MULTIDIMENSIONAL QUANTUM WALKS, RECURSION, AND QUANTUM DIVIDE & CONQUER

Stacey Jeffery

QUSOFT
CWI
UNIVERSITY OF AMSTERDAM

Galina Pass

QUSOFT
UNIVERSITY OF AMSTERDAM

POWER OF QUANTUM COMPUTERS

How powerful are quantum computers?

What problems allow quantum speedups?

Polynomial? Exponential?

TOOLS FOR CLASSICAL AND QUANTUM ALGORITHMS

CLASSICAL

- Greedy algorithms
- Dynamic programming
- Linear programming
- Heuristic algorithms
- Divide and conquer

QUANTUM

- Quantum amplitude amplification
- Quantum Fourier transform
- Quantum phase estimation
- Quantum walks
- Quantum divide and conquer

TOOLS FOR CLASSICAL AND QUANTUM ALGORITHMS

CLASSICAL

- Greedy algorithms
- Dynamic programming
- Linear programming
- Heuristic algorithms
- Divide and conquer

QUANTUM

- Quantum amplitude amplification
- Quantum Fourier transform
- Quantum phase estimation
- Quantum walks
- Quantum divide and conquer

THIS WORK

TOOLS FOR CLASSICAL AND QUANTUM ALGORITHMS

CLASSICAL

- Greedy algorithms
- Dynamic programming
- Linear programming
- Heuristic algorithms
- Divide and conquer

QUANTUM

- Quantum amplitude amplification
- Quantum Fourier transform
- Quantum phase estimation
- Quantum walks
- Quantum divide and conquer

