IEGULDĪJUMS TAVĀ NĀKOTNĒ

New developments in quantum algorithms

Andris Ambainis

University of Latvia

European Social Fund project "Datorzinātnes pielietojumi un tās saiknes ar kvantu fiziku"
Nr.2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044

What is quantum computation?

- New model of computing based on quantum mechanics.
- Quantum circuits, quantum Turing machines.
- More powerful than conventional models.
- Small-scale implementations exist (up to 12 quantum bits).

Shor’s algorithm

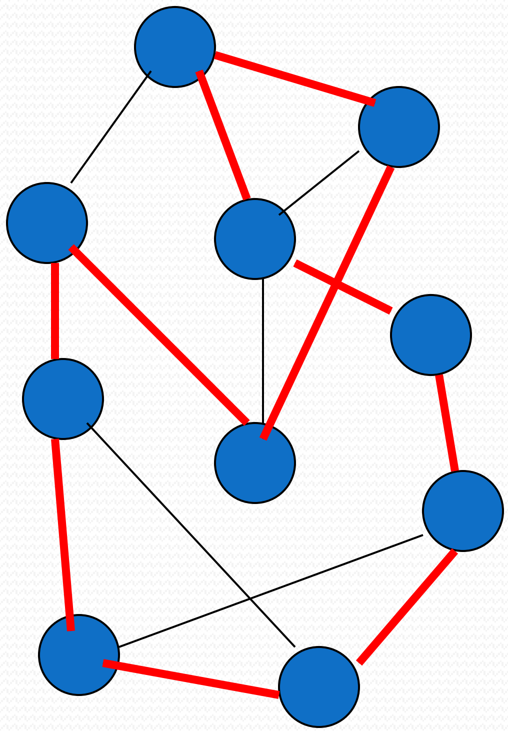
- Factoring: given N=pq, find p and q.
- Best algorithm - $2^{O(n^{1/3})}$, n – number of digits.
- Quantum algorithm - $O(n^3)$ [Shor, 94].
- Cryptosystems based on hardness of factoring/discrete log become insecure.

Grover's search

0	1	0	$\cdots$	0
x_1	x_2	x_3		x_N

- Find i such that $x_i=1$.
- Queries: ask i, get x_i.
- Classically, N queries required.
- Quantum: $O(\sqrt{N})$ queries [Grover, 96].
- Speeds up any search problem.

NP-complete problems



- Does this graph have a Hamiltonian cycle?

- Hamiltonian cycles are:
  - Easy to verify;
  - Hard to find (too many possibilities).

Quantum algorithm

0	1	0	…	0
x_1	x_2	x_3		x_N

- Let N – number of possible Hamiltonian cycles.
- Black box = algorithm that verifies if the i^{th} candidate – Hamiltonian cycle.
- Quantum algorithm with $O(\sqrt{N})$ steps.

Applicable to any search problem

Pell's equation

- Given d, find the smallest solution (x, y) to $x^2-dy^2=1$.
- Probably harder than factoring and discrete logarithm.
- Best classical algorithms:
  - for factoring;
  - $2^{O(\sqrt{N})}$ for discrete logarithm.

$$2^{O(N^{1/3})}$$

Hallgren, 2001: Quantum algorithm for Pell's equation.

Number theory and algebraic problems

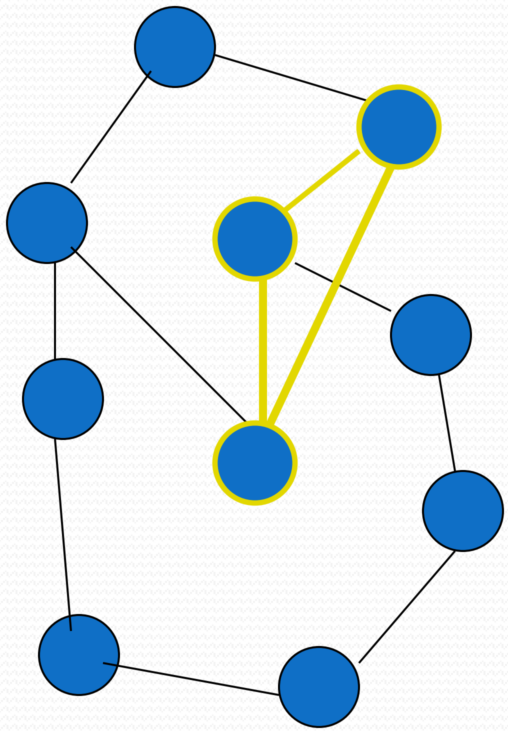
- Polynomial time quantum algorithms:
  - Factoring [Shor, 94]
  - Discrete logarithm [Shor, 94];
  - Pell's equation [Hallgren, 02].
  - Principal ideal problem [Hallgren, 02].
  - Computing the unit group [Hallgren, 05].

Element distinctness [A, 2004]

7	9	2	…	1
x_1	x_2	x_3		x_N

- Numbers $x_1, x_2, \ldots, x_N$.
- Determine if two of them are equal.
- Classically: N queries.
- Quantum: $O(N^{2/3})$.

Triangle finding [Magniez, Santha, Szegedy, 03]



- Graph G with n vertices.
- n^2 variables x_{ij}; $x_{ij}=1$ if there is an edge (i, j).
- Does G contain a triangle?
- Classically: $O(n^2)$.
- Quantum: $O(n^{1.3})$.

