiteness* of bounded-degree graphs in *sub-linear time*

	Classical: (Goldreich, Ron, Czumaj, Sohler, …)	Quantum:
Algorithms	$O(\sqrt{N})$	
Query lower bounds	$\Omega(\sqrt{N})$	

This talk

- Testing *expansion* and *bipartiteness* of bounded-degree graphs in *sub-linear time*

	Classical: (Goldreich, Ron, Czumaj, Sohler, …)	Quantum:
Algorithms	$O(\sqrt{N})$	$O(N^{1/3})$ for both problems, using Ambainis' algorithm for element distinctness
Query lower bounds	$\Omega(\sqrt{N})$	

This talk

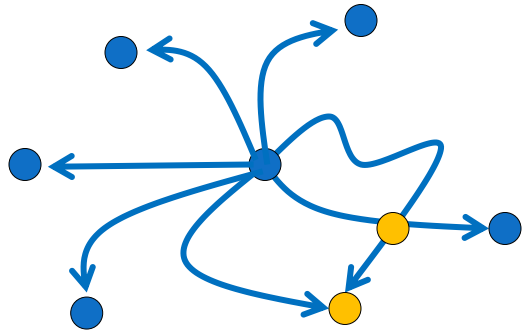
- Testing *expansion* and *bipartiteness* of bounded-degree graphs in *sub-linear time*

	Classical: (Goldreich, Ron, Czumaj, Sohler, …)	Quantum:
Algorithms	$O(\sqrt{N})$	$O(N^{1/3})$ for both problems, using Ambainis' algorithm for element distinctness
Query lower bounds	$\Omega(\sqrt{N})$	$\Omega(N^{1/4})$ for testing expansion, via polynomial method

Classical algorithms for testing bipartiteness and expansion

Classical algorithms for testing bipartiteness and expansion

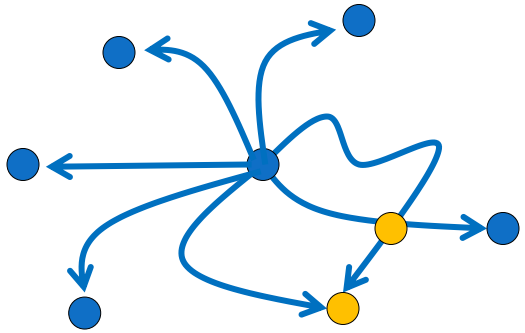
- Use random walks to characterize the structure of the graph (Goldreich and Ron)



Pick a random starting point.
Run $O(\sqrt{N})$ random walks, each of length poly(log N).
Look for *collisions* and *odd-length cycles*

Classical algorithms for testing bipartiteness and expansion

- Use random walks to characterize the structure of the graph (Goldreich and Ron)



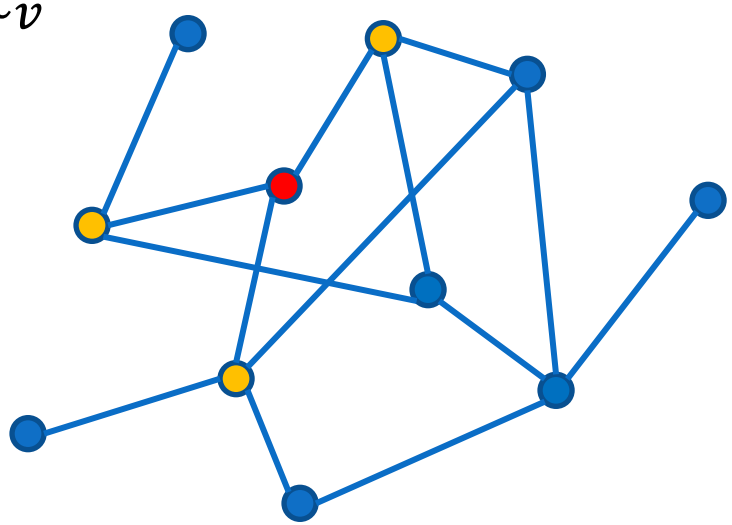
Pick a random starting point.
Run $O(\sqrt{N})$ random walks,
each of length poly(log N).
Look for *collisions* and
odd-length cycles

- G has few odd-length cycles iff G is close to bipartite
- Random walks have few collisions iff G is close to being an expander

How to define a quantum walk?

