Span-program-based quantum query algorithms

Mg.sc.comp. Agnis Āriņš
agnis.arins@lu.lv

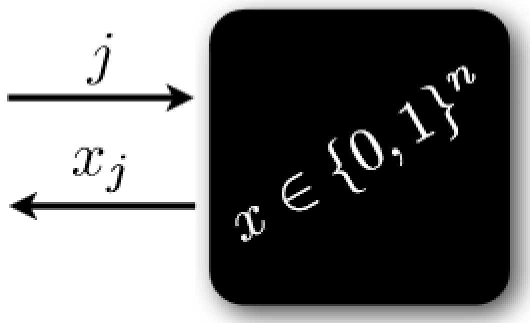
Ratnieki, 2014. gada 3. oktobris

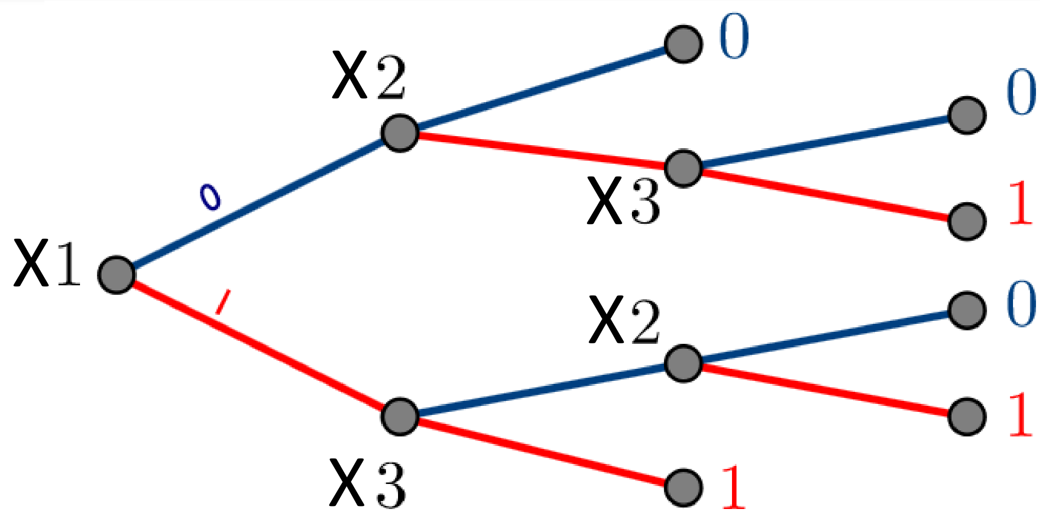
LATVIJAS UNIVERSITĀTE

History of span programs

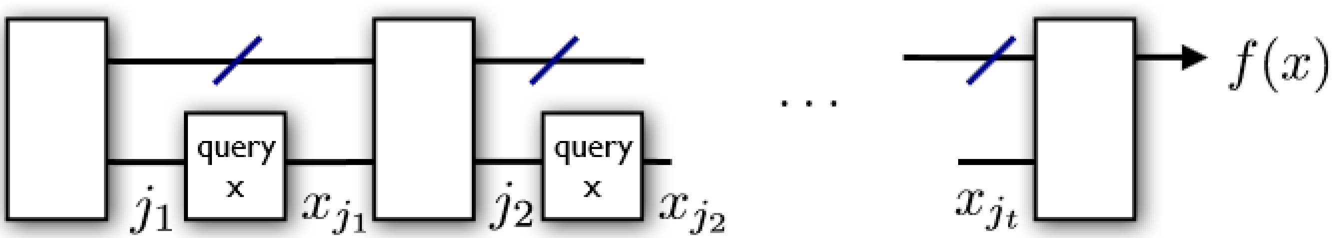
- First defined in 1993 by Karchmer, Wigderson
- Rediscovered in 2007, used for evaluating Boolean formulas by Reichardt, Špalek
- In 2010 Reichardt proved that span programs are equivalent to quantum query algorithms
- Used for s-t connectivity, triangle finding, k-distinctness, graph collision, ...
- Basis for Belovs's *Learning Graph* approach

