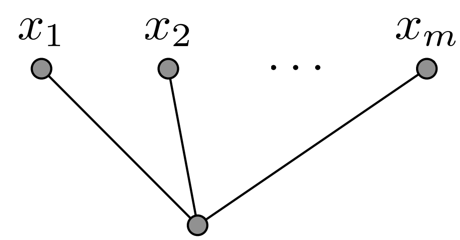
Any AND-OR formula of size N can be evaluated in time $N^{½+o(1)}$ on a quantum computer

Andrew Childs, Ben Reichardt, Robert Špalek, Shengyu Zhang

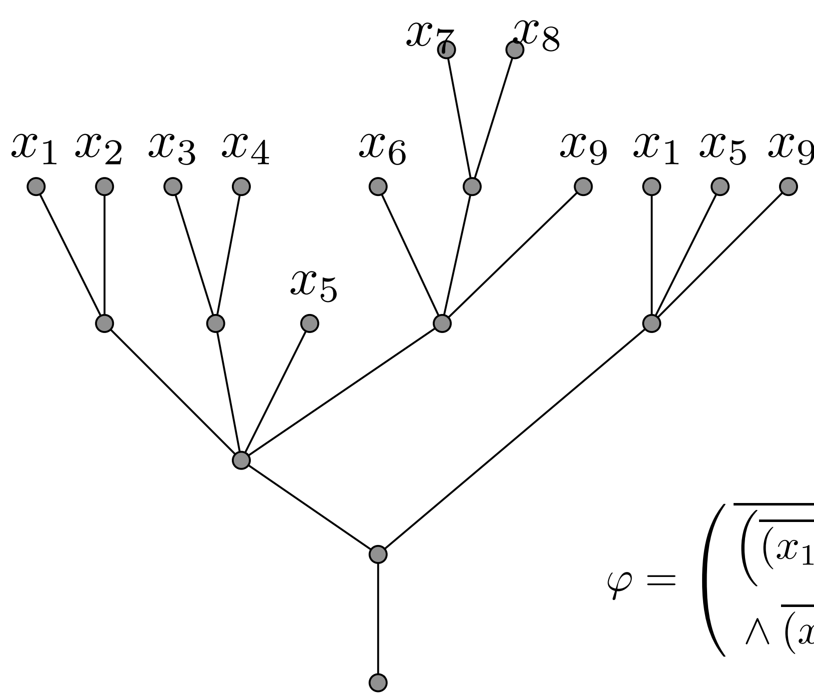
NAND formulas

- NAND gate (NOT-AND):



$$\neg(x_1 \wedge x_2 \wedge \cdots \wedge x_m)$$

- NAND "formula" = tree of nested NAND gates

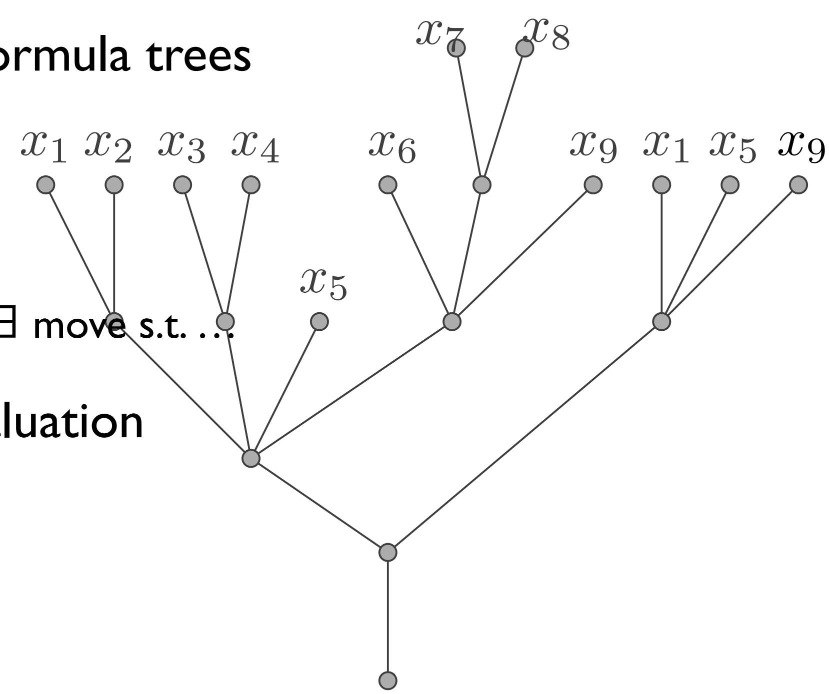


$$\varphi = \left(\overline{\left(\overline{(x_1 \wedge x_2)} \wedge \overline{(x_3 \wedge x_4)} \wedge x_5 \wedge \overline{(x_6 \wedge \overline{(x_7 \wedge x_8)} \wedge x_9)} \right)} \wedge \overline{(x_1 \wedge x_5 \wedge x_9)} \right)$$

- **Problem**: Evaluate φ(x), given a function for evaluating the x_i (oracle access to x).

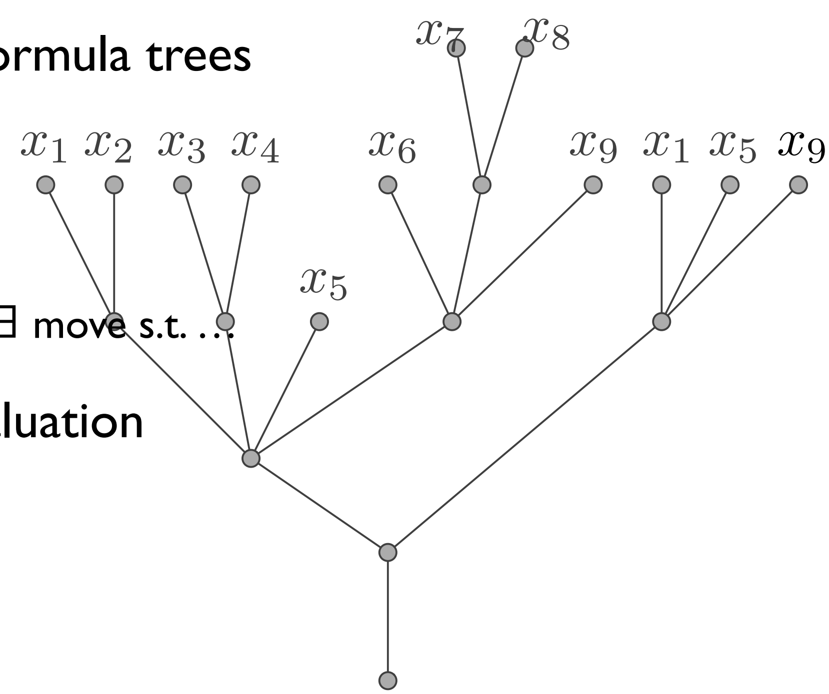
Problem motivations

- **Problem**: Evaluate φ(x), given a function for evaluating the x_i (oracle access to x).
- Motivations:
  - Equivalent to S = {AND, OR, NOT} formula trees
  - Playing "chess" (two-player games)
    - Nodes ↔ game histories
    - White wins if ∃ move s.t. ∀ black moves, ∃ move s.t. . . .
  - Decision version of min-max tree evaluation
    - inputs are real numbers
    - want to decide if minimax is ≥10 or not



Results

- **Problem**: Evaluate φ(x), given a function for evaluating the x_i (oracle access to x).
- Motivations:
  - Equivalent to S = {AND, OR, NOT} formula trees
  - Playing "chess" (two-player games)
    - Nodes ↔ game histories
    - White wins if ∃ move s.t. ∀ black moves, ∃ move s.t. ...
  - Decision version of min-max tree evaluation
    - inputs are real numbers
    - want to decide if minimax is ≥10 or not



- Results:
  - $N^{½+o(1)}$-time quantum algorithm (N= #leaves) for general trees (after efficient preprocessing independent of x)
  - O(√N)-query quantum algorithm for "approximately balanced" trees

Problem history (1/2)

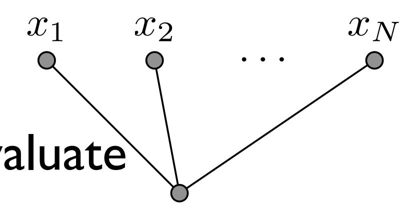
- Problem: Evaluate φ(x), given a function for evaluating the x_i.
- Results:
  - $N^{½+o(1)}$-time quantum algorithm (N= #leaves) for general trees (after efficient preprocessing independent of x)
  - $O(\sqrt{N})$-query quantum algorithm for "approximately balanced" trees (optimal!)
- Classical history
  - Deterministic algorithm requires time N
  - Randomized (Las Vegas) algorithm in E-time $O(N^{0.754})$ for balanced binary trees [Snir '85, Saks & Wigderson '86]

    $\log_2 \lambda_{max}(\begin{pmatrix} 0 & d \\ 1 & \frac{d-1}{2} \end{pmatrix})$

    - Flip coins to decide which subtree to evaluate next, short-circuit
    - Optimal [Santha '95]

Problem history (2/2)

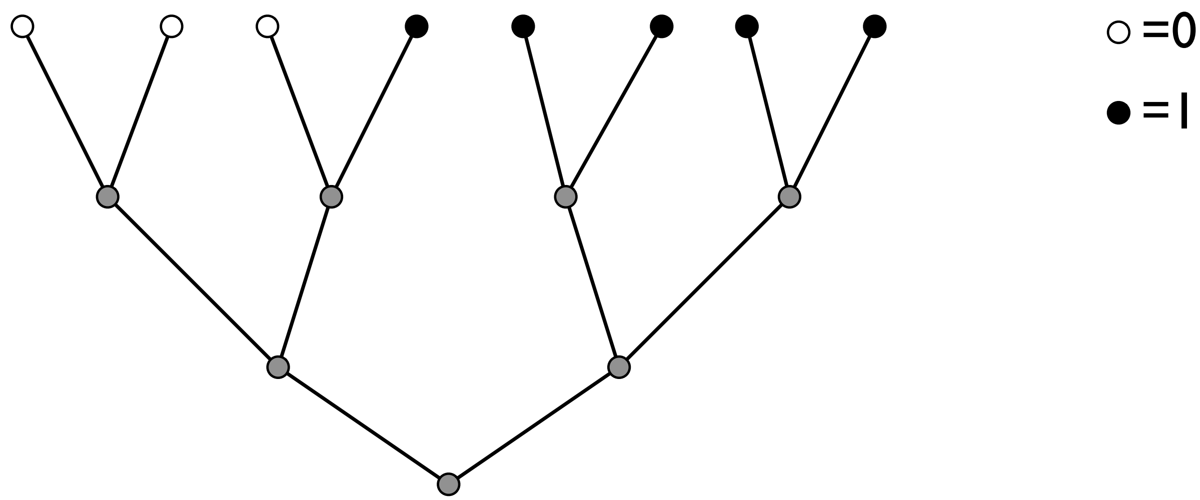
- Classical history
  - Deterministic algorithm requires time N
  - Randomized (Las Vegas) algorithm in E-time $\Theta(N^{0.754})$ for balanced binary trees [Snir '85, Saks & Wigderson '86, Santha '95]
- Quantum history



  - Grover search: $O(\sqrt{N})$-query quantum algorithm to evaluate (with constant error, $O(\sqrt{N} \log \log N)$-time) [Grover '96, '02]
    - Evaluates regular depth-d tree in $\sqrt{N}\, O(\log N)^{d-1}$ queries [BCW '98]
    - Extended to faulty oracles by [Høyer, Mosca, de Wolf '03] $\Rightarrow O(\sqrt{N}\, c^d)$ queries
  - Adversary lower bound $\Omega(\sqrt{N})$ queries [Barnum, Saks '04]
  - Farhi, Goldstone, Gutmann 2007: Breakthrough continuous-time quantum algorithm for evaluating balanced binary NAND tree in $N^{\frac{1}{2}+o(1)}$ queries & time