- Graph G, with N vertices and degree d
- For any vertex v, consider the uniform superposition over the neighbors of v,

$$|D_v\rangle = \frac{1}{\sqrt{d}} \sum_{u \sim v} |u\rangle$$

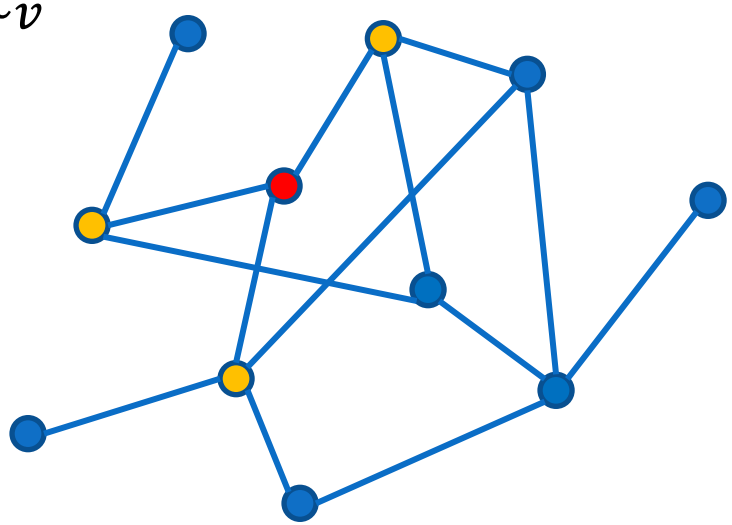


How to define a quantum walk?

- Graph G, with N vertices and degree d
- For any vertex v, consider the uniform superposition over the neighbors of v,

$$|D_v\rangle = \frac{1}{\sqrt{d}} \sum_{u \sim v} |u\rangle$$

- Can we do an operation that maps $|v\rangle$ to $|D_v\rangle$?
  - No, that's not unitary!
  - The states $|D_v\rangle$ are not orthogonal



How to define a quantum walk?

- Graph G, with N vertices and degree d
- For any vertex v, consider the uniform superposition over the neighbors of v,

$$|D_v\rangle = \frac{1}{\sqrt{d}} \sum_{u \sim v} |u\rangle$$

- Instead, use *two* quantum registers, and apply these "mixing" operations:

(Watrous)

$$C_1 = \sum_v |v\rangle\langle v| \times (I - 2|D_v\rangle\langle D_v|)$$

$$C_2 = \sum_v (I - 2|D_v\rangle\langle D_v|) \times |v\rangle\langle v|$$

The eigenvalues

- If the *classical* random walk has eigenvalues λ_v, then the *quantum* walk has eigenvalues

$$\lambda_v \pm i\sqrt{1-\lambda_v{}^2} = e^{\pm i\theta_v}, \quad \theta_v = \arccos(\lambda_v)$$

The eigenvalues

- If the *classical* random walk has eigenvalues λ_v, then the *quantum* walk has eigenvalues

$$\lambda_v \pm i\sqrt{1-\lambda_v{}^2} = e^{\pm i\theta_v}, \quad \theta_v = \arccos(\lambda_v)$$

- When $\lambda_v \approx 1$, we have $\lambda_v \approx 1 - (1/2)\,\theta_v{}^2$, which implies:

$$\theta_v \approx \sqrt{2(1-\lambda_v)}$$

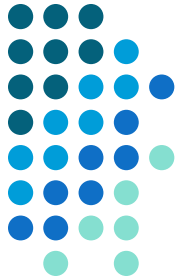
Spectral gap of quantum walk

Spectral gap of classical walk

(Szegedy)

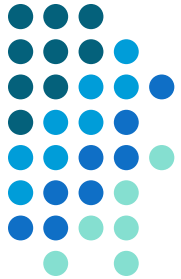
- This amplifies the spectral gap!

