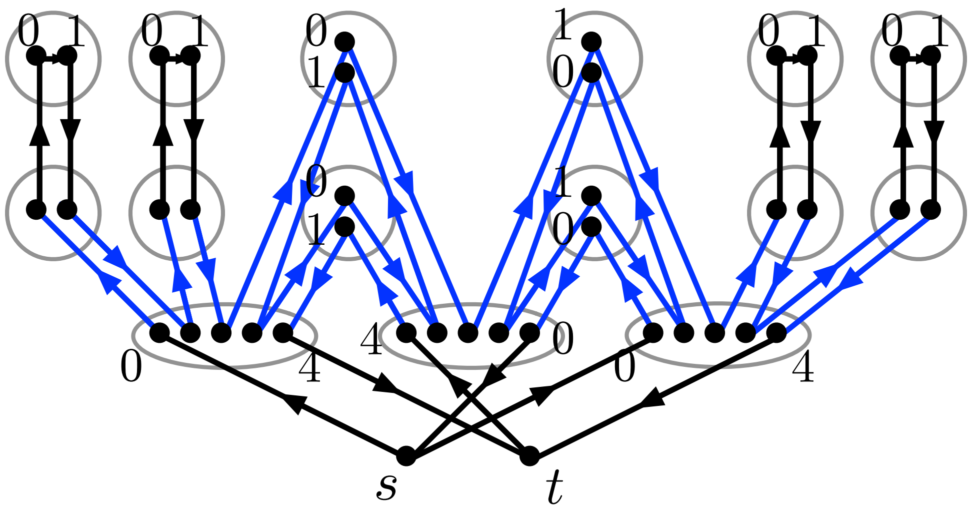
Quantum algorithm for deciding *st*-connectivity



Ben Reichardt
USC

Alexandrs Belovs
University of Latvia

arXiv:1203.2603 [quant-ph]

Workshop on Recent Progress in Quantum Algorithms, April 11, 2012

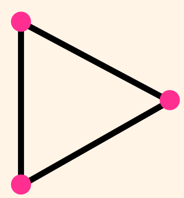
1 *st*-connectivity

Is there a path from s to t?

2 Path-detection

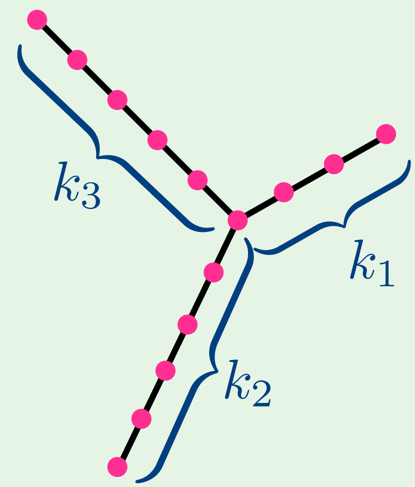
Is there *any* path of length k?

3 Triangle subgraph/not-a-minor promise problem



Does G contain a triangle, or is it acyclic?

4 Claw-detection



Detect a $\{k_1, k_2, k_3\}$ subdivided claw

Input: G's adjacency matrix, an $\binom{n}{2}$-bit string

Query complexity

Goal: Evaluate function f on input *x*

Resource:

$$\sum_j \alpha_j |j\rangle \longrightarrow \boxed{x \in \{0,1\}^n} \longrightarrow \sum_j \alpha_j (-1)^{x_j} |j\rangle$$

Algorithm:

st-connectivity (USTCON)

Classical algorithms

- Graph traversal (e.g., DFS), in $\Theta(n)$ space
- Randomized log-space [AKLLR'79]:
  Hitting time $H_{st} + H_{ts} = 2\, m\, R_{st} \leq n^3$

  \# edges effective resistance
- Derandomized by [Reingold '08]

Quantum query algorithm

[Dürr, Heiligman, Høyer, Mhalla '04]

1. List = {1}, {2}, …, {n}
2. Repeat:
   - Grover search for edge between two components
   - Merge those components
3. Output List

- Outputs all connected components
- O(n) bits of quantum RAM

$$\sqrt{\frac{n^2 \text{ possible edges}}{n \text{ edges}}} + \sqrt{\frac{n^2}{n-1}} + \cdots + \sqrt{\frac{n^2}{1}}$$

= $O(n^{3/2})$ *queries* to n×n adjacency matrix — optimal

__New *st*-connectivity quantum query algorithm__

$O(\sqrt{(\underbrace{M}_{\#\ \textit{possible}\text{ edges}} R_{st})}) = O(n\sqrt{d})$ queries,

if s and t are promised either to be within distance d, or be disconnected

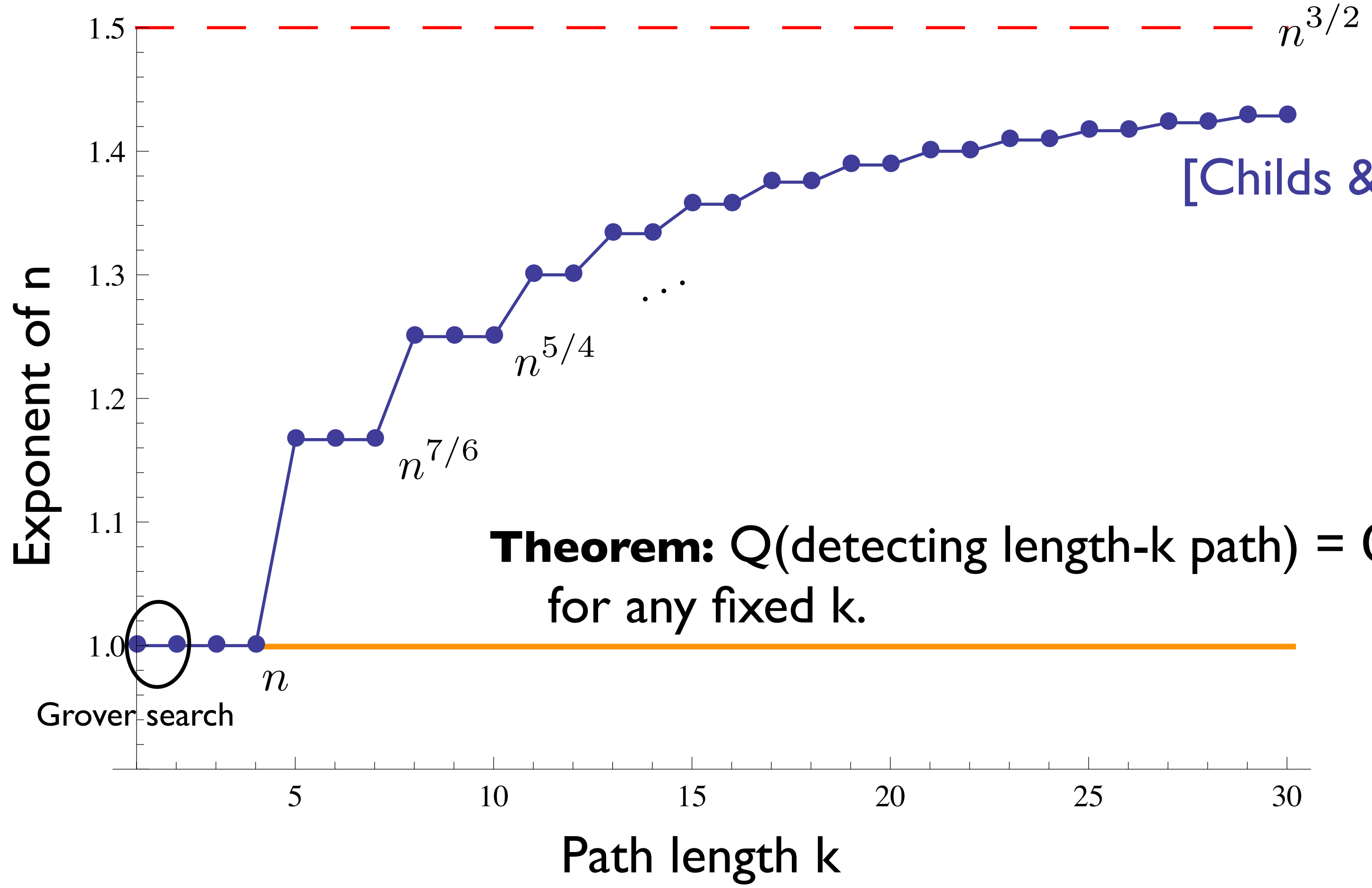
Applications
incl. learning graphs

Log space

Efficiently implementable

Connections to random & quantum walks

Application: Path detection
Does G contain a path of length k?