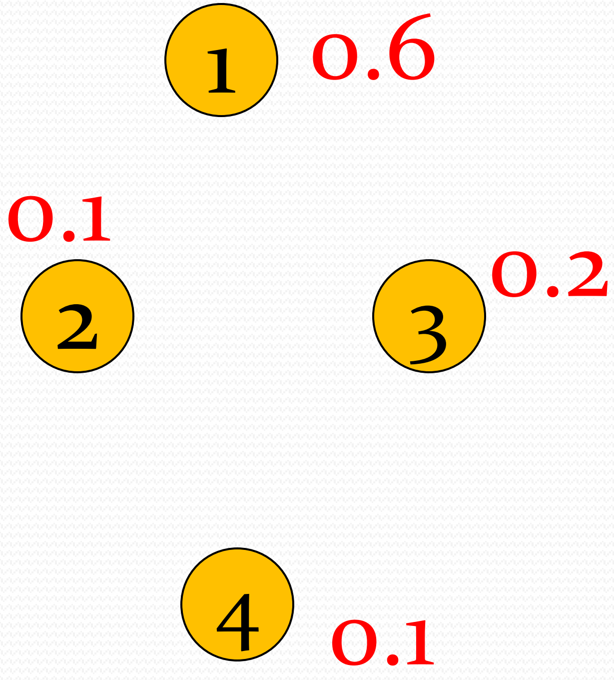
Talk outline

1. The model.
2. Recent developments in quantum algorithms.
   a) Formula evaluation;
   b) Systems of linear equations;

Part 1

The model

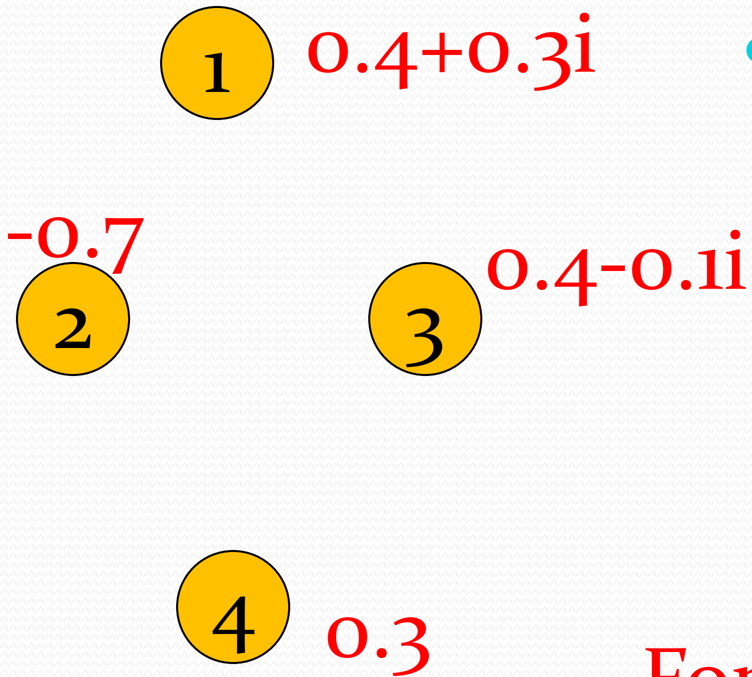
Probabilistic computation



- Probabilistic system with finite state space.
- Current state: probabilities p_i to be in state i.

$$\sum_i p_i = 1$$

Quantum computation

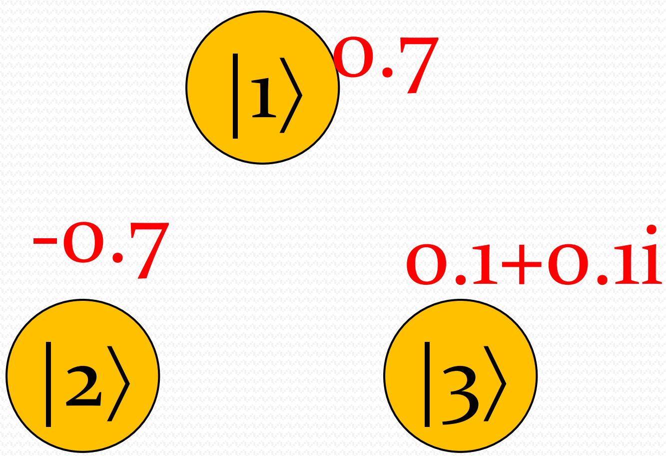


- Current state: amplitudes α_i to be in state i.

$$\sum_i |\alpha_i|^2 = 1$$

For most purposes, real (but negative) amplitudes suffice.

Notation

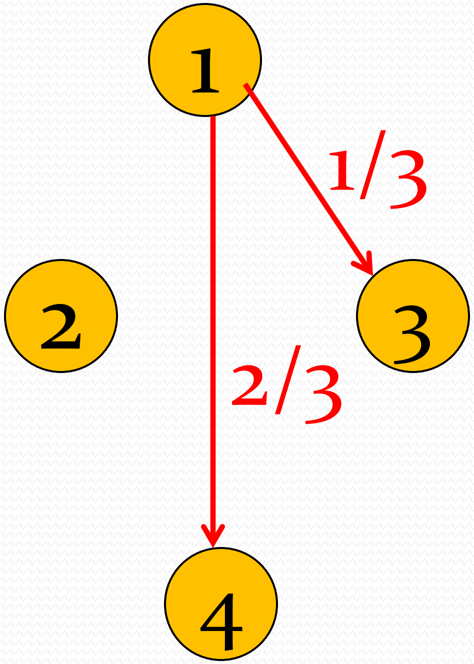


- Basis states $|1\rangle$, $|2\rangle$, $|3\rangle$.

$$|\Psi\rangle = 0.7\,|1\rangle - 0.7\,|2\rangle + (0.1+0.1i)|3\rangle$$

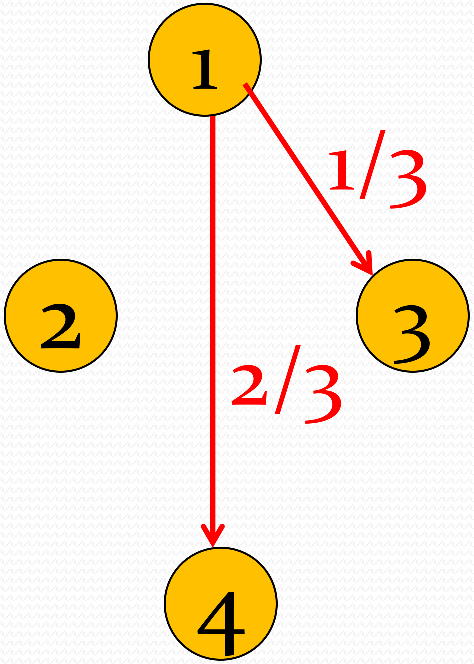
$$|\Psi\rangle = \begin{pmatrix} 0.7 \\ -0.7 \\ 0.1+0.1i \end{pmatrix}$$

Probabilistic computation



- Pick the next state, depending on the current one.

Probabilistic computation



- Transitions: r_{ij} - probabilities to move from i to j.

$$p'_j = \sum_i p_i r_{ij}$$

Probabilistic computation

- Probability vector $(p_1, \ldots, p_M)$.
- Transitions:

$$\begin{pmatrix} p'_1 \\ \ldots \\ p'_M \end{pmatrix} = \begin{pmatrix} r_{11} & \ldots & r_{1M} \\ \ldots & \ldots & \ldots \\ r_{M1} & \ldots & r_{MM} \end{pmatrix} \begin{pmatrix} p_1 \\ \ldots \\ p_M \end{pmatrix}$$

transition probabilities

after the transition

before the transition

Allowed transitions

$$\begin{pmatrix} p'_1 \\ \cdots \\ p'_M \end{pmatrix} = \begin{pmatrix} r_{11} & \cdots & r_{1M} \\ \cdots & \cdots & \cdots \\ r_{M1} & \cdots & r_{MM} \end{pmatrix} \begin{pmatrix} p_1 \\ \cdots \\ p_M \end{pmatrix}$$

- R –stochastic:
  - If $\Sigma_i p_i = 1$, then $\Sigma_i p'_i = 1$.