Quantum vs. classical walks



	Classical random walk	Quantum walk
Transition matrix is…	Stochastic	Unitary
Eigenvalues are…	Real	Complex w/ abs. value 1, of the form exp(iθ)
It does…	Diffusion	???
Which is…	Irreversible	Reversible
At a rate given by…	Spectral gap	Difference between phase angles θ

Algorithms using quantum walks



Algorithms using quantum walks

- **Searching** for a marked vertex in a graph
  - Get a quadratic quantum speed-up
  - (Can use this for triangle finding!) *(Magniez, Santha, Szegedy)*

Algorithms using quantum walks

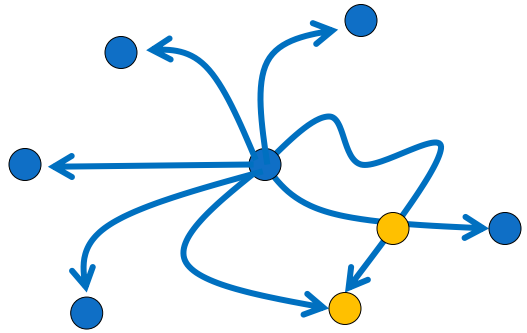
- **Searching** for a marked vertex in a graph
  - Get a quadratic quantum speed-up
  - (Can use this for triangle finding!) *(Magniez, Santha, Szegedy)*
- **Finding collisions** (element distinctness)
  - Function f defined on some set of N elements
  - Find a “collision”: a distinct pair (x,x’) s.t. f(x) = f(x’)
  - Classically, takes time $\Omega(N)$
  - Quantumly, can solve in time $O(N^{2/3})$ *(Ambainis)*

Quantum algorithms for testing bipartiteness and expansion

- 1st attempt: try to **search** through the graph?
  - If G is an expander, then we should be able to search through it quickly, using a quantum walk (Szegedy)
  - If G is far from an expander, then the search should fail. (Does it?)

Quantum algorithms for testing bipartiteness and expansion

- Better idea:
  - Use the classical strategy of Goldreich and Ron:

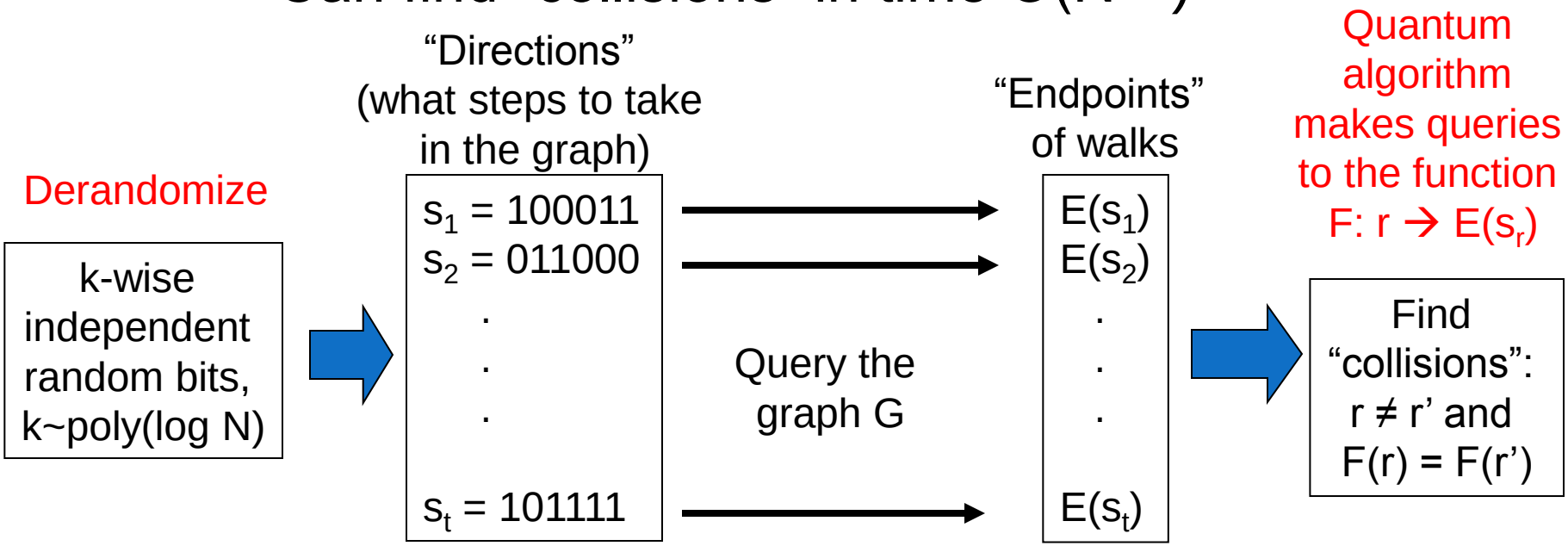


Pick a random starting point.
Run $O(\sqrt{N})$ random walks,
each of length poly(log N).
Look for *collisions* and
odd-length cycles