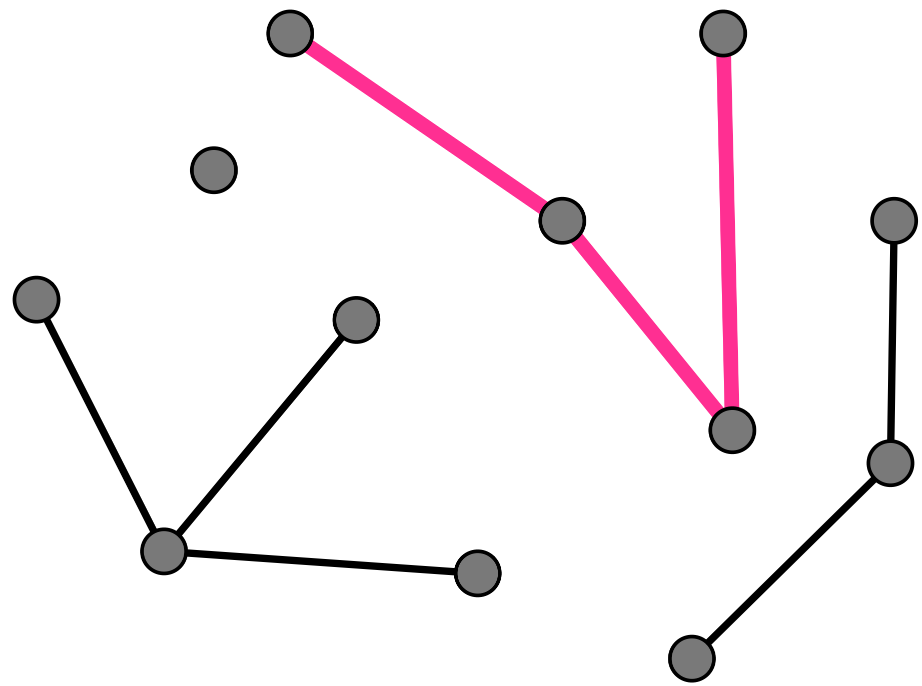
Theorem: Q(detecting length-k path) = Θ(n)
for any fixed k.

k^k

Application: Path detection
Does G contain  ?

<u>Algorithm</u>

- Randomly color vertices by {0,…,k}.
  Discard badly colored edges.
  (Hopefully a path is colored correctly.)

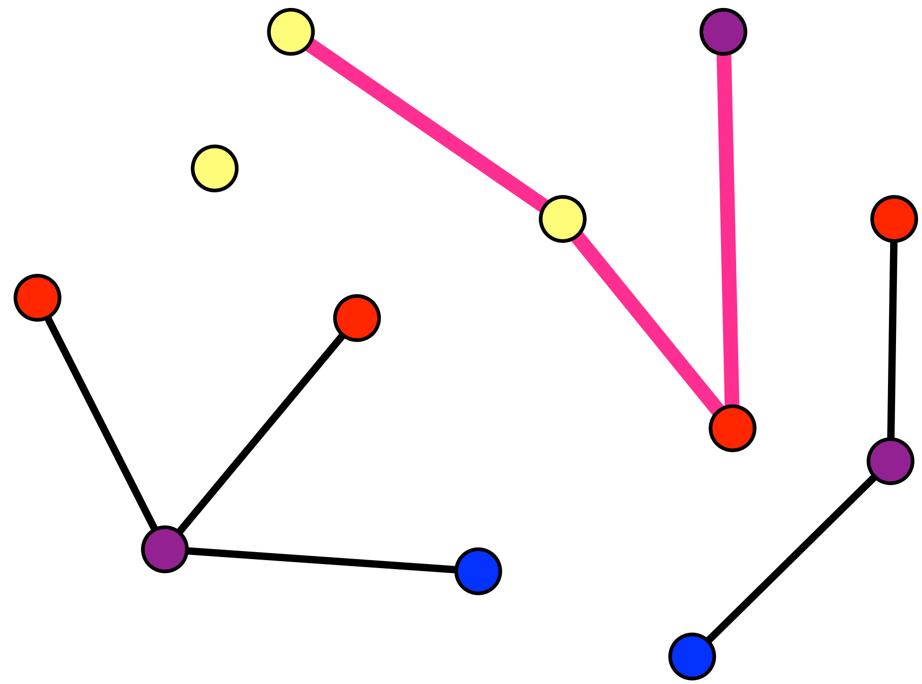


Application: Path detection

Does G contain  ?

<u>Algorithm</u>

- Randomly color vertices by {0,…,k}.
  Discard badly colored edges.
  (Hopefully a path is colored correctly.)

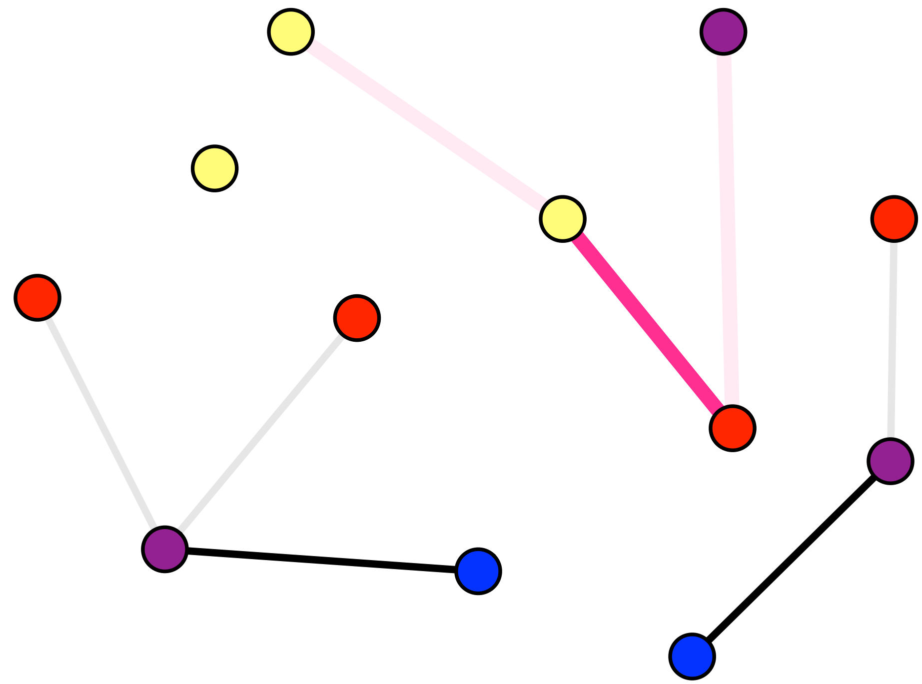


Application: Path detection

Does G contain  ?

<u>Algorithm</u>

- Randomly color vertices by {0,…,k}.
  Discard badly colored edges.
  (Hopefully a path is colored correctly.)

