The computational power of quantum walk

Andrew Childs

Department of Combinatorics & Optimization
and Institute for Quantum Computing

University of Waterloo

Why quantum computing?

Fast algorithms for classically hard problems

Fast algorithms for classically hard problems

Fast algorithms for classically hard problems

3107418240490043721350750035888567
9300373460228427275457201619488232
0644051808150455634682967172328678
2437916272838033415471073108501919
5485290073377248227835257423864540
14691736602477652346609

Fast algorithms for classically hard problems

3107418240490043721350750035888567
9300373460228427275457201619488232
0644051808150455634682967172328678
2437916272838033415471073108501919
5485290073377248227835257423864540
14691736602477652346609

=

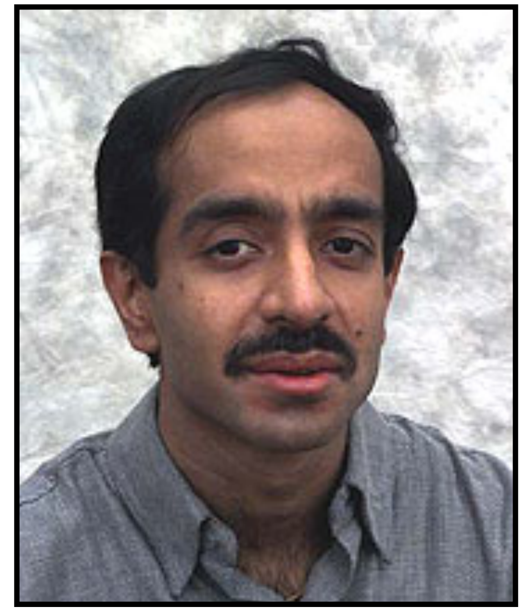
1634733645809253848443133883865090
8598417836700330923121811108523893
33100104508151212118167511579

×

1900871281664822113126851573935413
9754718967899685154936666385390880
271038021044989571912614655711

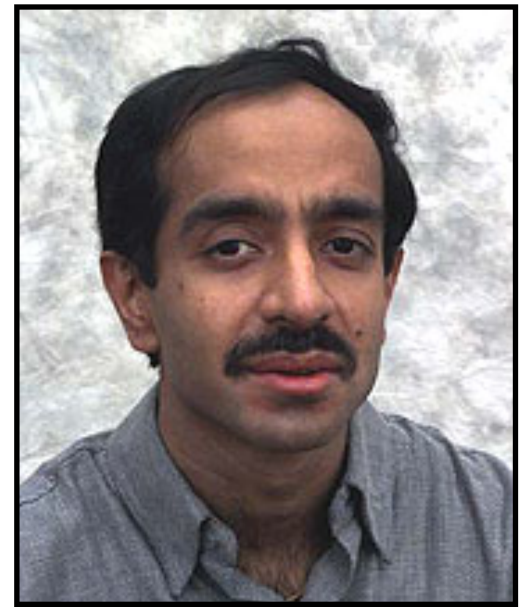
Fast algorithms for classically hard problems

3107418240490043721350750035888567
9300373460228427275457201619488232
0644051808150455634682967172328678
2437916272838033415471073108501919
5485290073377248227835257423864540
14691736602477652346609
=
1634733645809253848443133883865090
8598417836700330923121811108523893
33100104508151212118167511579
×
1900871281664822113126851573935413
9754718967899685154936666385390880
27103802104498957191261465571



Fast algorithms for classically hard problems

3107418240490043721350750035888567
9300373460228427275457201619488232
0644051808150455634682967172328678
2437916272838033415471073108501919
5485290073377248227835257423864540
14691736602477652346609
=
1634733645809253848443133883865090
859841783670033092312181110852389
3331001045081512121181675115790
×
1900871281664822113126851573935413
975471896789968515493666638539088
0271038021044989571912614655710



000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000001000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000

Fast algorithms for classically hard problems

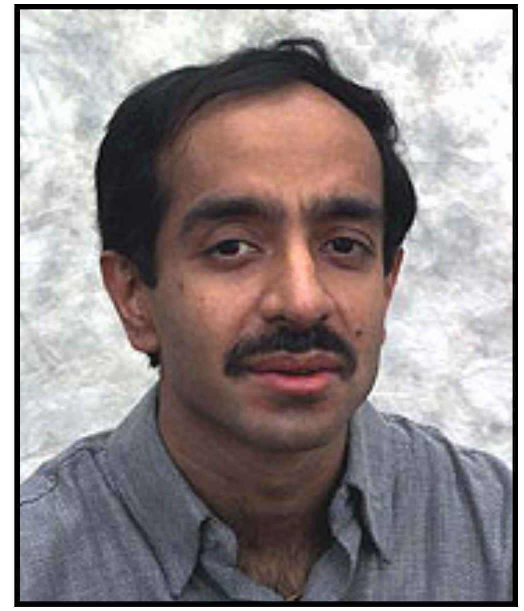
3107418240490043721350750035888567
9300373460228427275457201619488232
0644051808150455634682967172328678
2437916272838033415471073108501919
5485290073377248227835257423864540
146917366024776523466 09

=

1634733645809253848443133883865090
85984178367003309231218111085238
9333100104508151212118167511579

×

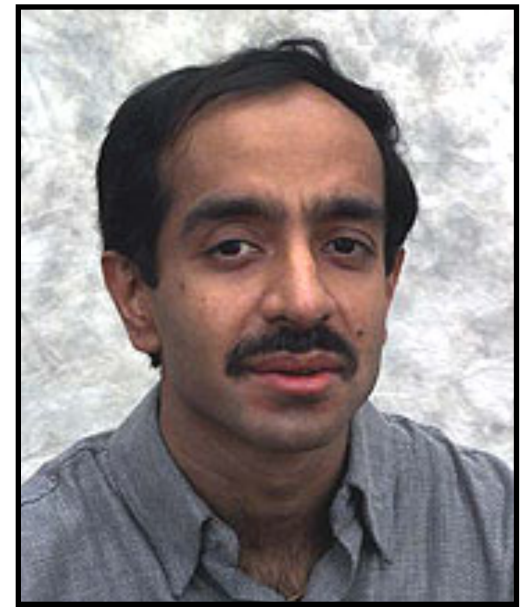
1900871281664822113126851573935413
975471896789968515493666638539088
027103802104498957191261465571



000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000001000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000

Fast algorithms for classically hard problems

3107418240490043721350750035888567
9300373460228427275457201619488232
0644051808150455634682967172328678
2437916272838033415471073108501919
5485290073377248227835257423864540
1469173660247765234609
=
1634733645809253848443133883865090
859841783670033092312181110852389
3331001045081512121181675115790
×
1900871281664822113126851573935413
9754718967899685154936666385390880
2710380210449895719126146557



000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000001000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000

- Computing discrete logarithms
- Decomposing Abelian groups
- Computations in number fields
- Approximating Gauss sums
- Shifted Legendre symbol
- Counting points on algebraic curves
- Approximating the Jones polynomial (and other topological invariants)
- Simulating quantum systems
- Linear systems
- Computing effective resistance
- …

- Formula evaluation
- Collision finding (k-distinctness, k-sum, etc.)
- Minimum spanning tree, connectivity, shortest paths, bipartiteness of graphs
- Network flows, maximal matchings
- Finding subgraphs
- Minor-closed graph properties
- Property testing (distance between distributions, bipartiteness/expansion of graphs, etc.)
- Checking matrix multiplication
- Group commutativity
- Subset sum
- …

Post-quantum cryptography

Much of the cryptography in use today (e.g., RSA, elliptic curves) could be broken by a quantum computer

Post-quantum cryptography

Much of the cryptography in use today (e.g., RSA, elliptic curves) could be broken by a quantum computer

One possible reaction:

Post-quantum cryptography

Much of the cryptography in use today (e.g., RSA, elliptic curves) could be broken by a quantum computer

One possible reaction:

Another reaction: Try to understand what quantum computers are good at so we can design cryptosystems they can't break

What can be computed efficiently?

What can be computed efficiently?

Nature is described by quantum mechanics, so to fully understand what can be computed in the real world, we have to understand the implications of quantum mechanics for computation.

What can be computed efficiently?

Nature is described by quantum mechanics, so to fully understand what can be computed in the real world, we have to understand the implications of quantum mechanics for computation.

Apparently nature can efficiently solve problems that a classical computer cannot.

What can be computed efficiently?

Nature is described by quantum mechanics, so to fully understand what can be computed in the real world, we have to understand the implications of quantum mechanics for computation.

Apparently nature can efficiently solve problems that a classical computer cannot.

Only two alternatives:

What can be computed efficiently?

Nature is described by quantum mechanics, so to fully understand what can be computed in the real world, we have to understand the implications of quantum mechanics for computation.

Apparently nature can efficiently solve problems that a classical computer cannot.

Only two alternatives:

- Classical computers can efficiently simulate quantum ones, or

What can be computed efficiently?

Nature is described by quantum mechanics, so to fully understand what can be computed in the real world, we have to understand the implications of quantum mechanics for computation.

Apparently nature can efficiently solve problems that a classical computer cannot.

Only two alternatives:

- Classical computers can efficiently simulate quantum ones, or
- Quantum mechanics is not a good description of nature

What can be computed efficiently?

Nature is described by quantum mechanics, so to fully understand what can be computed in the real world, we have to understand the implications of quantum mechanics for computation.

Apparently nature can efficiently solve problems that a classical computer cannot.

Only two alternatives:

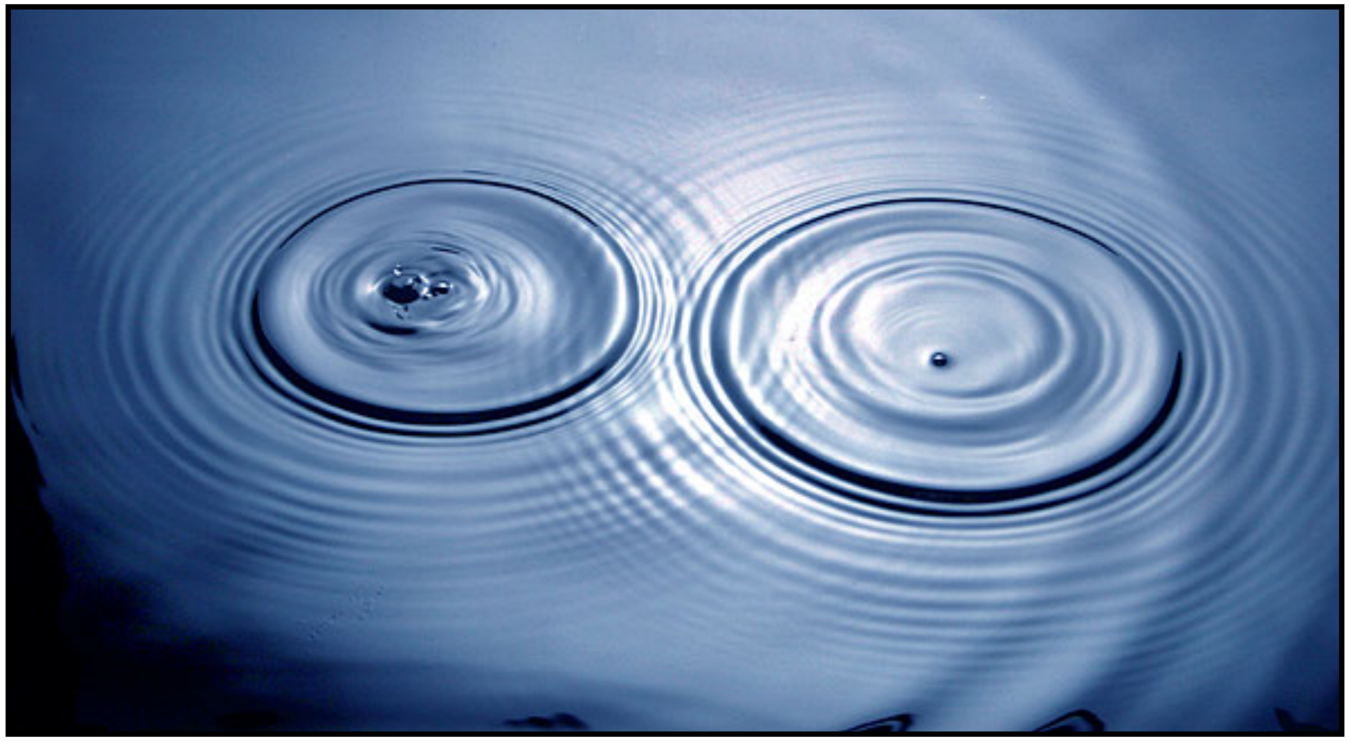
- Classical computers can efficiently simulate quantum ones, or
- Quantum mechanics is not a good description of nature

Main goal of my research: Understand the advantages of quantum over classical computation