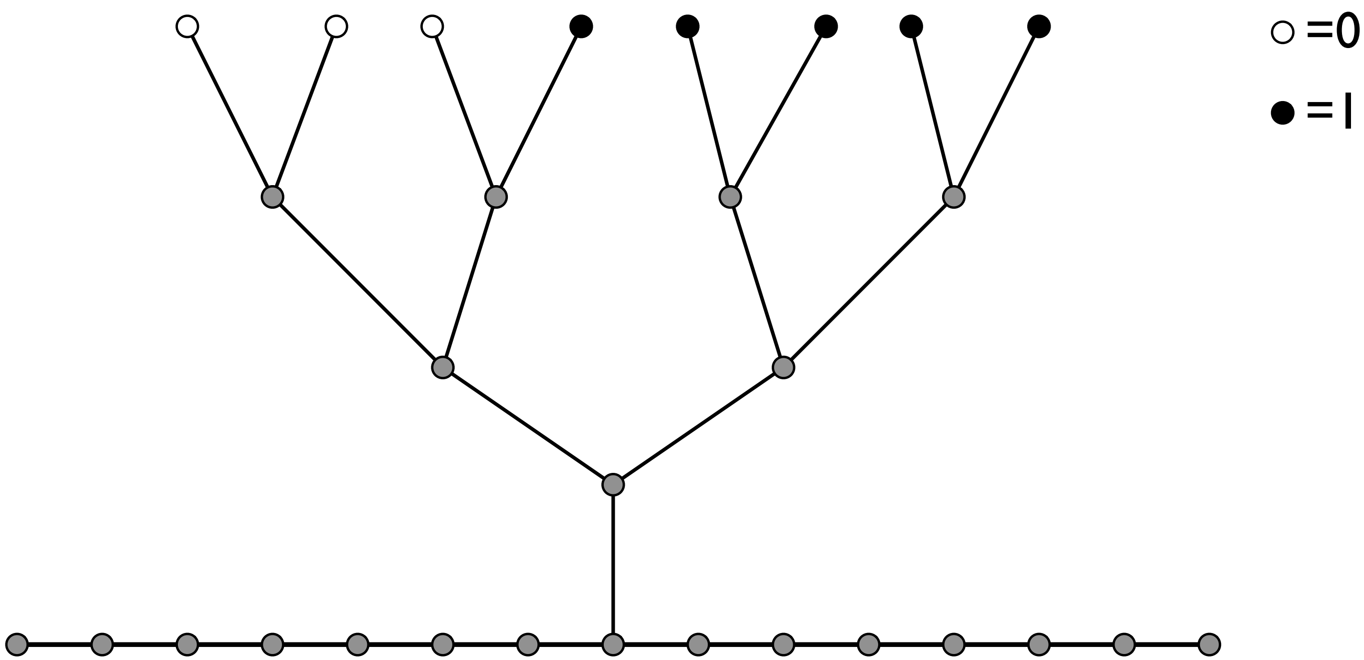
Farhi, Goldstone, Gutmann ‘07 algorithm

- **Theorem** ([FGG ‘07, CCJY ‘07]): A balanced binary NAND tree can be evaluated in time $N^{\frac{1}{2}+o(1)}$.
- Attach an infinite line to the root…



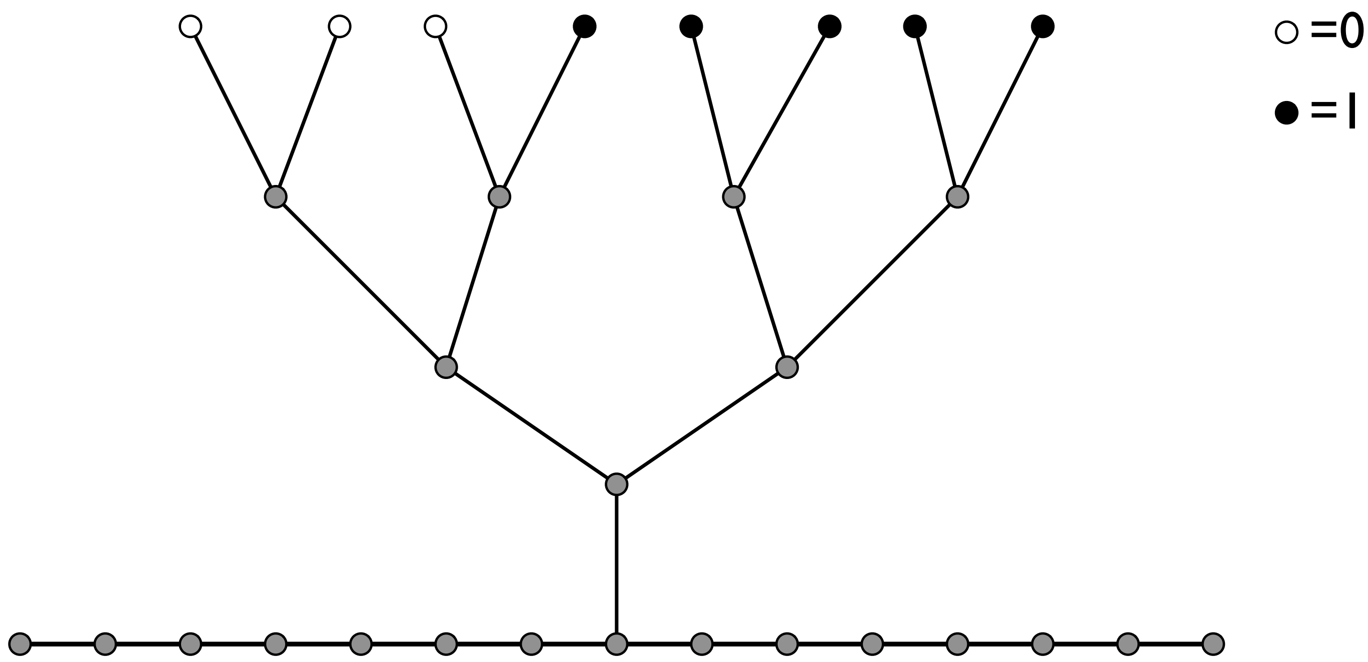
Farhi, Goldstone, Gutmann '07 algorithm

- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND tree can be evaluated in time $N^{1/2+o(1)}$.
- Attach an infinite line to the root…



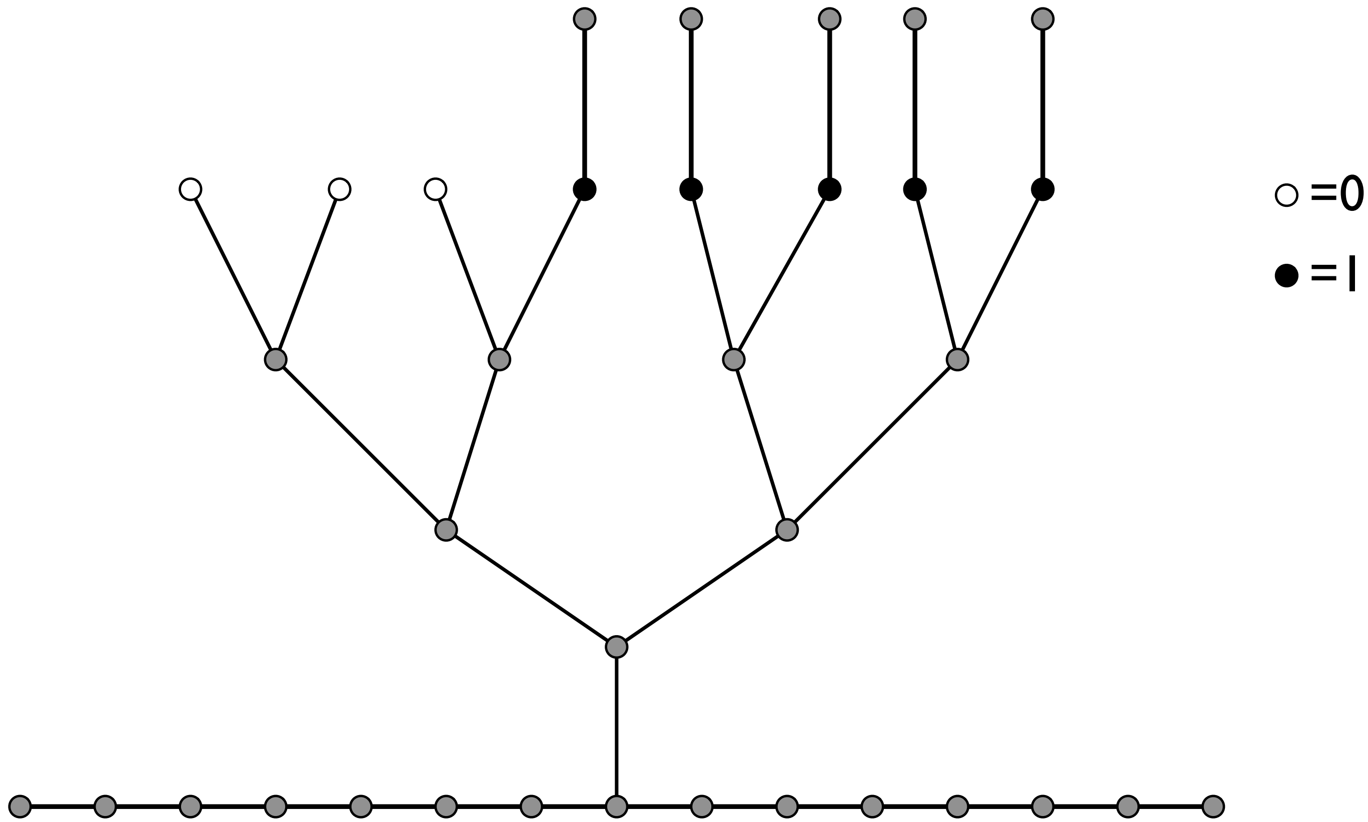
Farhi, Goldstone, Gutmann '07 algorithm

- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND tree can be evaluated in time $N^{½+o(1)}$.
- Attach an infinite line to the root
- Add edges above leaf nodes evaluating to one…



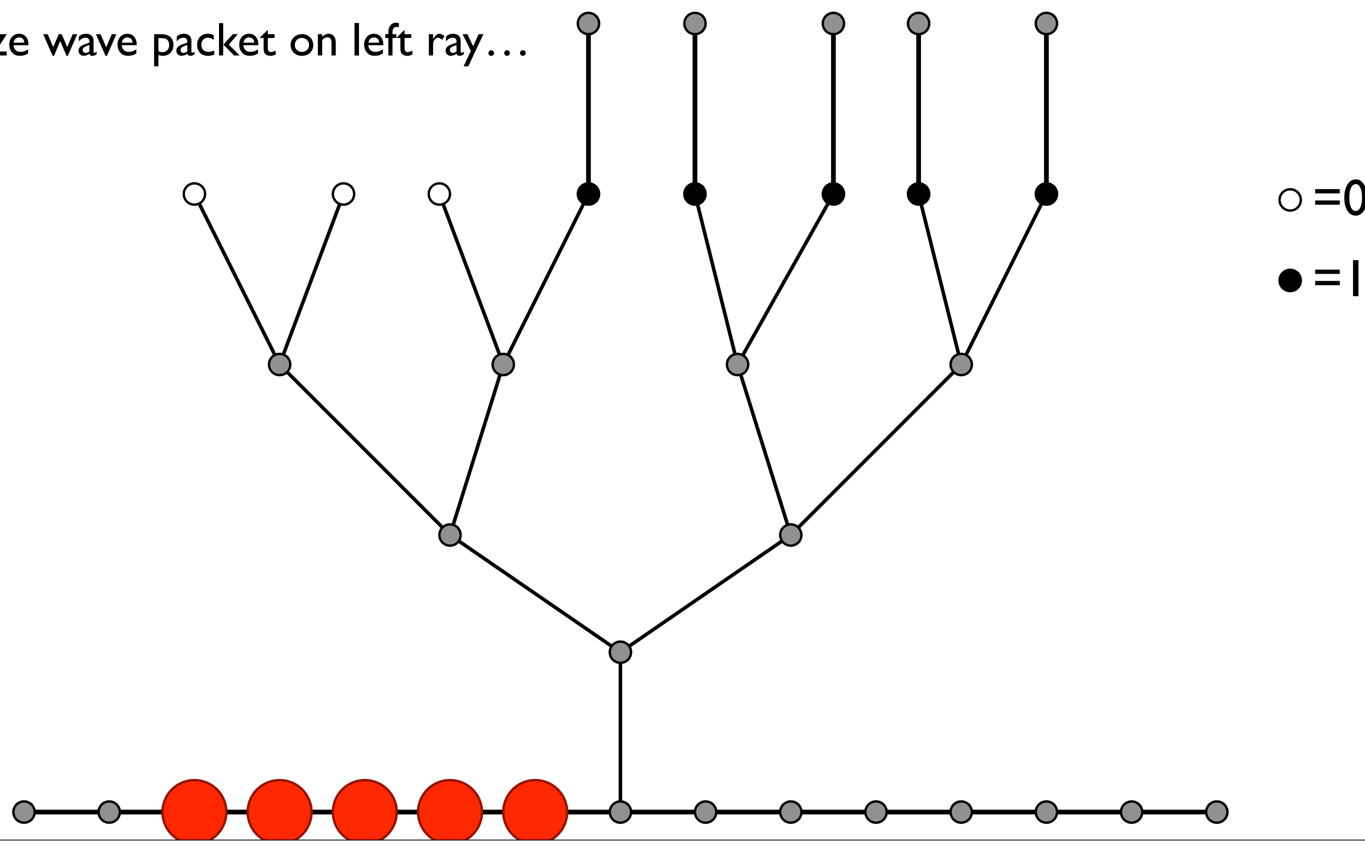
Farhi, Goldstone, Gutmann ‘07 algorithm

- **Theorem** ([FGG ‘07, CCJY ‘07]): A balanced binary NAND tree can be evaluated in time $N^{½+o(1)}$.
- Attach an infinite line to the root
- Add edges above leaf nodes evaluating to one…