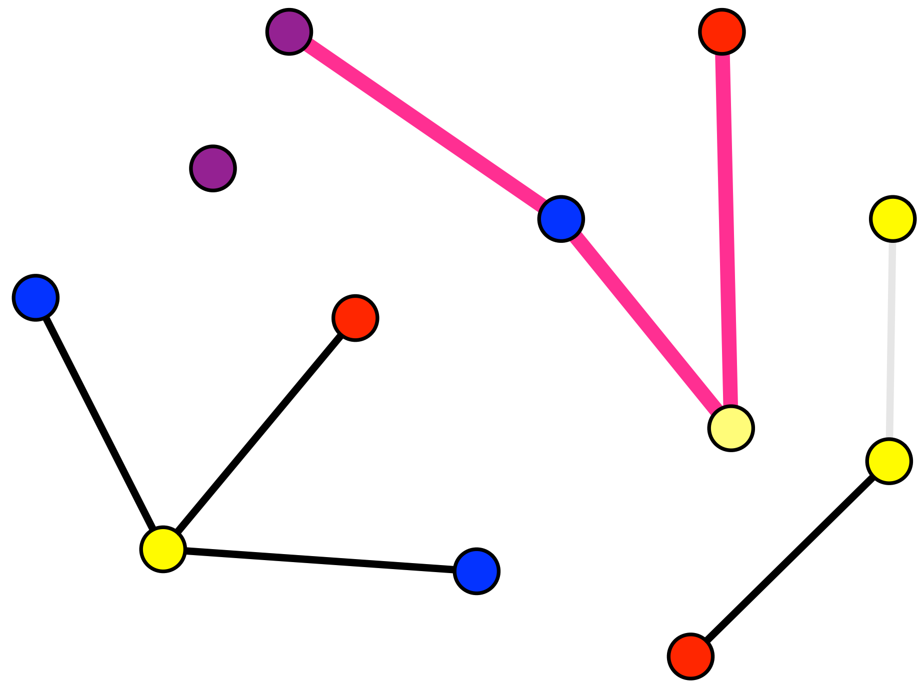


Application: Path detection

Does G contain  ?

<u>Algorithm</u>

- Randomly color vertices by {0,…,k}.
  Discard badly colored edges.
  (Hopefully a path is colored correctly.)



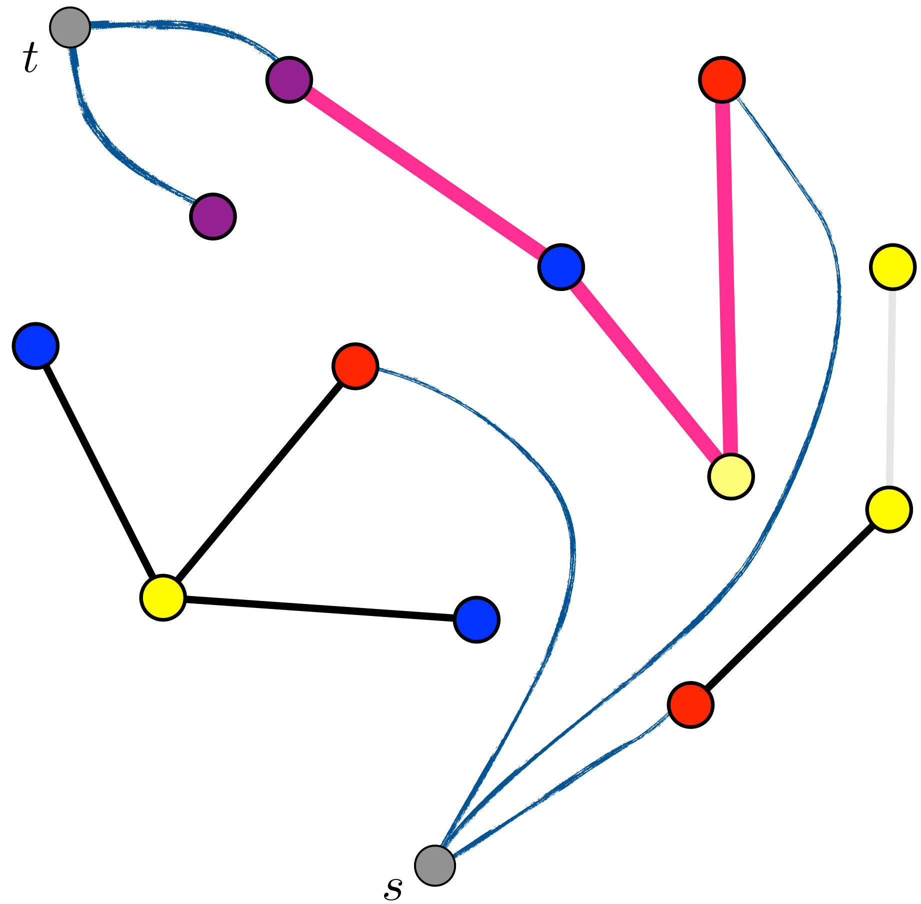
Application: Path detection

Does G contain  ?

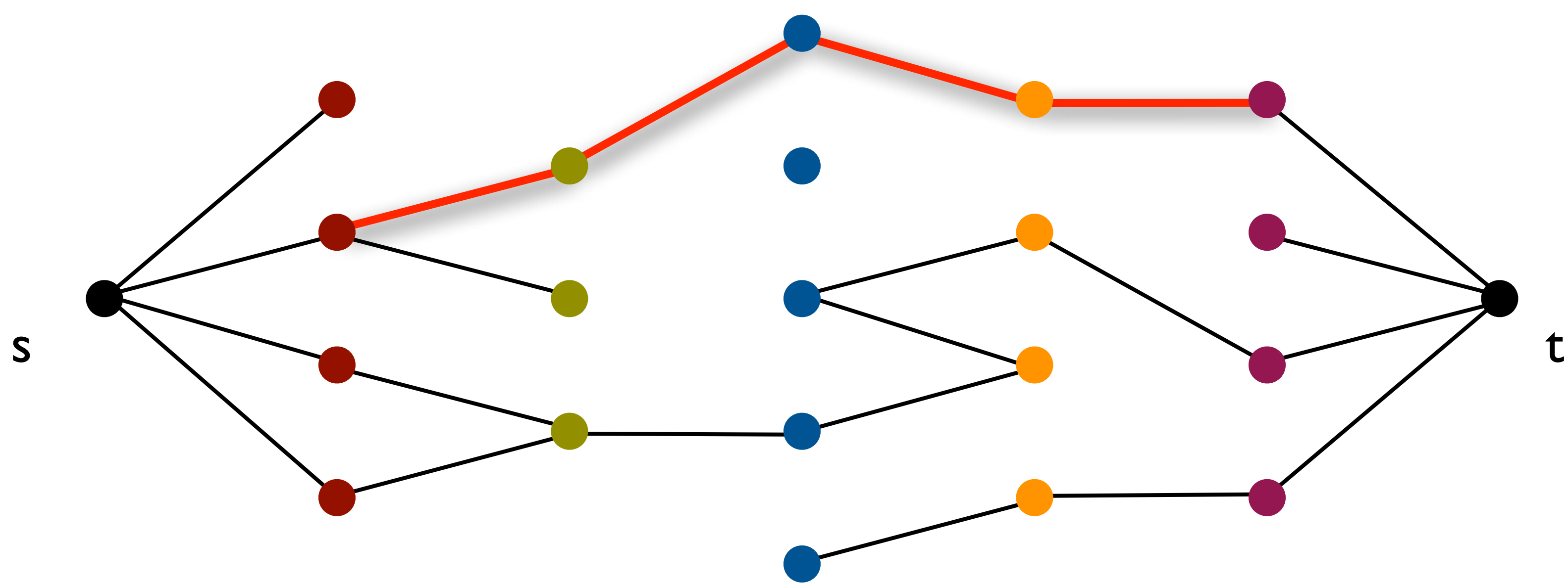
<u>Algorithm</u>

- Randomly color vertices by {0,…,k}. Discard badly colored edges. (Hopefully a path is colored correctly.)
- Attach s to color-0 vertices, and t to color-k vertices.
- Run s-t connectivity.