The origin of quantum speedup

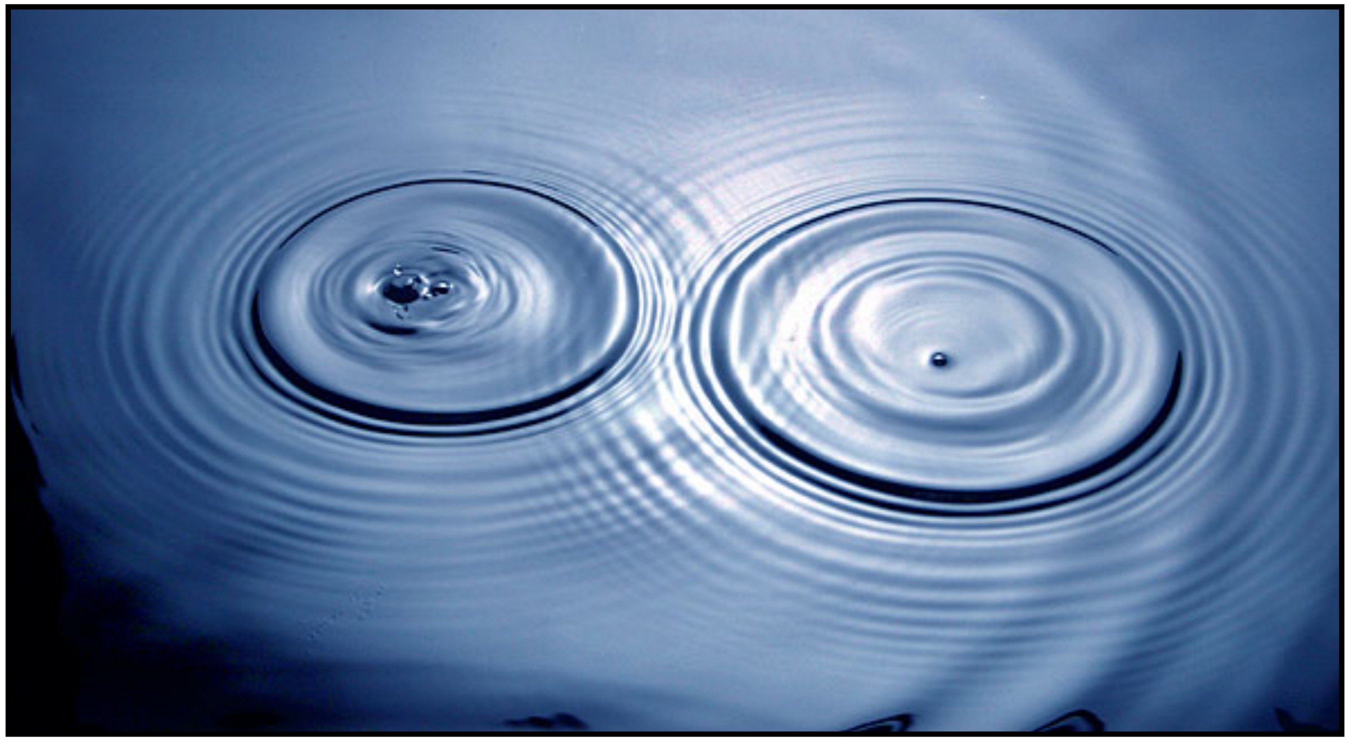
The origin of quantum speedup

Interference between computational paths



The origin of quantum speedup

Interference between computational paths

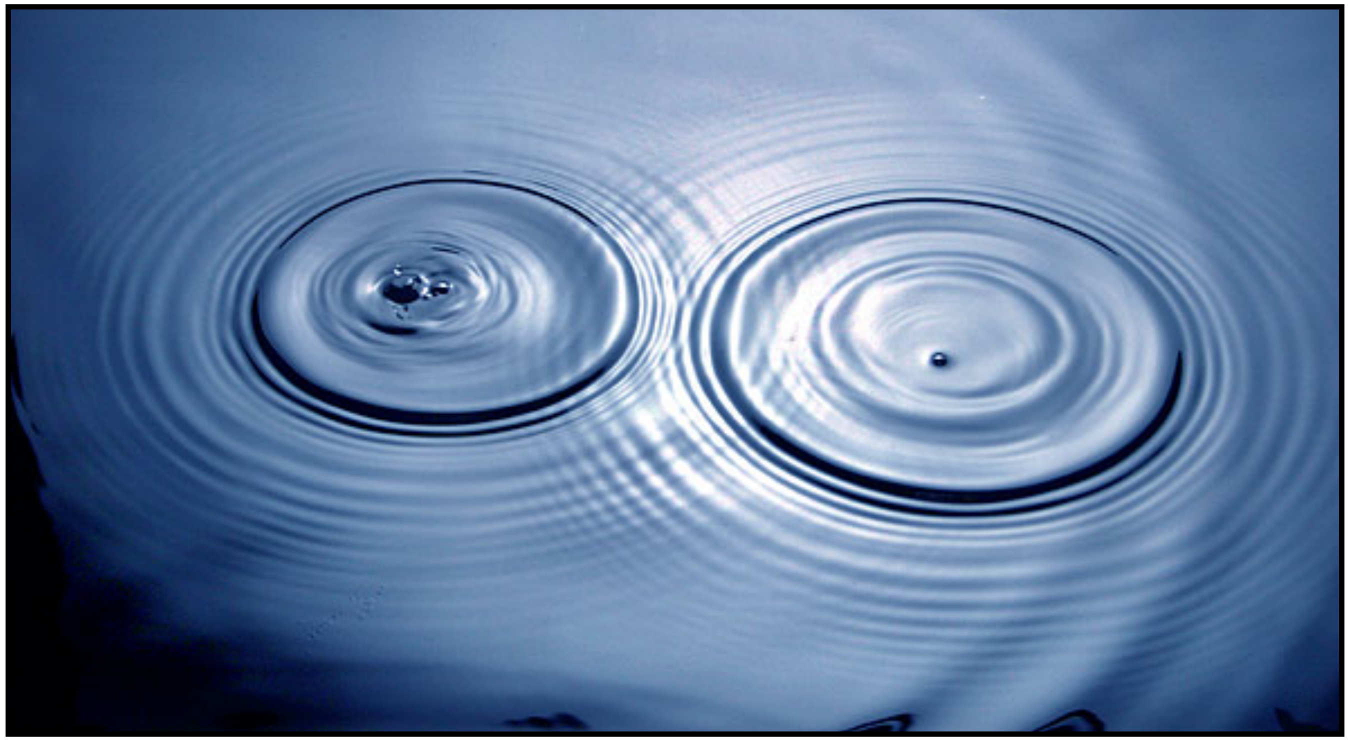


Arrange so that

- paths to the solution interfere constructively
- paths to non-solutions interfere destructively

The origin of quantum speedup

Interference between computational paths



Arrange so that
- paths to the solution interfere constructively
- paths to non-solutions interfere destructively

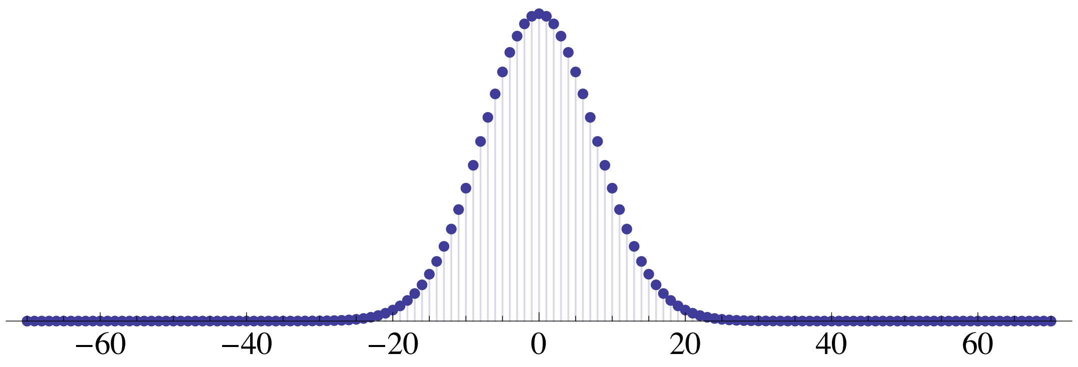
Quantum mechanics gives an efficient representation of complex interference phenomena

Quantum walk

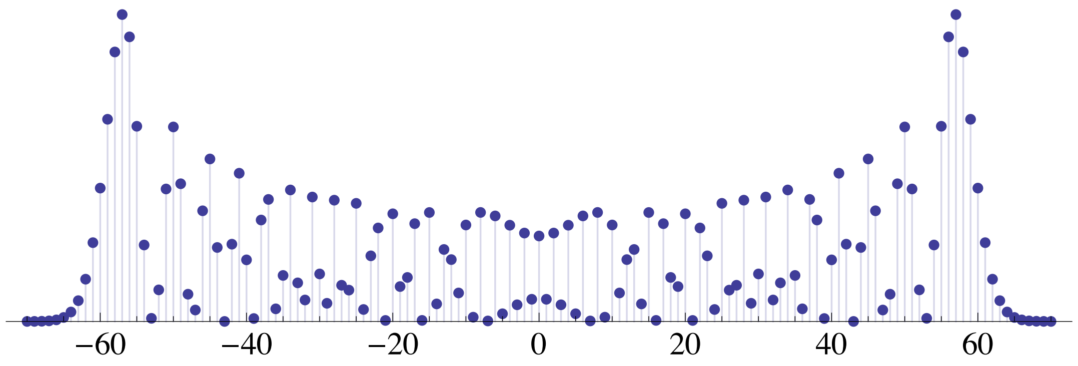
Quantum walk

Quantum analog of a random walk on a graph.

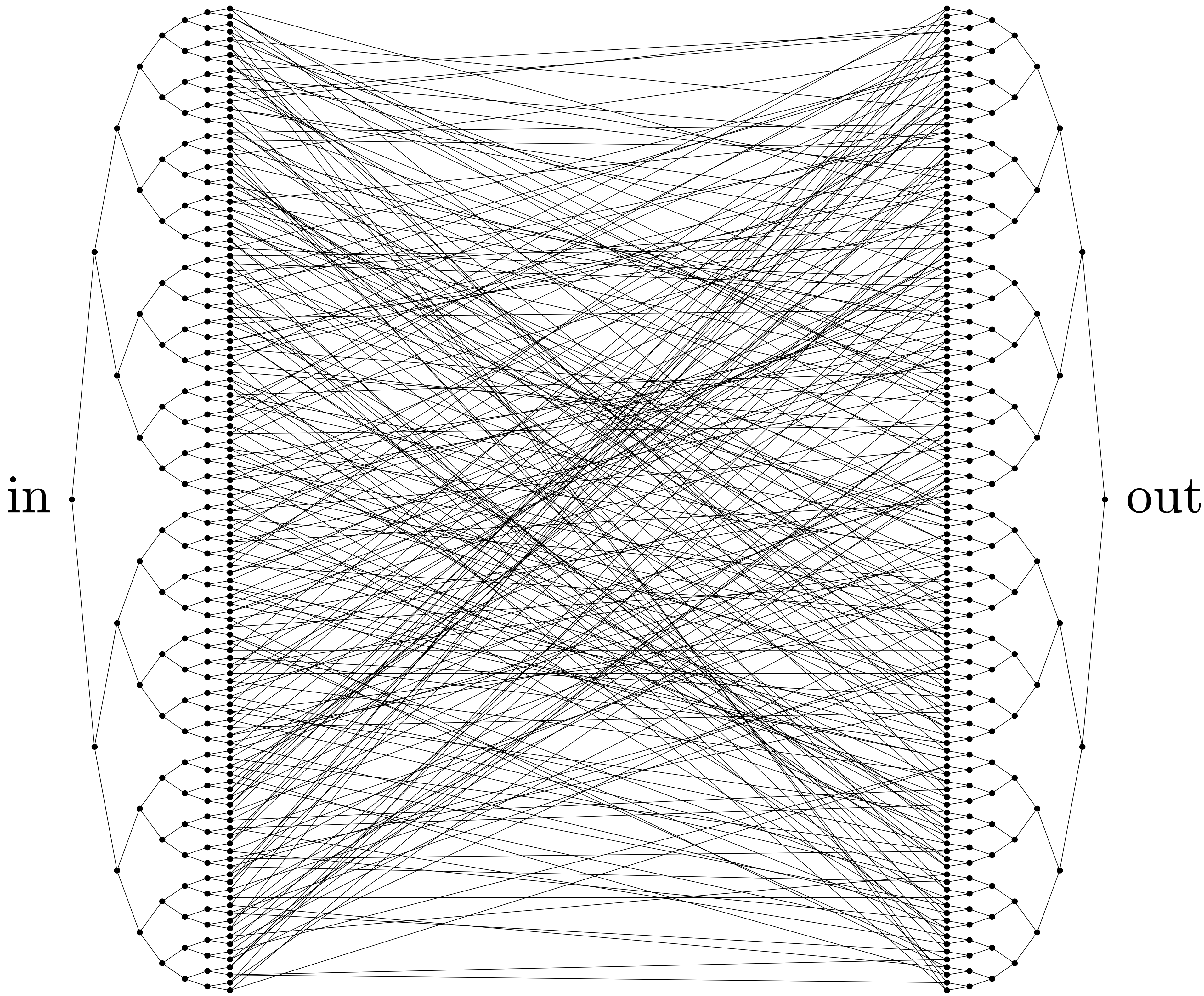
Idea: Replace probabilities by quantum amplitudes.
Interference can produce radically different behavior!



classical



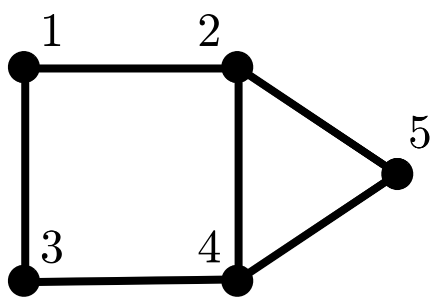
quantum



[Childs, Cleve, Deotto, Farhi, Gutmann, Spielman, STOC 2003]

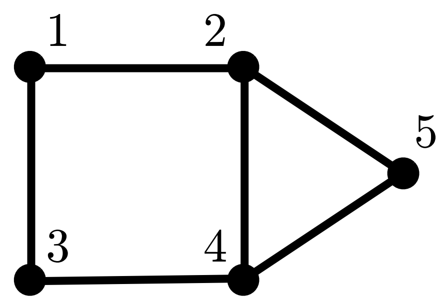
From random walk to quantum walk

Graph G:



From random walk to quantum walk

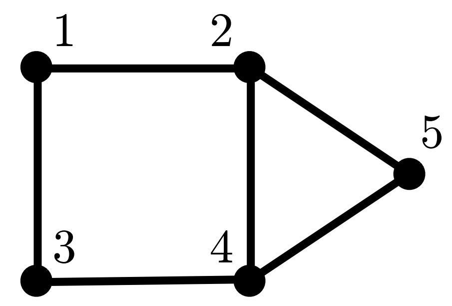
Graph G:



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

adjacency matrix

From random walk to quantum walk

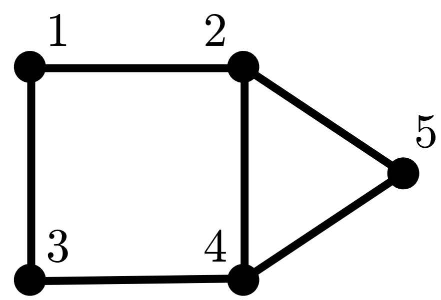
Graph G: 

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 \\ -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

adjacency matrix *Laplacian*

From random walk to quantum walk

Graph G:



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 \\ -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

adjacency matrix *Laplacian*

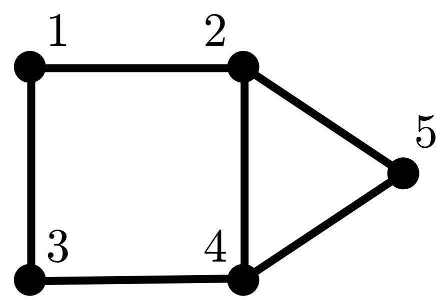
Random walk on G

State: Probability $p_v(t)$ of being at vertex v at time t

Dynamics: $\frac{\mathrm{d}}{\mathrm{d}t}\vec{p} = L\vec{p}$

From random walk to quantum walk

Graph G:



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 \\ -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

adjacency matrix *Laplacian*

Random walk on G

State: Probability $p_v(t)$ of being at vertex v at time t

Dynamics: $$\frac{\mathrm{d}}{\mathrm{d}t}\vec{p} = L\vec{p}$$

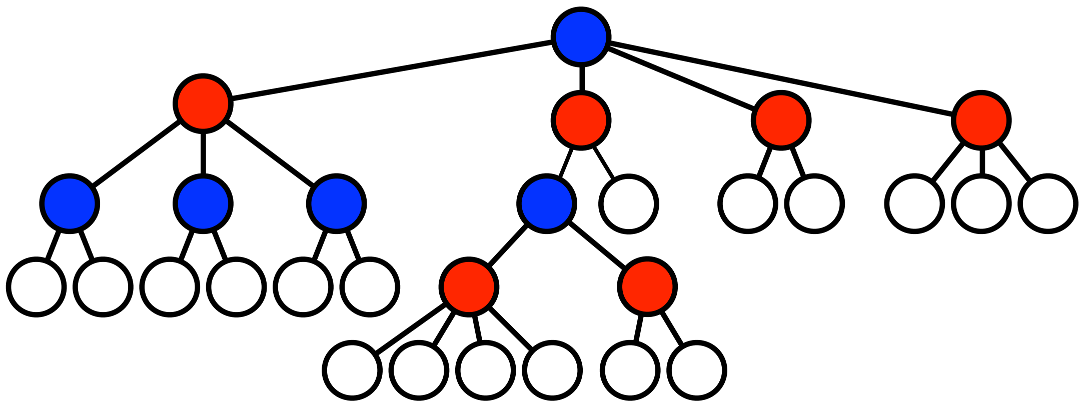
Quantum walk on G

State: Amplitude $a_v(t)$ to be at vertex v at time t

Dynamics: $$i\frac{\mathrm{d}}{\mathrm{d}t}\vec{a} = L\vec{a}$$

Outline

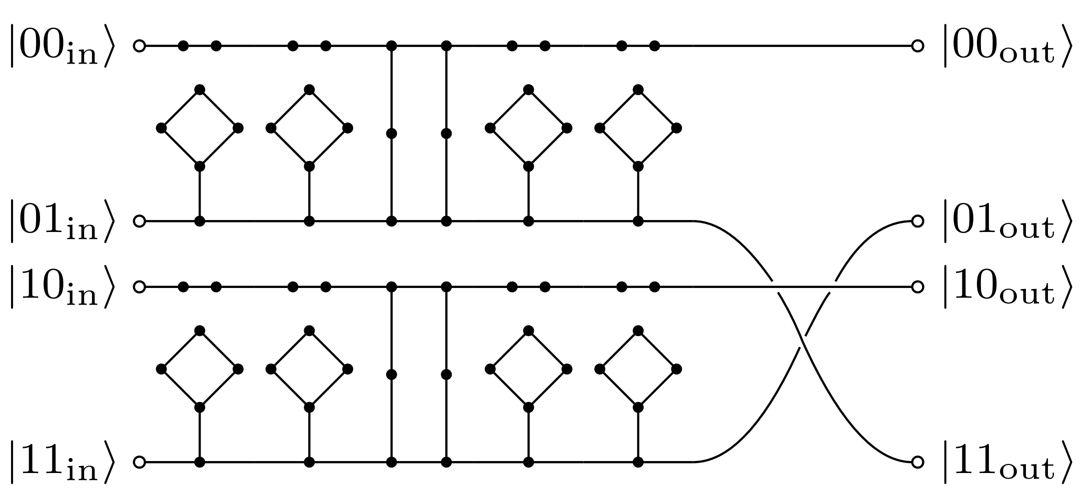
Formula evaluation



Quantum simulation

$$i \frac{\mathrm{d}}{\mathrm{d}t} \psi(t) = H\, \psi(t)$$

Universal computation

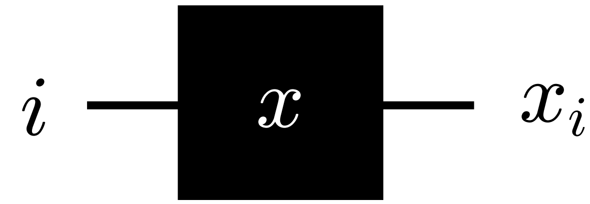


Formula evaluation

- Ambainis, Childs, Reichardt, Špalek, and Zhang, FOCS 2007, pp. 363–372; SIAM Journal on Computing **39**, 2513–2530 (2010)