quantum speedup for a class of problems

OUTLINE

CLASSICAL MOTIVATION

ATTEMPT TO USE QUANTUM PRIMITIVES

MAIN RESULT

APPLICATION TO DSTCON

OUTLINE

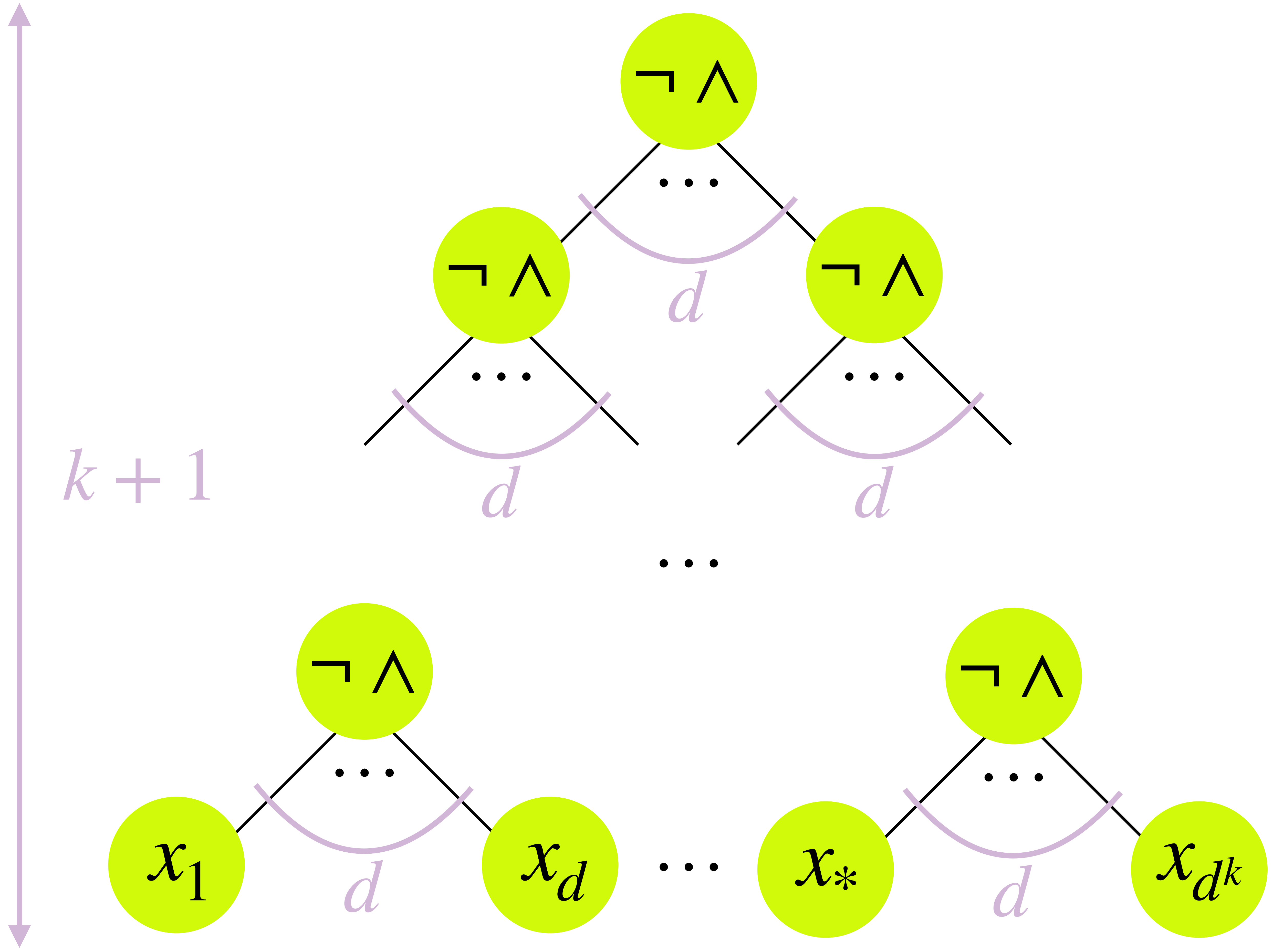
Classical motivation

ATTEMPT TO USE QUANTUM PRIMITIVES

MAIN RESULT

APPLICATION TO DSTCON

DIVIDE AND CONQUER APPROACH



NAND tree

DIVIDE AND CONQUER APPROACH

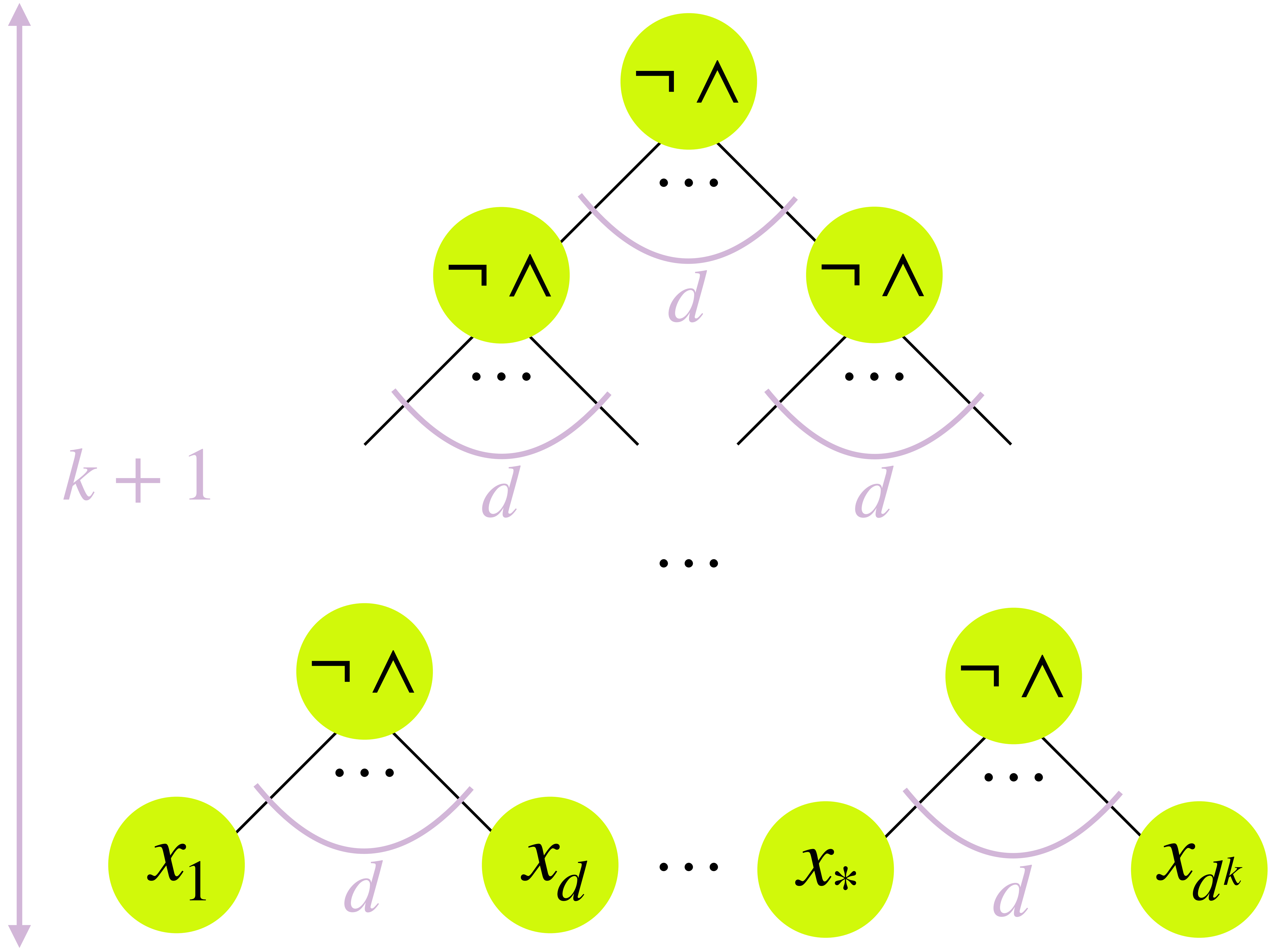
RECURSIVE RELATION

subproblems

$$T_{cl}(f_{k,d}) = dT_{cl}(f_{k-1,d}) + T_{aux}$$

subproblem complexity

combining



NAND tree

OUTLINE

CLASSICAL MOTIVATION

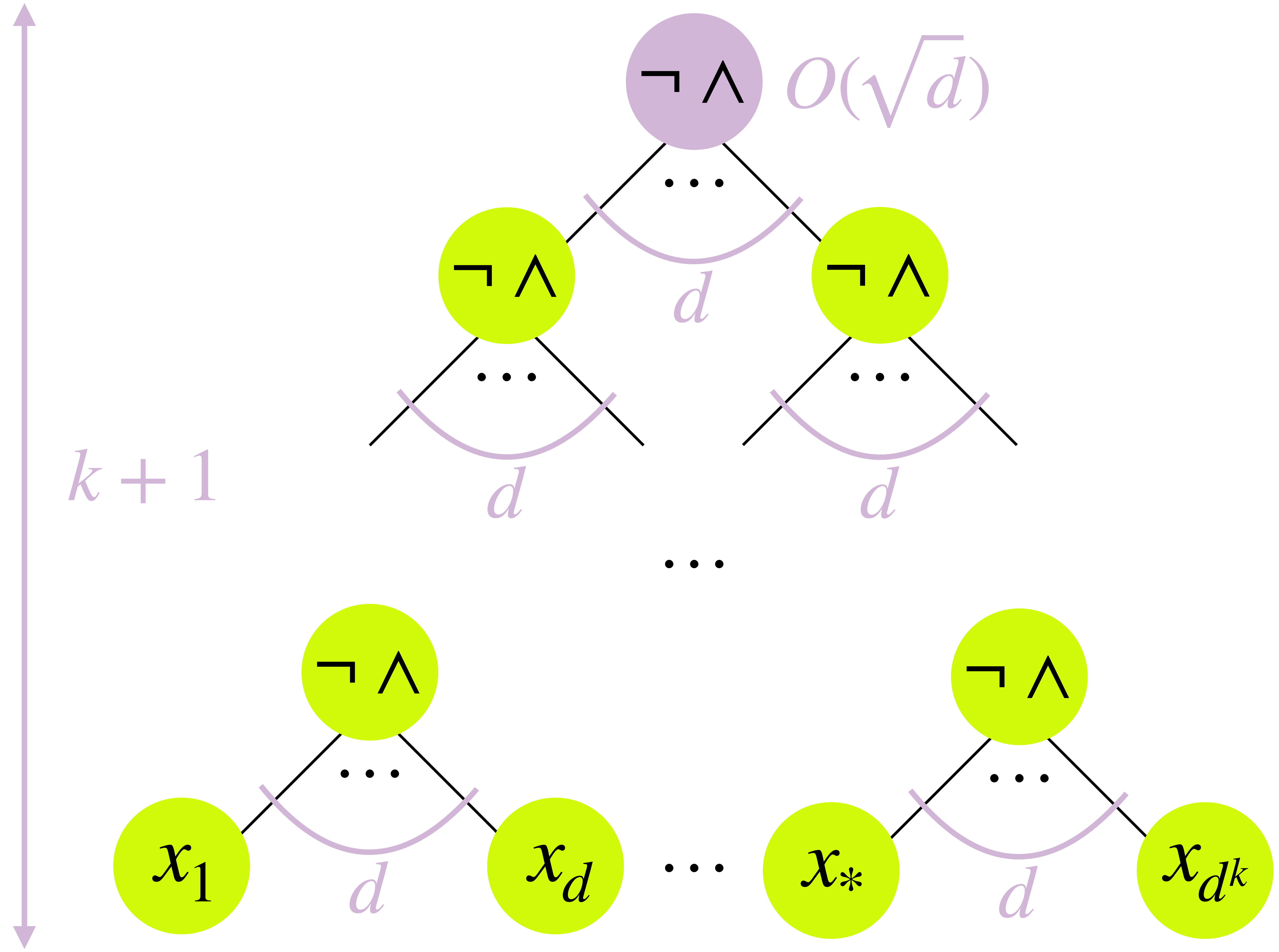
Attempt to use quantum primitives

MAIN RESULT

APPLICATION TO DSTCON

QUANTUM DIVIDE AND CONQUER ATTEMPT

- Grover's algorithm computes $\neg \wedge$ in $O(\sqrt{d})$ queries