$$O(n\sqrt{k+3}) = O(n)$$

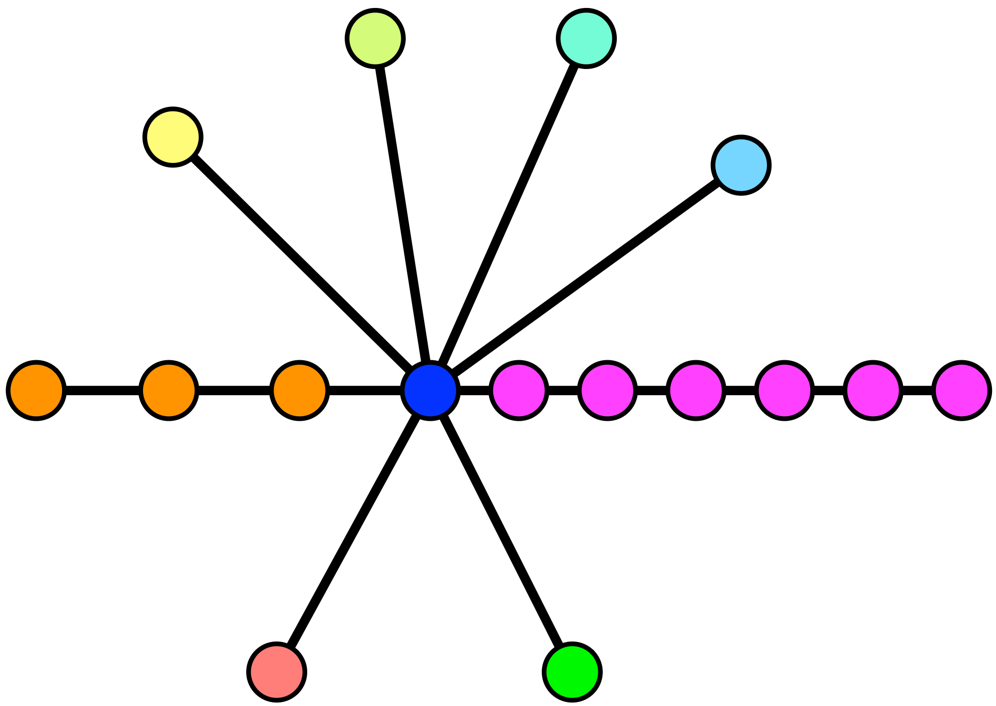


Application: Path detection
Does G contain a path of length k?



Application: Subgraph detection

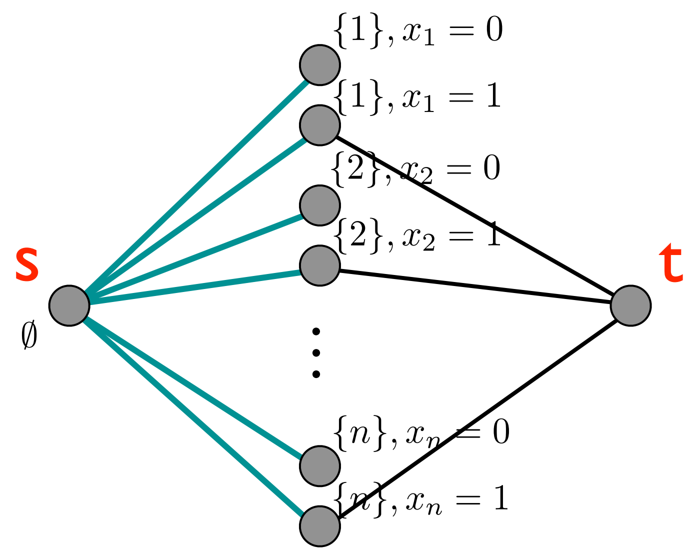
Star with two subdivided legs



Example application: Learning graphs

- Learning graphs (with input-independent weights) are a reduction to *st*-connectivity on graphs of a restricted form (not complete)

- Example: Grover search



- Complexity = $\sqrt{}$(max cut size) $\sqrt{}$($\max_x R_{st}(x)$)

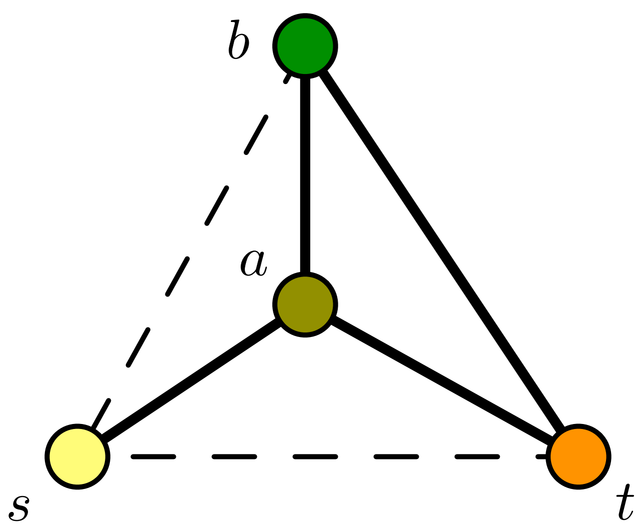
Span programs

- Span program P =
  - Target vector $|\tau\rangle$
  - Input vectors, each labeled by
    (index j, variable value b)
- P(x) := 1 iff $|\tau\rangle$ can be reached using input vectors labeled by
  (1, x_1), …, (n, x_n)

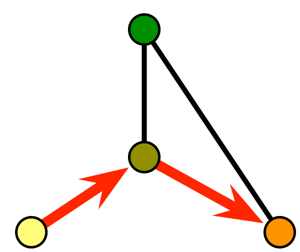
st-connectivity

- $|\tau\rangle = |t\rangle - |s\rangle \in \mathbf{R}^V$
- input vector $|u\rangle - |v\rangle$
  labeled by $(A_G)[u,v] = 1$, i.e., the input vector can be used if the edge is present

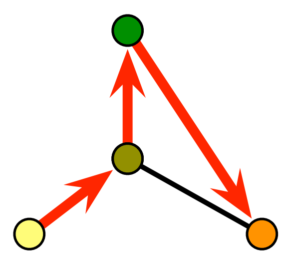
“Witnesses” to P(x)=1 ⇔ balanced s-t flows



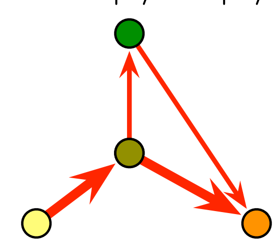
$$\begin{array}{rl} & |a\rangle - |s\rangle \\ + & |t\rangle - |a\rangle \\ \hline & |t\rangle - |s\rangle \end{array} \quad ✓$$



$$\begin{array}{rl} & |a\rangle - |s\rangle \\ & |b\rangle - |a\rangle \\ + & |t\rangle - |b\rangle \\ \hline & |t\rangle - |s\rangle \end{array} \quad ✓$$



$$\begin{array}{rl} & |a\rangle - |s\rangle \\ & \frac{1}{3}(|b\rangle - |a\rangle) \\ & \frac{1}{3}(|t\rangle - |b\rangle) \\ + & \frac{2}{3}(|t\rangle - |a\rangle) \\ \hline & |t\rangle - |s\rangle \end{array} \quad ✓$$