Decision tree model

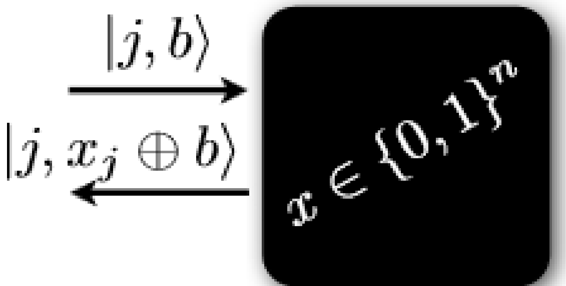




$\Rightarrow \quad \mathrm{MAJ}(x_1, x_2, x_3)$



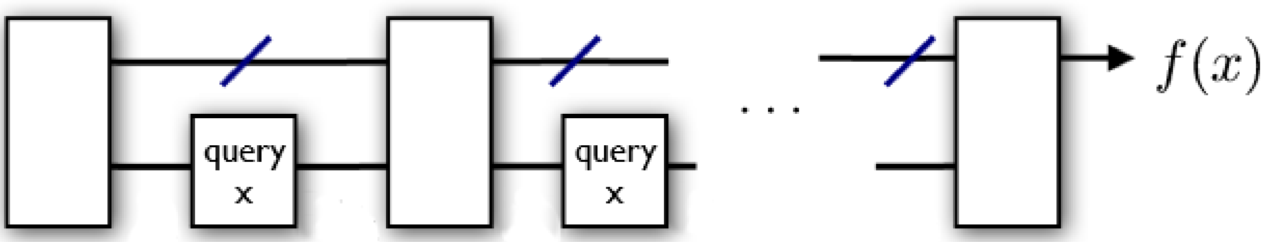
Quantum query model



Quantum queries

Unitary transformations

$$|x\rangle_I |\psi_x^t\rangle_A = U_t\, O\, U_{t-1} \ldots U_1 O\, U_0\, |x\rangle_I\, |1,0\rangle_Q\, |0\rangle_W$$



Notation

- $Span(S) = \left\{\sum_{i=1}^{k} \lambda_i v_i \mid k \in N,\ v_i \in S,\ \lambda_i \in \mathbb{F}\right\}$
- Bra: $\langle u| = (c_1, c_2, \dots, c_n)$ Ket: $|v\rangle = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix}$
- Inner product:

$$\langle u|v\rangle = a_1 c_1 + a_2 c_2 + \cdots + a_n c_n$$

- Norm: $\|v\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$

Span program P on n bits

Vector space: V

Target vector: $|t\rangle$

Sets of vectors: $V_{1,0}, V_{1,1}, \ldots, V_{n,0}, V_{n,1}, V_{free}$

If $x_i = b$ then vectors $V_{i,b}$ are available

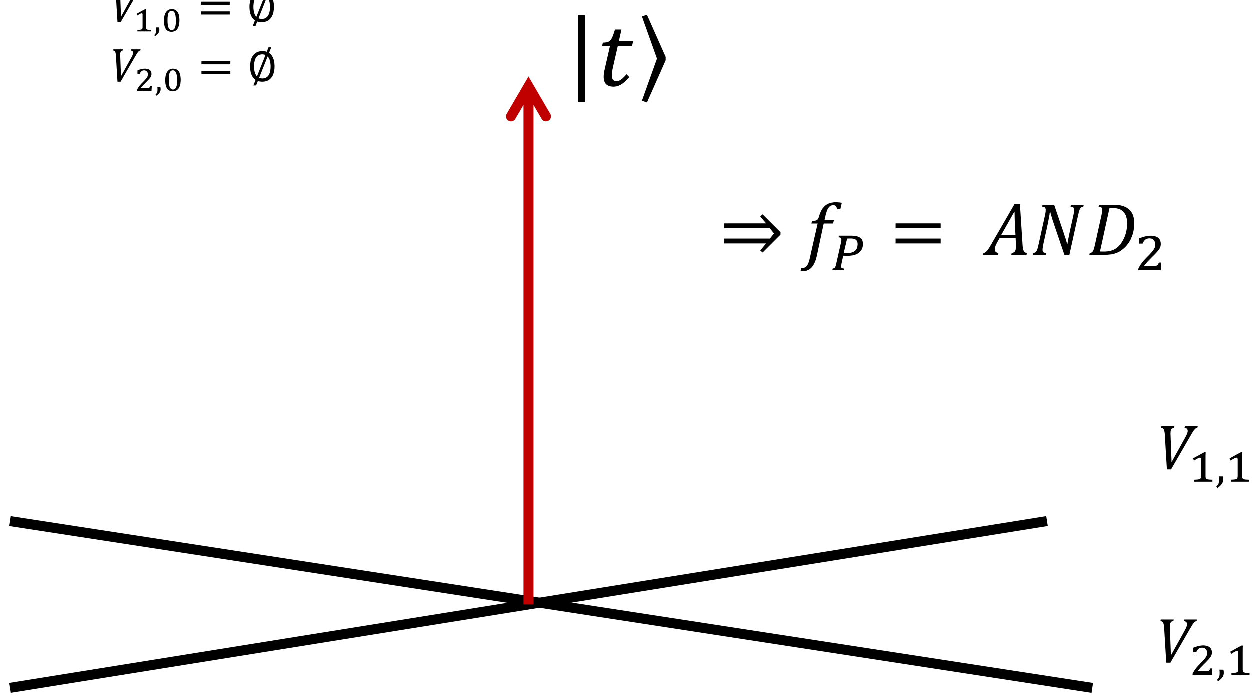
P «computes» $f_P : \{0,1\}^n \rightarrow \{0,1\}$