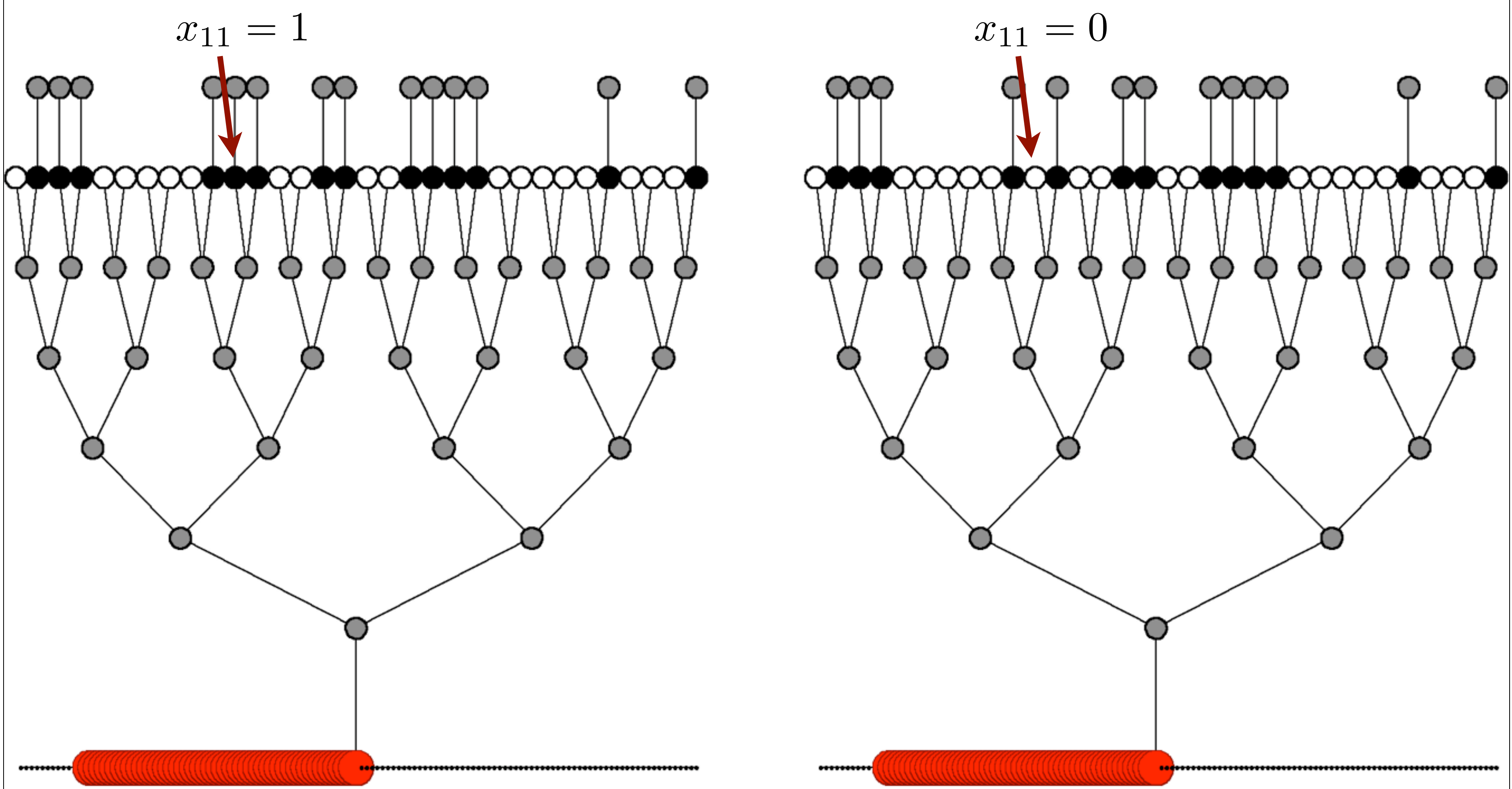


Farhi, Goldstone, Gutmann ‘07 algorithm

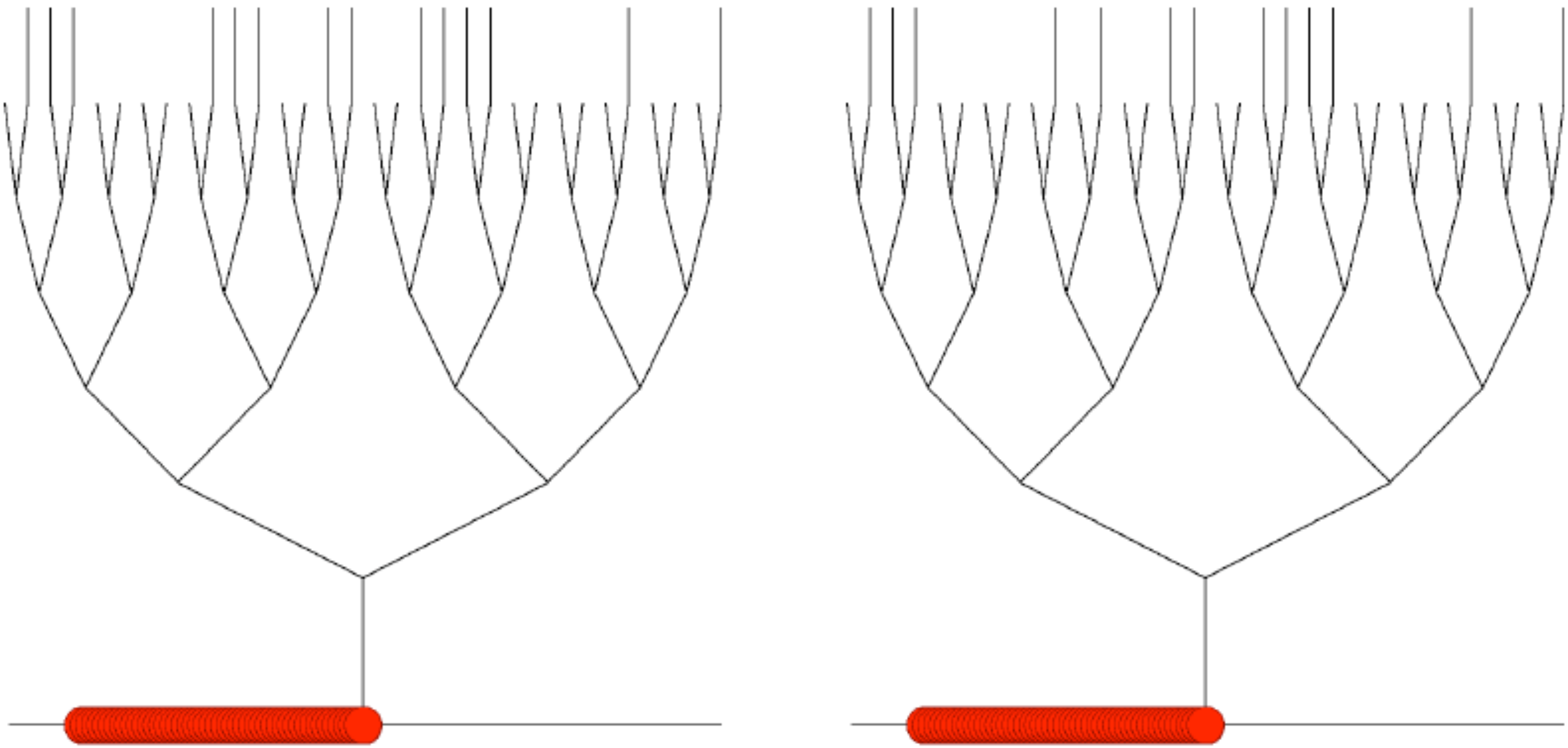
- **Theorem** ([FGG ‘07, CCJY ‘07]): A balanced binary NAND tree can be evaluated in time $N^{½+o(1)}$.
- Attach an infinite line to the root
- Add edges above leaf nodes evaluating to one
- Initialize wave packet on left ray…



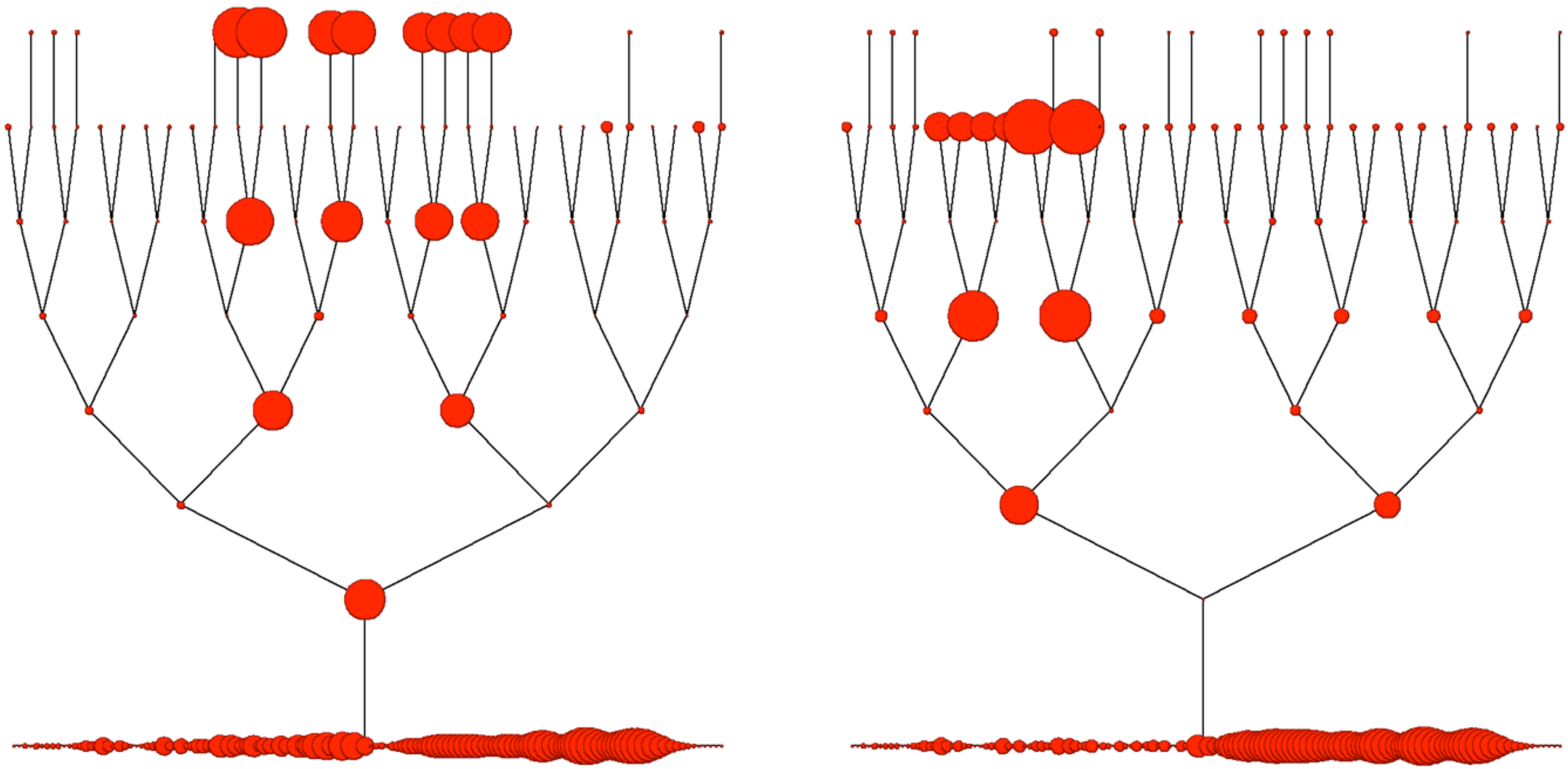
Continuous-time quantum walk [FGG '07]