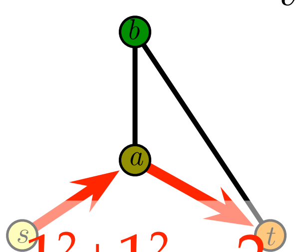
Witness size(P) = max$_x$ wsize(P,x)

Case P(x)=1

$$|\tau\rangle = \sum_{\substack{\text{available} \\ \text{input vectors } |v\rangle}} w_v |v\rangle$$

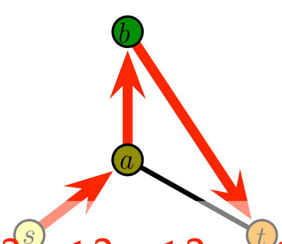
$$\mathrm{wsize}(P,x) = \min \sum_v |w_v|^2$$

$$\begin{array}{rl} & |a\rangle - |s\rangle \\ + & |t\rangle - |a\rangle \\ \hline & |t\rangle - |s\rangle \end{array} \quad \checkmark$$



$1^2+1^2 = 2$

$$\begin{array}{rl} & |a\rangle - |s\rangle \\ & |b\rangle - |a\rangle \\ + & |t\rangle - |b\rangle \\ \hline & |t\rangle - |s\rangle \end{array} \quad \checkmark$$



$1^2+1^2+1^2 = 3$

$$\begin{array}{rl} & |a\rangle - |s\rangle \\ & \frac{1}{3}(|b\rangle - |a\rangle) \\ & \frac{1}{3}(|t\rangle - |b\rangle) \\ + & \frac{2}{3}(|t\rangle - |a\rangle) \\ \hline & |t\rangle - |s\rangle \end{array} \quad \checkmark$$

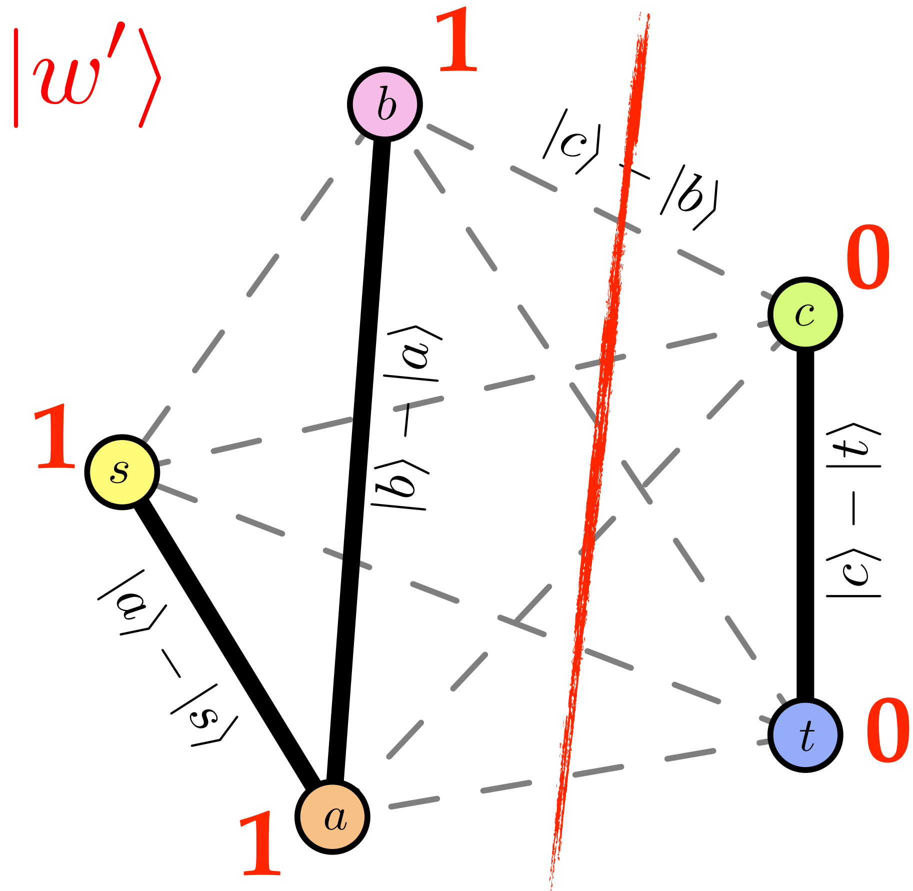
$1^2+(1/3)^2+(1/3)^2+(2/3)^2$

$= 5/3$

wsize(P_{STCONN}, G) = R_{st}(G)
≤ d(s,t) ≤ n

Witness size(P) = $\max_x$ wsize(P,x)

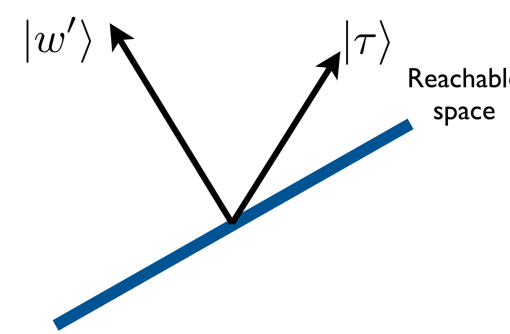
$$|\tau\rangle = |s\rangle - |t\rangle$$



Case P(x)=0

$$|\tau\rangle \notin \mathrm{Span}\big(\text{available input vectors } |v\rangle\big)$$

$$\Rightarrow \exists\ |w'\rangle \perp \text{available vectors}$$
$$\langle w'|\tau\rangle = 1$$



$$\mathrm{wsize}(P,x) = \min \sum_v |\langle w'|v\rangle|^2$$

$$\boxed{\text{wsize}(P_{\text{STCONN}}, G) = \text{cut size}(G) \leq n^2/4}$$

Witness size(P) = $\max_x$ wsize(P,x)

Case P(x)=1

$$|\tau\rangle = \sum_{\substack{\text{available} \\ \text{input vectors } |v\rangle}} w_v |v\rangle$$

$$\mathrm{wsize}(P, x) = \min \sum_v |w_v|^2$$

Case P(x)=0

$$\Rightarrow \exists\ |w'\rangle \perp \text{available vectors}$$
$$\langle w'|\tau\rangle = 1$$

$$\mathrm{wsize}(P, x) = \min \sum_v |\langle w'|v\rangle|^2$$

st-connectivity

s connected to *t*:

wsize(P_{STCONN}, G) = R_{st}(G) ≤ d(s,t) ≤ n

s not connected to *t*:

wsize(P_{STCONN}, G) ≤ n^2

$$\textbf{Witness size(P)} = \sqrt{\max_{x:\,P(x)=1} \text{wsize}(P,x) \max_{x:\,P(x)=0} \text{wsize}(P,x)}$$

Case P(x)=1

$$|\tau\rangle = \sum_{\substack{\text{available} \\ \text{input vectors } |v\rangle}} w_v |v\rangle$$

$$\text{wsize}(P,x) = \min \sum_v |w_v|^2$$

Case P(x)=0

$$\Rightarrow \exists\ |w'\rangle \perp \text{available vectors}$$
$$\langle w'|\tau\rangle = 1$$

$$\text{wsize}(P,x) = \min \sum_v |\langle w'|v\rangle|^2$$

st-connectivity

s connected to *t*:

wsize(P_{STCONN}, G) = R_{st}(G) ≤ d(s,t) ≤ n

s not connected to *t*:

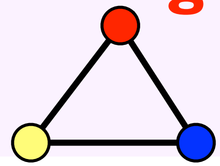
wsize(P_{STCONN}, G) ≤ n^2

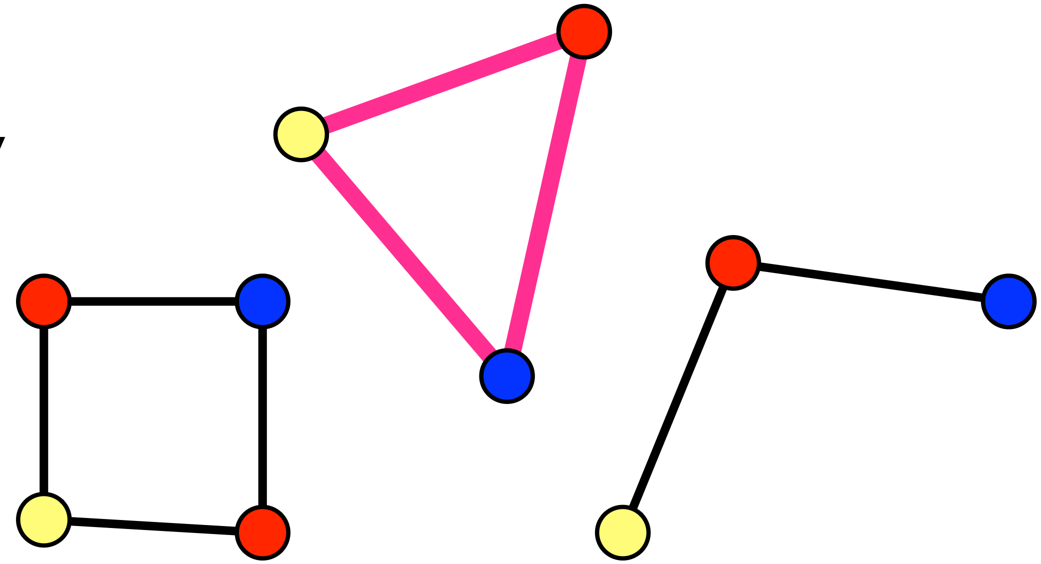
Theorem:

$$Q(f) = \Theta\Big(\min_{P \text{ eval to } f} \text{wsize}(P) \Big)$$

space: # qubits = log(# input vectors)

Application?: Triangle detection

Does G contain  ?



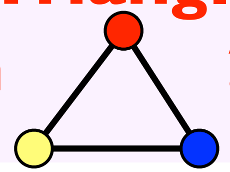
Algorithm?

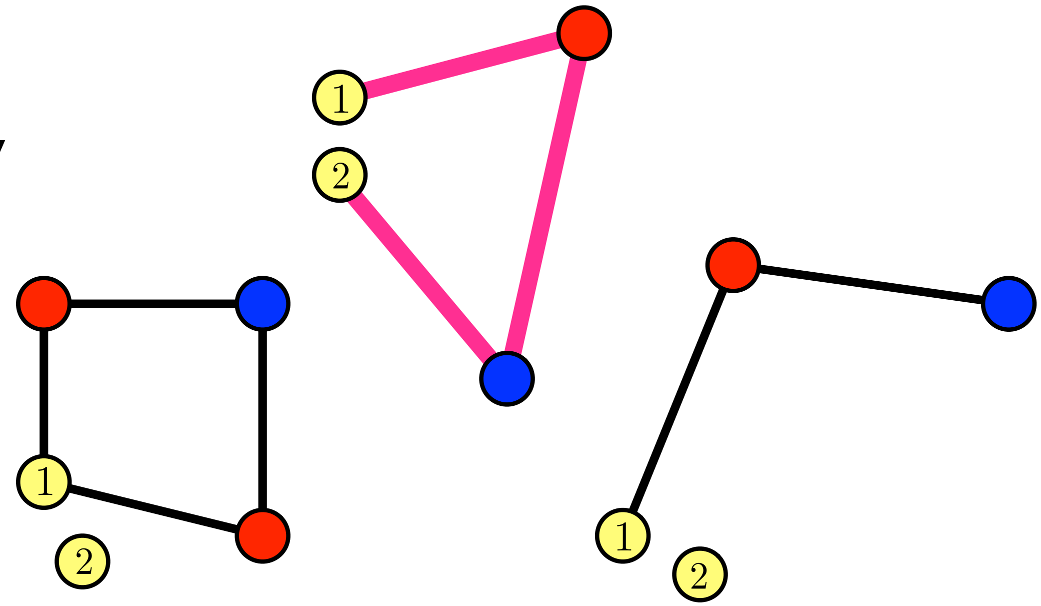
- Randomly color vertices. Split yellow vertices in two. Keep only edges
  

- Attach s to vertices 1,
  t to vertices 2

- Run s-t connectivity.

Application?: Triangle detection

Does G contain  ?



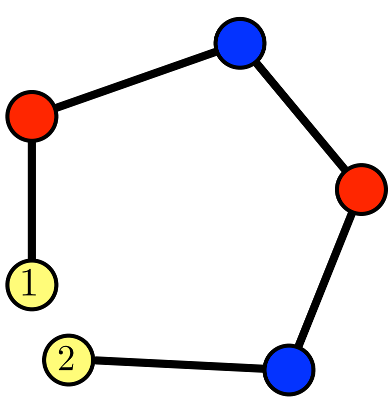
Algorithm?

- Randomly color vertices. Split yellow vertices in two. Keep only edges

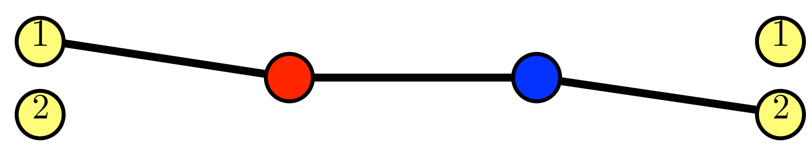
- Attach s to vertices 1,
  t to vertices 2
- Run s-t connectivity.

It doesn't work! It correctly detects triangles, but also:

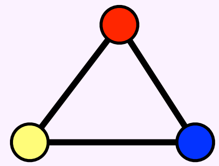
Odd cycles (triangle is a minor)



Paths (first & last vertices needn't match)



We'll fix this…



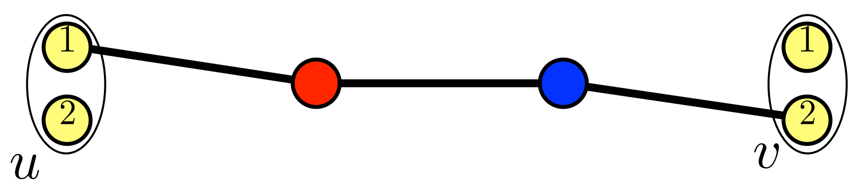
detection

Algorithm

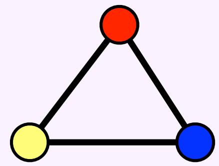
- Randomly color vertices. Split yellow vertices. Keep edges 

- Attach s to vertices 1, t to vertices 2
- Run s-t connectivity with breadcrumbs

$$|u,1\rangle - |s\rangle + |\text{crumb } u\rangle$$
$$|t\rangle - |u,2\rangle - |\text{crumb } u\rangle$$



$$|t\rangle - |s\rangle + |\text{crumb } u\rangle - |\text{crumb } v\rangle \neq |\tau\rangle$$

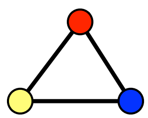


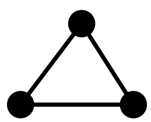
detection

Algorithm

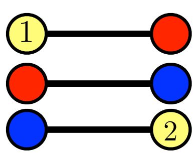
- Randomly color vertices. Split yellow vertices.
  Keep edges 
- Attach s to vertices 1, t to vertices 2
- Run s-t connectivity with breadcrumbs

$$|u,1\rangle - |s\rangle + |\text{crumb } u\rangle$$
$$|t\rangle - |u,2\rangle - |\text{crumb } u\rangle$$