$$f_P(x) = 1 \iff |t\rangle \in Span\left(V_{free} \cup \bigcup_j V_{j,x_j}\right)$$

Span program for «AND» function

Geometric example: $\boldsymbol{V} = \mathbb{R}^2$

$V_{1,0} = \emptyset$
$V_{2,0} = \emptyset$



$\Rightarrow f_P = \ AND_2$

LATVIJAS UNIVERSITATE

Span program for «XOR» function

- $f(x_1, x_2) = x_1 \oplus x_2$
- Pick the target vector: $|t\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
  - if $x_1 = 1$ then make available the vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$
  - if $x_1 = 0$ then make available the vector $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$
  - if $x_2 = 1$ then make available the vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$
  - if $x_2 = 0$ then make available the vector $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Span program for «XOR» function

- $f(x_1, x_2) = x_1 \oplus x_2$
- Vector space: $V = \mathbb{R}^2$
- Basis vectors: $\{|0\rangle, |1\rangle\}$
- Pick the target vector: $|t\rangle = |0\rangle + |1\rangle$
  - $x_1 = 1 : \{ |0\rangle \}$
  - $x_1 = 0 : \{ |1\rangle \}$
  - $x_2 = 1 : \{ |0\rangle \}$
  - $x_2 = 0 : \{ |1\rangle \}$

Span program for «OR_n» function

- $f(x_1, x_2, \dots, x_n) = x_1 \vee x_2 \vee \cdots \vee x_n$
- Vector space: $V = \mathbb{R}$
- Basis vectors: $\{|0\rangle\}$
- Pick the target vector: $|t\rangle = |0\rangle$
- For every $i \in [1..n]$:
  - $x_i = 1 : \ \{\ |0\rangle\ \}$
  - $x_i = 0 : \ \{\}$

Case $f_P(x) = 1$: positive witness

- A – column matrix of the available vectors
- $|w\rangle$ - the positive witness
- $wsize_1(P, x) = \min_{|w\rangle:\, A|w\rangle = |t\rangle} \| |w\rangle \|^2$
- $wsize_1(P, x)$ – essentially a sum of squared coefficients for those vectors that are used to express the target vector
- $wsize_1(P) = \max_{x \in \{0,1\}^n} wsize_1(P, x)$

«OR_n»: positive witness

- Target vector: $|t\rangle = |0\rangle$
- For every $i \in [1..n]$:
  - $x_i = 1 : \{ |0\rangle \}$
  - $x_i = 0 : \{\}$
- If for some $x_i = 1$ then the target vector can be expressed: $|t\rangle = 1 * |0\rangle$
- $wsize_1(P) \leq 1^2$

Case $f_P(x) = 0$: negative witness

- $V(x) = V_{free} \cup \bigcup_{j \in [1..n]} V_{j, x_j}$
- $V(\neg x) = V_{free} \cup \bigcup_{j \in [1..n]} V_{j, (1-x_j)}$
- $|w'\rangle$ - the negative witness
- Need to find such $|w'\rangle$ that $\langle w'|t\rangle = 1$ and $\forall\ |y\rangle \in V(x)\ \ (|w'\rangle \perp |y\rangle)$
- $\mathrm{wsize}_0(\mathrm{P}, \mathrm{x}) = \min_{|w'\rangle} \sum_{|y\rangle \in V(\neg x)} \langle w'|y\rangle^2$
- $wsize_0(P) = \max_{x \in \{0,1\}^n} wsize_0(P, x)$