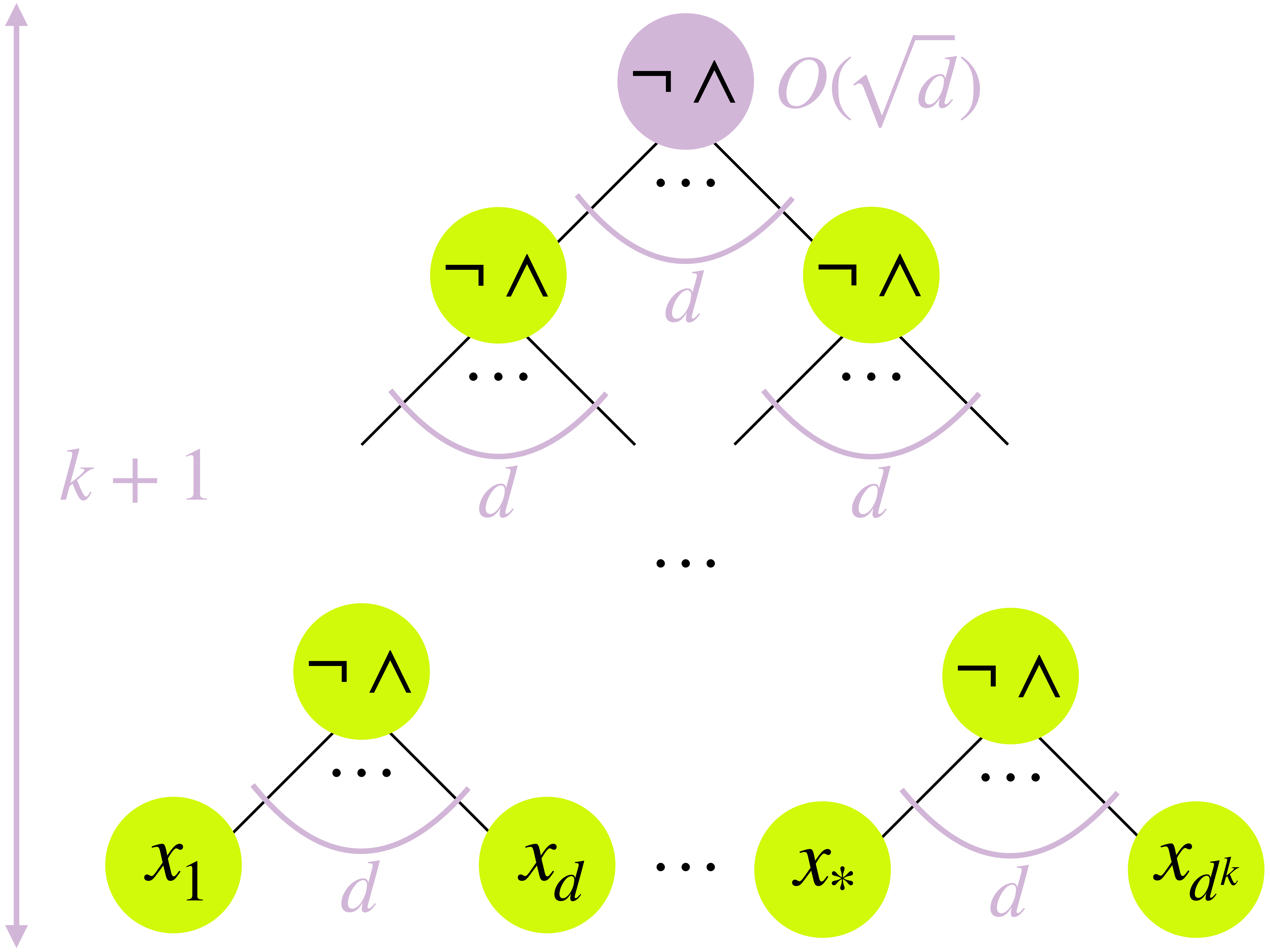
NAND tree

QUANTUM DIVIDE AND CONQUER ATTEMPT

- Grover's algorithm computes $\neg\wedge$ in $O(\sqrt{d})$ queries

if Grover's alg is perfect

- $T_q(f_{k,d}) = c\sqrt{d}T_q(f_{k-1,d})$



NAND tree

QUANTUM DIVIDE AND CONQUER ATTEMPT

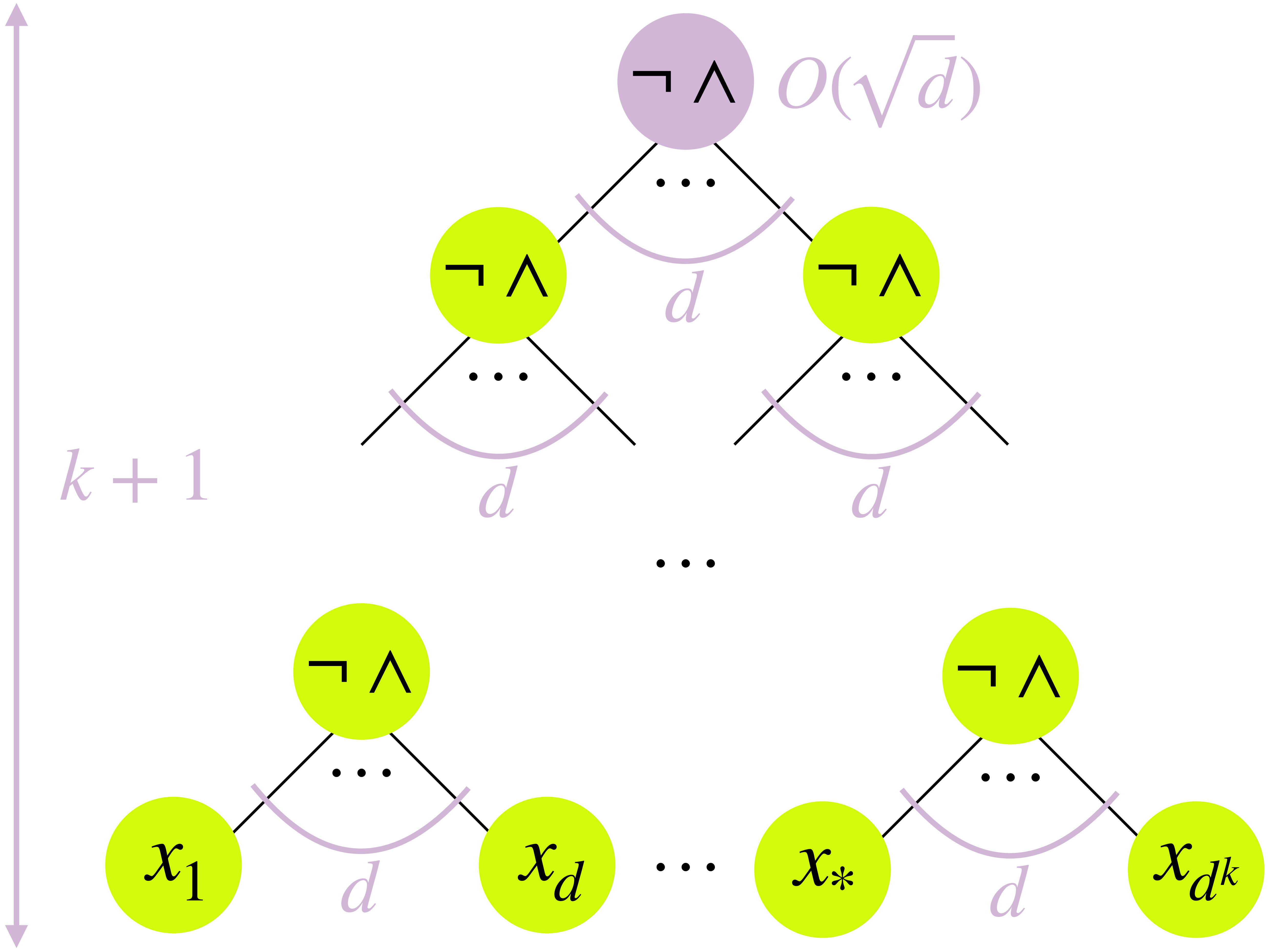
- Grover's algorithm computes $\neg\wedge$ in $O(\sqrt{d})$ queries

if Grover's alg is perfect

- $T_q(f_{k,d}) = c\sqrt{d}\,T_q(f_{k-1,d})$

$$= c^k(\sqrt{d})^k$$

kills the speedup



NAND tree

QUANTUM DIVIDE AND CONQUER ATTEMPT

- Grover's algorithm computes $\neg\wedge$ in $O(\sqrt{d})$ queries

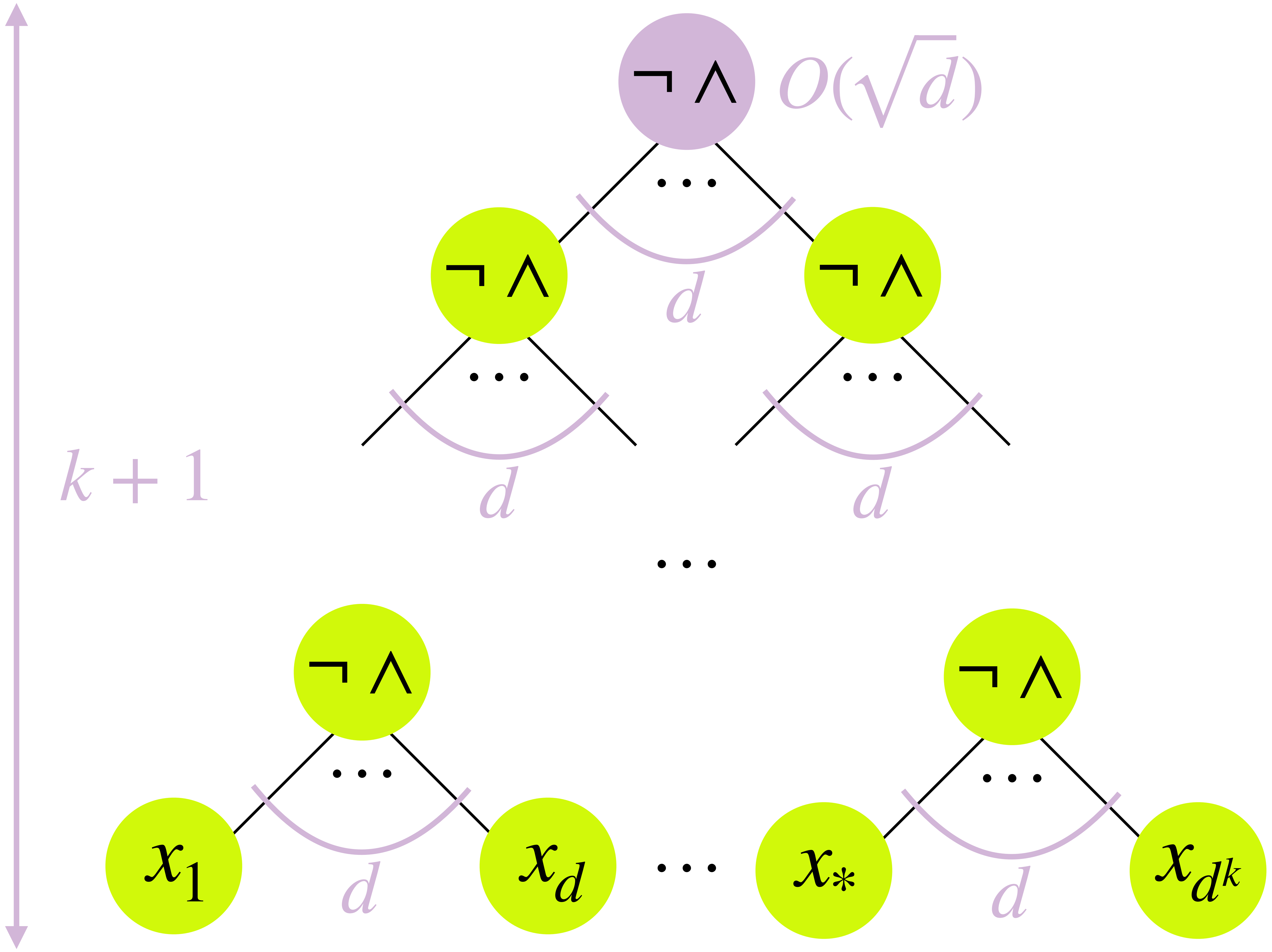
if Grover's alg is perfect

- $T_q(f_{k,d}) = c\sqrt{d}\,T_q(f_{k-1,d})$

$$= c^k(\sqrt{d})^k$$

kills the speedup

- But we know $O(\sqrt{d^k})$ [Rei11]