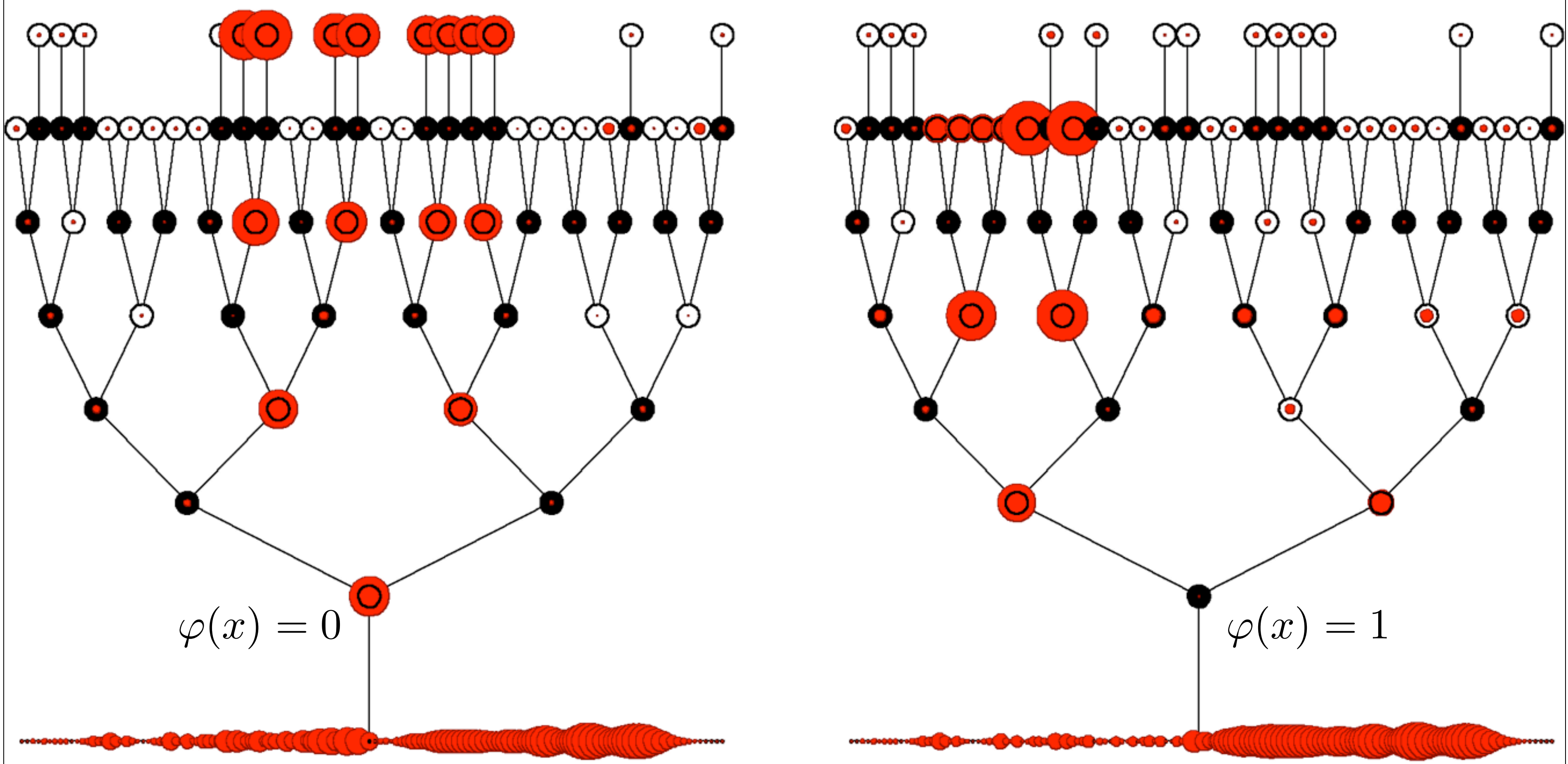
Continuous-time quantum walk [FGG '07]



Continuous-time quantum walk [FGG '07]



Continuous-time quantum walk [FGG '07]

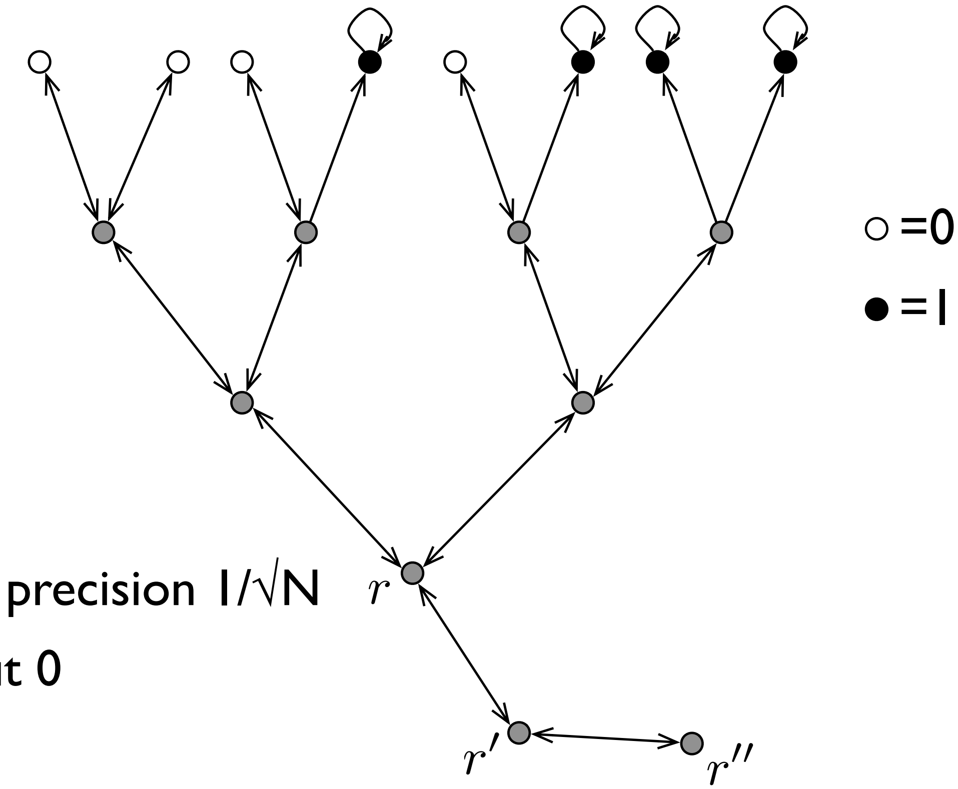


Talk outline

- Introduction
  - Motivation, classical & quantum problem history
  - Farhi, Goldstone, Gutmann breakthrough algorithm
- Results [Childs, Reichardt, Špalek, Zhang '07]:
  - $O(\sqrt{N})$-query quantum algorithm for "approximately balanced" trees
  - $N^{½+o(1)}$-time quantum algorithm (N= #leaves) for general trees (after efficient preprocessing independent of x)
- Optimal balanced tree algorithm
- Proof sketch
  - Szegedy correspondence
  - Zero-energy proof sketch
- Extension to unbalanced trees
  - Preprocessing
  - Weights
- Extensions & Open problems

Optimal balanced tree algorithm

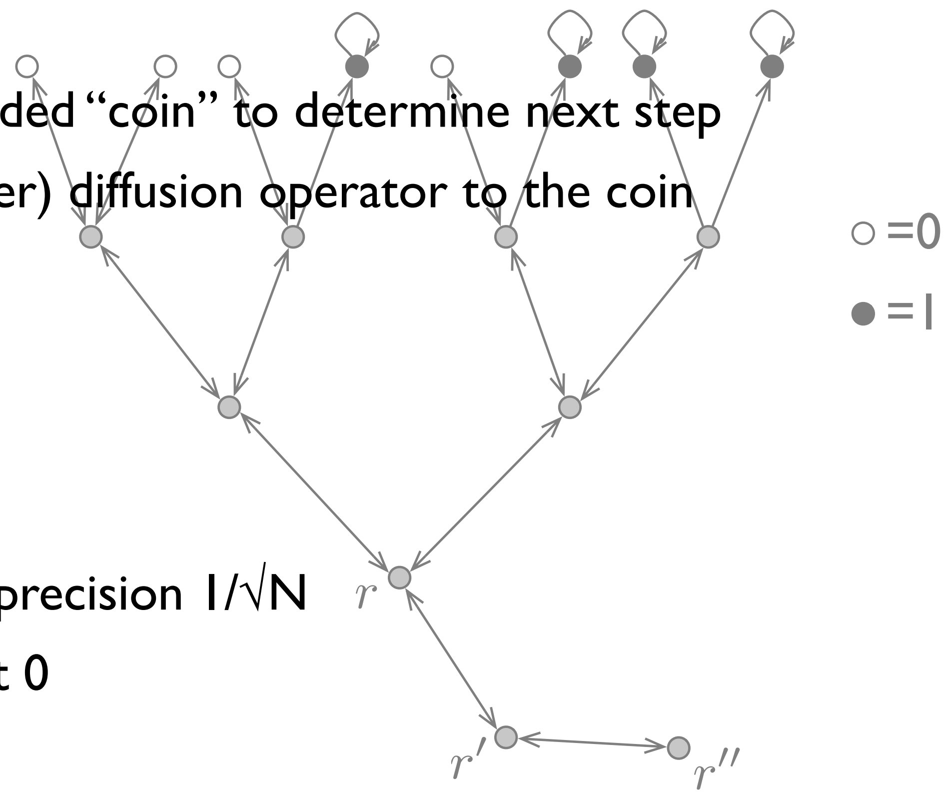
- Start with classical uniform random walk on balanced tree
  - Make leaves (inputs) evaluating to 1 probability *sinks*
  - Add two nodes r' and r'' at bottom, bias the coin at r'
- Quantize this walk…



- Start at r''
- Apply phase estimation to precision 1/√N
  - If phase is 0 or π, output 0
  - Otherwise output 1

Optimal balanced tree algorithm

- Start with classical uniform random walk on balanced tree
  - Make leaves (inputs) evaluating to 1 probability *sinks*
  - Add two nodes r' and r'' at bottom, bias the coin at r'
- Quantize this walk…
  - Classically, flip a three-sided "coin" to determine next step
  - Quantumly, apply (Grover) diffusion operator to the coin



- Start at r''
- Apply phase estimation to precision 1/√N
  - If phase is 0 or π, output 0
  - Otherwise output 1

Quantum walks

- Hilbert space $\mathbf{C}^{\vec{E}} = \left\langle |v, w\rangle : (v, w) \in E \right\rangle \subset \mathbf{C}^{V \times V} = \left\langle |v, w\rangle : v, w \in V \right\rangle$
- $U = \mathrm{Step} \cdot \mathrm{Flip}$
  - $\mathrm{Step} = \sum_{v,w} |v, w\rangle\langle w, v|$ switches direction of edges
  - $\mathrm{Flip} = \sum_{v} |v\rangle\langle v| \otimes \mathrm{Reflection}(|p(v)\rangle)$ diffuses outgoing edges from v
    - $|p(v)\rangle = \sum_{w \sim v} \sqrt{p_{v,w}}|w\rangle \qquad \left(\sum_{w} p_{v,w} = 1\right)$

$$P = \sum_{v,w} \sqrt{p_{v,w}\, p_{w,v}}|v\rangle\langle w|$$

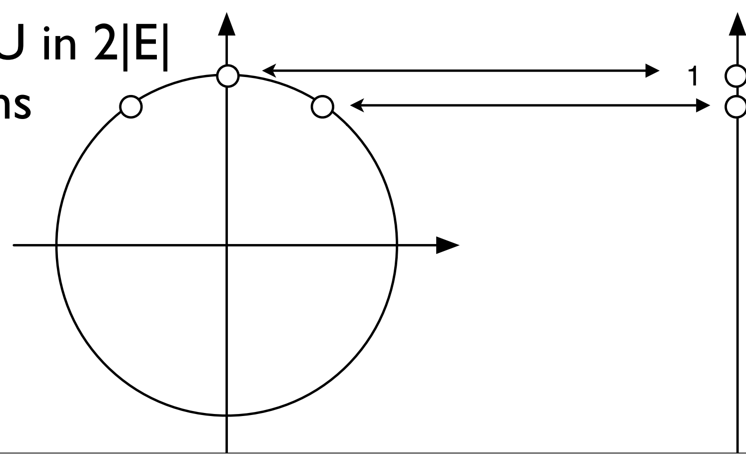
Proof: Szegedy correspondence

- Hilbert space $\mathbf{C}^{\vec{E}} = \left\langle |v,w\rangle : (v,w) \in E \right\rangle \subset \mathbf{C}^{V \times V} = \left\langle |v,w\rangle : v,w \in V \right\rangle$
- $U = \mathrm{Step} \cdot \mathrm{Flip}$
  - $\mathrm{Step} = \sum_{v,w} |v,w\rangle\langle w,v|$ switches direction of edges
  - $\mathrm{Flip} = \sum_{v} |v\rangle\langle v| \otimes \mathrm{Reflection}(|p(v)\rangle)$ diffuses outgoing edges from v

$$|p(v)\rangle = \sum_{w \sim v} \sqrt{p_{v,w}} |w\rangle \qquad \left(\sum_{w} p_{v,w} = 1 \right)$$