LATVIJAS UNIVERSITATE

«OR_n»: negative witness

- Target vector: $|t\rangle = |0\rangle$
- For every $i \in [1..n]$:
  - $x_i = 1 :\ \{\, |0\rangle \,\}$
  - $x_i = 0 :\ \{\}$
- If all x_i are equal to 0 then can take negative witness: $|w'\rangle = |0\rangle$
- $wsize_0(P) \leq 1^2 + 1^2 + \cdots + 1^2 = n$

Span program complexity

- $wsize(P) = \sqrt{wsize_1(P) * wsize_0(P)}$
- $Q(f)$ - quantum query complexity
- $Q(f) = O\left(wsize\left(P_f\right)\right)$
- For «OR_n»:

$$wsize(P) = \sqrt{1 * n} = \sqrt{n}$$

Span program for «OR_n» function

- If $\mathrm{m} = \#i: (x_i = 1)$ then the target vector can be expressed: $|t\rangle = \frac{1}{m}|0\rangle + \frac{1}{m}|0\rangle + \cdots + \frac{1}{m}|0\rangle$
- $wsize_1(P) \leq m\left(\frac{1}{m}\right)^2 = \frac{1}{m}$
- $wsize_0(P) \leq 1^2 + 1^2 + \cdots + 1^2 = n$
- $wsize(P) = \sqrt{wsize_1(P) * wsize_0(P)} = \sqrt{\frac{n}{m}}$

Graph bipartiteness

- n – number of vertices in the given graph
- We will construct an $\boldsymbol{\theta}\left(\boldsymbol{n}\sqrt{\boldsymbol{n}}\right)$ algorithm for testing if the given graph is bipartite
- It's asymptotically optimal
- Any classical algorithm would need to consider all edges – in worst case $\Omega(n^2)$ queries
- Main idea: look for an odd cycle
- An undirected graph is bipartite iff it has no odd cycles

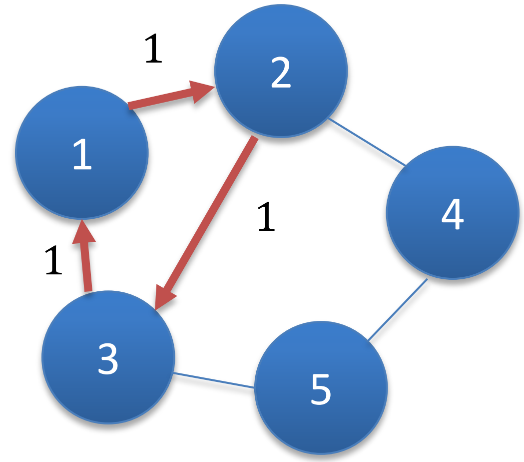
Graph bipartiteness

- Vector space: $V = \mathbb{R}^{2n^2+1}$
- Basis vectors: $\{|0\rangle\} \cup \left\{\left|v_{k,b}\right\rangle | v, k \in [1..n], b \in \{0,1\}\right\}$
- Target vector: $|t\rangle = |0\rangle$
- For every $k \in [1..n]$: can use the free vector $|0\rangle + \left|k_{k,0}\right\rangle + \left|k_{k,1}\right\rangle$
- For every $k \in [1..n]$:
  - for every edge $u - v$ (where input $x_{u,v} = 1$) make available the vectors: $\left|u_{k,0}\right\rangle + \left|v_{k,1}\right\rangle$ and $\left|u_{k,1}\right\rangle + \left|v_{k,0}\right\rangle$

Graph bipartiteness

$$|t\rangle = \left(|0\rangle + \left|1_{1,0}\right\rangle + \left|1_{1,1}\right\rangle\right) - \left(\left|1_{1,0}\right\rangle + \left|2_{1,1}\right\rangle\right) + \left(\left|2_{1,1}\right\rangle + \left|3_{1,0}\right\rangle\right) - \left(\left|3_{1,0}\right\rangle + \left|1_{1,1}\right\rangle\right)$$