NAND tree

OUTLINE

CLASSICAL MOTIVATION

ATTEMPT TO USE QUANTUM PRIMITIVES

Main result

APPLICATION TO DSTCON

QUANTUM DIVIDE AND CONQUER

THEOREM. Let $f_{l,n} = \phi(\underbrace{f_{\frac{l}{b},n}, \ldots, f_{\frac{l}{b},n}}_{a}) \vee f_{aux,l,n}$

Then the quantum time complexity of $f_{l,n}$ is $\tilde{O}(T(l,n))$, where

$$T(l,n) \leq \sqrt{a}T(l/b,n) + T^{q}_{aux}$$

and space complexity $O(S^{q}_{aux}(l,n) + \log T(l,n))$.

vs classical $T(l,n) \leq aT(l/b,n) + T^{cl}_{aux}$

QUANTUM DIVIDE AND CONQUER

$f_{l,n} : D_{l,n} \to \{0,1\}$

THEOREM. Let $f_{l,n} = \phi(\underbrace{f_{\frac{l}{b},n}, \ldots, f_{\frac{l}{b},n}}_{a}) \vee f_{aux,l,n}$

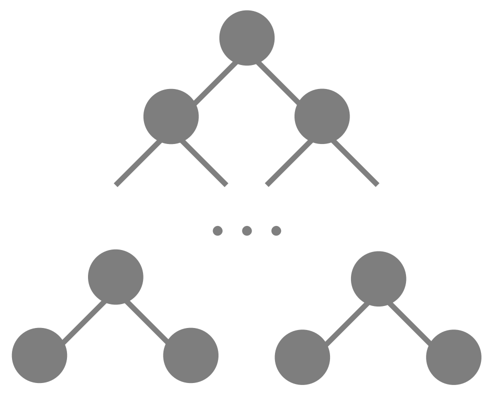
Then the quantum time complexity of $f_{l,n}$ is $\tilde{O}(T(l,n))$, where

$$T(l,n) \leq \sqrt{a}T(l/b,n) + T^q_{aux}$$

and space complexity $O(S^q_{aux}(l,n) + \log T(l,n))$.

vs classical $T(l,n) \leq aT(l/b,n) + T^{cl}_{aux}$

QUANTUM DIVIDE AND CONQUER



$f_{l,n} : D_{l,n} \to \{0,1\}$ symmetric Boolean formula

THEOREM. Let $f_{l,n} = \phi(\underbrace{f_{\frac{l}{b},n}, \ldots, f_{\frac{l}{b},n}}_{a}) \vee f_{aux,l,n}$

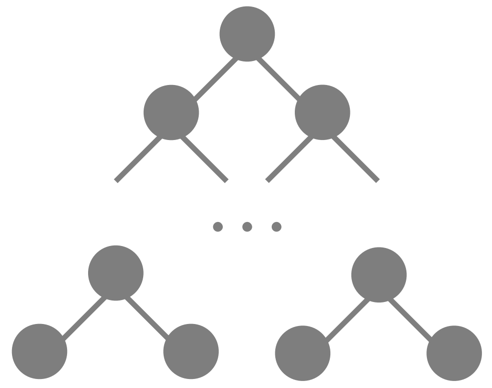
Then the quantum time complexity of $f_{l,n}$ is $\tilde{O}(T(l,n))$, where

$$\boxed{T(l,n) \leq \sqrt{a}T(l/b,n) + T^q_{aux}}$$

and space complexity $O(S^q_{aux}(l,n) + \log T(l,n))$.

vs classical $T(l,n) \leq aT(l/b,n) + T^{cl}_{aux}$

QUANTUM DIVIDE AND CONQUER



$f_{l,n} : D_{l,n} \to \{0,1\}$ symmetric Boolean formula T^q_{aux}, S^q_{aux}

THEOREM. Let $f_{l,n} = \phi(\underbrace{f_{\frac{l}{b},n}, \dots, f_{\frac{l}{b},n}}_{a}) \vee f_{aux,l,n}$

Then the quantum time complexity of $f_{l,n}$ is $\tilde{O}(T(l,n))$, where

$$T(l,n) \leq \sqrt{a}T(l/b,n) + T^q_{aux}$$

and space complexity $O(S^q_{aux}(l,n) + \log T(l,n))$.

vs classical $T(l,n) \leq aT(l/b,n) + T^{cl}_{aux}$

QUERY VS TIME COMPLEXITY

- Assume oracle access to the input $x \in \{0,1\}^m$

$$|i\rangle|0\rangle \rightarrow \boxed{O_x} \rightarrow |i\rangle|x_i\rangle$$

QUERY VS TIME COMPLEXITY

can encode e.g. a function or an adjacency matrix

- Assume oracle access to the input $x \in \{0,1\}^m$

$$|i\rangle|0\rangle \rightarrow \boxed{O_x} \rightarrow |i\rangle|x_i\rangle$$

QUERY VS TIME COMPLEXITY

can encode e.g. a function or an adjacency matrix

- Assume oracle access to the input $x \in \{0,1\}^m$

$$|i\rangle|0\rangle \rightarrow \boxed{O_x} \rightarrow |i\rangle|x_i\rangle$$

- Often the most expensive operation of the algorithm

QUERY VS TIME COMPLEXITY

can encode e.g. a function or an adjacency matrix

- Assume oracle access to the input $x \in \{0,1\}^m$

$$|i\rangle|0\rangle \rightarrow \boxed{O_x} \rightarrow |i\rangle|x_i\rangle$$

- Often the most expensive operation of the algorithm

- QUERY COMPLEXITY: only count oracle calls

QUERY VS TIME COMPLEXITY

can encode e.g. a function or an adjacency matrix

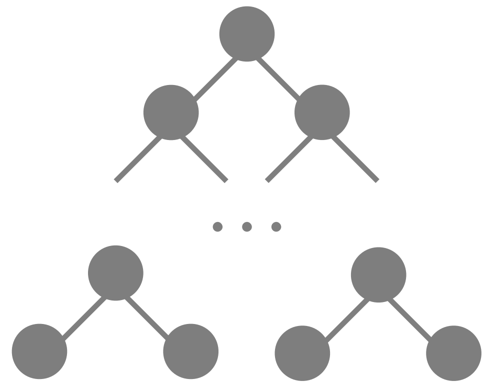
- Assume oracle access to the input $x \in \{0,1\}^m$

$$|i\rangle|0\rangle \rightarrow \boxed{O_x} \rightarrow |i\rangle|x_i\rangle$$

- Often the most expensive operation of the algorithm
- QUERY COMPLEXITY: only count oracle calls
- TIME COMPLEXITY: count all the operations

STRONGER

QUANTUM DIVIDE AND CONQUER



$f_{l,n} : D_{l,n} \to \{0,1\}$ symmetric Boolean formula T^q_{aux}, S^q_{aux}

THEOREM. Let $f_{l,n} = \phi(\underbrace{f_{\frac{l}{b},n}, \ldots, f_{\frac{l}{b},n}}_{a}) \vee f_{aux,l,n}$

Then the quantum time complexity of $f_{l,n}$ is $\tilde{O}(T(l,n))$, where

$$T(l,n) \leq \sqrt{a}T(l/b,n) + T^q_{aux}$$

and space complexity $O(S^q_{aux}(l,n) + \log T(l,n))$.

vs classical $T(l,n) \leq aT(l/b,n) + T^{cl}_{aux}$

CONTEXT

QUERY COMPLEXITY

[CKKD+22]

- Arbitrary Boolean formulas & more general functions
- Only query complexity
- Not constructive