Quantum computation

- Amplitude vector $(\alpha_1, \ldots, \alpha_M)$, $\cdot \sum_i |\alpha_i|^2 = 1$
- Transitions:

$$\begin{pmatrix} \alpha'_1 \\ \ldots \\ \alpha'_M \end{pmatrix} = \begin{pmatrix} u_{11} & \ldots & u_{1M} \\ \ldots & \ldots & \ldots \\ u_{M1} & \ldots & u_{MM} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \ldots \\ \alpha_M \end{pmatrix}$$

transition matrix

after the transition

before the transition

Allowed transitions

$$\begin{pmatrix} \alpha'_1 \\ \cdots \\ \alpha'_M \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1M} \\ \cdots & \cdots & \cdots \\ u_{M1} & \cdots & u_{MM} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_M \end{pmatrix}$$

- U – unitary:
  - If $\sum_i |\alpha_i|^2 = 1$, then $\sum_i |\alpha'_i|^2 = 1$.

Equivalent to $UU^+=I$.

Quantum computing vs. nature

Quantum computing

- Unitary transformations U.
- Transformation U performed in one step.
- No intermediate states.

Quantum physics

- Physical evolution – continuous time.
- Forces acting on a physical system – Hamiltonian H.

Evolution for time t:

$$U = e^{-iHt}$$

Summary so far

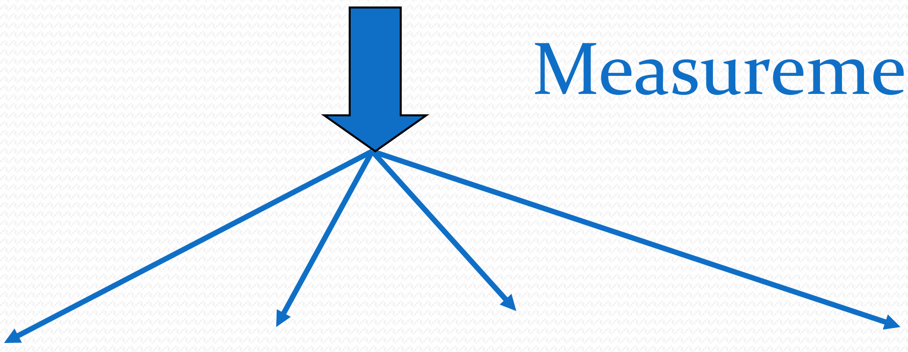
- Quantum ≈ probabilistic with complex probabilities.
- Instead of $\Sigma_i p_i = 1$ we have $\sum_i |\alpha_i|^2 = 1$ (l_2 norm instead of l_1).

How do we go from quantum world to conventional world?

Measurement

Quantum state:

$$\alpha_1 |1\rangle + \alpha_2 |2\rangle + \ldots + \alpha_M |M\rangle$$



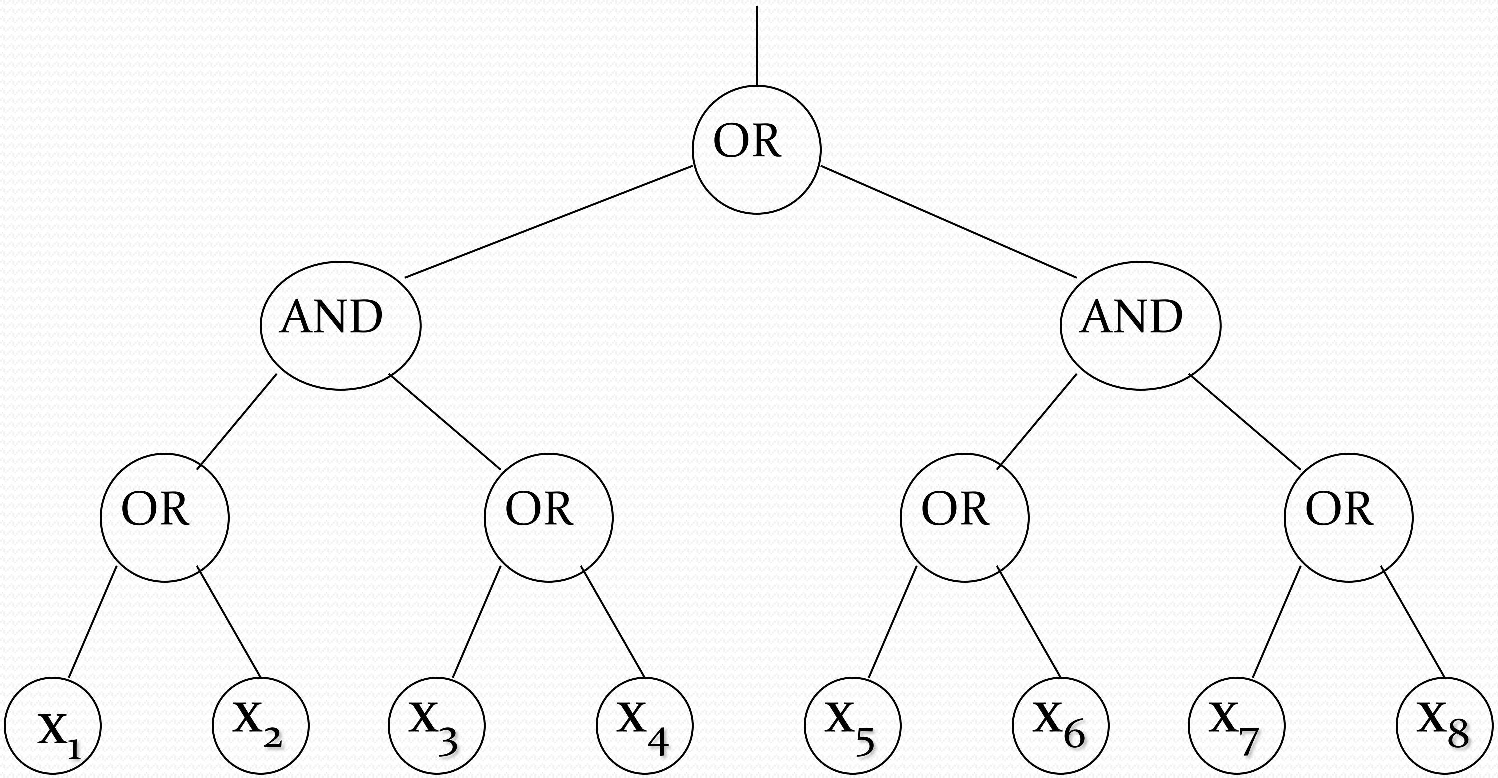
Measurement

	1	2	…	M
prob.	$\lvert\alpha_1\rvert^2$	$\lvert\alpha_2\rvert^2$		$\lvert\alpha_M\rvert^2$