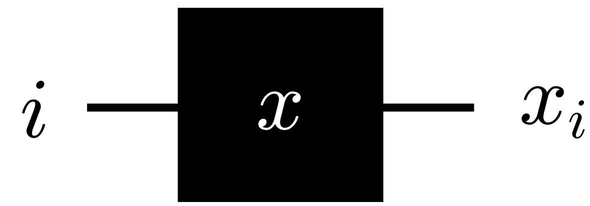
Query complexity of formula evaluation

Query model: given a black box for a string $x \in \{0,1\}^n$



Query complexity of formula evaluation

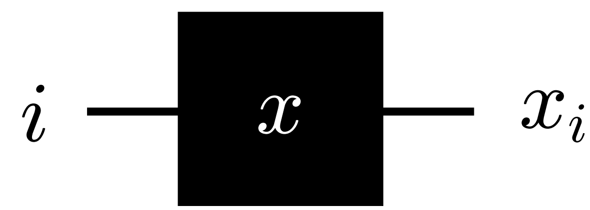
Query model: given a black box for a string $x \in \{0,1\}^n$



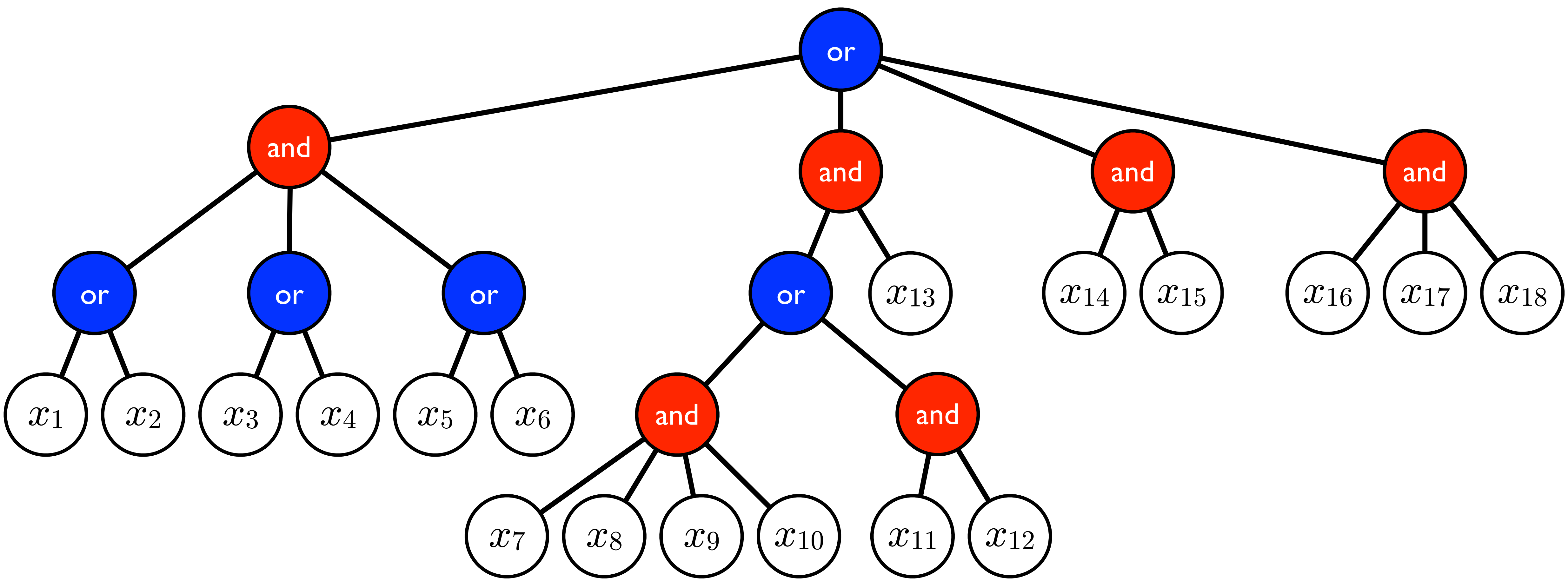
Compute some function of x using as few queries as possible

Query complexity of formula evaluation

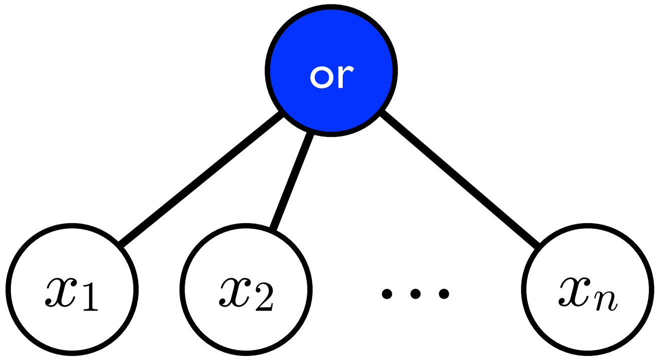
Query model: given a black box for a string $x \in \{0,1\}^n$



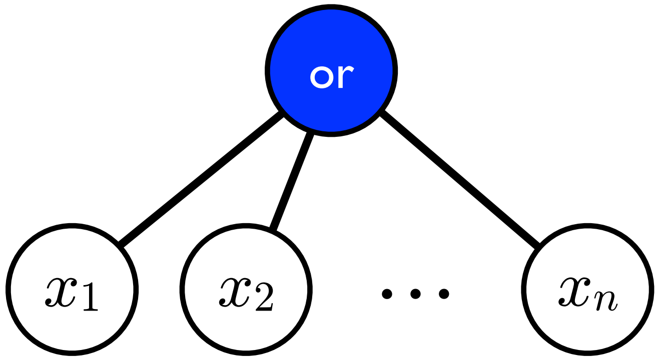
Compute some function of x using as few queries as possible



A single OR gate

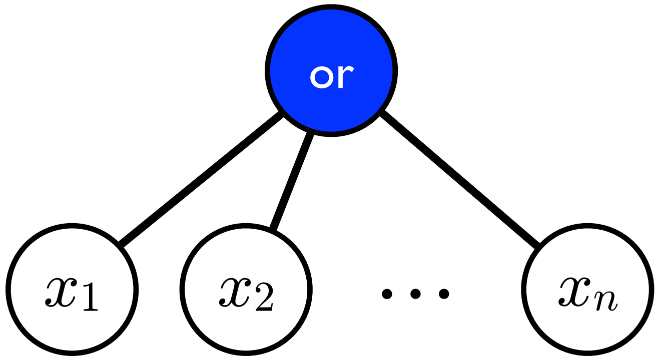


A single OR gate



Classical complexity: $\Theta(n)$

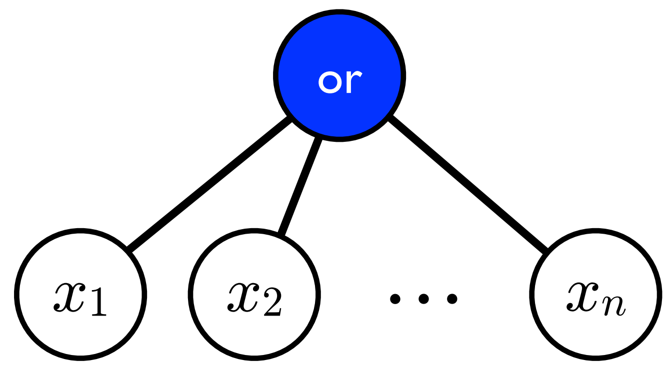
A single OR gate



Classical complexity: $\Theta(n)$

Quantum algorithm [Grover 1996]: $O(\sqrt{n})$

A single OR gate

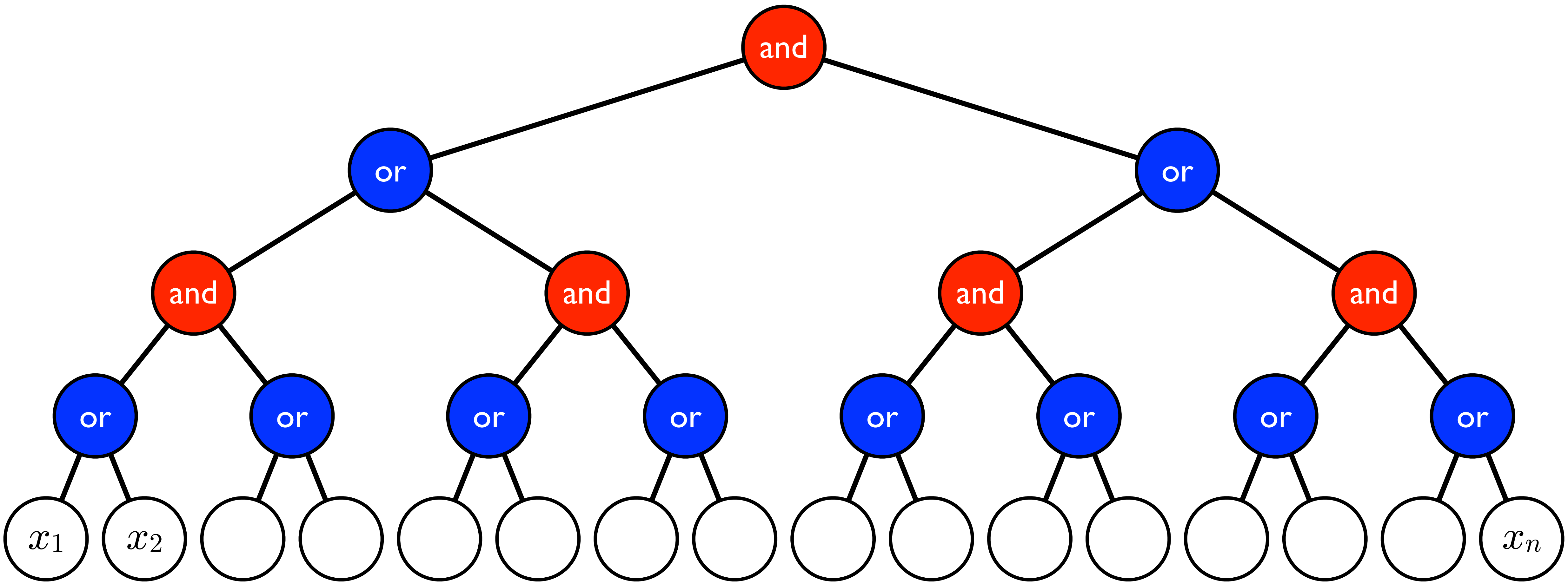


Classical complexity: $\Theta(n)$

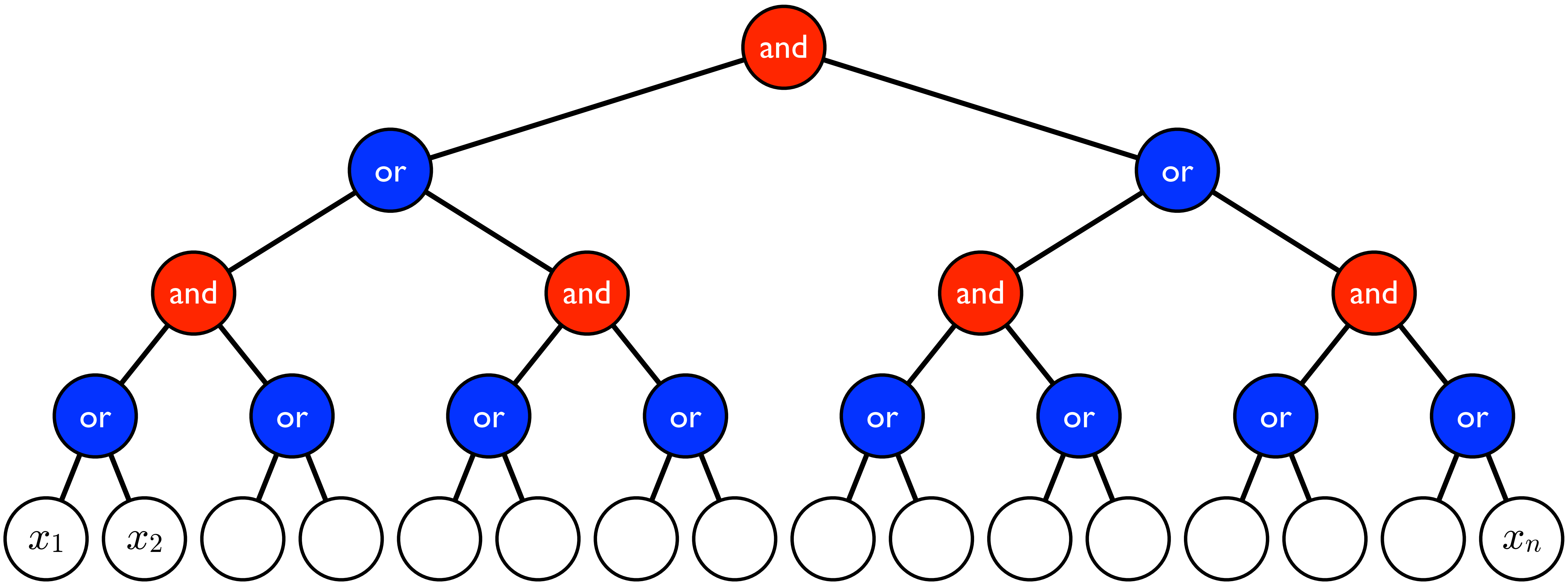
Quantum algorithm [Grover 1996]: $O(\sqrt{n})$