TIME COMPLEXITY

[ABB+23]

- AND or OR
- MIN or MAX

OUTLINE

CLASSICAL MOTIVATION

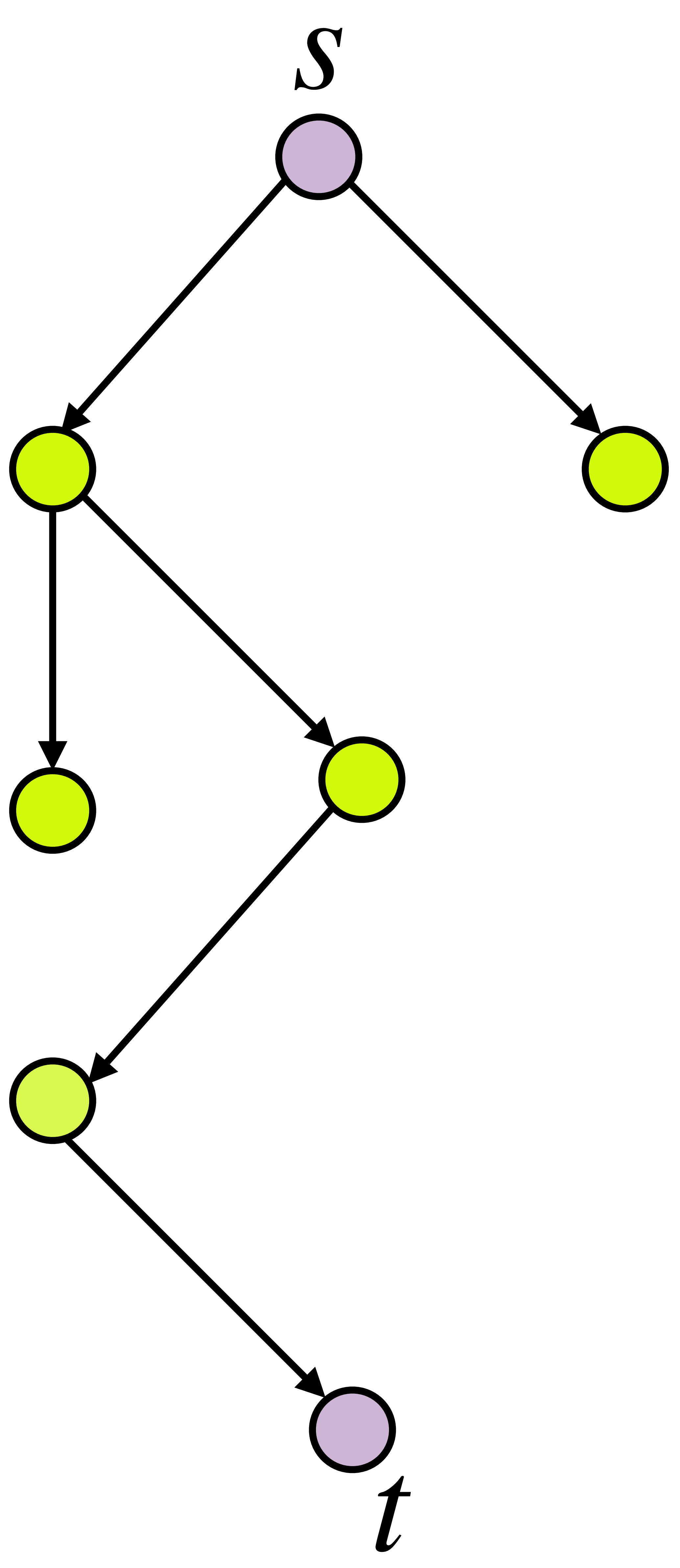
ATTEMPT TO USE QUANTUM PRIMITIVES

MAIN RESULT

Application to DSTCON

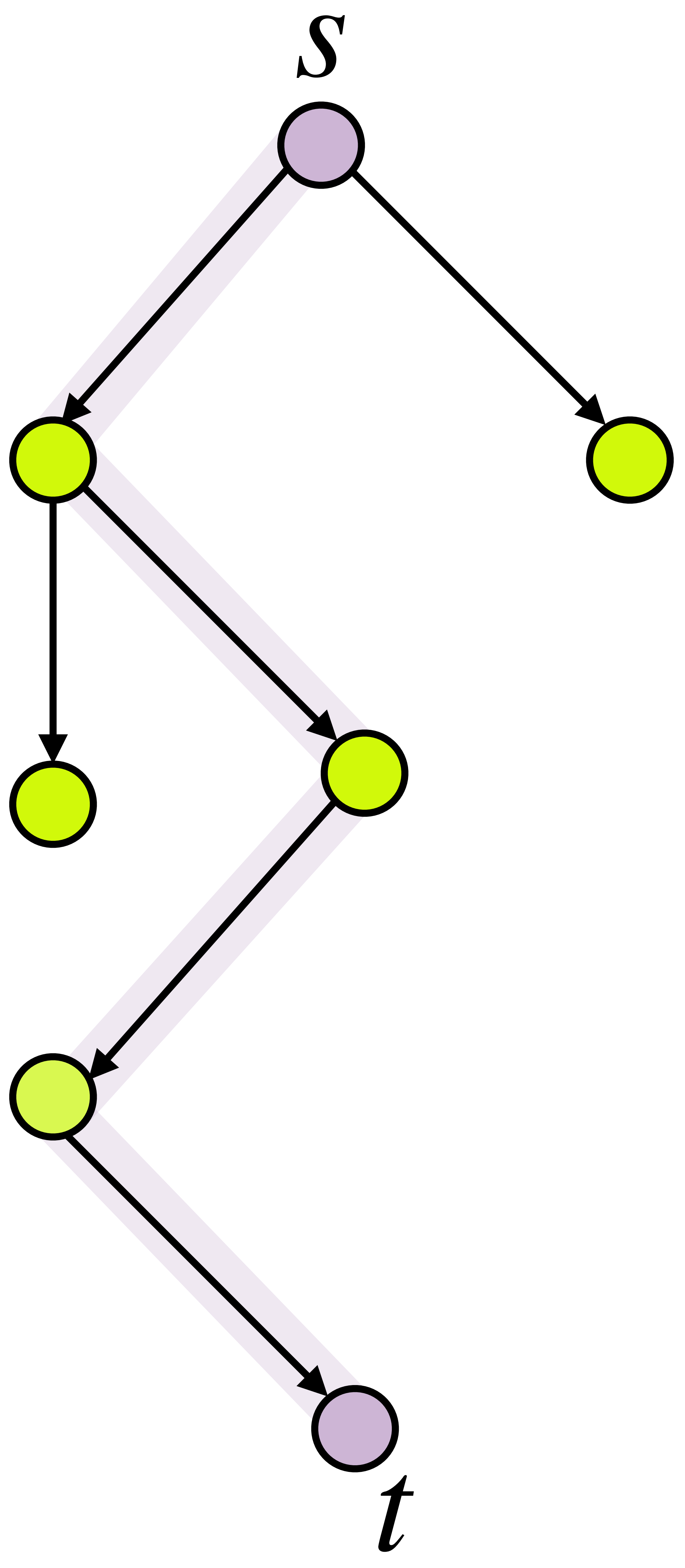
DIRECTED ST-CONNECTIVITY

- Directed graph $G = (V, E)$
- Special vertices $s, t \in V$
- Is there a path from s to t?