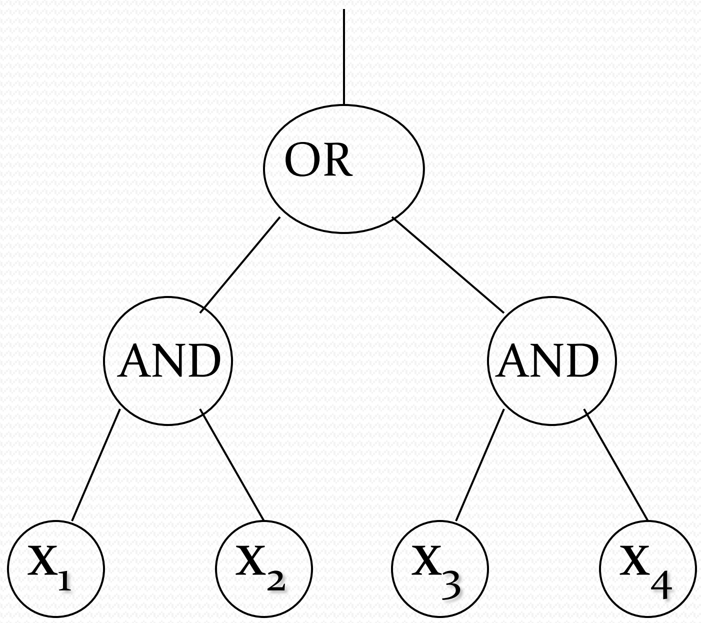
Part 2a

Formula evaluation

AND-OR tree



Evaluating AND-OR trees

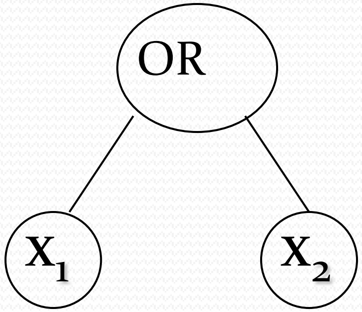


- Variables x_i accessed by queries to a black box:
  - Input i;
  - Black box outputs x_i.
- Quantum case:

$$\sum_i a_i |i\rangle \rightarrow \sum_i a_i (-1)^{x_i} |i\rangle$$

- Evaluate T with the smallest number of queries.

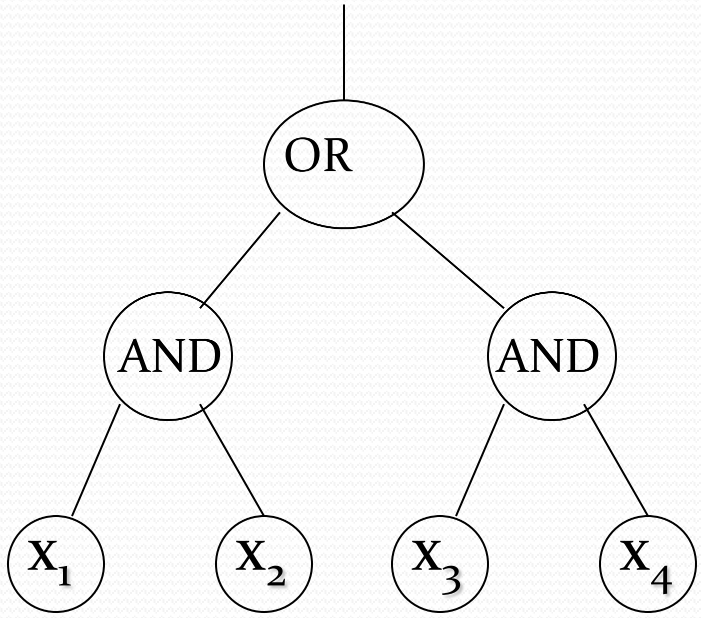
Motivation



- Vertices = chess positions;
- Leaves = final positions;
- x_i=1 if the 1st player wins;
- At internal vertices, AND/OR evaluates whether the player who makes the move can win.

How well can we play chess if we only know the position tree?

Results (up to 2007)

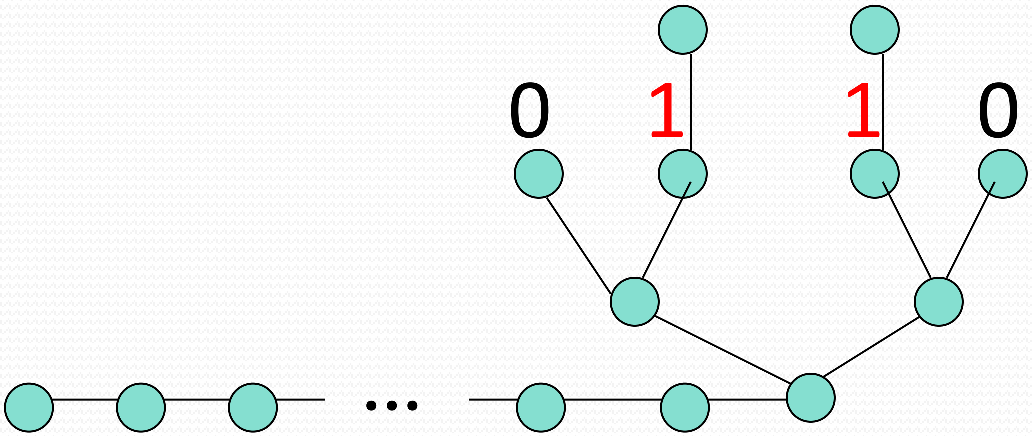


- Full binary tree of depth d.
- $N=2^d$ leaves.
- Deterministic: $\Omega(N)$.
- Randomized [SW,S]: $\Theta(N^{0.753\ldots})$.
- Quantum?
- Easy q. lower bound: $\Omega(\sqrt{N})$.

New results

- [Farhi, Gutman, Goldstone, 2007]:O(√N) time algorithm for evaluating full binary trees in Hamiltonian query model.
- [A, Childs, Reichardt, Spalek, Zhang, 2007]: $O(N^{1/2+o(1)})$ time algorithm for evaluating any formulas in the usual query model.