Quantum lower bound [BBBV 1996]: $\Omega(\sqrt{n})$

Balanced binary AND-OR trees

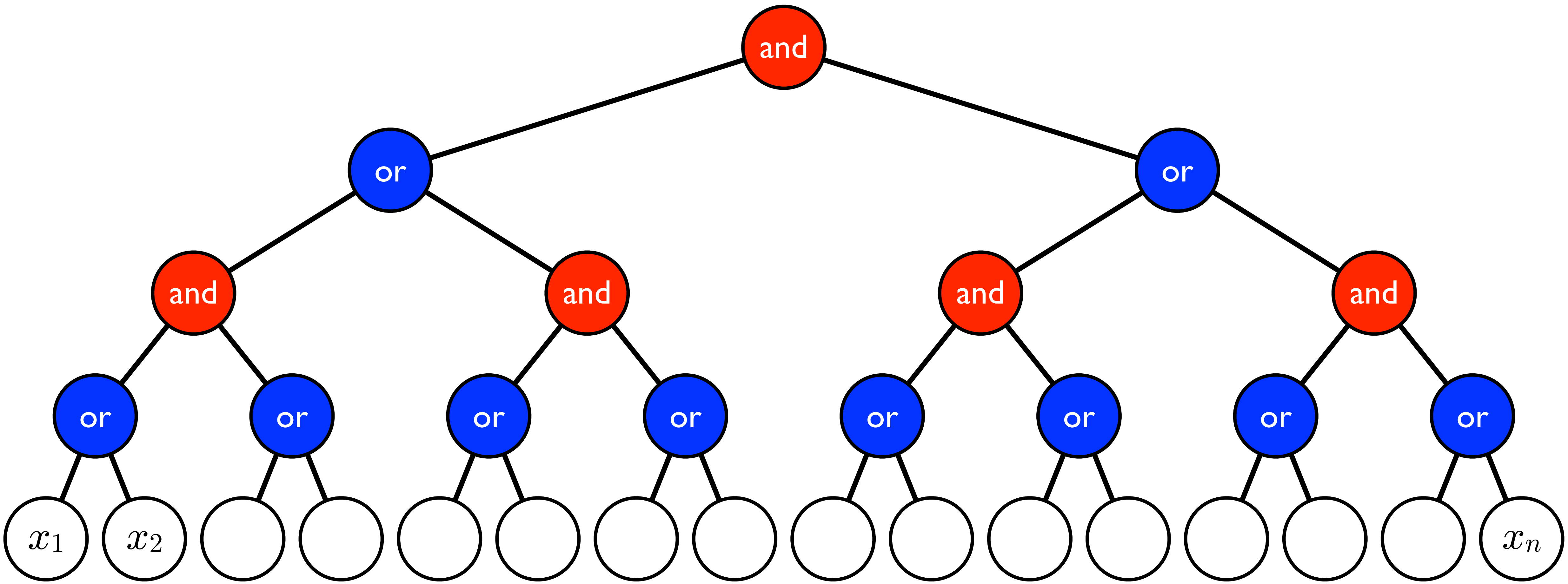


Balanced binary AND-OR trees



Classical complexity [Snir 85; Saks, Wigderson 86; Santha 95]: $\Theta(n^{0.753})$

Balanced binary AND-OR trees



Classical complexity [Snir 85; Saks, Wigderson 86; Santha 95]: $\Theta(n^{0.753})$

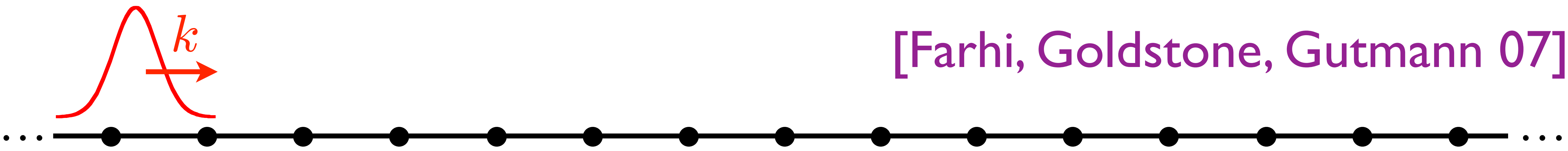
Quantum lower bound [Barnum, Saks 02]: $\Omega(\sqrt{n})$
(holds for arbitrary AND-OR formulas)

Formula evaluation by scattering

[Farhi, Goldstone, Gutmann 07]

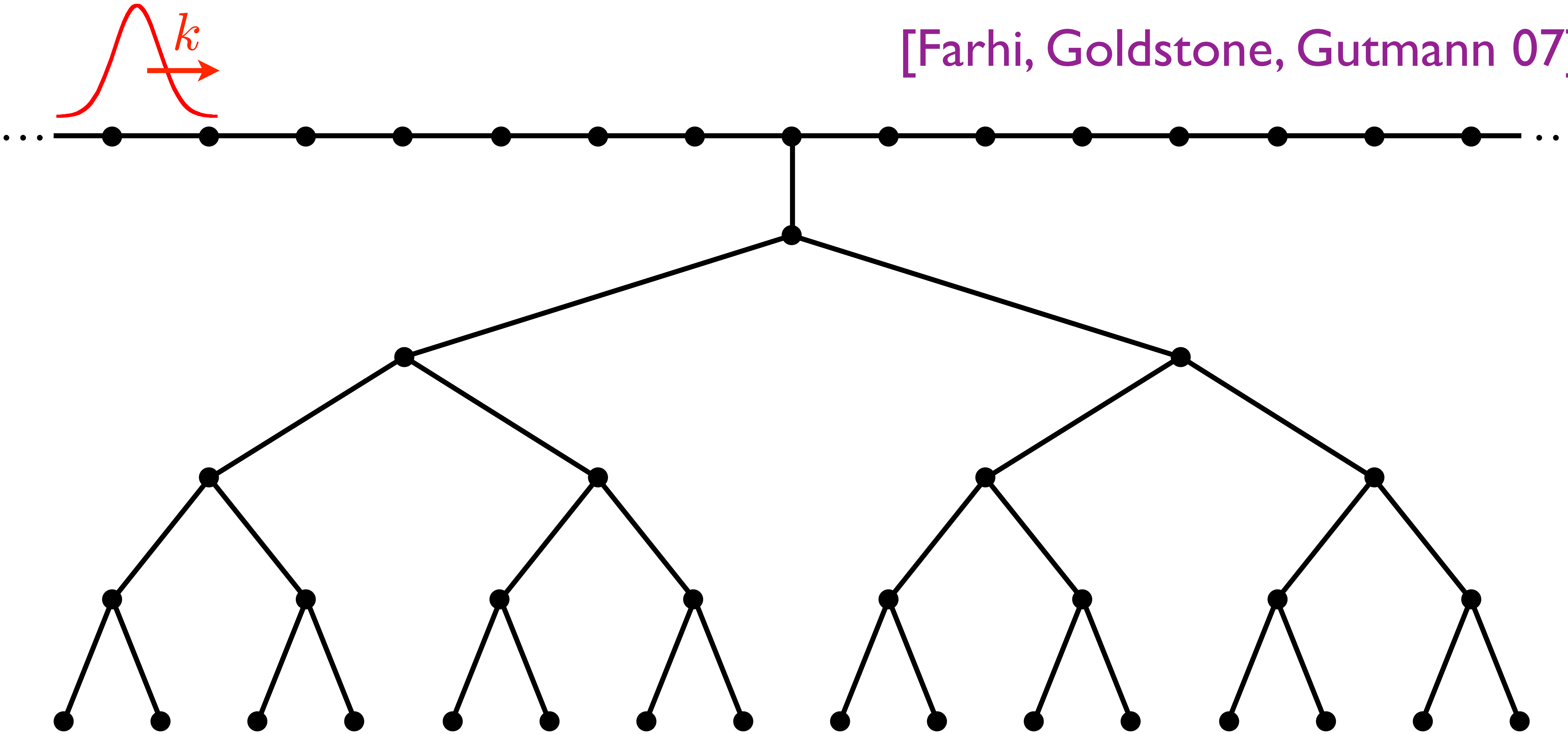
Formula evaluation by scattering

[Farhi, Goldstone, Gutmann 07]



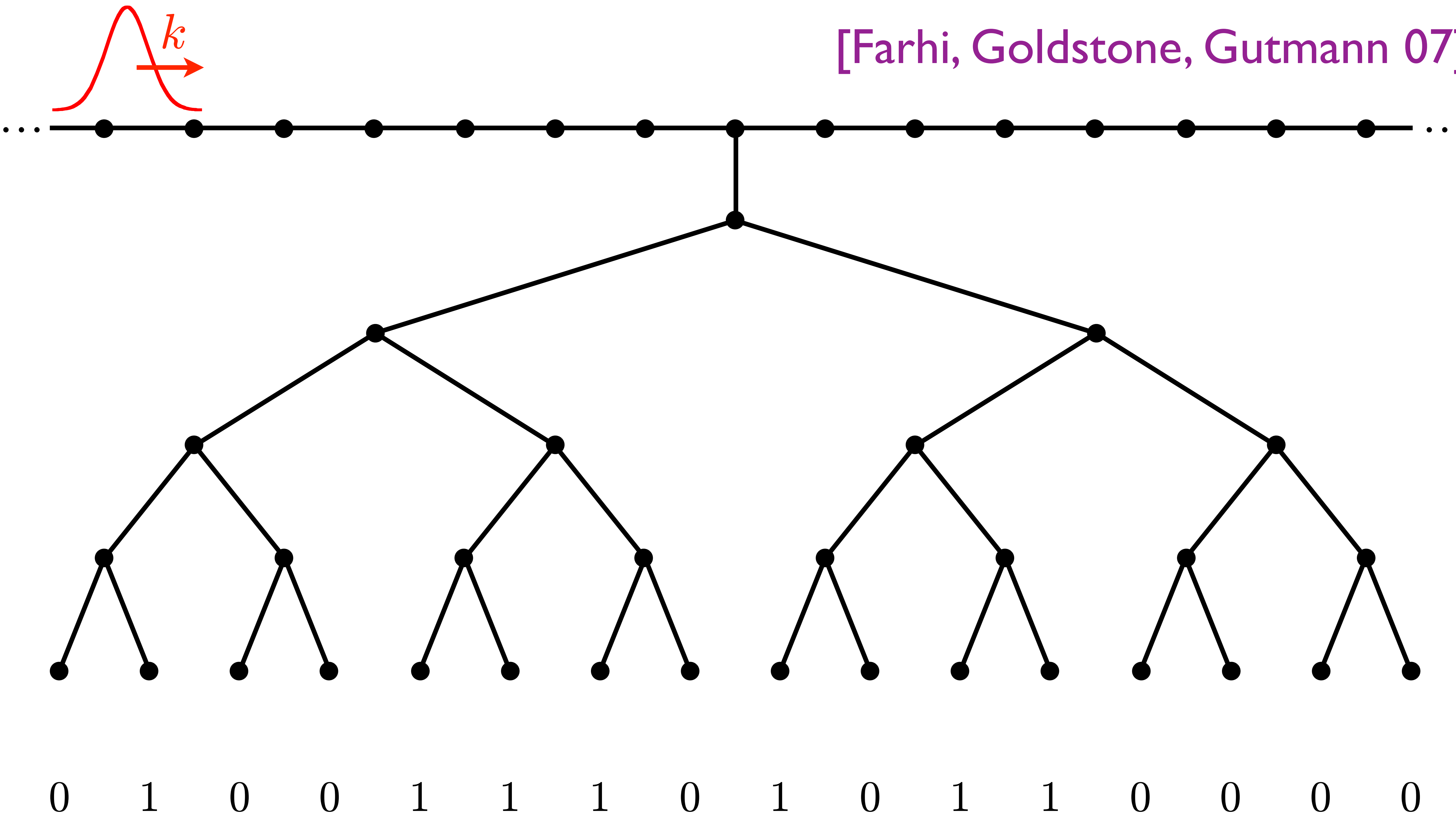
Formula evaluation by scattering

[Farhi, Goldstone, Gutmann 07]



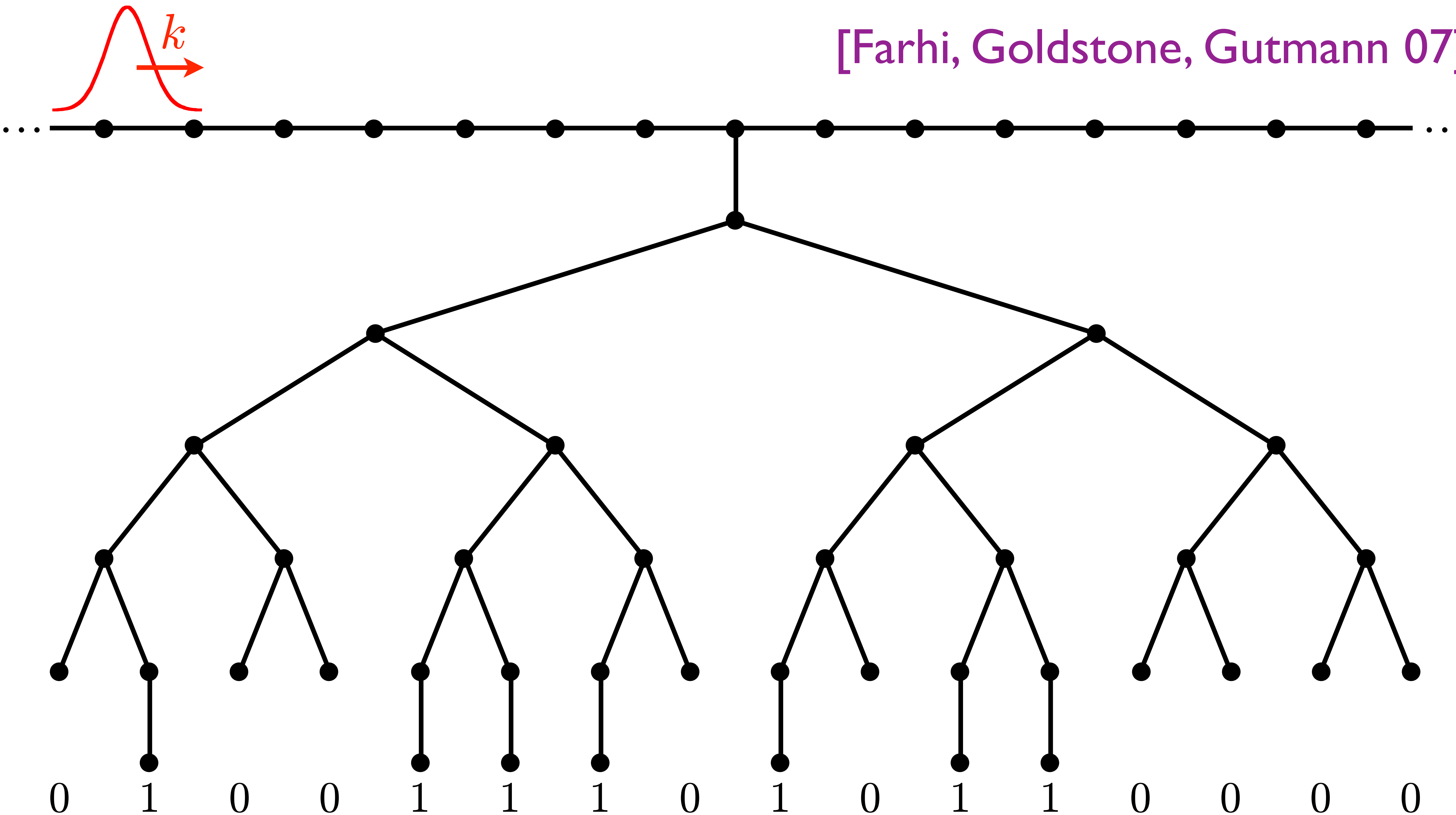
Formula evaluation by scattering

[Farhi, Goldstone, Gutmann 07]



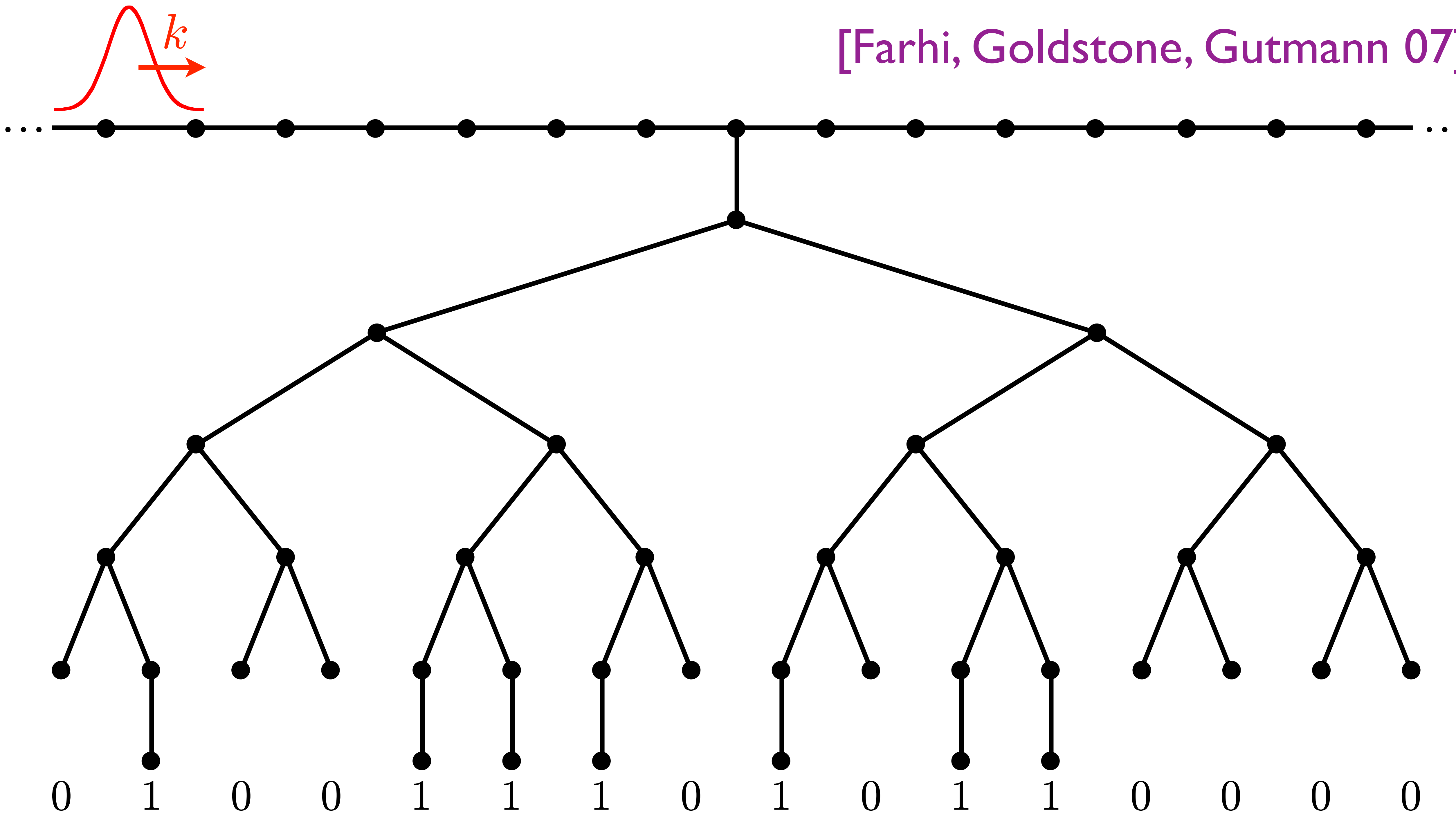
Formula evaluation by scattering

[Farhi, Goldstone, Gutmann 07]



Formula evaluation by scattering

[Farhi, Goldstone, Gutmann 07]



Claim: For small k, the wave is transmitted if the formula (translated into NAND gates) evaluates to 0, and reflected if it evaluates to 1.