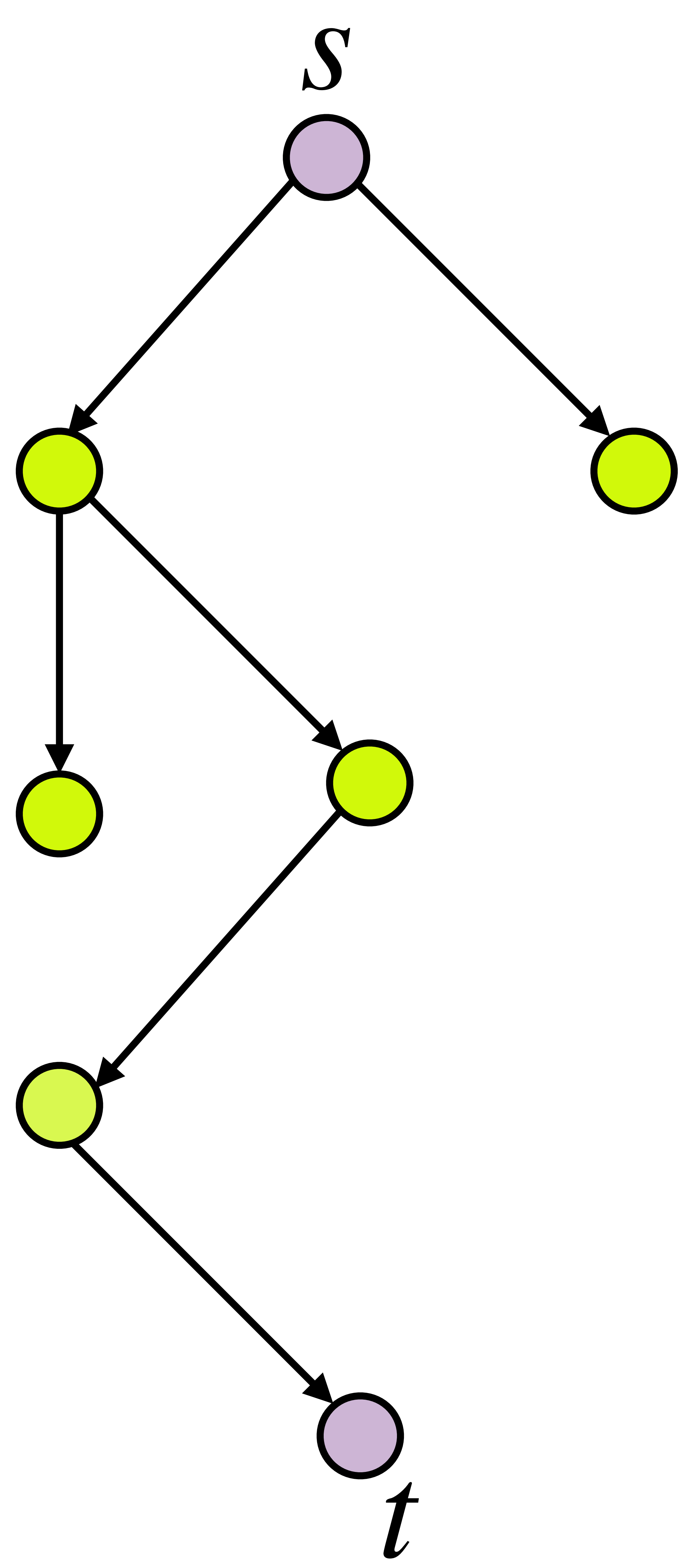
DIRECTED ST-CONNECTIVITY

- Directed graph $G = (V, E)$
- Special vertices $s, t \in V$
- Is there a path from s to t?



DIRECTED ST-CONNECTIVITY

- Directed graph $G = (V, E)$
- Special vertices $s, t \in V$
- Is there a path from s to t?



SAVITCH'S ALGORITHM [Sav70]

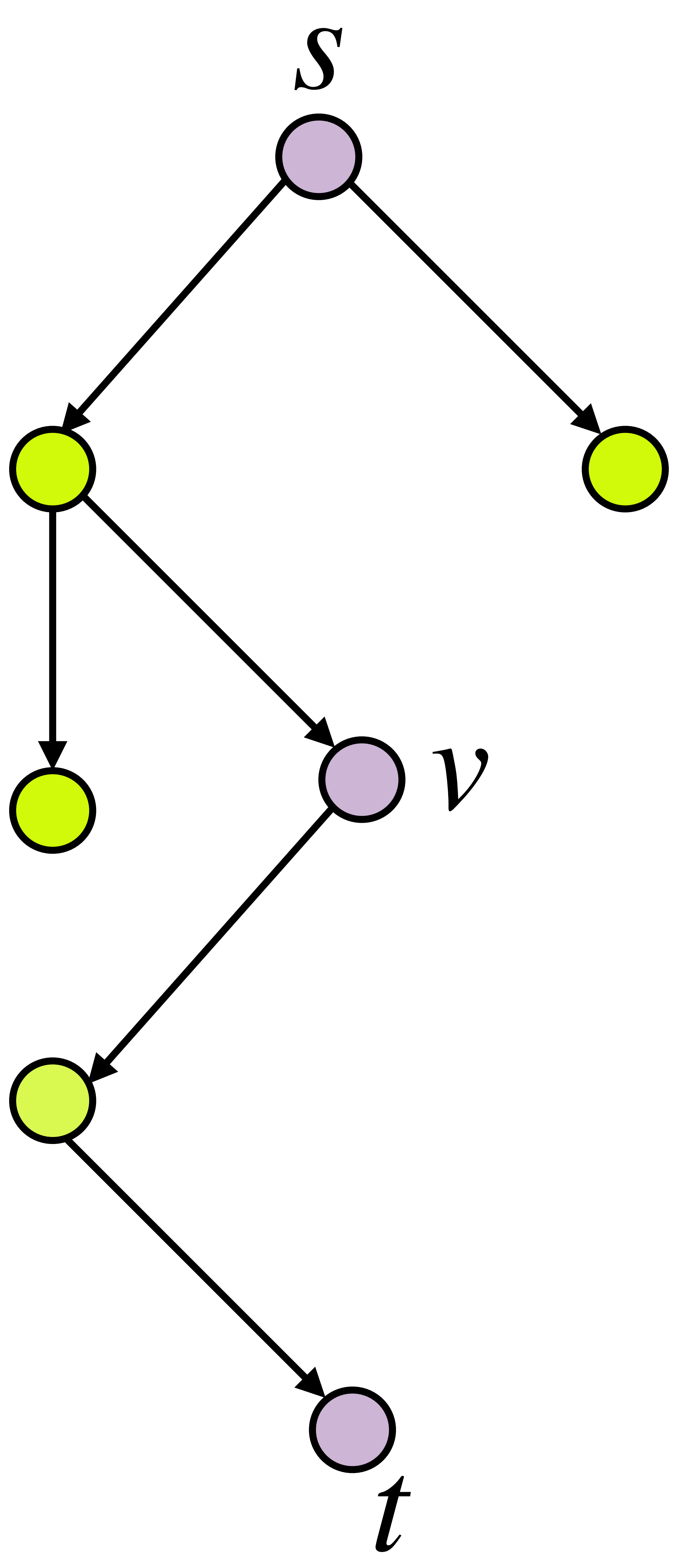
For $v \in V$:

Check if there is a path $s \to v$ of length $n/2$

Check if there is a path $v \to t$ of length $n/2$

DIRECTED ST-CONNECTIVITY

- Directed graph $G = (V, E)$
- Special vertices $s, t \in V$
- Is there a path from s to t?



SAVITCH'S ALGORITHM [Sav70]

For $v \in V$:

Check if there is a path $s \to v$ of length $n/2$

Check if there is a path $v \to t$ of length $n/2$

DIRECTED ST-CONNECTIVITY

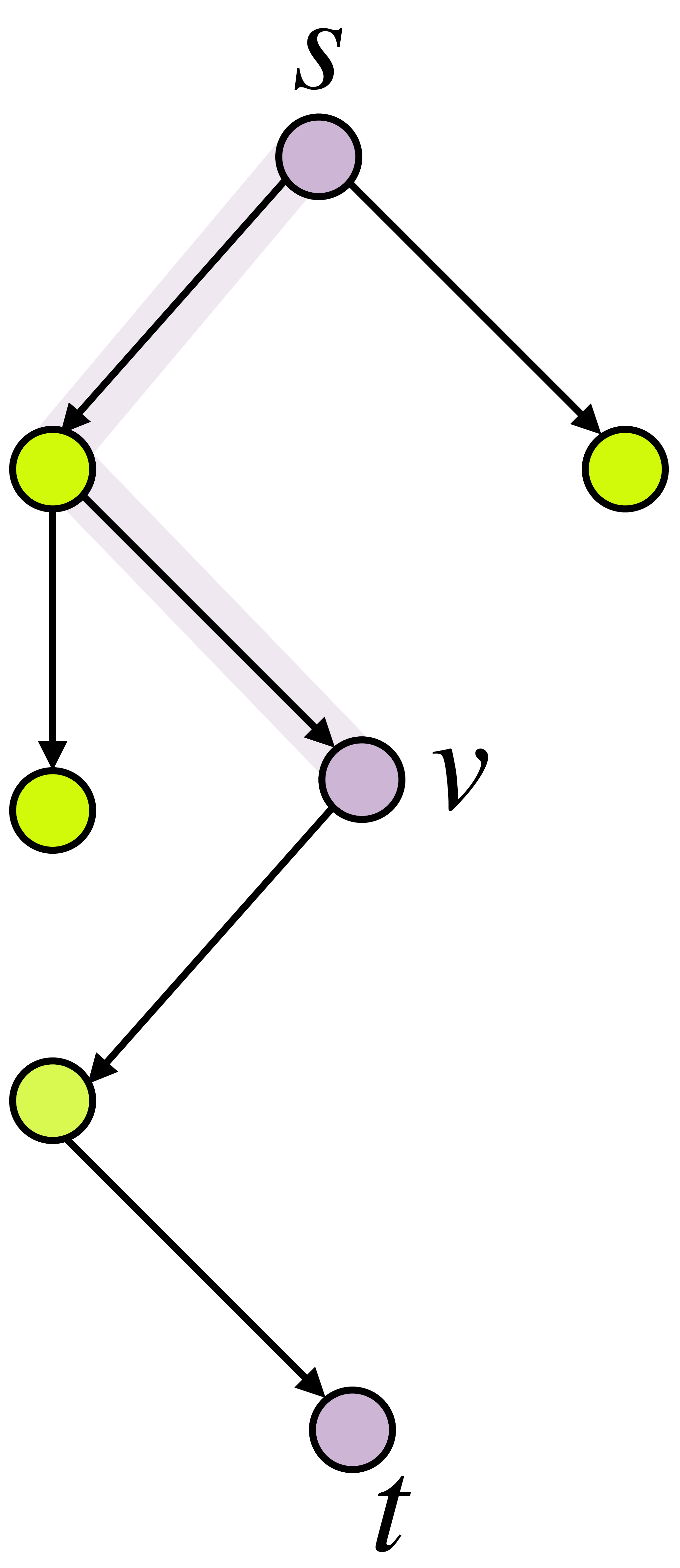
- Directed graph $G = (V, E)$
- Special vertices $s, t \in V$
- Is there a path from s to t?