- Correspondence between spectrum and eigenvalues of

Coined unitary U in 2|E| dimensions



Symmetric matrix P in |V| dimensions

$$P = \sum_{v,w} \sqrt{p_{v,w}\, p_{w,v}} |v\rangle\langle w|$$

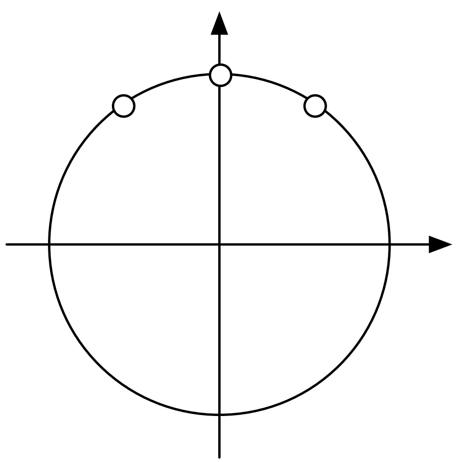
Proof: Szegedy correspondence

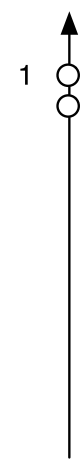
- Correspondence between spectrum and eigenvalues of

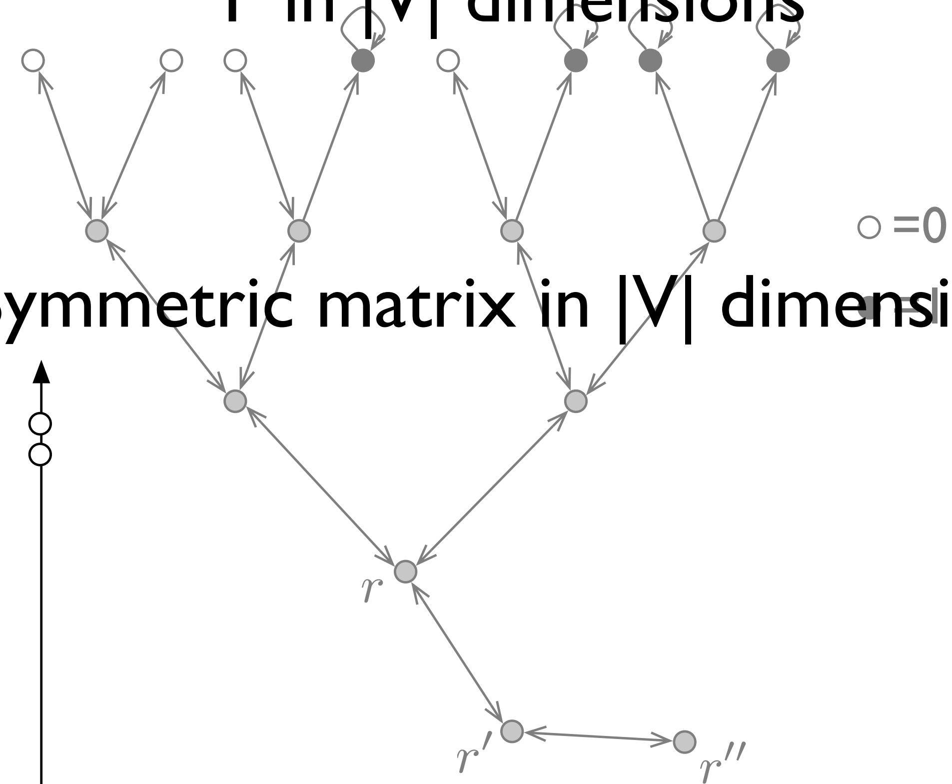
Coined unitary U in 2|E| dimensions

Classical random walk transition matrix P in |V| dimensions

Symmetric matrix in |V| dimensions

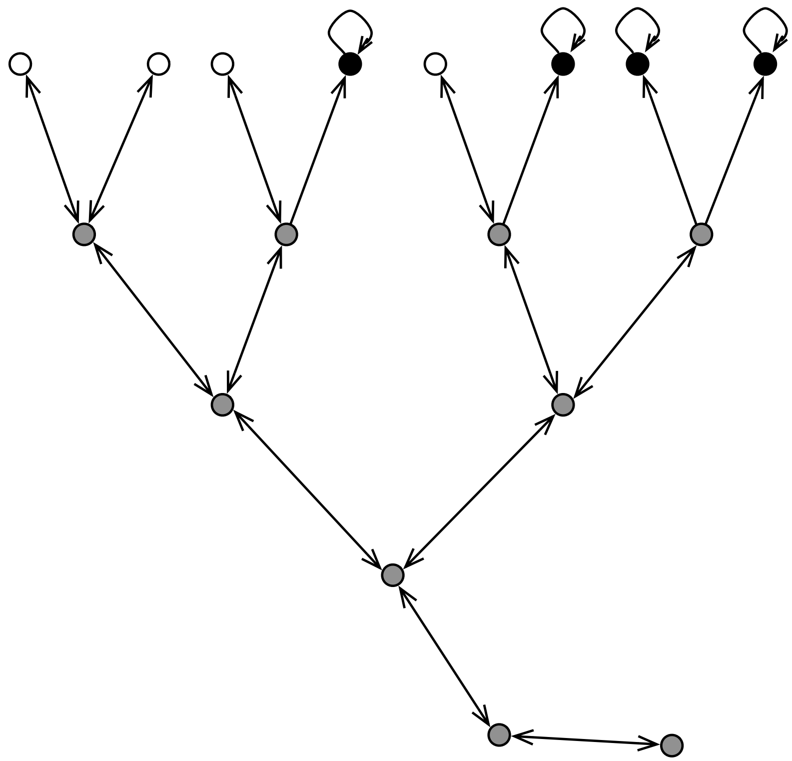






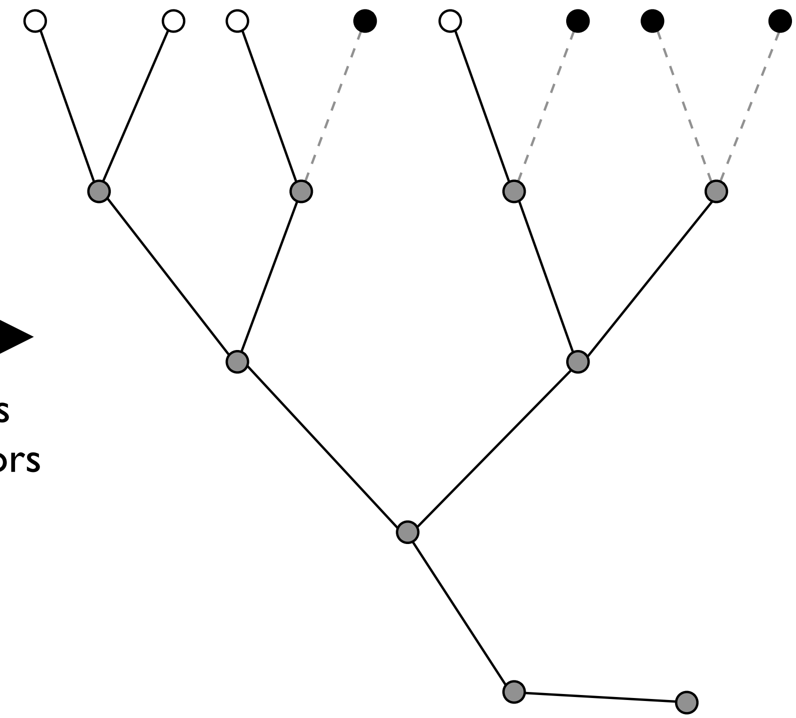
“Szegedization” applied

Quantum coined walk U on:



eigenvalues
& eigenvectors

Adjacency matrix A_G of:

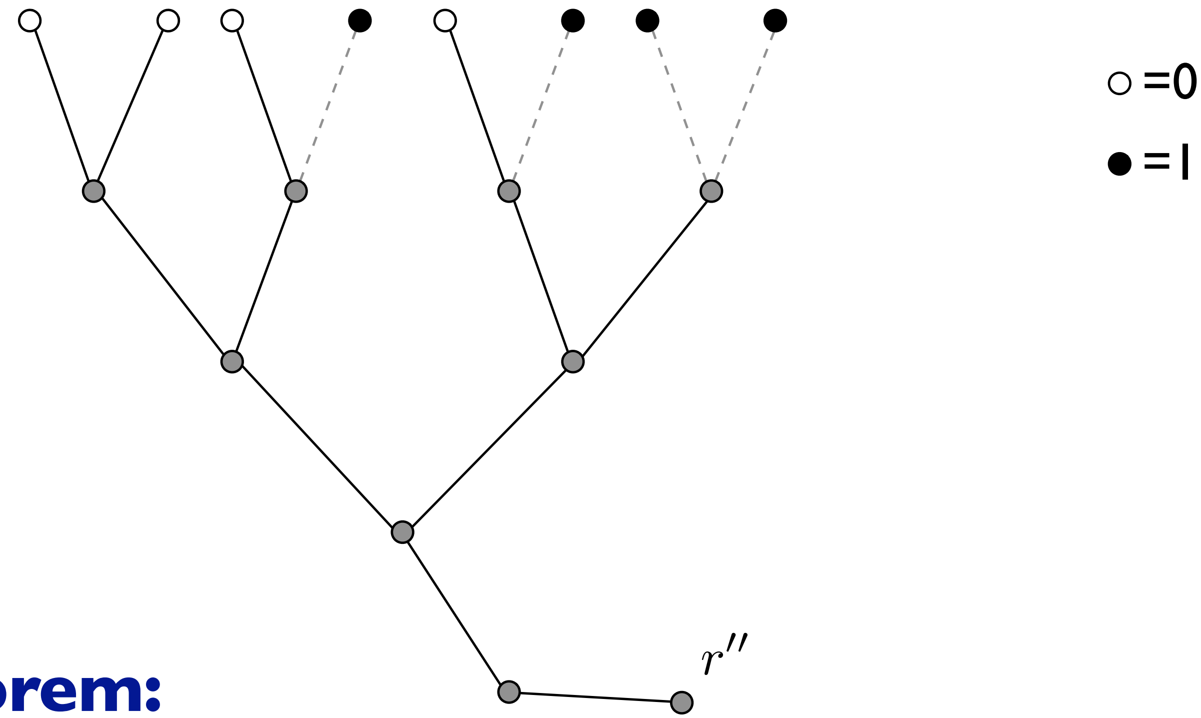


○ =0

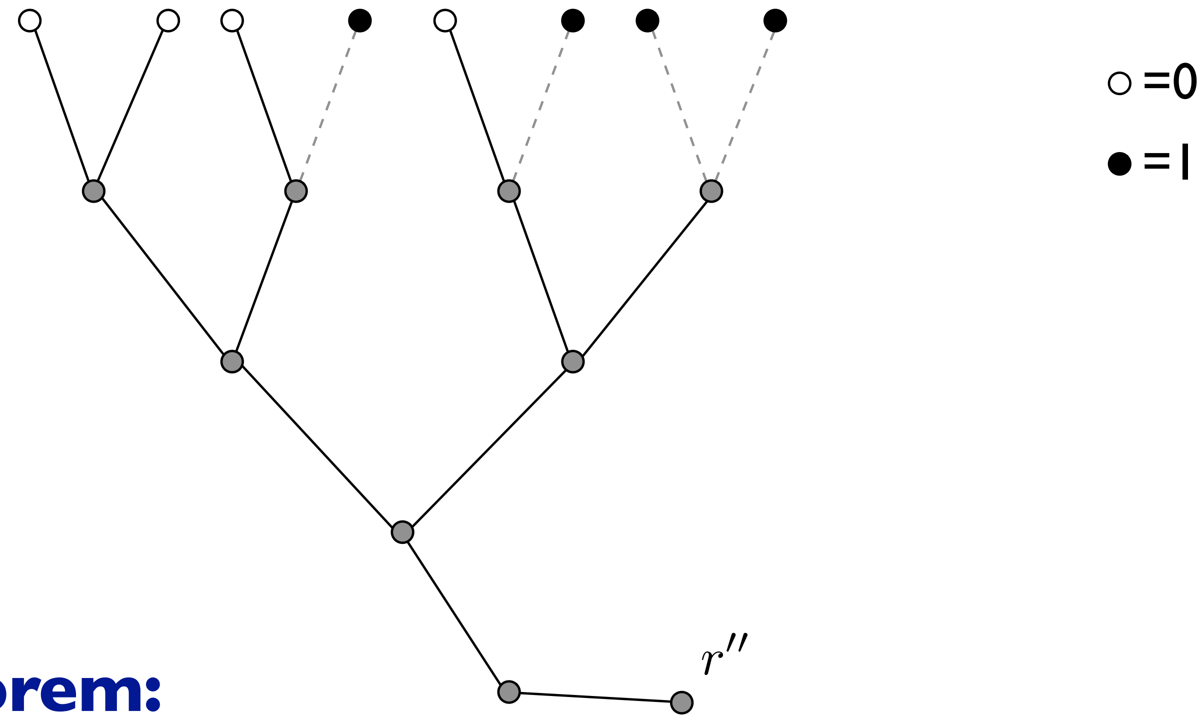
● =1

2|E| dimensions

|V| dimensions



- **Main Theorem:**
  - Adjacency matrix A_G has eigenvalue E=0 eigenvector with $\Omega(1)$ support on r" when $\varphi(x)=0$.
  - A_G has no eigenvalues $E \in (-1/\sqrt{N}, 1/\sqrt{N})$ with support on r" when $\varphi(x)=0$.