General formulas

This simple strategy does not work for general formulas.

General formulas

This simple strategy does not work for general formulas.

To get a general algorithm:

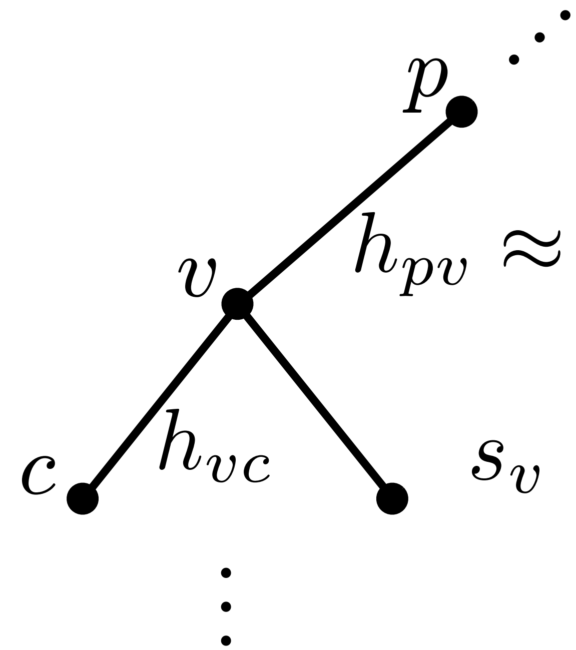
- Rewrite the formula to be "approximately balanced"
- Assign weights to the edges of the tree
- Show that eigenvectors are related to the function value

General formulas

This simple strategy does not work for general formulas.

To get a general algorithm:

- Rewrite the formula to be "approximately balanced"
- Assign weights to the edges of the tree
- Show that eigenvectors are related to the function value



$$h_{pv} \approx \sqrt[4]{\frac{s_v}{s_p}}$$

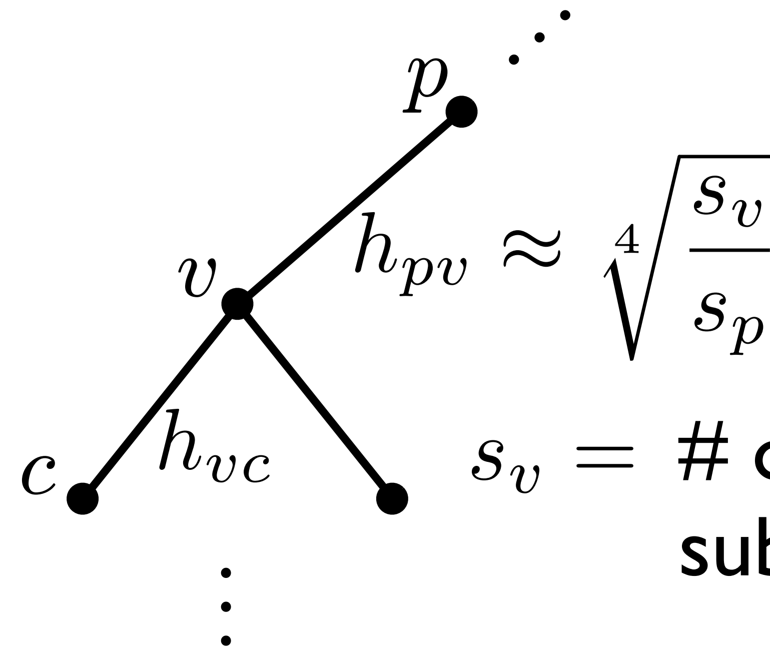
$s_v =$ # of inputs in subformula under v

General formulas

This simple strategy does not work for general formulas.

To get a general algorithm:

- Rewrite the formula to be "approximately balanced"
- Assign weights to the edges of the tree
- Show that eigenvectors are related to the function value



$s_v =$ # of inputs in subformula under v

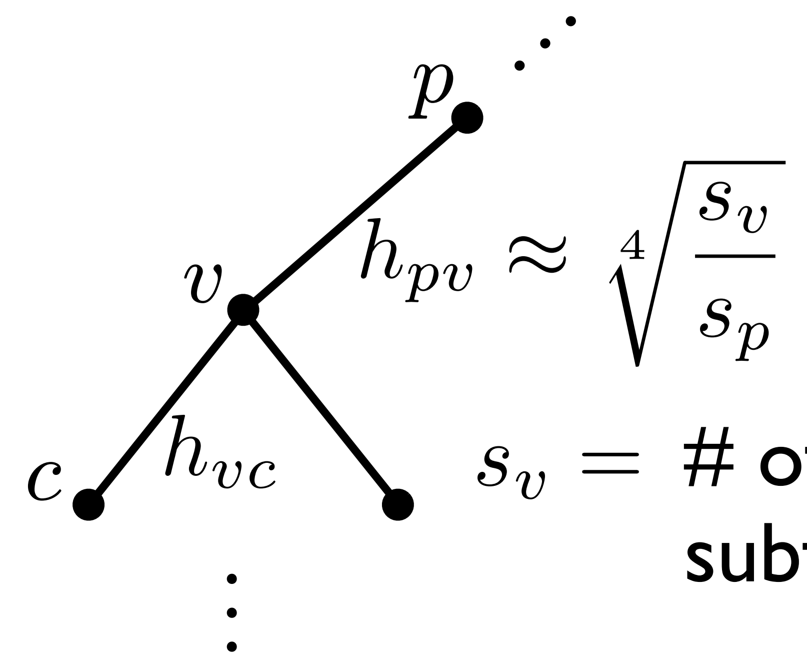
Lemma: If the formula evaluates to 0, then the tree has an eigenstate with eigenvalue 0 that has constant overlap on the root. If the formula evaluates to 1, then all eigenstates with eigenvalue $O(1/\sqrt{n})$ have no overlap on the root.

General formulas

This simple strategy does not work for general formulas.

To get a general algorithm:

- Rewrite the formula to be "approximately balanced"
- Assign weights to the edges of the tree
- Show that eigenvectors are related to the function value



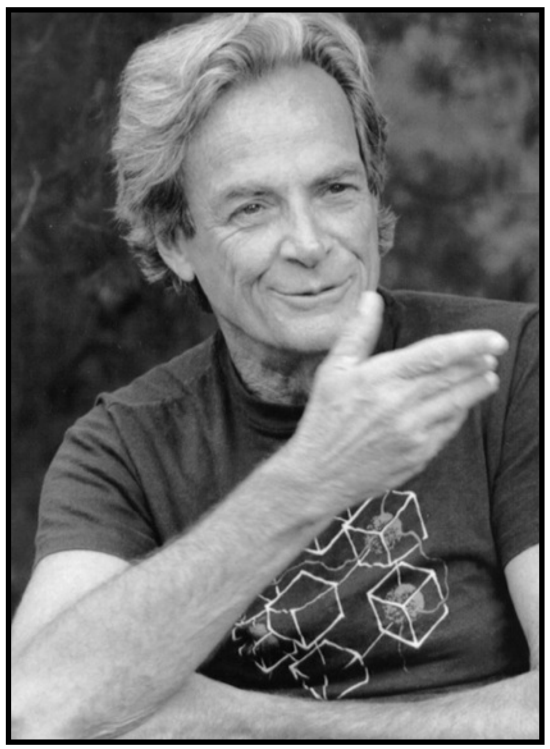
$s_v =$ # of inputs in subformula under v

> **Lemma:** If the formula evaluates to 0, then the tree has an eigenstate with eigenvalue 0 that has constant overlap on the root. If the formula evaluates to 1, then all eigenstates with eigenvalue $O(1/\sqrt{n})$ have no overlap on the root.

The quantum query complexity of evaluating any AND-OR formula is $O(n^{\frac{1}{2}+\epsilon})$ (subsequently improved to $O(\sqrt{n})$ [Reichardt 10])

Quantum simulation

- Childs, Communications in Mathematical Physics **294**, 581–603 (2010)
- Berry and Childs, Quantum Information and Computation **12**, 29–62 (2012)



"... nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy."

Richard Feynman
Simulating physics with computers (1981)

Why simulate quantum mechanics?

Why simulate quantum mechanics?

Computational chemistry/physics

- chemical reactions
- properties of materials

Why simulate quantum mechanics?

Computational chemistry/physics

- chemical reactions
- properties of materials

Implementation of quantum algorithms

- continuous-time quantum walk
- adiabatic quantum computation
- linear equations

Quantum dynamics

The dynamics of a quantum system are determined by its *Hamiltonian*.

$$i \frac{\mathrm{d}}{\mathrm{d}t} \psi(t) = H \, \psi(t)$$

Quantum dynamics

The dynamics of a quantum system are determined by its *Hamiltonian*.

$$i\frac{\mathrm{d}}{\mathrm{d}t}\psi(t) = H\,\psi(t)$$

Quantum simulation problem: Given a description of the Hamiltonian H, an evolution time t, and an initial state $\psi(0)$, produce the final state $\psi(t)$ (to within some error tolerance)

Quantum dynamics

The dynamics of a quantum system are determined by its *Hamiltonian*.

$$i\frac{\mathrm{d}}{\mathrm{d}t}\psi(t) = H\,\psi(t)$$

Quantum simulation problem: Given a description of the Hamiltonian H, an evolution time t, and an initial state $\psi(0)$, produce the final state $\psi(t)$ (to within some error tolerance)

A classical computer cannot even represent the state efficiently

Quantum dynamics

The dynamics of a quantum system are determined by its *Hamiltonian*.

$$i \frac{\mathrm{d}}{\mathrm{d}t} \psi(t) = H \, \psi(t)$$

Quantum simulation problem: Given a description of the Hamiltonian H, an evolution time t, and an initial state $\psi(0)$, produce the final state $\psi(t)$ (to within some error tolerance)

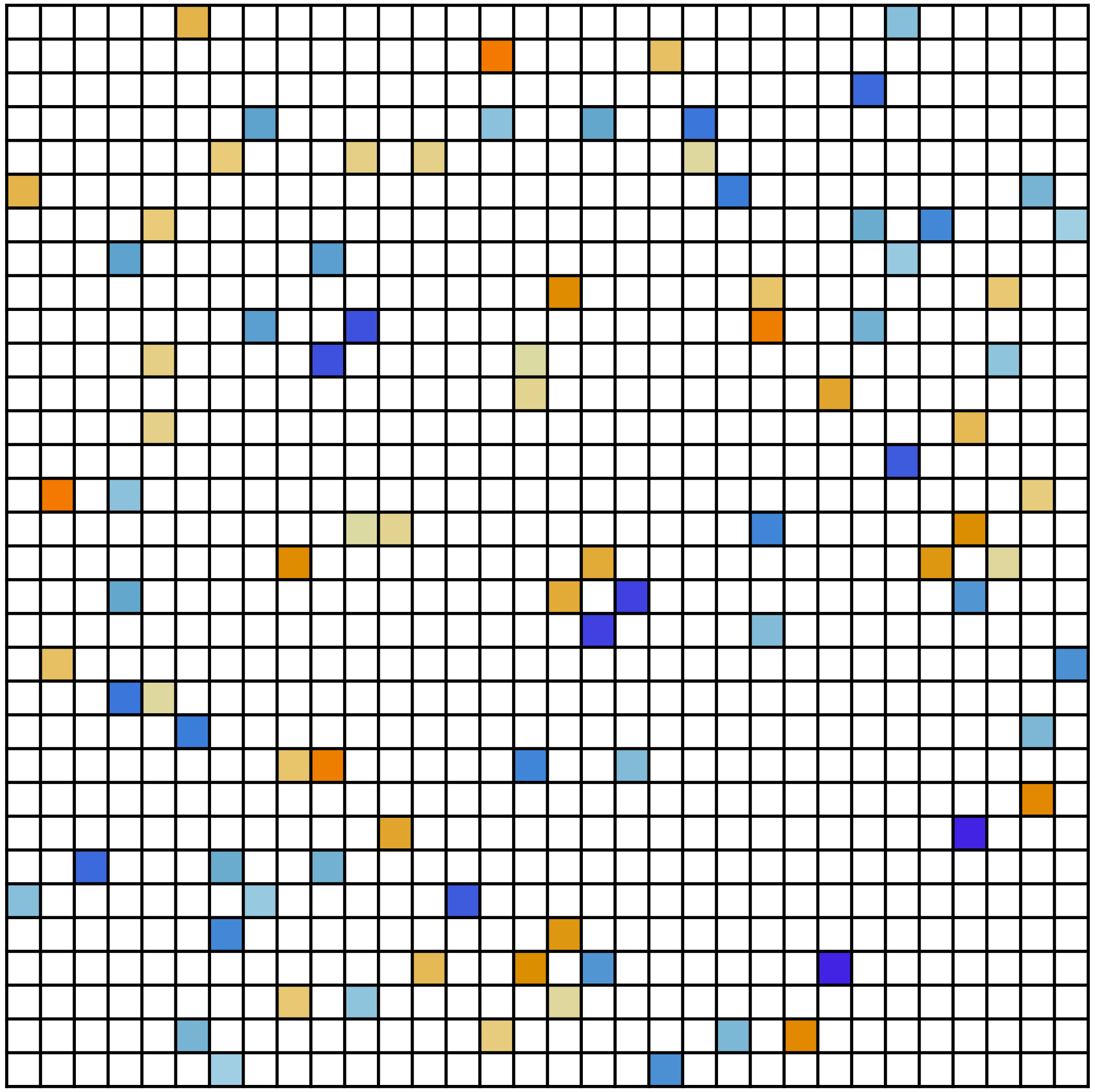
A classical computer cannot even represent the state efficiently

By performing measurements on the final state, a quantum computer can efficiently answer questions that (apparently) a classical computer cannot