SAVITCH'S ALGORITHM [Sav70]

For $v \in V$:

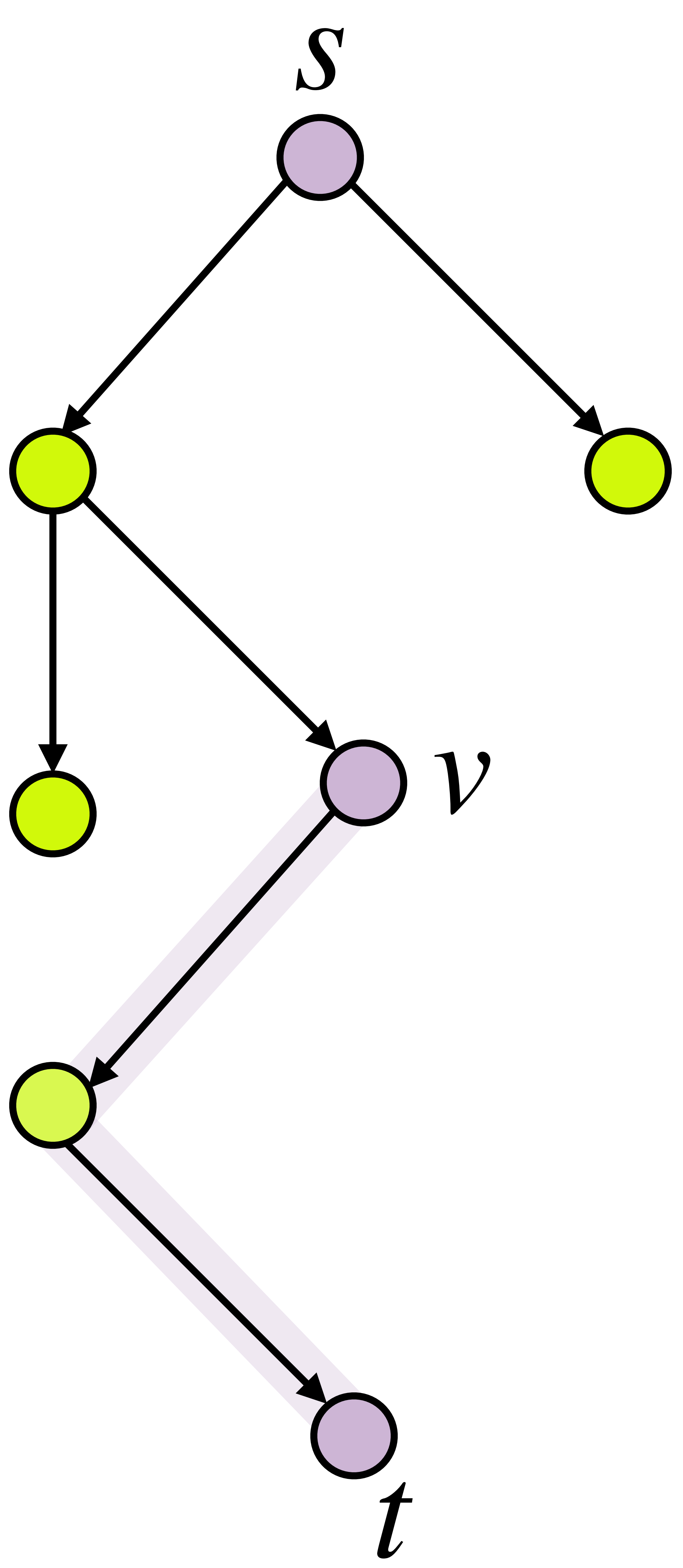
  Check if there is a path $s \to v$ of length $n/2$

  Check if there is a path $v \to t$ of length $n/2$



DIRECTED ST-CONNECTIVITY

- Directed graph $G = (V, E)$
- Special vertices $s, t \in V$
- Is there a path from s to t?



SAVITCH'S ALGORITHM [Sav70]

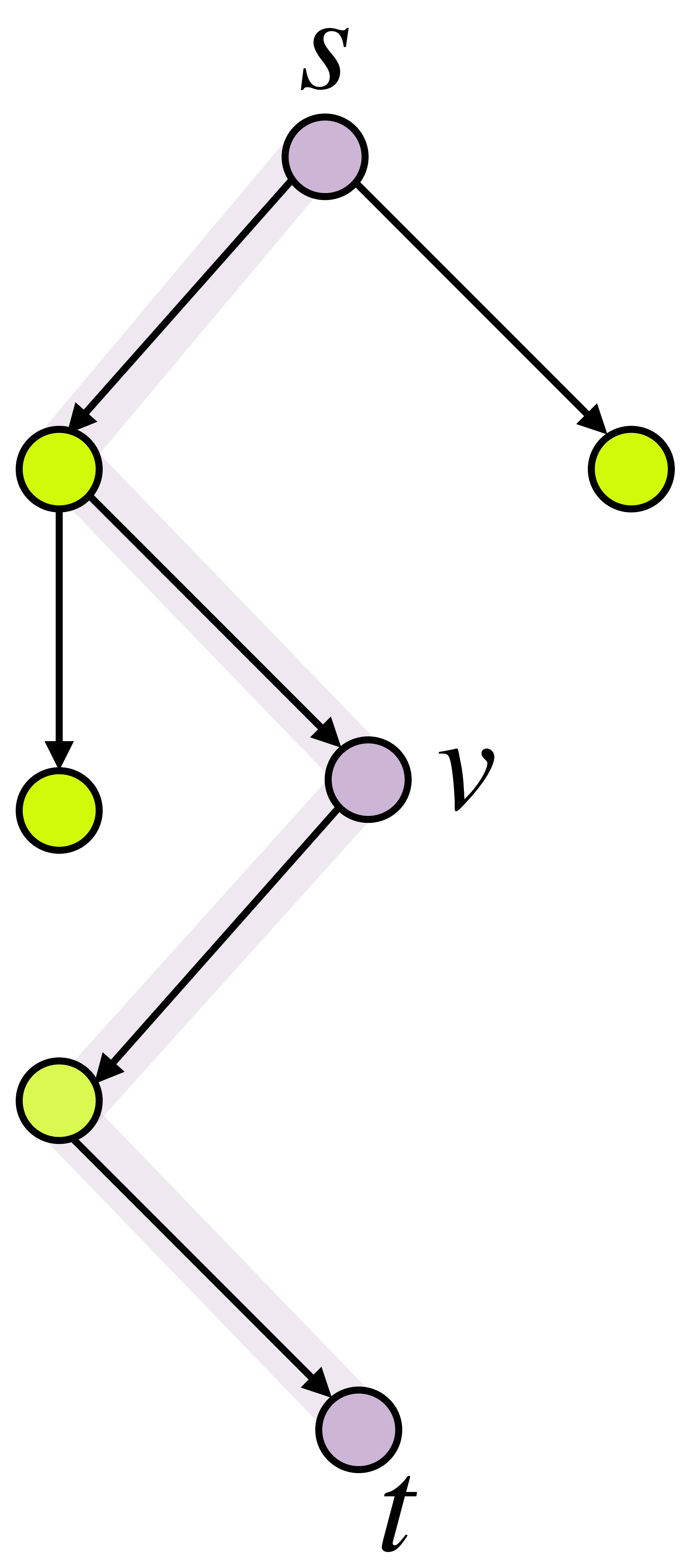
For $v \in V$:

  Check if there is a path $s \to v$ of length $n/2$

  Check if there is a path $v \to t$ of length $n/2$

DIRECTED ST-CONNECTIVITY

- Directed graph $G = (V, E)$
- Special vertices $s, t \in V$
- Is there a path from s to t?