Augmented tree

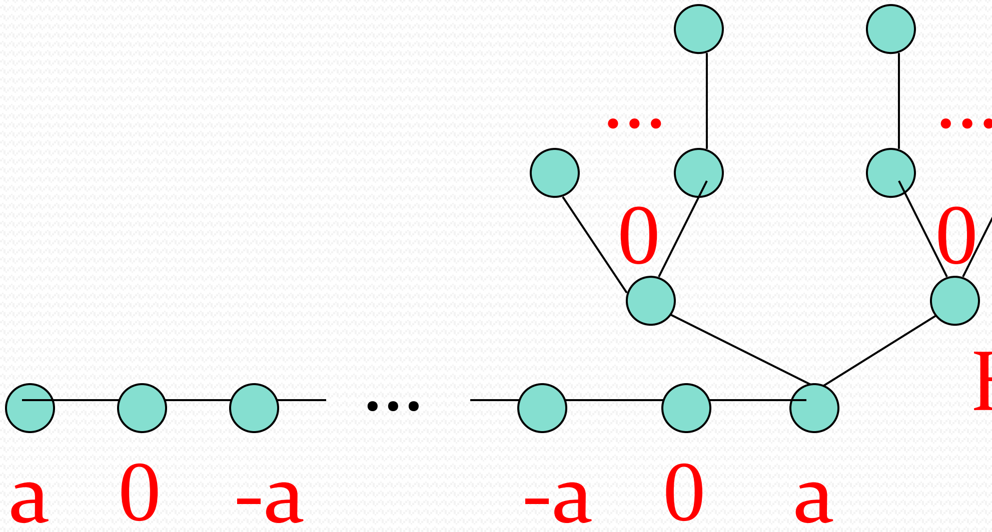


Finite "tail" in one direction

Finite tail algorithm

Starting state:

$$|\Psi_{start}\rangle = \sum_j (-1)^j a|2j\rangle$$



Hamiltonian H,

H – adjacency matrix

What happens?

- If T=0, the state stays almost unchanged.
- If T=1, the state "scatters" into the tree.

Run for $O(\sqrt{N})$ time, check if the state $|\Psi\rangle$ is close to the starting state $|\Psi_{start}\rangle$.

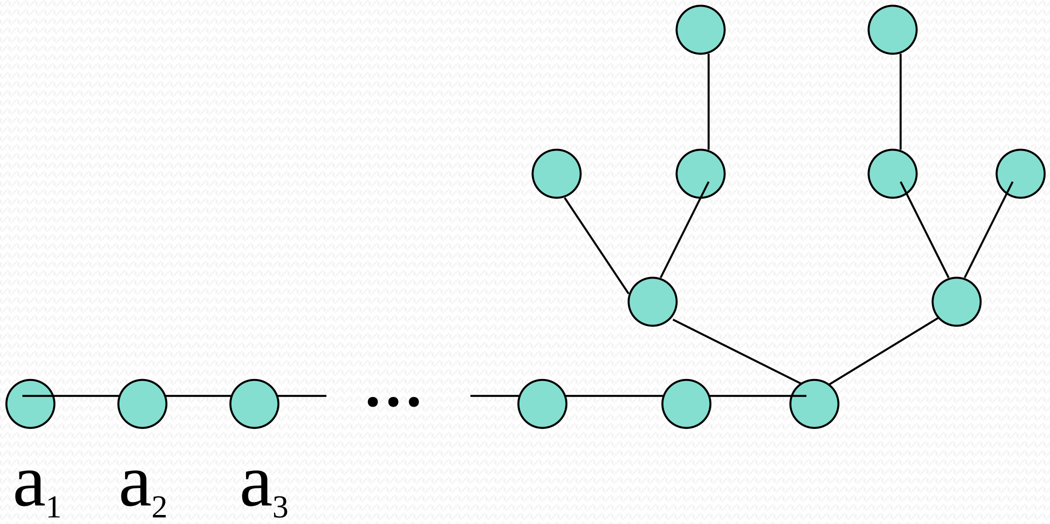
When is the state unchanged?

- H – forces acting on the system.
- (State $|\Psi\rangle$ unchanged) $\leftrightarrow$ $H|\Psi\rangle=0$.

$$e^{-iHt}\,|\Psi\rangle = |\Psi\rangle \Leftrightarrow H\,|\Psi\rangle = 0.$$

What does $H|\Psi\rangle = 0$ mean?