- **Main Theorem:**
  - Adjacency matrix A_G has eigenvalue E=0 eigenvector with Ω(1) support on r" when φ(x)=0.
  - A_G has no eigenvalues E∈(-1/√N,1/√N) with support on r" when φ(x)=0.

- ∴ Phase estimation to precision 1/√N (time √N), starting at r",

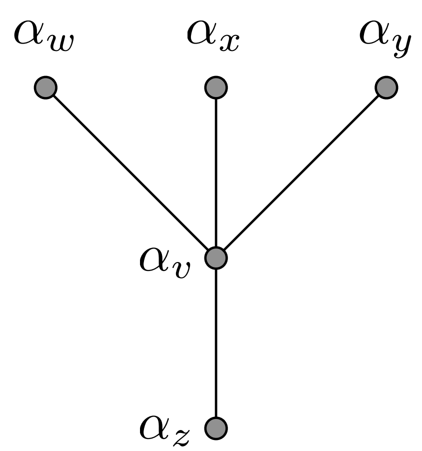
  evaluates φ(x).

- **Theorem:** $\varphi(x)=0 \Longleftrightarrow \exists$ an E=0 eigenstate of A_G supported on root r.

- **Theorem:** φ(x)=0 $\Longleftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0 constraint…



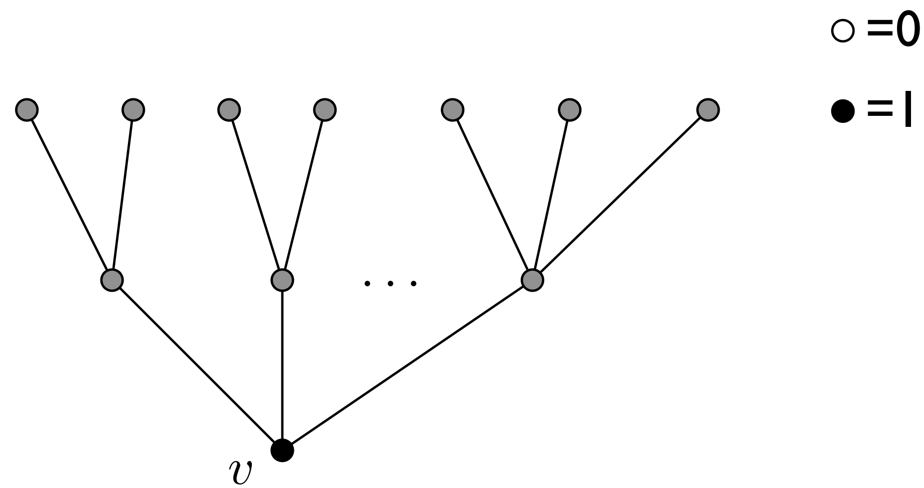
$$\alpha_w + \alpha_x + \alpha_y + \alpha_z = E\alpha_v = 0$$

- **Theorem:** $\varphi(x)=0 \Longleftrightarrow \exists$ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v : \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If $\varphi(v)=1$, every E=0 eigenstate $|\alpha\rangle$ of T_v has $\alpha_v=0$.



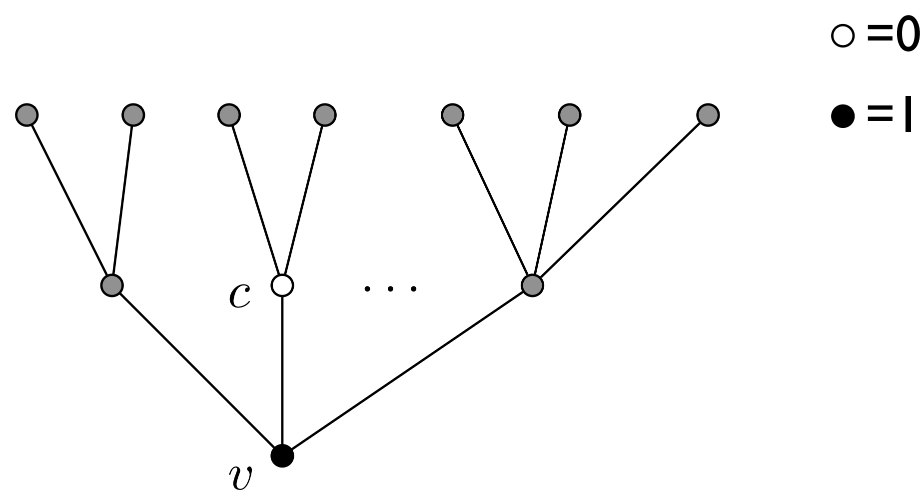
$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

- **Theorem:** $\varphi(x)=0 \iff \exists$ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v: \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If $\varphi(v)=1$, every E=0 eigenstate $|\alpha\rangle$ of T_v has $\alpha_v=0$.



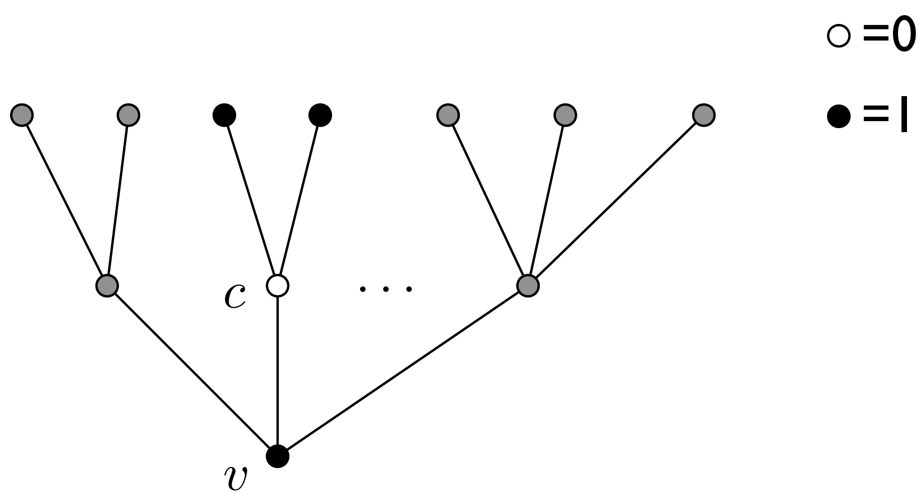
$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

- **Theorem:** $\varphi(x)=0 \iff \exists$ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v : \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If $\varphi(v)=1$, every E=0 eigenstate $|\alpha\rangle$ of T_v has $\alpha_v=0$.



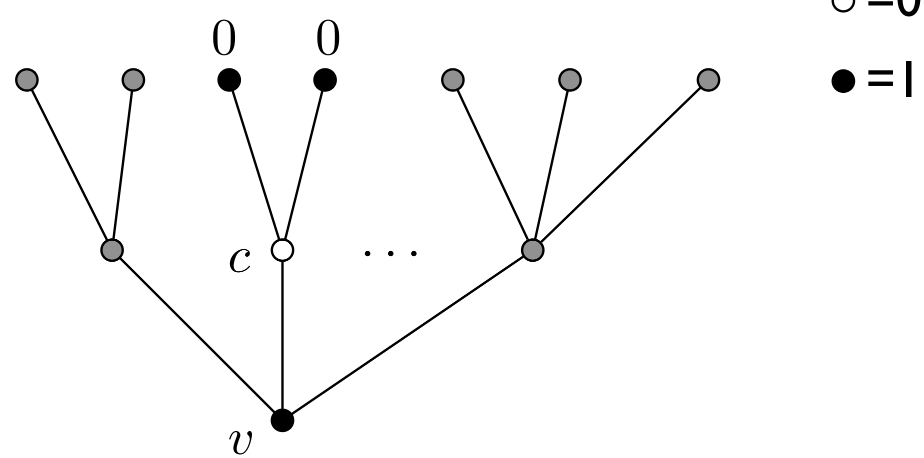
$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

- **Theorem:** φ(x)=0 $\Longleftrightarrow$ $\exists$ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v : \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0.



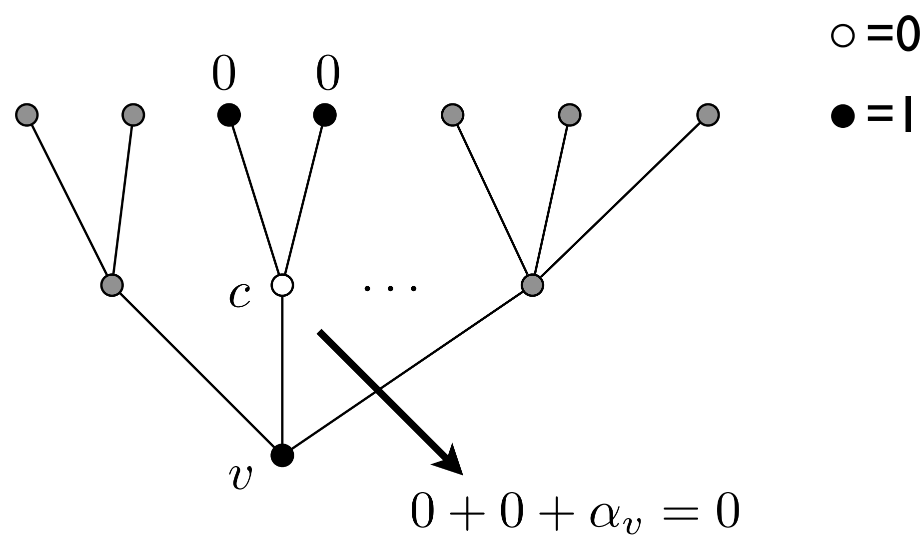
$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

- **Theorem:** φ(x)=0 $\Longleftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v: \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓



$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

- **Theorem:** φ(x)=0 $\Longleftrightarrow$ $\exists$ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v: \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓

- Claim 0: If φ(v)=0, $\exists$ E=0 eigenstate $|\alpha\rangle$ of T_v with $\alpha_v \neq 0$.

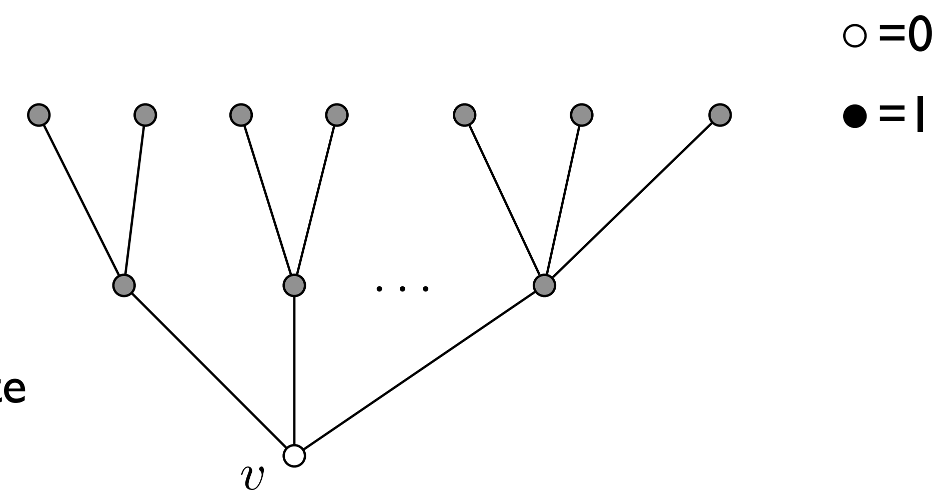
- **Theorem:** φ(x)=0 $\Longleftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v : \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓

- Claim 0: If φ(v)=0, ∃ E=0 eigenstate $|\alpha\rangle$ of T_v with $\alpha_v \neq 0$.