Sparse Hamiltonians

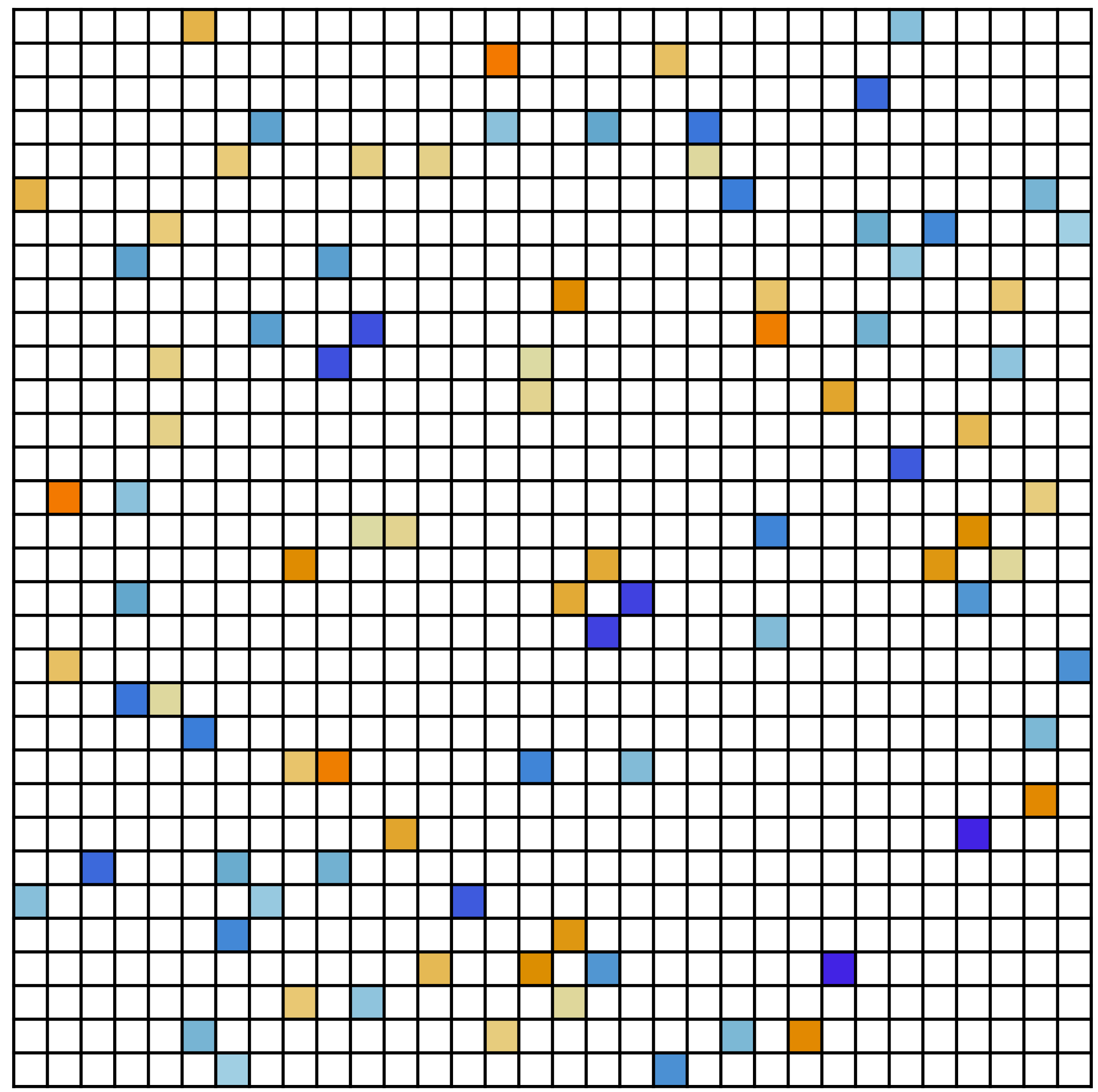
At most d nonzero entries per row (here $d = 4$)

$$H = $$



Sparse Hamiltonians

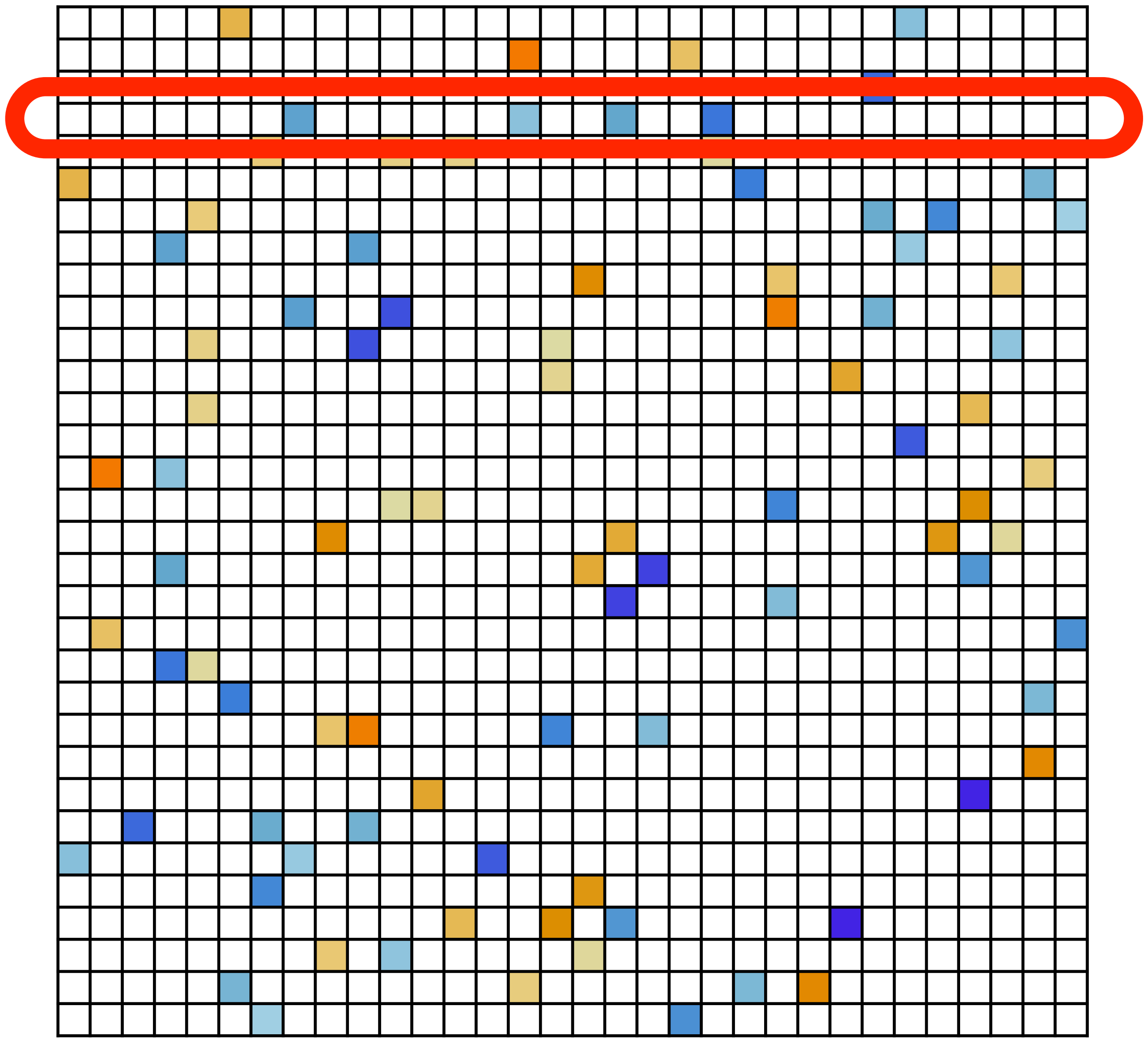
At most d nonzero entries per row (here $d = 4$)

$H =$ 

Assumption: we can efficiently compute locations and values of nonzero entries in any given row

Sparse Hamiltonians

At most d nonzero entries per row (here $d = 4$)

$H =$ 

Assumption: we can efficiently compute locations and values of nonzero entries in any given row

Simulation via product formulas

Original approach to sparse Hamiltonian simulation:

- Decompose $H = \sum_j H_j$ where each H_j is 1-sparse
  (distributed edge coloring)
- Recombine terms
  (product formulas, e.g., $e^{-i(A+B)t} \approx (e^{-iAt/r} e^{-iBt/r})^r$)

[AT 03, CCDFGS 03, BACS 07, CK 10]

Simulation via product formulas

Original approach to sparse Hamiltonian simulation:

- Decompose $H = \sum_j H_j$ where each H_j is 1-sparse (distributed edge coloring)
- Recombine terms (product formulas, e.g., $e^{-i(A+B)t} \approx (e^{-iAt/r} e^{-iBt/r})^r$)

Running time of the best approach of this kind:

- Superlinear in evolution time t
- Cubic in sparsity d

[AT 03, CCDFGS 03, BACS 07, CK 10]

Discrete-time quantum walk

Can we define a quantum walk that takes discrete steps?

Discrete-time quantum walk

Can we define a quantum walk that takes discrete steps?

In general, locality and unitarity are incompatible

Discrete-time quantum walk

Can we define a quantum walk that takes discrete steps?

In general, locality and unitarity are incompatible

Workaround: define a walk on the *directed edges* (a "coined walk")

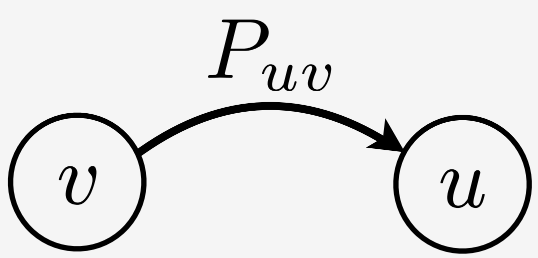
Discrete-time quantum walk

Can we define a quantum walk that takes discrete steps?

In general, locality and unitarity are incompatible

Workaround: define a walk on the *directed edges* (a "coined walk")

Szegedy 05: For a stochastic transition matrix P,



- Reflect about $\mathrm{span}\{\psi_v : v \in V\}$

  where $(\psi_v)_{(w,u)} = \begin{cases} \sqrt{P_{uv}} & \text{if } v = w \\ 0 & \text{otherwise} \end{cases}$

- Swap the edge direction

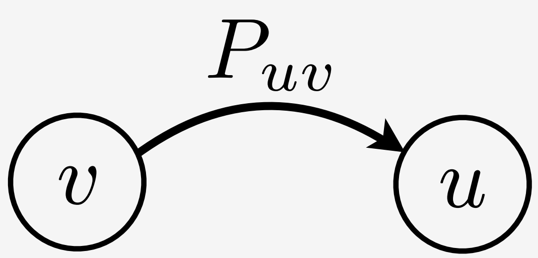
Discrete-time quantum walk

Can we define a quantum walk that takes discrete steps?

In general, locality and unitarity are incompatible

Workaround: define a walk on the *directed edges* (a "coined walk")

Szegedy 05: For a stochastic transition matrix P,



- Reflect about $\mathrm{span}\{\psi_v : v \in V\}$

  where $(\psi_v)_{(w,u)} = \begin{cases} \sqrt{P_{uv}} & \text{if } v = w \\ 0 & \text{otherwise} \end{cases}$

- Swap the edge direction

This gives a quantum analog of the Markov chain P

Hamiltonian simulation by quantum walk

1. Define an analog of Szegedy's walk for any Hamiltonian H (in place of the stochastic matrix P)
2. Show how to perform steps of this walk using queries to the sparse Hamiltonian
3. Relate the spectrum of the walk to the spectrum of H
4. Infer information about the spectrum of the walk (and hence of H) using quantum phase estimation
5. Introduce the appropriate phase $e^{-i\phi t}$ for each eigenstate of H with eigenvalue ϕ

Hamiltonian simulation by quantum walk

1. Define an analog of Szegedy's walk for any Hamiltonian H (in place of the stochastic matrix P)
2. Show how to perform steps of this walk using queries to the sparse Hamiltonian
3. Relate the spectrum of the walk to the spectrum of H
4. Infer information about the spectrum of the walk (and hence of H) using quantum phase estimation
5. Introduce the appropriate phase $e^{-i\phi t}$ for each eigenstate of H with eigenvalue ϕ

Theorem: This running time of this approach is $O(dt)$.

Hamiltonian simulation by quantum walk

1. Define an analog of Szegedy's walk for any Hamiltonian H (in place of the stochastic matrix P)
2. Show how to perform steps of this walk using queries to the sparse Hamiltonian
3. Relate the spectrum of the walk to the spectrum of H
4. Infer information about the spectrum of the walk (and hence of H) using quantum phase estimation
5. Introduce the appropriate phase $e^{-i\phi t}$ for each eigenstate of H with eigenvalue ϕ

Theorem: This running time of this approach is $O(dt)$.

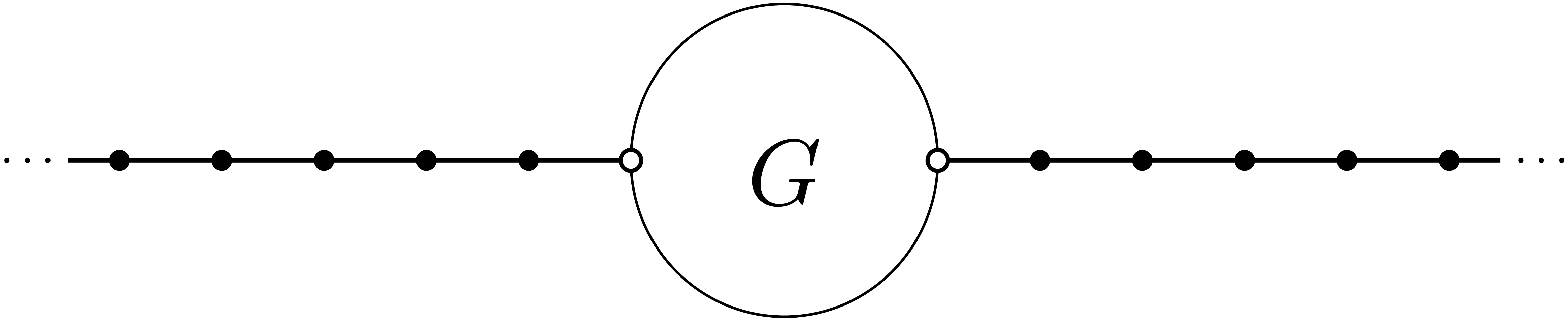
This algorithm is optimal with respect to either d or t alone

Universal computation

- Childs, Physical Review Letters **102**, 180501 (2009)
- Childs, Gosset, and Webb, Science **339**, 791–794 (2013)

Scattering on graphs

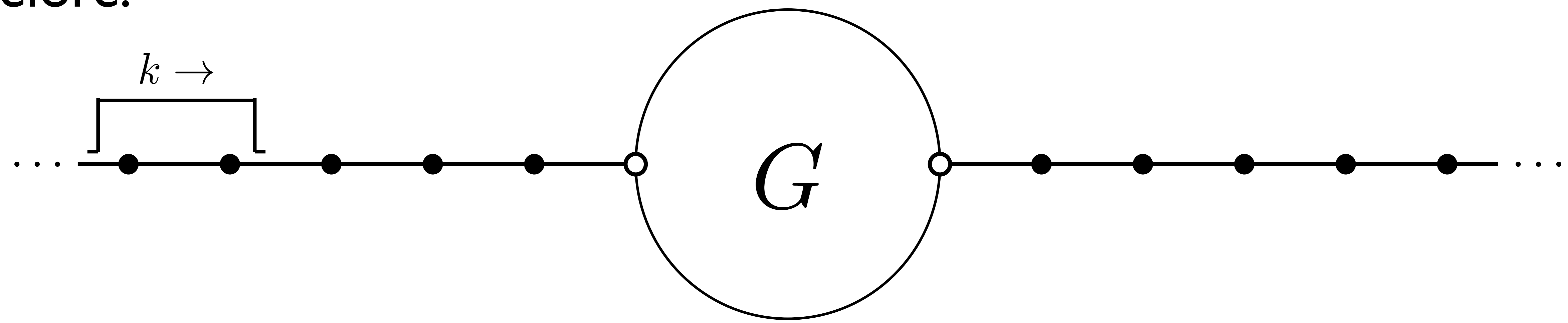
Attach semi-infinite paths to two vertices of an arbitrary finite graph.