Witness size G contains  : wsize = $1^2+1^2+1^2$ = O(1)

G does not contain  minor ⇒ G acyclic ⇒ G a forest:

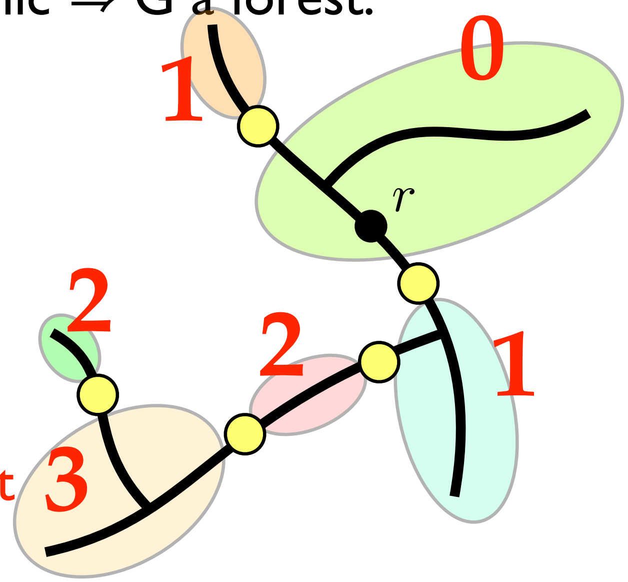
Set $\langle s|w'\rangle = 1$, $\langle t|w'\rangle = 0 \Rightarrow \langle \tau|w'\rangle = 1$ ✓

Across edges , set $\langle u|w'\rangle = \langle v|w'\rangle$

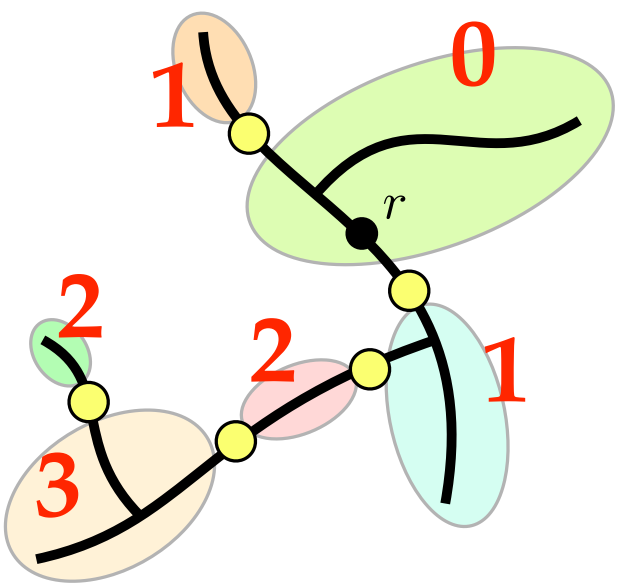
Set $\langle \text{crumb } u|w'\rangle = -\langle u,2|w'\rangle$

$\langle u,1|w'\rangle = 1 + \langle u,2|w'\rangle$

to fix all coefficients up to additive constant



Witness size (acyclic case)

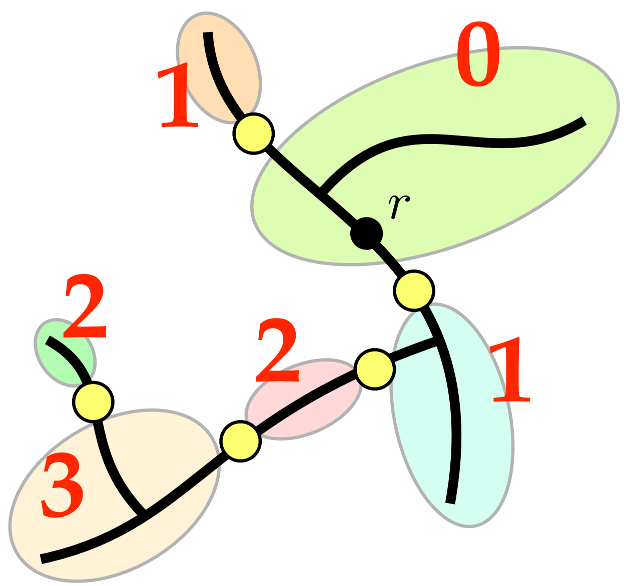


$$\text{wsize} = \min_{\text{witnesses } |w'\rangle} \sum_{\text{input vectors } |v\rangle} |\langle w'|v\rangle|^2$$

$$\leq \underbrace{n^2}_{\#\text{ input vectors}} \cdot \max_v |\langle v|w'\rangle|^2 \leq n^4$$

No good

Witness size (acyclic case)



$$\text{wsize} = \min_{\text{witnesses } |w'\rangle} \sum_{\text{input vectors } |v\rangle} |\langle w'|v\rangle|^2$$

$$\leq \sum_{\text{vertices } u} |\langle u|w'\rangle|^2 \cdot \underbrace{n}_{\substack{\text{\# input vectors}\\ \text{incident to u}}}$$

Discarding edges

$\Rightarrow$ Each edge removed with probability ⅓

$$\Rightarrow \underset{\substack{\text{random}\\ \text{coloring}}}{\mathbb{E}}[\text{wsize}] \leq \sum_{\text{vertices } u} \left(\sum_{j=1}^{\infty} j^2 (2/3)^j \right) \cdot n = O(n^2)$$

Theorem: There exists an O(n)-query quantum algorithm that distinguishes between:

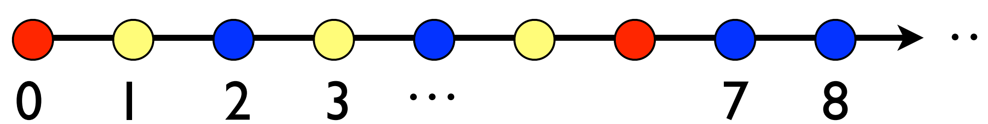
- graphs containing a triangle subgraph, and
- graphs that do not contain a triangle as a minor.

Space complexity

- Algorithms need to look up the color of a vertex
- For detecting a length-k path P_{k+1}, (k+1)-wise independence of the coloring suffices
  $\Rightarrow$ log(n) space for the hash function

- But analysis for triangle-detection algorithm requires full independence
  $\Rightarrow$ Θ(n) space of quantum RAM

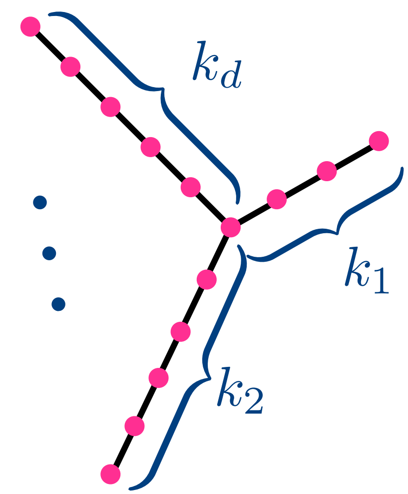
$$\underset{\substack{\text{random}\\ \text{coloring}}}{\mathbb{E}}[\mathrm{wsize}] \leq \sum_{\text{vertices } u} \underbrace{\left(\sum_{j=1}^{\infty} j^2 (2/3)^j \right)}_{\substack{\text{E[(distance from 0 before two}\\ \text{vertices are given same color)}^2\text{]}}} \cdot n = O(n^2)$$