H – adjacency matrix

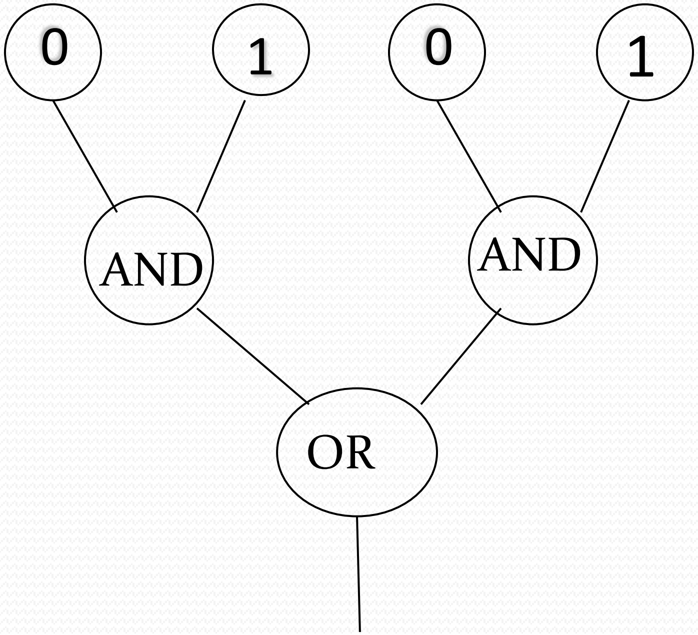


$$H|\Psi\rangle = (b_i),$$

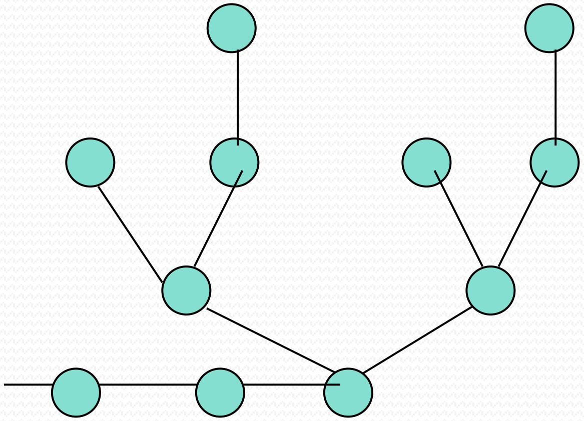
$$b_i = \sum_{(i,j)-edge} a_j$$

$H|\Psi\rangle = 0 \leftrightarrow$ for each i: $\sum_{(i,j)-edge} a_j = 0$

Example

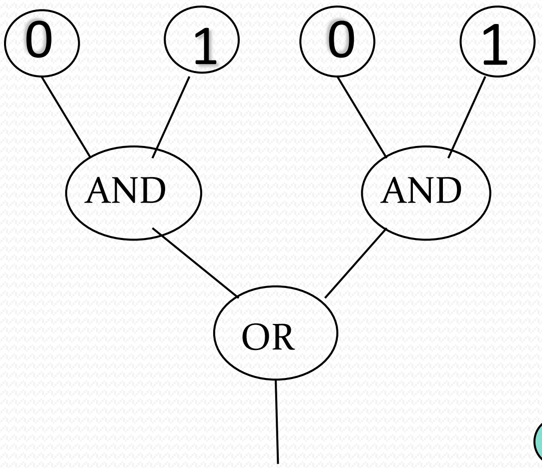


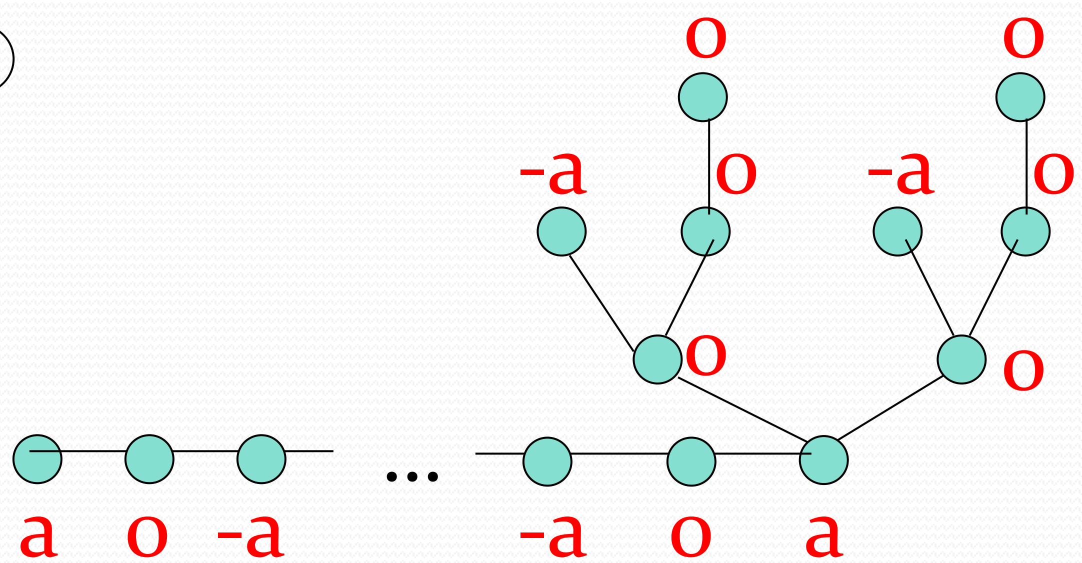
Formula



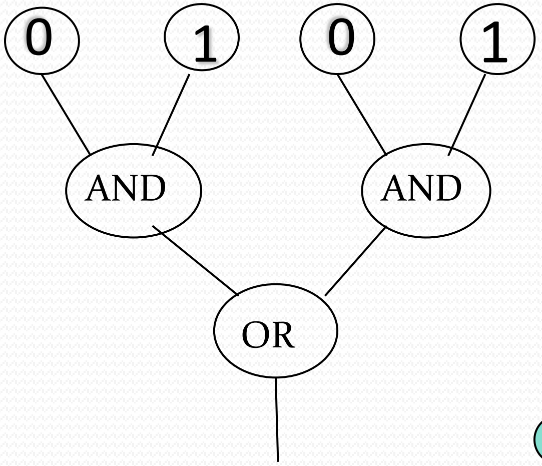
Augmented tree

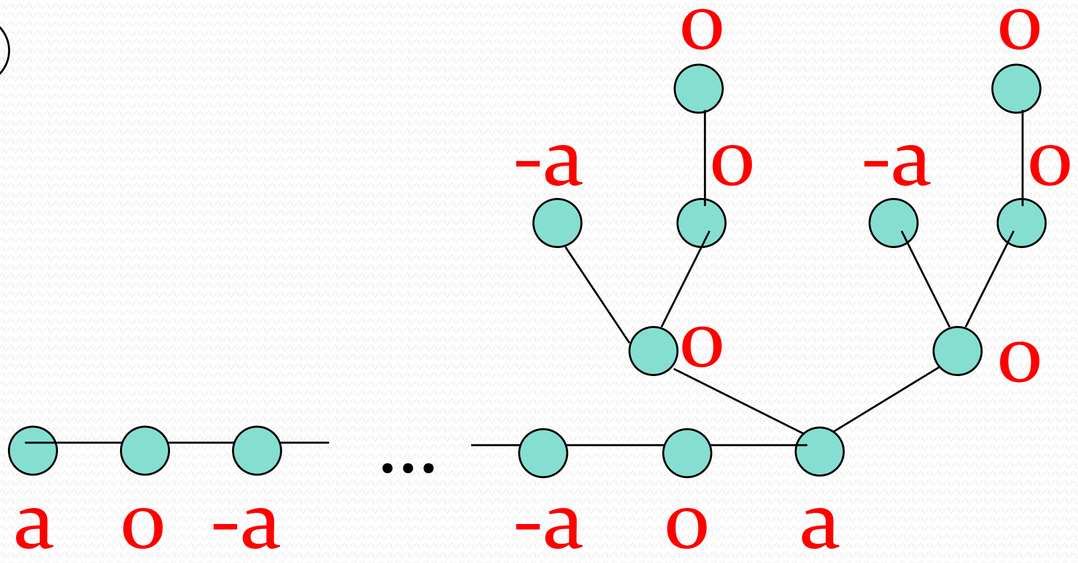
$H|\Psi\rangle = 0$ state





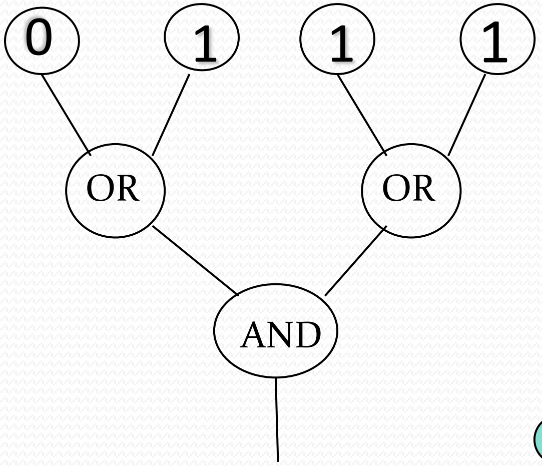
General property

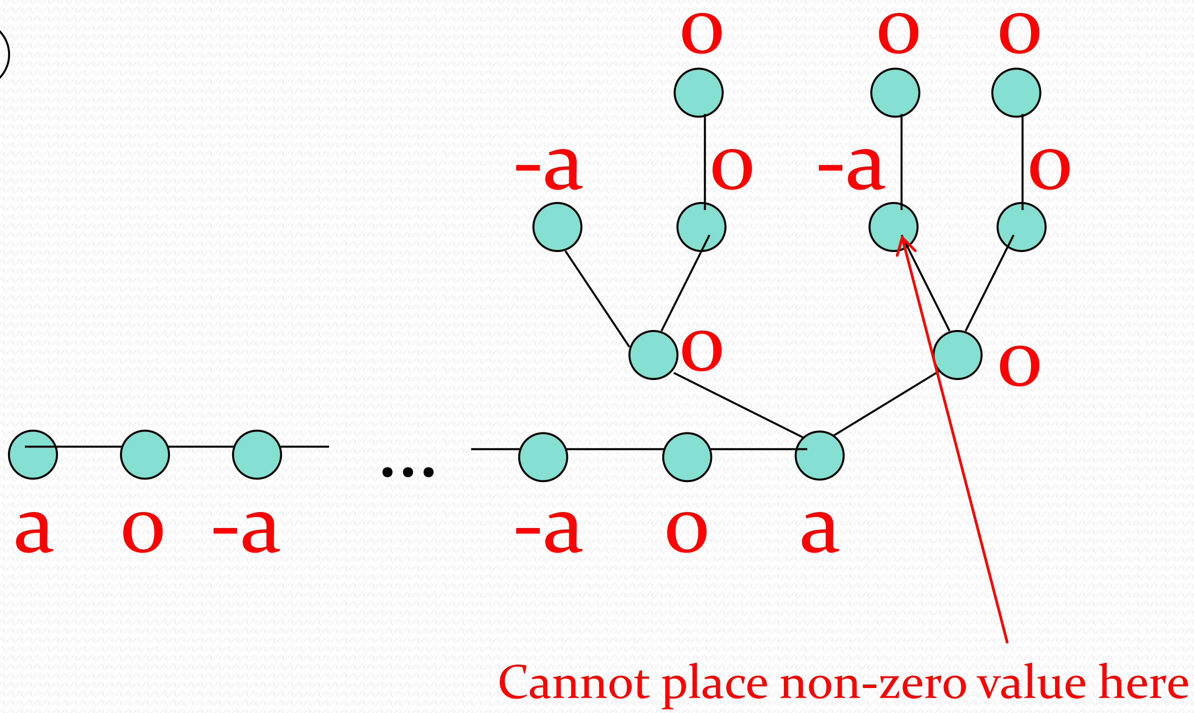




Leaves with non-zero a_i form a certificate of T=0.

T=1 case





No |Ψ⟩ with H|Ψ⟩=0.

Summary

- [Farhi, Gutman, Goldstone, 2007] Hamiltonian algorithm;
- [A, Childs, et al., 2007] Discrete time algorithm.
- $O(\sqrt{N})$ time for full binary tree;
- $O(\sqrt{N}d)$ for any formula of depth d;
- $O(N^{1/2+o(1)})$ for any formula.
- Improved to $O(\sqrt{N} \log N)$ by [Reichardt, 2010].

Span programs [Karchmer, Wigderson, 1993]

- Target vector v.
- Input $x_1, \ldots, x_N \rightarrow$ vectors $v_1, \ldots, v_M$.
- Output $F(x_1, \ldots, x_N) = 1$ if there exist $v_{i1}, v_{i2}, \ldots, v_{ik}$:

$$v = v_{i1} + v_{i2} + \ldots + v_{ik}.$$

Span program example

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \alpha \end{pmatrix} \quad \begin{pmatrix} 1 \\ \beta \end{pmatrix}$$