$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

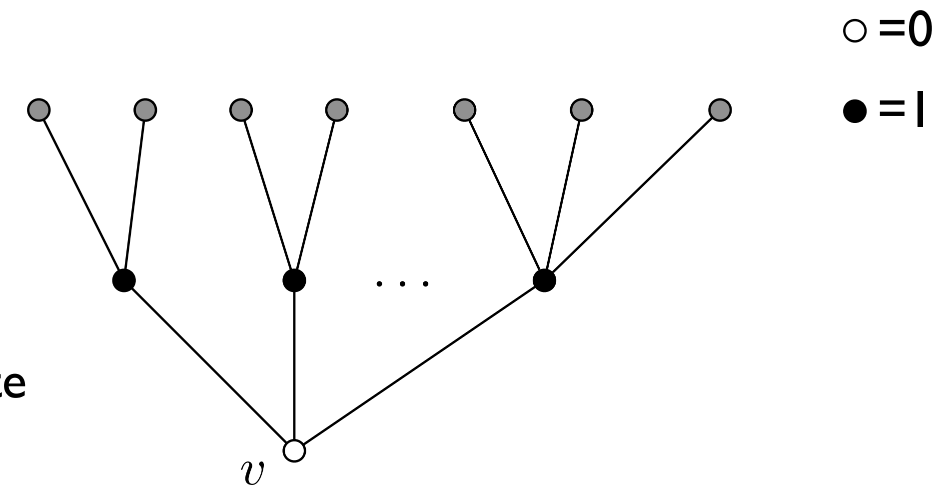
- **Theorem:** φ(x)=0 $\Longleftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v : \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓

- Claim 0: If φ(v)=0, ∃ E=0 eigenstate $|\alpha\rangle$ of T_v with $\alpha_v \neq 0$.



$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

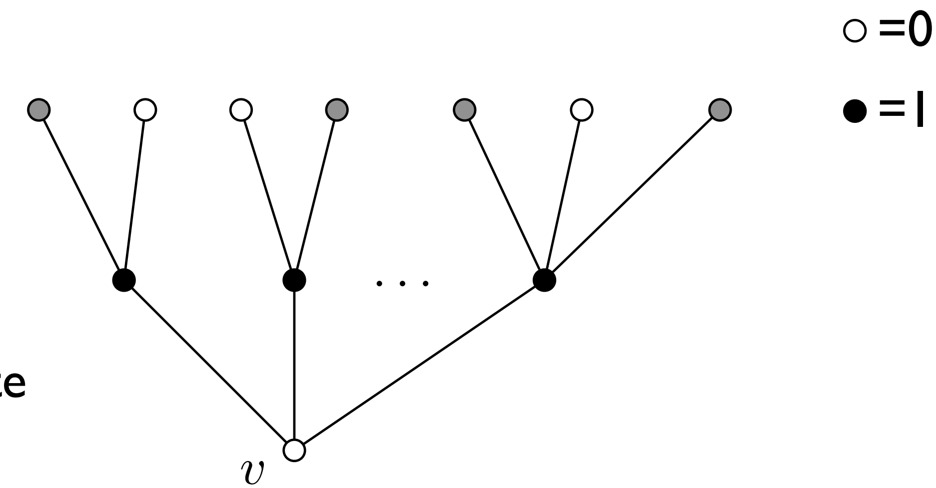
- **Theorem:** φ(x)=0 $\Longleftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v: \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓

- Claim 0: If φ(v)=0, ∃ E=0 eigenstate $|\alpha\rangle$ of T_v with $\alpha_v \neq 0$.



$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

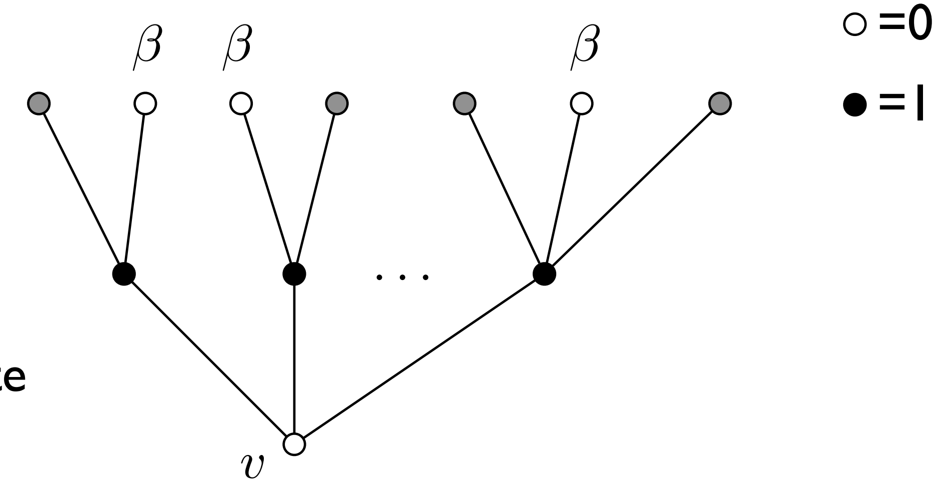
- **Theorem:** φ(x)=0 $\Longleftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v: \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓

- Claim 0: If φ(v)=0, ∃ E=0 eigenstate $|\alpha\rangle$ of T_v with α_v≠0.



$$\text{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

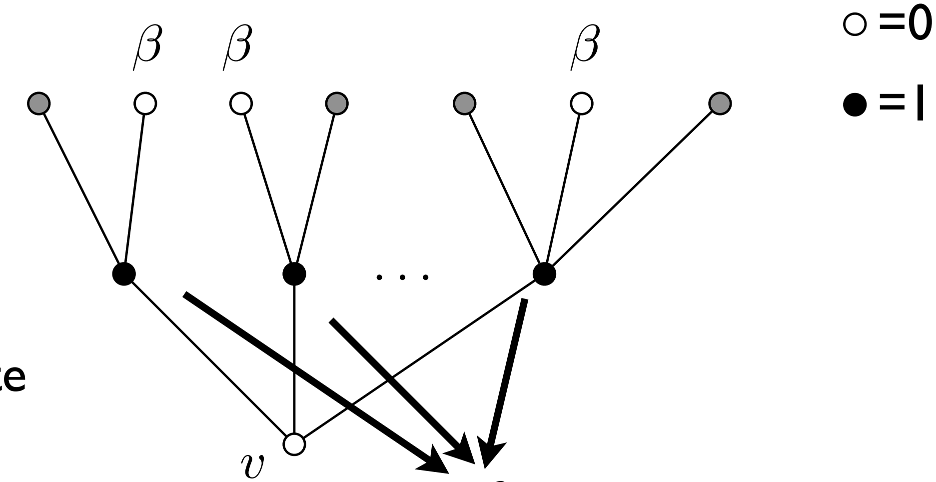
- **Theorem:** φ(x)=0 ⟺ ∃ an E=0 eigenstate of A_G supported on root r.

Proof

- E=0: $\forall v : \quad \sum_{w \sim v} \alpha_w = 0$

- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓

- Claim 0: If φ(v)=0, ∃ E=0 eigenstate $|\alpha\rangle$ of T_v with $\alpha_v \neq 0$.



$\beta + \alpha_v = 0$

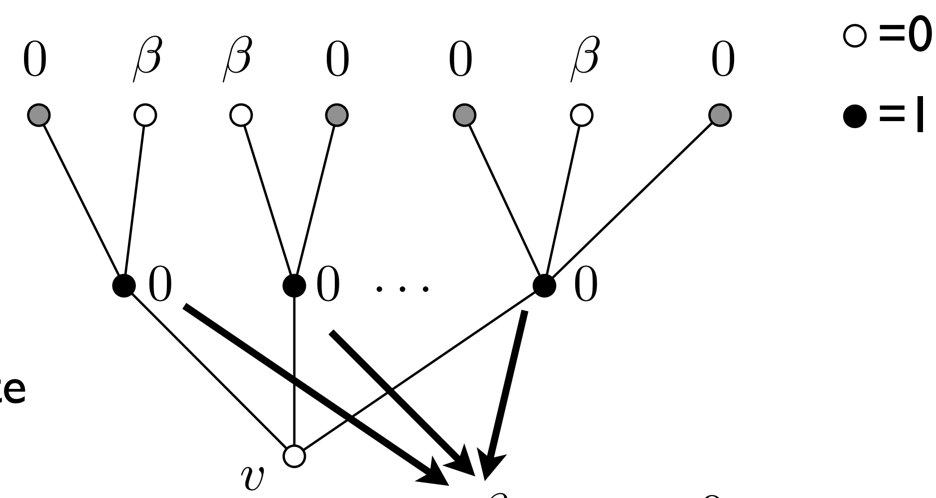
$\Rightarrow$ Set $\alpha_v = -\beta$.

$$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$$

- **Theorem:** φ(x)=0 $\Longleftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r. ✓

Proof

- E=0: $\forall v: \quad \sum_{w \sim v} \alpha_w = 0$
- Claim 1: If φ(v)=1, every E=0 eigenstate $|\alpha\rangle$ of T_v has α_v=0. ✓
- Claim 0: If φ(v)=0, ∃ E=0 eigenstate $|\alpha\rangle$ of T_v with $\alpha_v \neq 0$. ✓



$\beta + \alpha_v = 0$

$\Rightarrow$ Set $\alpha_v = -\beta$.

$\mathrm{NAND}(x_1, \ldots, x_k) = 1 - \prod_i x_i$

- **Theorem:** $\varphi(x)=0 \iff \exists$ an E=0 eigenstate of A_G supported on root r. ✓
  - Claim 1: If $\varphi(v)=1$, every E=0 eigenstate $|\alpha\rangle$ of T_v has $\alpha_v=0$.
  - Claim 0: If $\varphi(v)=0$, $\exists$ E=0 eigenstate $|\alpha\rangle$ of T_v with $\alpha_v \neq 0$.

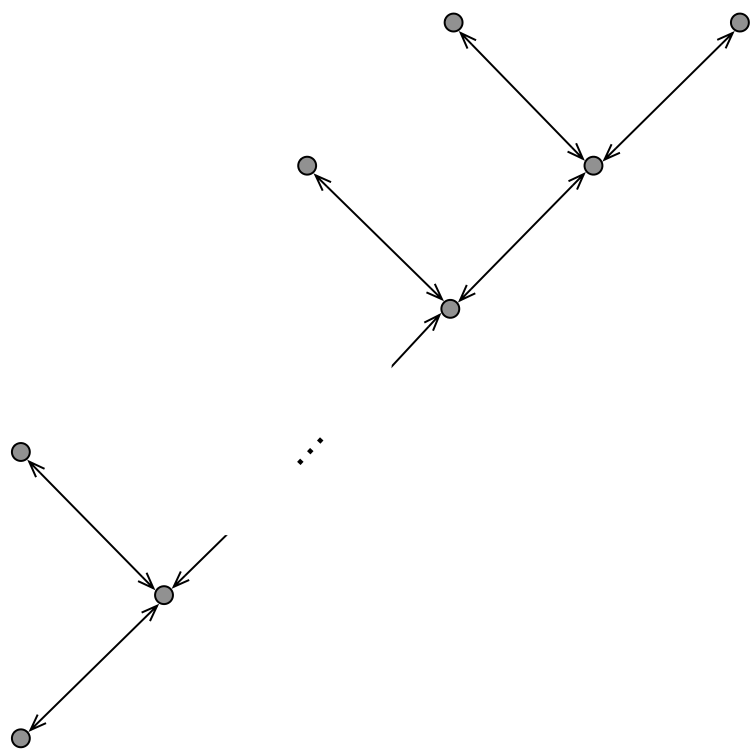
- **Main Theorem:**
  - Adjacency matrix A_G has eigenvalue E=0 eigenvector with $\Omega(1)$ support on r" when $\varphi(x)=0$.
  - A_G has no eigenvalues $E \in (-1/\sqrt{N}, 1/\sqrt{N})$ with support on r" when $\varphi(x)=0$.