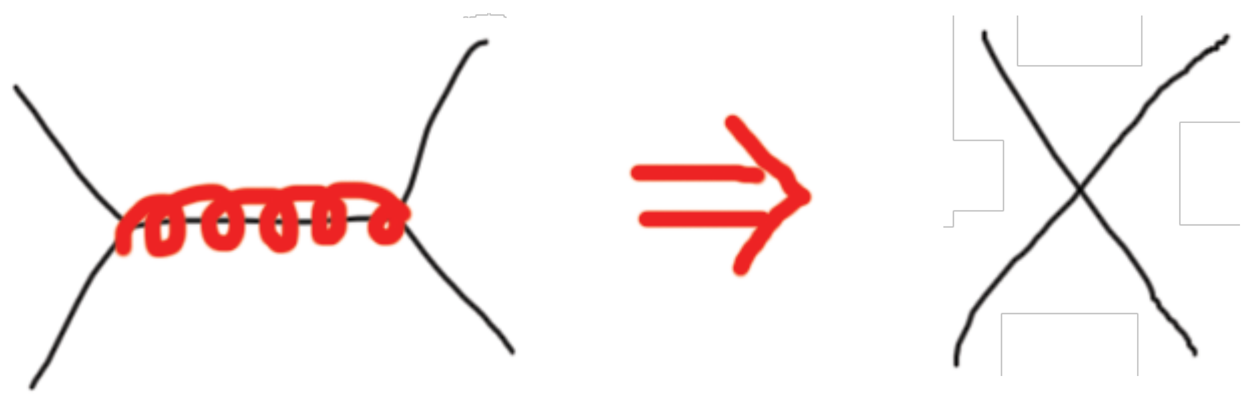
Subdivided stars

Theorem: For any fixed $k_1, \ldots, k_d$, here exists an O(n)-query quantum algorithm that distinguishes between:

- graphs containing a $\{k_1, \ldots, k_d\}$ subdivided star, and
- graphs not containing said subdivided star as a minor.



Corollary: O(n)-query quantum algorithm for detecting $\{k_1, k_2, k_3\}$ subdivided star ("claw").



Subdivided star subgraph/not-a-minor problem

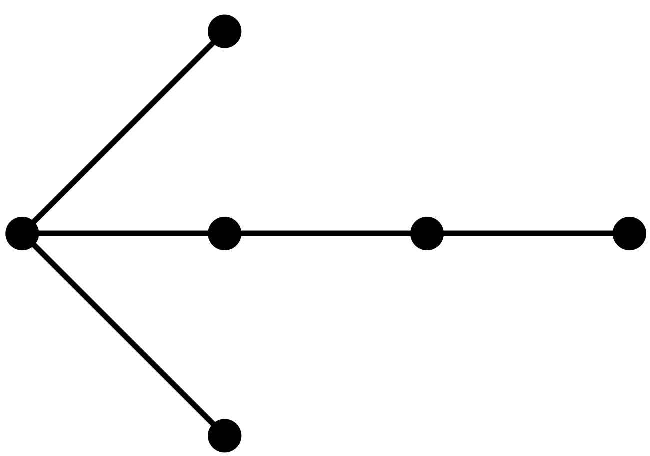
Algorithm

- Randomly color vertices of G by vertices of subdivided star. Keep only correctly colored edges.
- Evaluate the span program for s-t connectivity with breadcrumbs…

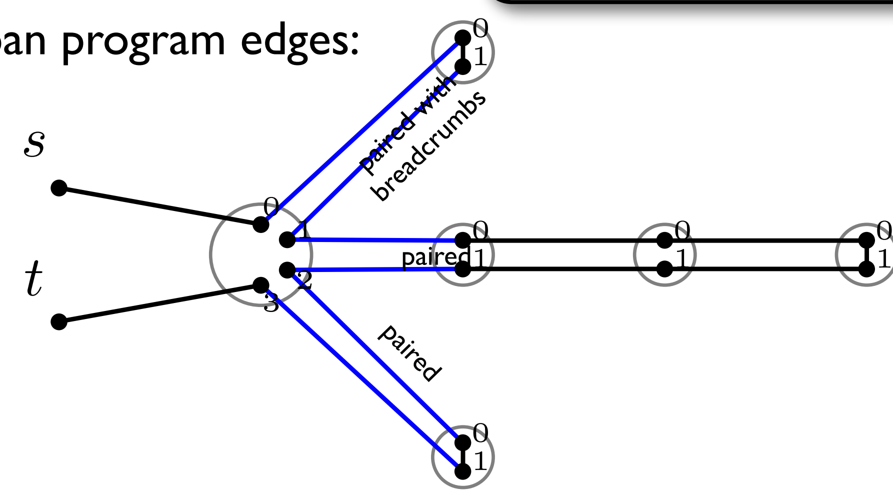
Open problems:

- Does the same algorithm work for any trees/forests?
- Characterize for what graphs breadcrumb trick works.
- Is the query complexity of the subgraph/not-a-minor problem always O(n)?

Example: G=T



span program edges:



Idea: *s*-*t* path traverses the subdivided star, out and back each leg.
Paired edges force path to use same return edge.

Time-efficient implementations

Span-program evaluation algorithm

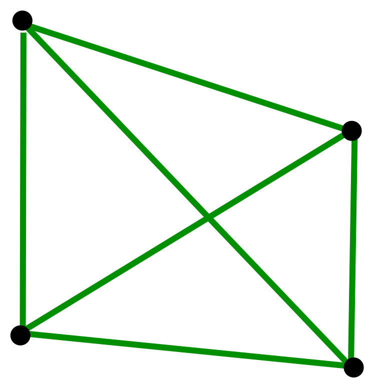
Run phase estimation on

$$\mathcal{O}_x \cdot \; \mathrm{Ref}\left(\mathrm{Ker}\left(\begin{array}{c|c|c|c|c} \text{target vector} & & & \text{input vectors} & \end{array}\right)\right)$$

st-connectivity

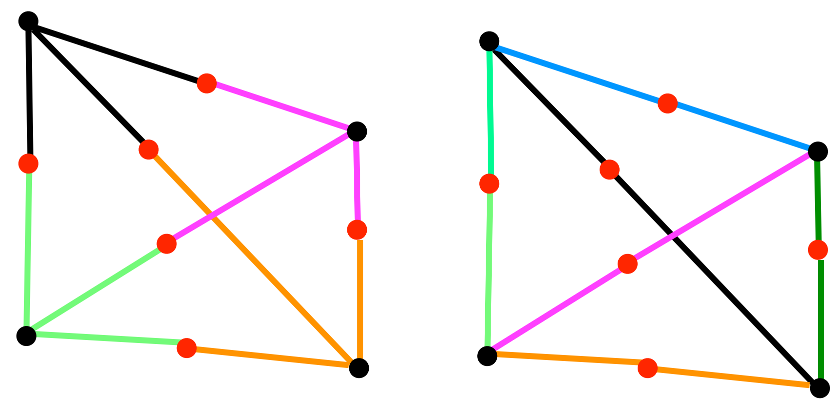
$$\mathrm{Ker}\begin{pmatrix} 1 & & 1 & \\ \text{-}1 & 1 & & \\ & \text{-}1 & \text{-}1 & \ldots \end{pmatrix}$$ ={balanced flows in K_n}

Implementing the reflection about the set of balanced flows in K_n



1. Factor the solution

   - into constraints on original vertices & on edge vertices—now commuting



2. Use phase estimation to isolate the +1 eigenspace, reflect, uncompute

Open problems

- Is $O(n\sqrt{d})$ promise st-connectivity query complexity optimal?
- Efficient implementations of learning graphs
- Combine learning graphs with breadcrumb trick
- Connection to quantum walks
- Algorithms based on the superposition over the electrical flow —are exponential speedups possible?