Target

$x_1=1$ $x_2=1$ $x_3=1$

Span program example

x1=1, x2=1, x3=0

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \alpha \end{pmatrix} \quad \begin{pmatrix} 1 \\ \beta \end{pmatrix}$$

Target $\qquad x_1=1 \quad x_2=1 \quad x_3=1$

Output = 1.

Span program example

x1=1, x2=0, x3=0

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ \alpha \end{pmatrix} \qquad \begin{pmatrix} 1 \\ \beta \end{pmatrix}$$

Target $\qquad x_1=1 \qquad x_2=1 \qquad x_3=1$

Output = 0.

Span program example

$$\begin{pmatrix}0\\1\end{pmatrix} \qquad \begin{pmatrix}1\\1\end{pmatrix} \quad \begin{pmatrix}1\\\alpha\end{pmatrix} \quad \begin{pmatrix}1\\\beta\end{pmatrix}$$

Target $\qquad x_1=1 \qquad x_2=1 \qquad x_3=1$

Output = "yes" if ≥2 of $x_i=1$.

Composing span programs

- Span program S_1 with target t_1.
- Span program S_2 with target t_2.

Span program $S_1 \cup S_2$ with target t_1+t_2.

Answers 1 if both S_1 and S_2 answer 1.

F_1, F_2 ➔ F_1 AND F_2

Span programs [Reichardt, Špalek, 2008]

Logic formula of size T

↓

Span program with witness size T

↓

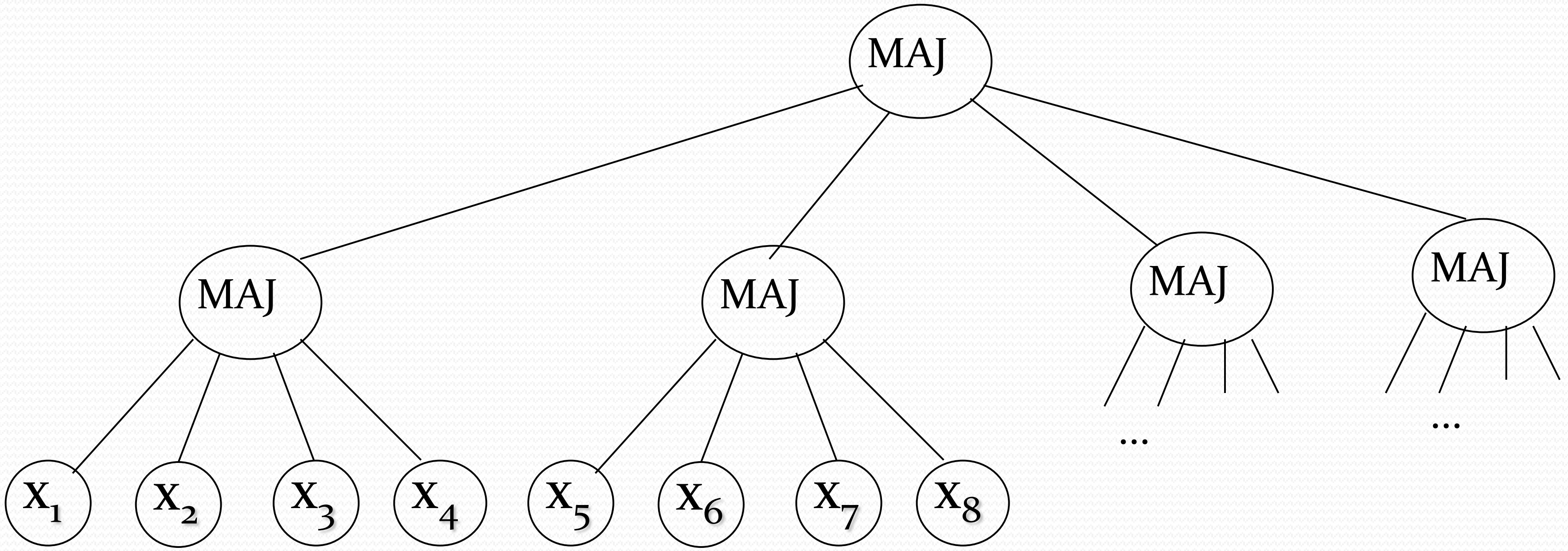
O(√T) query quantum algorithm

Far-reaching generalization of formula evaluation

Example

- MAJ(x_1, x_2, x_3 , x_4)=1 if at least 2 x_i are equal to 1.
- Formula size: 8.
- Span program: 6.