Scattering on graphs

Attach semi-infinite paths to two vertices of an arbitrary finite graph.

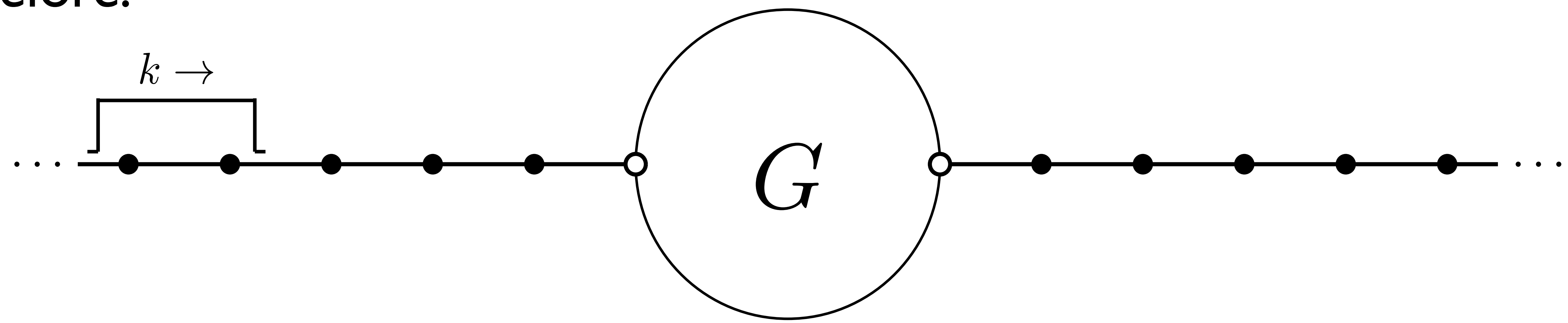
Before:



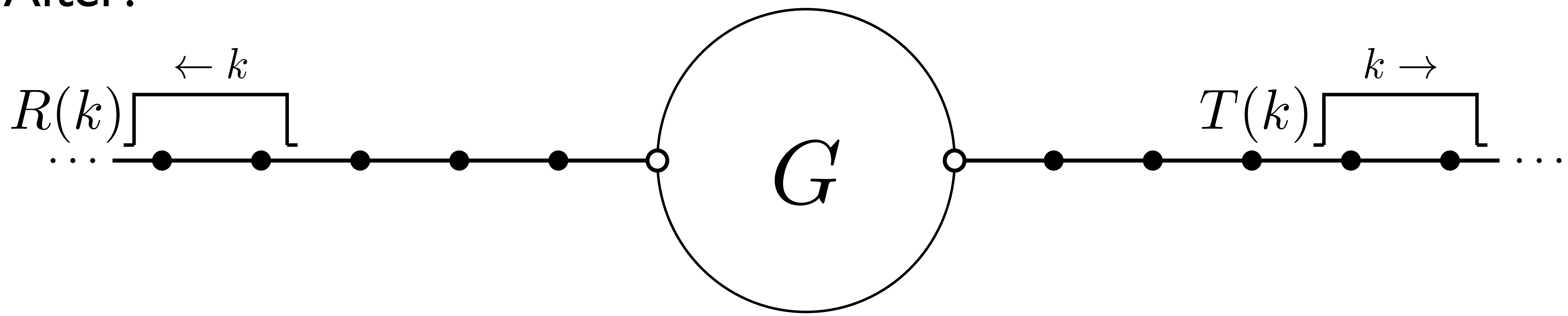
Scattering on graphs

Attach semi-infinite paths to two vertices of an arbitrary finite graph.

Before:



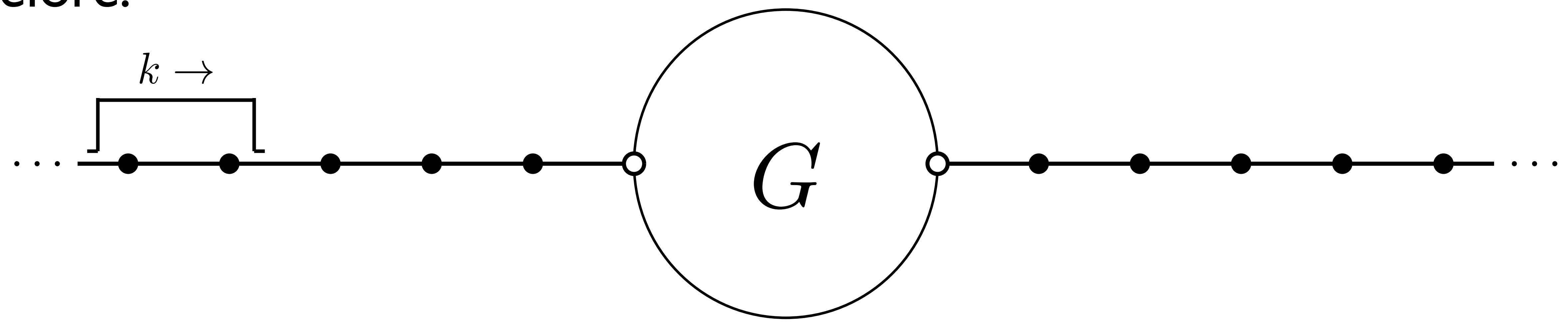
After:



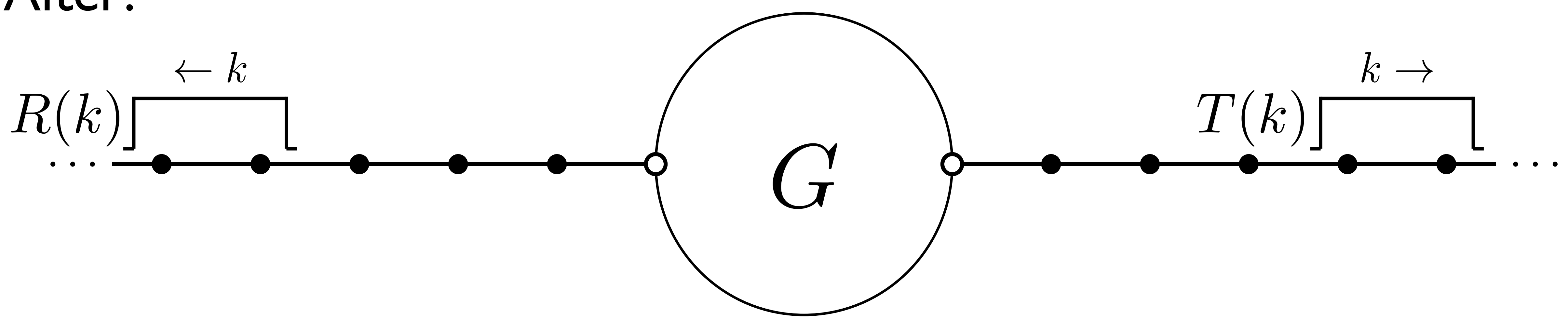
Scattering on graphs

Attach semi-infinite paths to two vertices of an arbitrary finite graph.

Before:

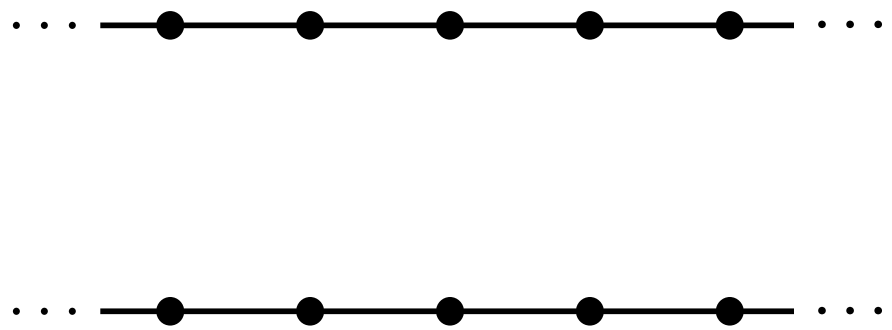


After:

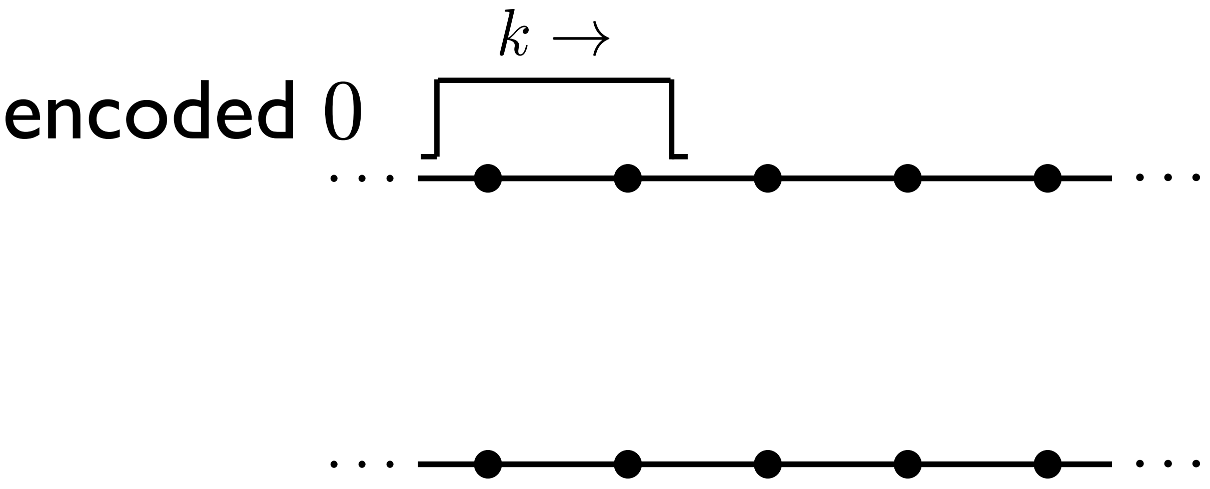


More generally, attach any number of semi-infinite paths. The scattering behavior is described a unitary matrix called the *S-matrix*.

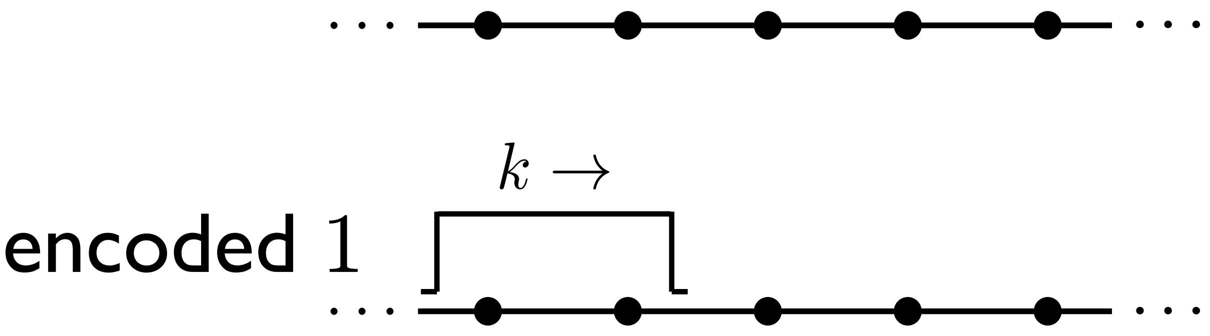
Implementing a gate



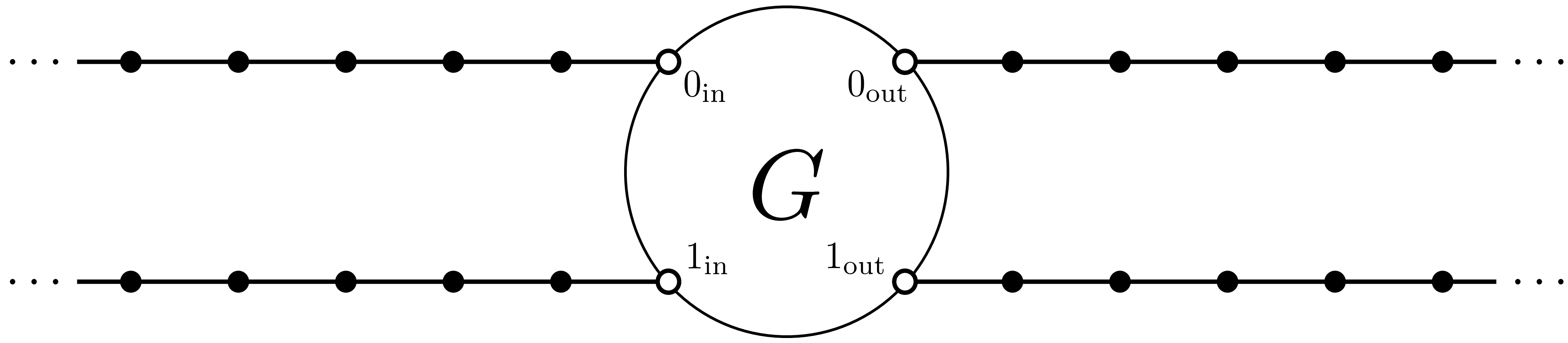
Implementing a gate



Implementing a gate

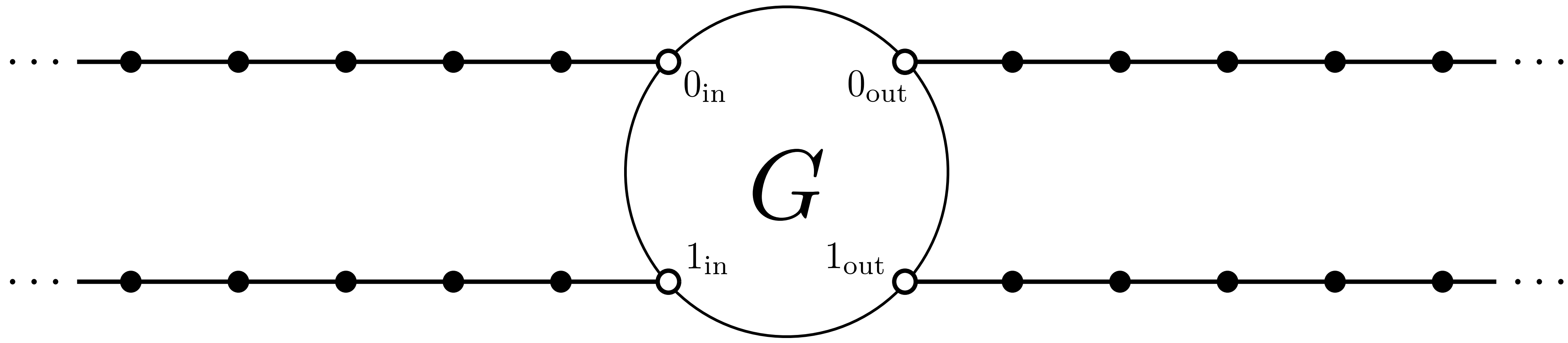


Implementing a gate



To perform a gate, design a graph whose S-matrix implements the desired transformation U at the momentum used for the encoding.