  - Use Ambainis' quantum algorithm to **find collisions**
    - Running time: $O(N^{1/2}) \rightarrow O(N^{1/3})$

Quantum algorithms for testing bipartiteness and expansion

- Consider t random walks $s_1,\ldots,s_t$, where $t\sim\sqrt{N}$
- Derandomize, using k-wise independent rv's
- => Can find "collisions" in time $O(N^{1/3})$

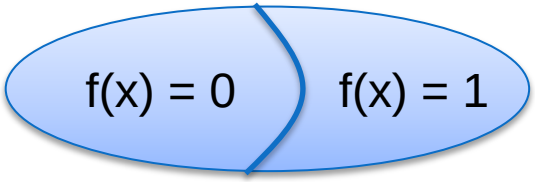


Exponential quantum speedup?

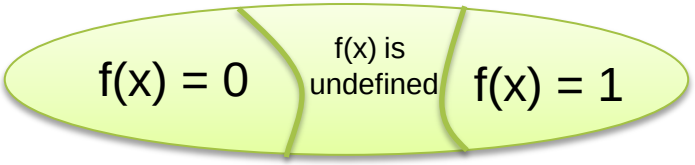
- Lower-bound the number of oracle queries?

Exponential quantum speedup?

- Lower-bound the number of oracle queries?
  - For computing total functions:
    best possible speedup is polynomial (Beals et al.)



  - For property testing: no such obstacle;
    exponential speedups are possible! (Buhrman et al.)



Exponential quantum speedup?

- Bad news:
  Testing graph expansion requires $\Omega(N^{1/4})$ queries
- Proof: polynomial method

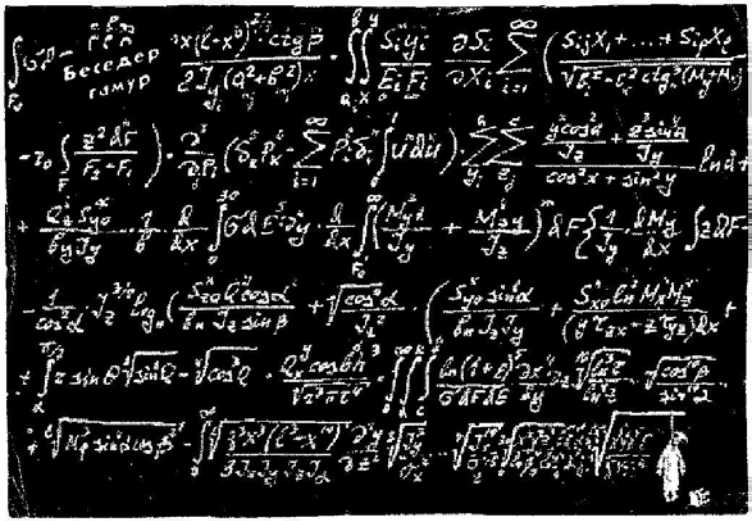


Image: http://www.math.missouri.edu/~rudelson/papers/publications.html

Exponential quantum speedup?

- Bad news:
  Testing graph expansion requires $\Omega(N^{1/4})$ queries
- Proof: polynomial method

Note about proof techniques:

- Standard q. adversary method:
  "property testing barrier"
- Polynomial method:
  hard to apply to graph problems

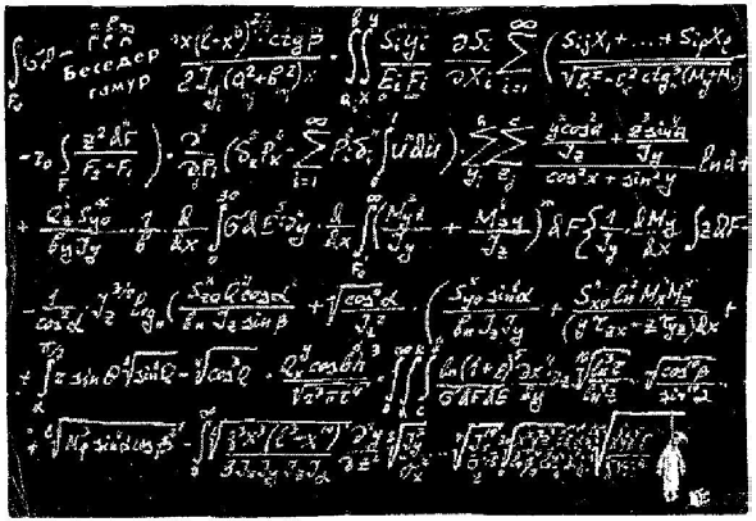
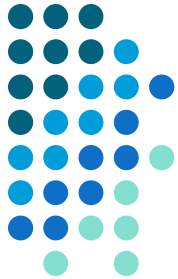