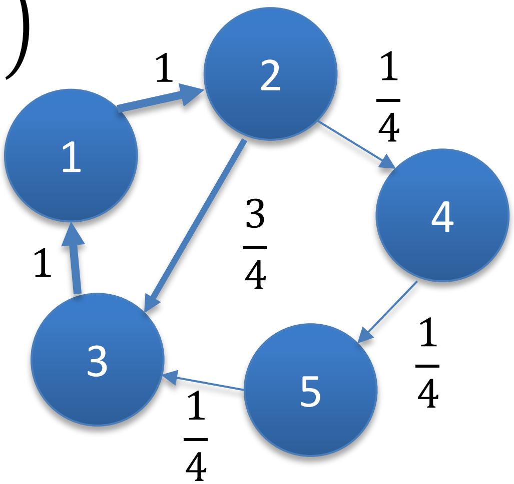
$$wsize_1(P,x) \leq 1^2 + (-1)^2 + 1^2 + (-1)^2 = 4$$



LATVIJAS UNIVERSITATE

Graph bipartiteness

$$
\begin{aligned}
|t\rangle \; = & \left(|0\rangle + \left|1_{1,0}\right\rangle + \left|1_{1,1}\right\rangle\right) - \left(\left|1_{1,0}\right\rangle + \left|2_{1,1}\right\rangle\right) \\
& + \frac{3}{4}\left(\left|2_{1,1}\right\rangle + \left|3_{1,0}\right\rangle\right) \\
& + \frac{1}{4}\Big(\left(\left|2_{1,1}\right\rangle + \left|4_{1,0}\right\rangle\right) - \left(\left|4_{1,0}\right\rangle + \left|5_{1,1}\right\rangle\right) \\
& + \left(\left|5_{1,1}\right\rangle + \left|3_{1,0}\right\rangle\right)\Big) - \left(\left|3_{1,0}\right\rangle + \left|1_{1,1}\right\rangle\right)
\end{aligned}
$$



$$
\begin{aligned}
& wsize_1(P, x) \\
& \quad \leq 1^2 + (-1)^2 + \left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(-\frac{1}{4}\right)^2 \\
& \quad + \frac{1^2}{4} + (-1)^2 = 3 + \frac{12}{16}
\end{aligned}
$$

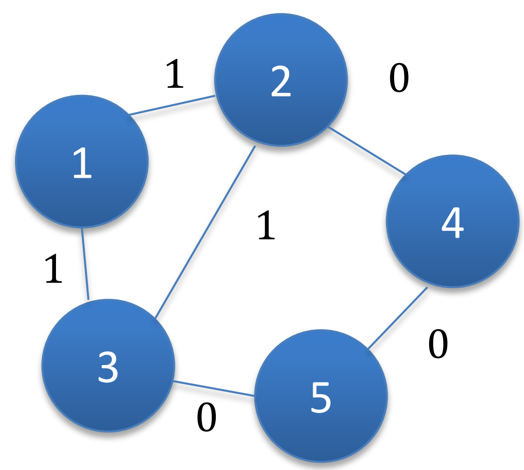
LATVIJAS UNIVERSITATE

Graph bipartiteness

$$|t\rangle =$$
$$\frac{1}{3}\left(\left(|0\rangle + \left|1_{1,0}\right\rangle + \left|1_{1,1}\right\rangle\right) - \left(\left|1_{1,0}\right\rangle + \left|2_{1,1}\right\rangle\right) + \left(\left|2_{1,1}\right\rangle + \left|3_{1,0}\right\rangle\right) - \left(\left|3_{1,0}\right\rangle + \left|1_{1,1}\right\rangle\right)\right)$$
$$+\frac{1}{3}\left(\left(|0\rangle + \left|2_{2,0}\right\rangle + \left|2_{2,1}\right\rangle\right) - \left(\left|2_{2,0}\right\rangle + \left|3_{2,1}\right\rangle\right) + \left(\left|3_{2,1}\right\rangle + \left|1_{2,0}\right\rangle\right) - \left(\left|1_{2,0}\right\rangle + \left|2_{2,1}\right\rangle\right)\right)$$
$$+\frac{1}{3}\left(\left(|0\rangle + \left|3_{3,0}\right\rangle + \left|3_{3,1}\right\rangle\right) - \left(\left|3_{3,0}\right\rangle + \left|1_{3,1}\right\rangle\right) + \left(\left|1_{3,1}\right\rangle + \left|2_{3,0}\right\rangle\right) - \left(\left|2_{3,0}\right\rangle + \left|3_{3,1}\right\rangle\right)\right)$$