- Remains to show support α_r is large ($\Omega(1)$) when $\varphi(r)=0$, and that there is a large spectral gap ($1/\sqrt{N}$) away from E=0 when $\varphi(r)=1$.
- Proofs by same induction but quantitative.

Algorithm for unbalanced trees

- Main idea: Consider quantization of same classical random walk, except with biased coins $\mathrm{weight}(p, v) = s_v^{\beta} / s_p^{1/2-\beta}$ (β arbitrary)

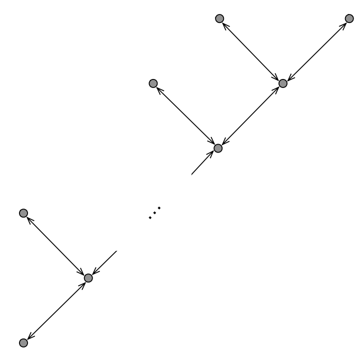
- Problem: Walk might not even reach the bottom of a deep formula in time √N



Algorithm for unbalanced trees

- Main idea: Consider quantization of same classical random walk, except with biased coins $\mathrm{weight}(p, v) = s_v^{\beta} / s_p^{1/2-\beta}$ (β arbitrary)

- Problem: Walk might not even reach the bottom of a deep formula in time √N



- Solution: Rebalance the formula tree

  Theorem: ([Bshouty, Cleve, Eberly '91, Bonet & Buss '94]) For any NAND formula φ and k ≥ 2, can efficiently construct an equivalent NAND formula φ' with

  - depth(φ') = O(k log N)
  - size(φ') ≤ $N^{1+1/\log k}$

  size-depth tradeoff (set $k=2^{\sqrt{(\log N)}}$ to balance size*depth)

Extension: Formulas on different gate sets

- What is the cost of evaluating a formula that uses other gates besides {AND, OR, NOT, NAND}?
- Example: 3-bit majority $\mathrm{MAJ3}(x_1, x_2, x_3) = \begin{cases} 1, & \text{if } x_1 + x_2 + x_3 \geq 2 \\ 0, & \text{otherwise} \end{cases}$

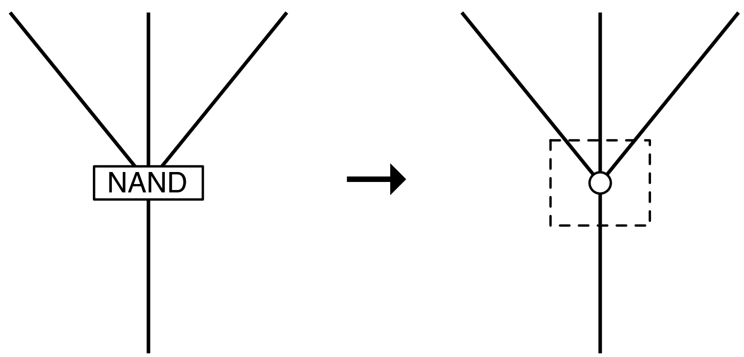
- Classical complexity to evaluate recursive MAJ3-gate tree is unknown:
  - for a depth-d balanced tree it is $\Omega\Big((7/3)^d\Big)$ and $O\Big((2.655\ldots)^d\Big)$
    [Jayram, Kumar & Sivakumar '03]
- Quantum complexity lower bound is $\Omega\Big(\sqrt{C_0(f)C_1(f)}\Big) = \Omega(2^d)$
- Quantum upper bound
  - expand into {AND, OR} gates:
    $$\mathrm{MAJ3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$$
  - size → 5^d, ∴ $O(\sqrt{5^d})=O(2.24^d)$-query algorithm

Different gate sets: Gate gadgets

- Classical complexity to evaluate recursive MAJ3-gate tree is unknown:
  - for depth-d balanced tree: $\Omega\Big((7/3)^d\Big)$ and $O\Big((2.655\ldots)^d\Big)$ [Jayram, Kumar & Sivakumar '03]
- Quantum lower bound: $\Omega\Big(\sqrt{C_0(f)C_1(f)}\Big) = \Omega(2^d)$
- Quantum upper bound: $\mathrm{MAJ3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$ ⇒ $O(\sqrt{5}^d)$

- Gate gadgets:

recall…

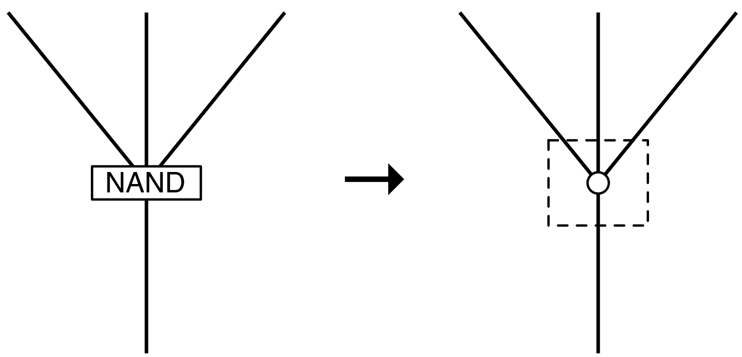


Different gate sets: Gate gadgets

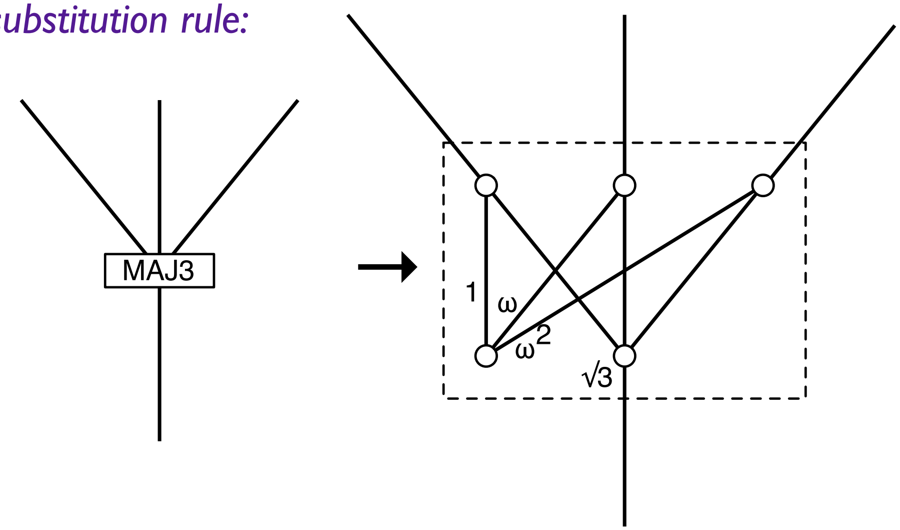
- Classical complexity to evaluate recursive MAJ3-gate tree is unknown:
  - for depth-d balanced tree: $\Omega\Big(\big(7/3\big)^d\Big)$ and $O\Big(\big(2.655\ldots\big)^d\Big)$ [Jayram, Kumar & Sivakumar '03]
- Quantum lower bound: $\Omega\Big(\sqrt{C_0(f)C_1(f)}\Big) = \Omega(2^d)$
- Quantum upper bound: $\mathrm{MAJ3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$ ⇒ O(√5^d)

- Gate gadgets:

recall…



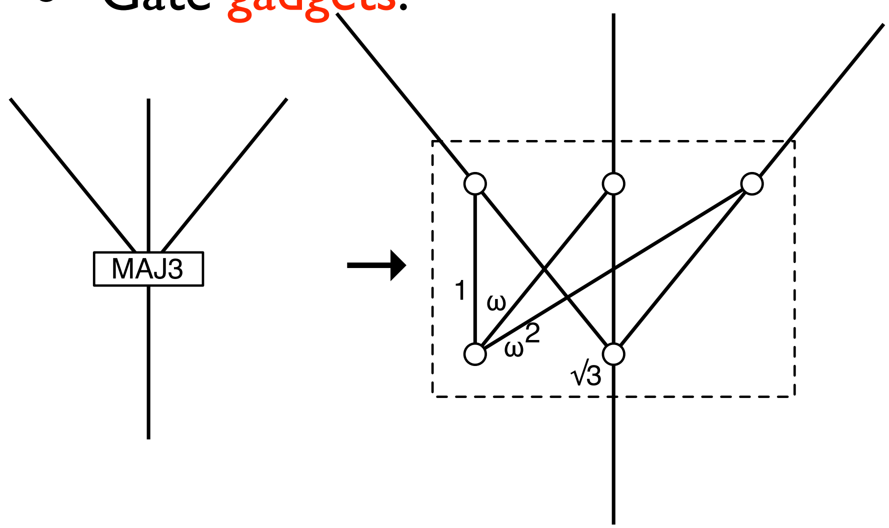
new substitution rule:



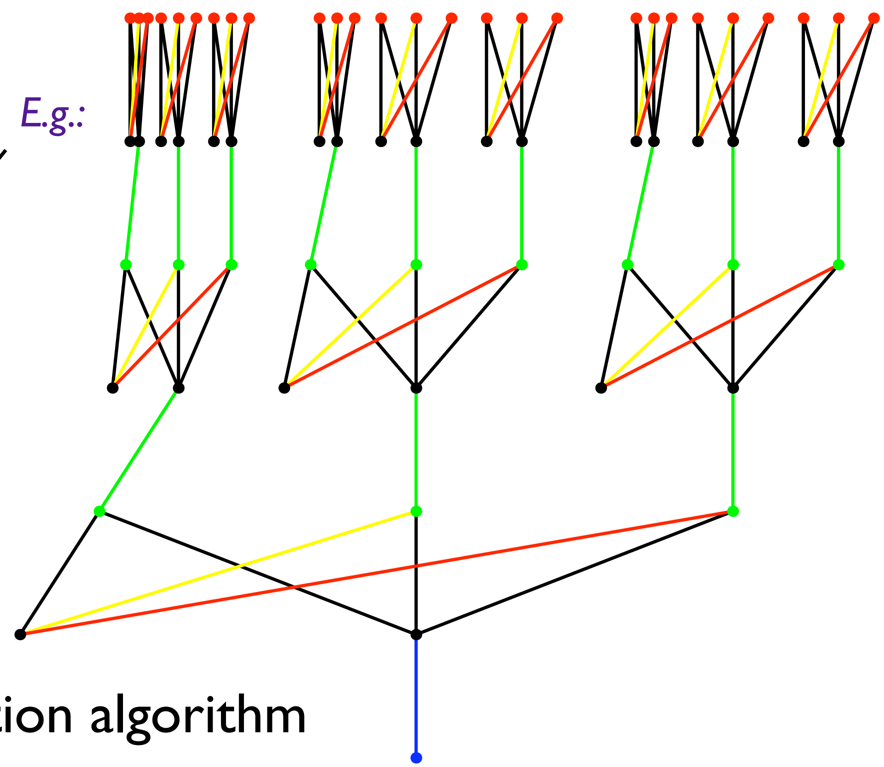
Different gate sets: Gate gadgets

- Classical complexity to evaluate recursive MAJ3-gate tree is unknown:
  - for depth-d balanced tree: $\Omega\Big(\big(7/3\big)^d\Big)$ and $O\Big(\big(2.655\ldots\big)^d\Big)$ [Jayram, Kumar & Sivakumar '03]
- Quantum lower bound: $\Omega\Big(\sqrt{C_0(f)C_1(f)}\Big) = \Omega(2^d)$
- Quantum upper bound: $\mathrm{MAJ3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$ ⇒ $O(\sqrt{5}^d)$

- Gate gadgets:



E.g.:



⇒Optimal recursive MAJ3 evaluation algorithm

Open problems

- Results:
  - $N^{½+o(1)}$-time quantum algorithm (N= #leaves) for general {AND, OR, NOT} trees (after efficient preprocessing independent of x)
  - $O(\sqrt{N})$-query qu. alg. for "approximately balanced" {AND, OR, NOT} trees (optimal!)
  - $O(N^{\log_3 2})$-query qu. alg. for balanced MAJ-3 formula trees (optimal!)
- Open: Extension to allow other gates, e.g.,
  - 3-bit not-all-equal = $NOR(AND(x_1,x_2,x_3), AND(x_1^*,x_2^*,x_3^*))$
  - 6-bit (monotone modified) Kushilevitz's function
  - Of interest to understand quantum lower bound separation ADV versus ADV± [Høyer, Lee, Špalek '07]
- Open: Noisy oracle inputs (à la [Høyer, Mosca, de Wolf '03])?
- Open Classical ?: Is [BCE'91] formula rebalancing optimal?
  - Does there exist formula φ, k such that every equivalent φ' of depth at most k log N has size(φ') ≥ $N^{1+1/\log k}$?