Iterated thresholds



d levels – formula of size 8^d, span program 6^d.

$O(\sqrt{6^d})$ quantum algorithm

Span programs [Reichardt, 2009]

Span program with witness size T

|||

O(√T) query quantum algorithm

Adversary bound [A, 2001, Hoyer, Lee, Špalek, 2007]

- Boolean function $f(x_1, \ldots, x_N)$;
- Inputs $x = (x_1, \ldots, x_N)$;
- Matrix A: $A[x, y] \neq 0$ only if $f(x) \neq f(y)$
- <u>Theorem</u> Computing f requires

$$\frac{\lambda(A)}{\max_i \lambda(A \bullet D_i)}$$

quantum queries

Span programs [Reichardt, 2009]

Optimal adversary bound

↓

Semidefinite program (SDP)

↓

Dual SDP

↓

Optimal span program

Span programs [Reichardt, 2009]

Span program with witness size T

$\equiv$

$O(\sqrt{T})$ query quantum algorithm

Summary

- Span programs = optimal quantum algorithms.
- Open problem: how to design good span programs?
- Quantum algorithm for perfect matchings?

Part 2b

Solving systems of linear equations

The problem

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2$$
$$\ldots$$
$$a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N = b_N$$

- Given a_{ij} and b_i, find x_i.
- Best classical algorithm: $O(N^{2.37\ldots})$.

Obstacles to quantum algorithm

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N &= b_2 \\
&\ldots \\
a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N &= b_N
\end{aligned}
$$

- Obstacle 1: takes time $O(N^2)$ to read all a_{ij}.
- Solution: query access to a_{ij}.
- Grover: search N items with $O(\sqrt{N})$ quantum queries.
- Obstacle 2: takes time O(N) to output all x_i.

Harrow, Hassidim, Lloyd, 2008

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2$$
$$\ldots$$
$$a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N = b_N$$

$$\text{Output} = \sum_{i=1}^{N} x_i |i\rangle$$

- Measurement $\rightarrow$ i with probability x_i^2.
- Estimating $c_1x_1+c_2x_2+\ldots+c_Nx_N$.

  Seems to be difficult classically.

Harrow, Hassidim, Lloyd, 2008

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2$$
$$\ldots$$
$$a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N = b_N$$

- Running time for producing $\sum_{i=1}^{N} x_i |i\rangle$: O(log^c N), but with dependence on two other parameters.
- Exponential speedup, if the other parameters are good.

The main ideas

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2$$

$$\ldots$$

$$a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N = b_N$$

$$\sum_{i=1}^{N} b_i |i\rangle \longrightarrow \sum_{i=1}^{N} x_i |i\rangle$$

Easy-to-prepare — Solution

The main ideas

$$Ax = b$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_N \end{pmatrix}$$

$$\sum_{i=1}^{N} b_i |i\rangle \xrightarrow{x = A^{-1}b} \sum_{i=1}^{N} x_i |i\rangle$$

How do we apply A^{-1}?

Eigenvectors

- $|\Psi\rangle$ - eigenvector if $A|\Psi\rangle = \lambda|\Psi\rangle$.
- λ - eigenvalue.
- Assume: A – Hermitian (A=A*).

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_N \end{pmatrix} \Rightarrow x = \sum_i c_i v_i$$

v_i – eigenvector of A

The main ideas

$$Ax = b$$

$$x = \sum_i c_i v_i \qquad Av_i = \lambda_i v_i$$

$$Ax = \sum_i c_i \lambda_i v_i$$

The main ideas

$$x = \sum_i c_i v_i \longrightarrow b = \sum_i c_i \lambda_i v_i$$

$$b = \sum_i a_i v_i \longrightarrow x = \sum_i a_i \lambda_i^{-1} v_i$$

Implement a quantum transformation

$$|v_i\rangle \rightarrow \lambda_i^{-1} |v_i\rangle$$

$$|b\rangle \rightarrow |x\rangle$$

Eigenvalue estimation

- Subroutine in Shor's quantum algorithm for factoring.
- Explicitly defined in Kitaev, 1995.
- Input: A and $|v_i\rangle$: $A|v_i\rangle = \lambda_i|v_i\rangle$.
- Output: $|v_i\rangle\,|\lambda'_i\rangle$, $\lambda'_i \approx \lambda_i$.

$$|v_i\rangle \xrightarrow{\text{EE}} |v_i\rangle|\lambda'_i\rangle \rightarrow \frac{1}{\lambda'_i}|v_i\rangle|\lambda'_i\rangle \xrightarrow{\text{EE}^{-1}} \frac{1}{\lambda'_i}|v_i\rangle$$

Caveat

$$|v_i\rangle|\lambda'_i\rangle \rightarrow \frac{1}{\lambda_i^{'}}|v_i\rangle|\lambda'_i\rangle$$

is not unitary!

Solution: perform

$$|v_i\rangle|\lambda'_i\rangle \rightarrow |v_i\rangle|\lambda'_i\rangle\left(\frac{C}{\lambda_i^{'}}|succ\rangle + \sqrt{1-\left(\frac{C}{\lambda_i^{'}}\right)^2}\,|fail\rangle\right)$$

Running time

1. Size of system N $\rightarrow$ O($\log^c$ N).
2. Time to implement A – O(1) for sparse matrices A, O(N) generally.
3. Condition number of A.

$$k = \frac{\mu_{\max}}{\mu_{\min}}$$

μ_{max} and μ_{min} – biggest and smallest eigenvalues of A

$$Time - O\left(\kappa^2 \log^c N\right)$$

Dependence on condition number

- Classical algorithms for sparse A: $O(N\sqrt{k})$.
- [Harrow, Hassidim, Llyod, 2008]: $O(k^2 \log^c N)$.
- [A, 2010]: $O(k^{1+o(1)} \log^c N)$, via improved version of eigenvalue estimation.
- [HHL, 2008]: $\Omega(k^{1-o(1)})$, unless BQP=PSPACE.

Open problem

- What problems can we reduce to systems of linear equations (with $\sum_i x_i|i\rangle$ as the answer)?

- Examples:
  - Search;
  - Perfect matchings in a graph;
  - Graph bipartiteness.

Biggest issue: condition number.