Image: http://www.math.missouri.edu/~rudelson/papers/publications.html

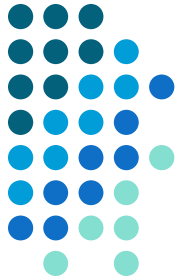
Quantum lower bound



- Construct a random graph G as follows:
  - *M vertices*, divided equally into *L components*
  - On each component, construct a constant number of random perfect matchings, and take their union

  - L = 1 => G is an expander
  - L ≥ 2 => G is far from being an expander

Quantum lower bound



- Take any quantum algorithm Q, that makes *T queries* and correctly tests expansion
  - Key lemma: Pr_G [Q(G) accepts] ≈ f(M,L), where f is a bivariate polynomial, deg(f) = O(T log T)
  - Know f(M,1) ≥ 2/3, f(M,2) ≤ 1/3, |f(M,L)| ≤ 1 => lower bound on deg(f) (Aaronson, Shi)
  - (Omitting some details!)

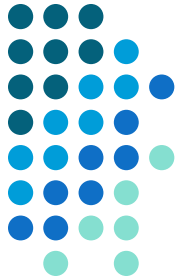
The messy details

- Dependencies among vertices in the graph G
  - Partition the M vertices into L components
    - Vertices u,v end up in same component =>
      less likely that u,w end up in same component
  - Solution: re-sample the graph
    - Take G, and choose a random subgraph of size N
    - Repeat N times:
      choose one component at random (iid),
      then sample one vertex from that component
      (w/o replacement)

The messy details

- Key lemma: Pr_G [Q(G) accepts] ≈ f(M,L), a bivariate polynomial of degree O(T log T)
  - Edges in the graph G are described by a set of *Boolean* variables x_{uvj} (u,v in [N], j in [d])
  - Pr_G [Q(G) accepts] is a multivariate polynomial P in the x_{uvj}, of degree 2T
  - Each term in P corresponds to a subset of edges; this defines a graph H
  - Each component in H corresponds to a monomial X in x_{uvj}
  - Pr_G [X=1] is a simple *rational function* of M and L
  - Pr_G [Q(G) accepts] is a *sum of rational functions* of M and L

The messy details



- Key lemma: Pr_G [Q(G) accepts] ≈ f(M,L), a bivariate polynomial of degree O(T log T)
  - Pr_G [Q(G) accepts] is a *sum of rational functions* of M and L
  - *Magic*: most of the stuff in the denominators gets cancelled out by the numerators
    - Combinatorics…
  - What's left can be approximated away
    - Provided M in $[aN - \delta^{3/2}, aN + \delta^{3/2}]$, and L in $[0,\delta]$
    - where $\delta = O(\sqrt{N})$, $a \approx 1$
  - Implies a lower bound: $\deg(f) = \Omega(\sqrt{\delta}) = \Omega(N^{1/4})$
    - So algorithm needs $T = \Omega(N^{1/4} / \log N)$ queries!

This talk

- Testing *expansion* and *bipartiteness* of bounded-degree graphs in *sub-linear time*

	Classical: (Goldreich, Ron, Czumaj, Sohler, …)	Quantum:
Algorithms	$O(\sqrt{N})$	$O(N^{1/3})$ for both problems, using Ambainis' algorithm for element distinctness
Query lower bounds	$\Omega(\sqrt{N})$	$\Omega(N^{1/4})$ for testing expansion, via polynomial method

Outlook

- Quantum algorithms can make use of ideas from classical algorithms

Quantum walks	Random walks
Quantum adiabatic algorithm	Simulated annealing, go-with-the-winners heuristic

- Query lower bounds can rule out the possibility of exponential quantum speedups
  - Total functions: No
  - Property testing: Sometimes