$$wsize_1(P,x) \leq 3 * \left(\frac{1}{3}\right)^2 * (1^2 + (-1)^2 + 1^2 + (-1)^2)$$
$$= \frac{4}{3} < 3 + \frac{12}{16}$$

$$\boldsymbol{wsize_1(P) = O(1)}$$



LATVIJAS UNIVERSITĀTE

Graph bipartiteness: complexity

- Need to find such $|w'\rangle$ that $\langle w'|0\rangle = 1$ and $\forall\ |y\rangle \in V(x)\ \ (|w'\rangle \perp |y\rangle)$
- Must have $\langle w'|\big(|0\rangle + \left|k_{k,0}\right\rangle + \left|k_{k,1}\right\rangle\big)\rangle = 0$
- For every k set $\langle w'|k_{k,0}\rangle = 0$ and $\langle w'|k_{k,1}\rangle = -1$
  - For every u $- v$ set $\langle w'|u_{k,b}\rangle = -\langle w'|v_{k,1-b}\rangle$
- For any given vector v value $\langle w'|v\rangle^2 \leq 1$
- The total number of vectors does not exceed $n + n^3$
- $wsize_0(P) = O(n^3)$
- $wsize(P) = \sqrt{wsize_1(P) * wsize_0(P)} = \boldsymbol{\theta}\big(\boldsymbol{n}\sqrt{\boldsymbol{n}}\big)$

LATVIJAS UNIVERSITATE

Graph connectivity

- Main idea: is every vertex reachable from vertex 1?
- Target vector: $|t\rangle = |0_2\rangle + |0_3\rangle + \cdots + |0_N\rangle$
- For every $k \in [2..n]$: can use the free vector

$$|0_k\rangle + |1_k\rangle - |k_k\rangle$$

- For every $k \in [2..n]$:
  - for every edge $u - v$ (where input $x_{u,v} = 1$) make available the vector:

$$|u_k\rangle - |v_k\rangle$$

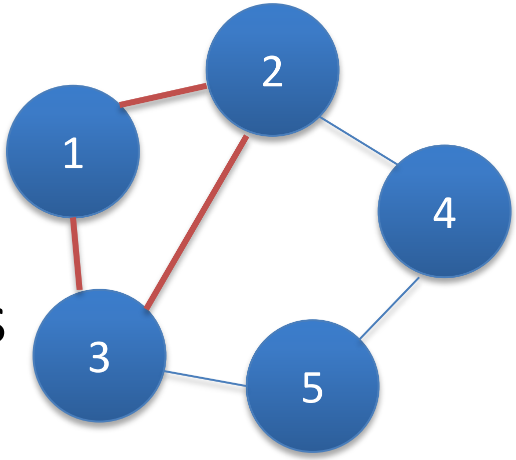
- $\boldsymbol{wsize(P) = \theta(n\sqrt{n})}$

Eulerian cycle

- Main idea: check for vertex with an odd degree
- Target vector: $|t\rangle = \ |0\rangle$
- For every $k \in [1..n]$ can use vectors:
  - free vetor $|0\rangle + |k_{0,0}\rangle - |k_{n,1}\rangle$
  - For every $i \in [1..n]$:
    - If $(x_{k,i} = 1)$ then can use vectors: $-|k_{i-1,0}\rangle + |k_{i,1}\rangle$ and $-|k_{i-1,1}\rangle + |k_{i,0}\rangle$
    - if $(x_{k,i} = 0)$ then can use vectors: $-|k_{i-1,0}\rangle + |k_{i,0}\rangle$ and $-|k_{i-1,1}\rangle + |k_{i,1}\rangle$

Open problems

- Graph collision: a graph is known beforehand but the vertex marking is not, check if two marked vertices are adjacent. Best lower bound is $\Omega(\sqrt{n})$.
- Triangle finding: check if in a given graph there are three vertices which are pairwise adjacent. Best lower bound is $\Omega(n)$.
- Perfect matching: check if can take exactly one entry with value «1» in each column and row from the graph's adjacency matrix



LATVIJAS UNIVERSITATE

THANKS FOR LISTENING. QUESTIONS?

LATVIJAS UNIVERSITĀTE

END.

LATVIJAS UNIVERSITATE