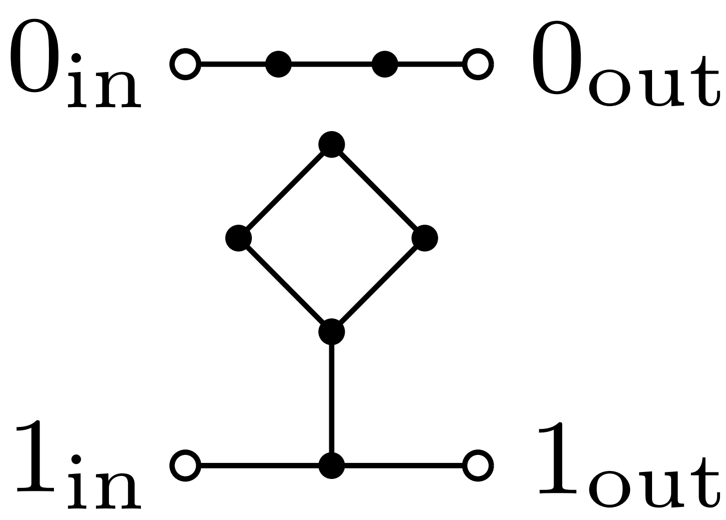
Implementing a gate

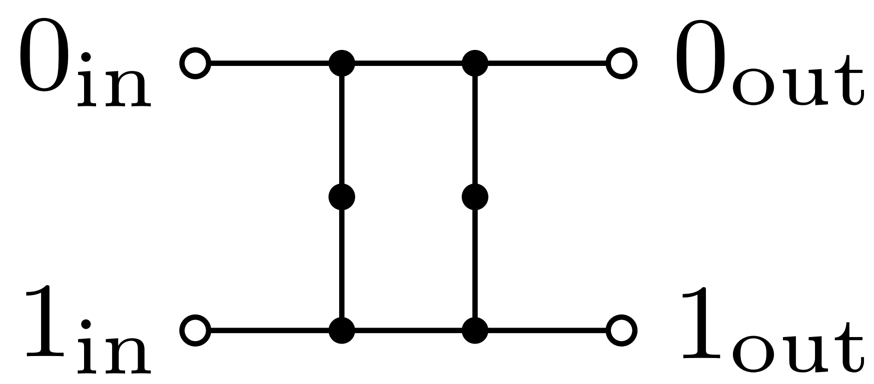


To perform a gate, design a graph whose S-matrix implements the desired transformation U at the momentum used for the encoding.

$$S(k) = \begin{pmatrix} 0 & V \\ U & 0 \end{pmatrix}$$

Universal set of single-qubit gates





$$\begin{pmatrix} 1 & 0 \\ 0 & \sqrt{i} \end{pmatrix}$$

$$-\frac{1}{\sqrt{2}}\begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix}$$

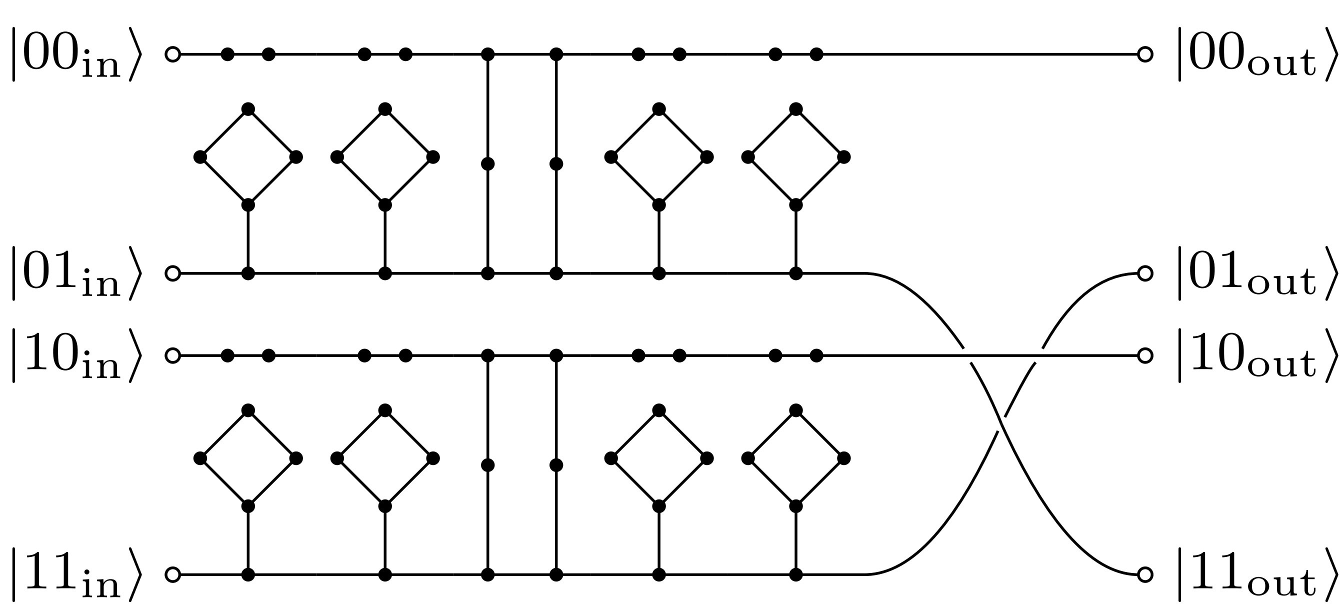
momentum for logical states: $k = \pi/4$

Universality construction

With an appropriate encoding of n-qubit states, two-qubit gates are trivial.

Implement sequences of gates by concatenation.

Result: Any n-qubit circuit can be simulated by some graph.



Quantum walk is computationally universal

In principle, any quantum computation can be performed by a quantum walk (of a very restricted form: unweighted, simple graph with maximum degree 3).

Quantum walk is computationally universal

In principle, any quantum computation can be performed by a quantum walk (of a very restricted form: unweighted, simple graph with maximum degree 3).

Quantum walks are computationally powerful!

Quantum walk is computationally universal

In principle, any quantum computation can be performed by a quantum walk (of a very restricted form: unweighted, simple graph with maximum degree 3).

Quantum walks are computationally powerful!

The construction suggests an algorithmic technique (scattering on graphs)…

Quantum walk is computationally universal

In principle, any quantum computation can be performed by a quantum walk (of a very restricted form: unweighted, simple graph with maximum degree 3).

Quantum walks are computationally powerful!

The construction suggests an algorithmic technique (scattering on graphs)…

… but not a new architecture (the graph is necessarily exponentially large).

Quantum walks with many walkers

Consider a quantum walk with many walkers that interact locally

Quantum walks with many walkers

Consider a quantum walk with many walkers that interact locally

With m walkers on an n-vertex graph, there are n^m states

Quantum walks with many walkers

Consider a quantum walk with many walkers that interact locally

With m walkers on an n-vertex graph, there are n^m states

Theorem: Any n-qubit, g-gate quantum circuit can be simulated by a quantum walk with $n+1$ walkers interacting for time $\mathrm{poly}(n, g)$ on an unweighted planar graph with $\mathrm{poly}(n, g)$ vertices.

Quantum walks with many walkers

Consider a quantum walk with many walkers that interact locally

With m walkers on an n-vertex graph, there are n^m states

Theorem: Any n-qubit, g-gate quantum circuit can be simulated by a quantum walk with $n+1$ walkers interacting for time $\mathrm{poly}(n, g)$ on an unweighted planar graph with $\mathrm{poly}(n, g)$ vertices.

Consequences:

Quantum walks with many walkers

Consider a quantum walk with many walkers that interact locally

With m walkers on an n-vertex graph, there are n^m states

Theorem: Any n-qubit, g-gate quantum circuit can be simulated by a quantum walk with $n+1$ walkers interacting for time $\mathrm{poly}(n,g)$ on an unweighted planar graph with $\mathrm{poly}(n,g)$ vertices.

Consequences:

- Quantum walks with many interacting walkers (on small graphs) are also computationally powerful

Quantum walks with many walkers

Consider a quantum walk with many walkers that interact locally

With m walkers on an n-vertex graph, there are n^m states

Theorem: Any n-qubit, g-gate quantum circuit can be simulated by a quantum walk with $n+1$ walkers interacting for time $\mathrm{poly}(n,g)$ on an unweighted planar graph with $\mathrm{poly}(n,g)$ vertices.

Consequences:

- Quantum walks with many interacting walkers (on small graphs) are also computationally powerful
- New architecture for a quantum computer (with no time-dependent control)

Quantum walks with many walkers

Consider a quantum walk with many walkers that interact locally

With m walkers on an n-vertex graph, there are n^m states

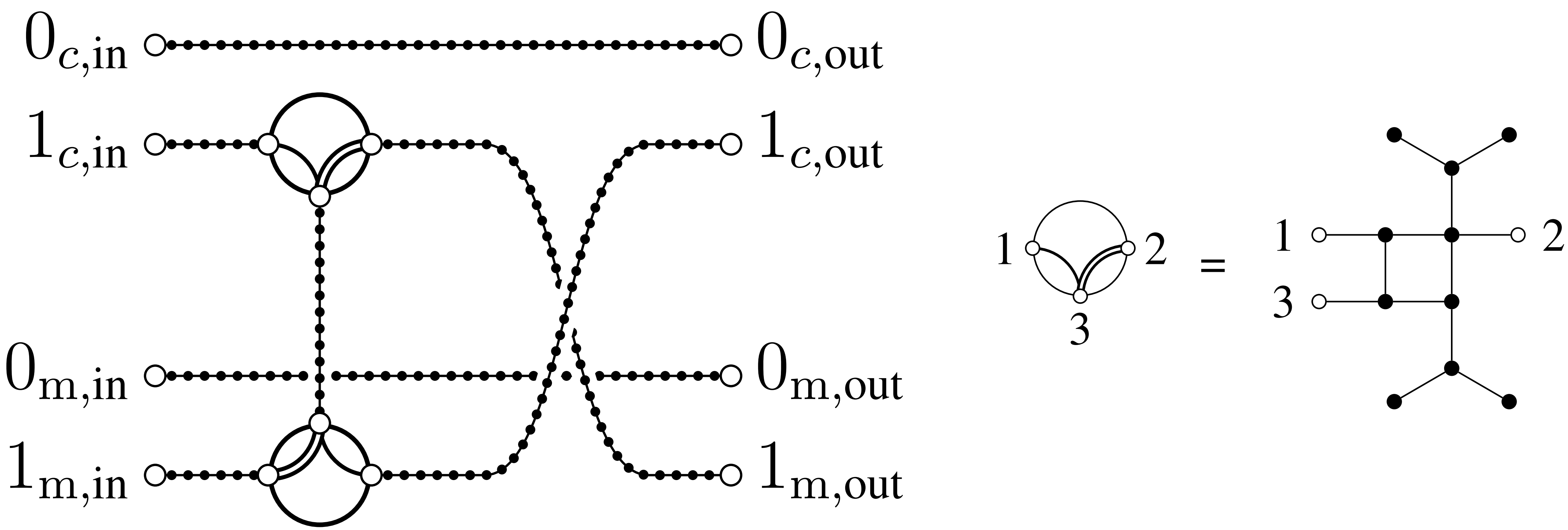
Theorem: Any n-qubit, g-gate quantum circuit can be simulated by a quantum walk with $n+1$ walkers interacting for time $\mathrm{poly}(n,g)$ on an unweighted planar graph with $\mathrm{poly}(n,g)$ vertices.

Consequences:

- Quantum walks with many interacting walkers (on small graphs) are also computationally powerful
- New architecture for a quantum computer (with no time-dependent control)
- Simulating the dynamics of interacting many-body systems is BQP-hard (e.g., the "Bose-Hubbard model" on a sparse, unweighted, planar graph)

Universal computation with many walkers

Main new idea: a gadget that implements a two-qubit interaction via momentum-dependent routing

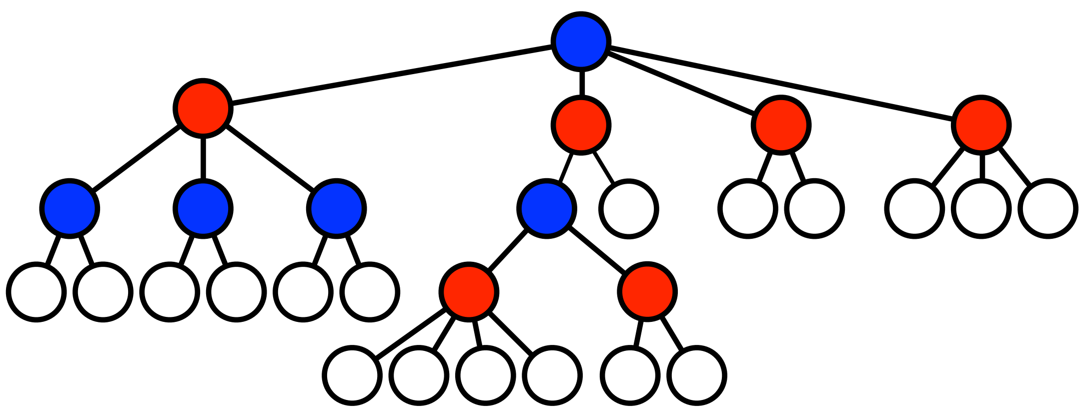


(+ extensive analysis to show the strategy works on a finite graph with small error)

Summary

Quantum walk is a powerful algorithmic tool.

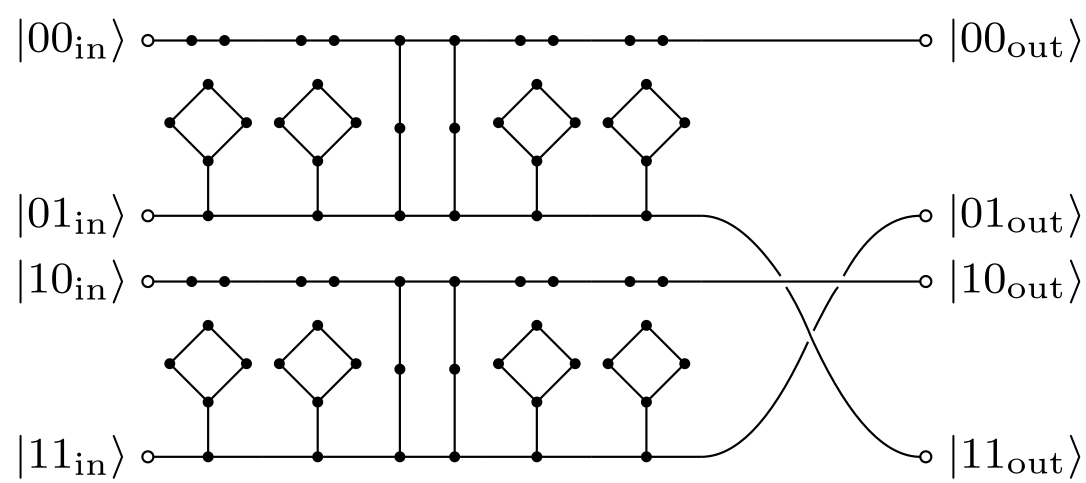
Formula evaluation



Quantum simulation

$$i \frac{\mathrm{d}}{\mathrm{d}t} \psi(t) = H\, \psi(t)$$

Universal computation



Outlook

When will we have large-scale quantum computers?

Outlook

When will we have large-scale quantum computers?

“Prediction is very difficult, especially about the future.” –Niels Bohr

Outlook

When will we have large-scale quantum computers?

“Prediction is very difficult, especially about the future.” –Niels Bohr

We can (and should!) address many crucial questions now:

Outlook

When will we have large-scale quantum computers?

"Prediction is very difficult, especially about the future." –Niels Bohr

We can (and should!) address many crucial questions now:

- How can we design cryptosystems that resist quantum attacks?

Outlook

When will we have large-scale quantum computers?

“Prediction is very difficult, especially about the future.” –Niels Bohr

We can (and should!) address many crucial questions now:

- How can we design cryptosystems that resist quantum attacks?
- How efficiently can quantum computers simulate quantum systems?

Outlook

When will we have large-scale quantum computers?

"Prediction is very difficult, especially about the future." –Niels Bohr

We can (and should!) address many crucial questions now:

- How can we design cryptosystems that resist quantum attacks?
- How efficiently can quantum computers simulate quantum systems?
- What other problems have fast quantum algorithms?

Outlook

When will we have large-scale quantum computers?

"Prediction is very difficult, especially about the future." –Niels Bohr

We can (and should!) address many crucial questions now:

- How can we design cryptosystems that resist quantum attacks?
- How efficiently can quantum computers simulate quantum systems?
- What other problems have fast quantum algorithms?
- What other tools are useful for building quantum algorithms?

Outlook

When will we have large-scale quantum computers?

"Prediction is very difficult, especially about the future." –Niels Bohr

We can (and should!) address many crucial questions now:

- How can we design cryptosystems that resist quantum attacks?
- How efficiently can quantum computers simulate quantum systems?
- What other problems have fast quantum algorithms?
- What other tools are useful for building quantum algorithms?
- What problems are hard even for quantum computers?