SAVITCH'S ALGORITHM [Sav70]

For $v \in V$:

Check if there is a path $s \rightarrow v$ of length $n/2$

Check if there is a path $v \rightarrow t$ of length $n/2$

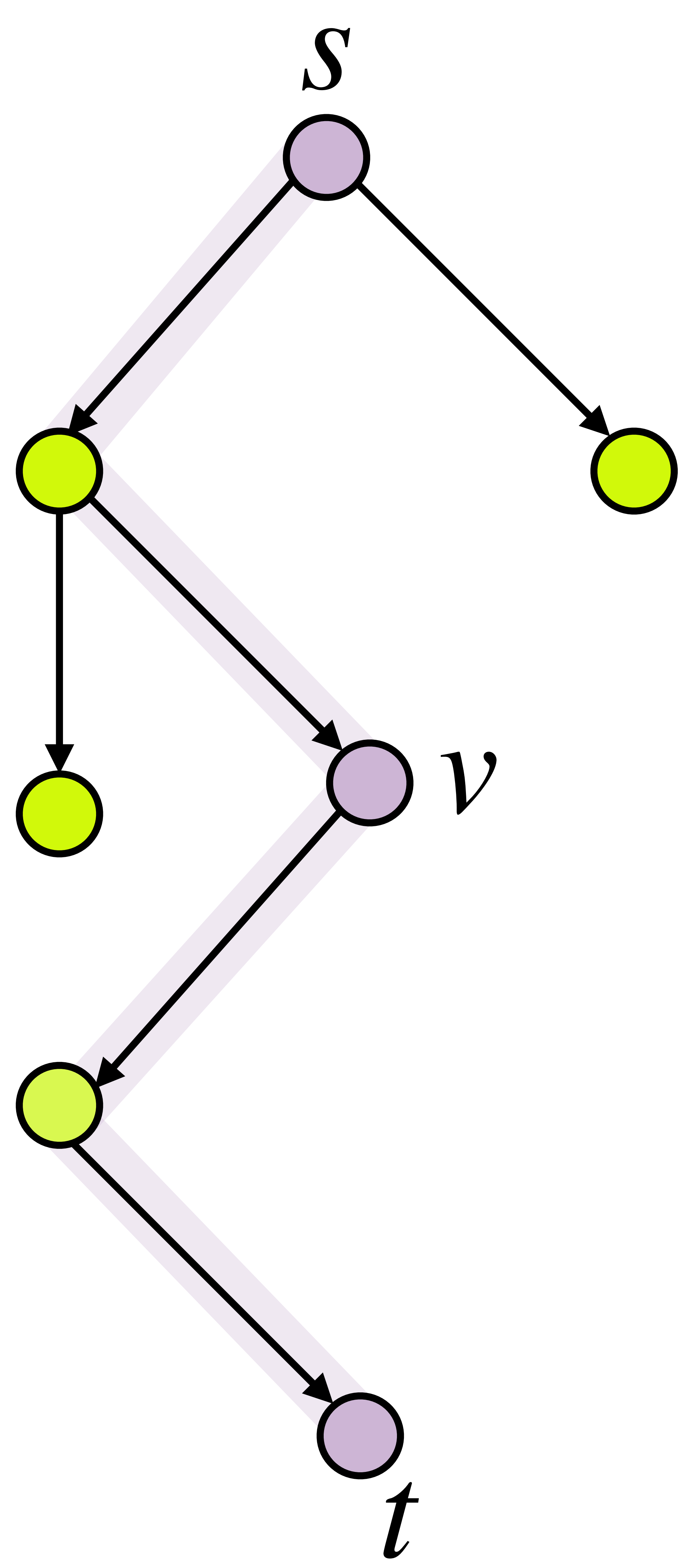
DIRECTED ST-CONNECTIVITY

SAVITCH'S ALGORITHM [Sav70]

For $v \in V$:

Check if there is a path $s \to v$ of length $n/2$

Check if there is a path $v \to t$ of length $n/2$



DIRECTED ST-CONNECTIVITY

SAVITCH'S ALGORITHM [Sav70]

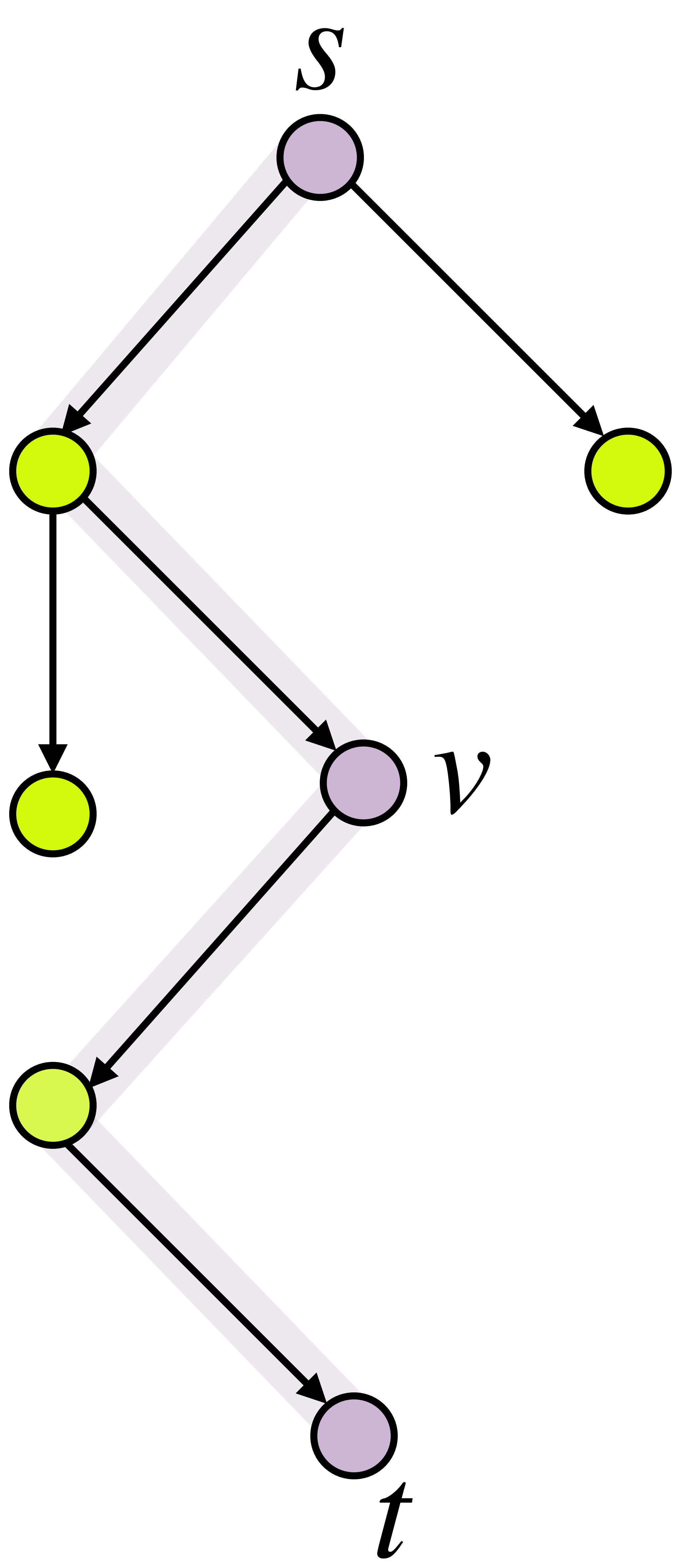
For $v \in V$:

  Check if there is a path $s \to v$ of length $n/2$

  Check if there is a path $v \to t$ of length $n/2$

TIME	$O\left((2n)^{\log n}\right)$
SPACE	$O\left(\log^2 n\right)$

$|V| = n$



DIRECTED ST-CONNECTIVITY

SAVITCH'S ALGORITHM [Sav70]

For $v \in V$:

Check if there is a path $s \to v$ of length $n/2$

Check if there is a path $v \to t$ of length $n/2$

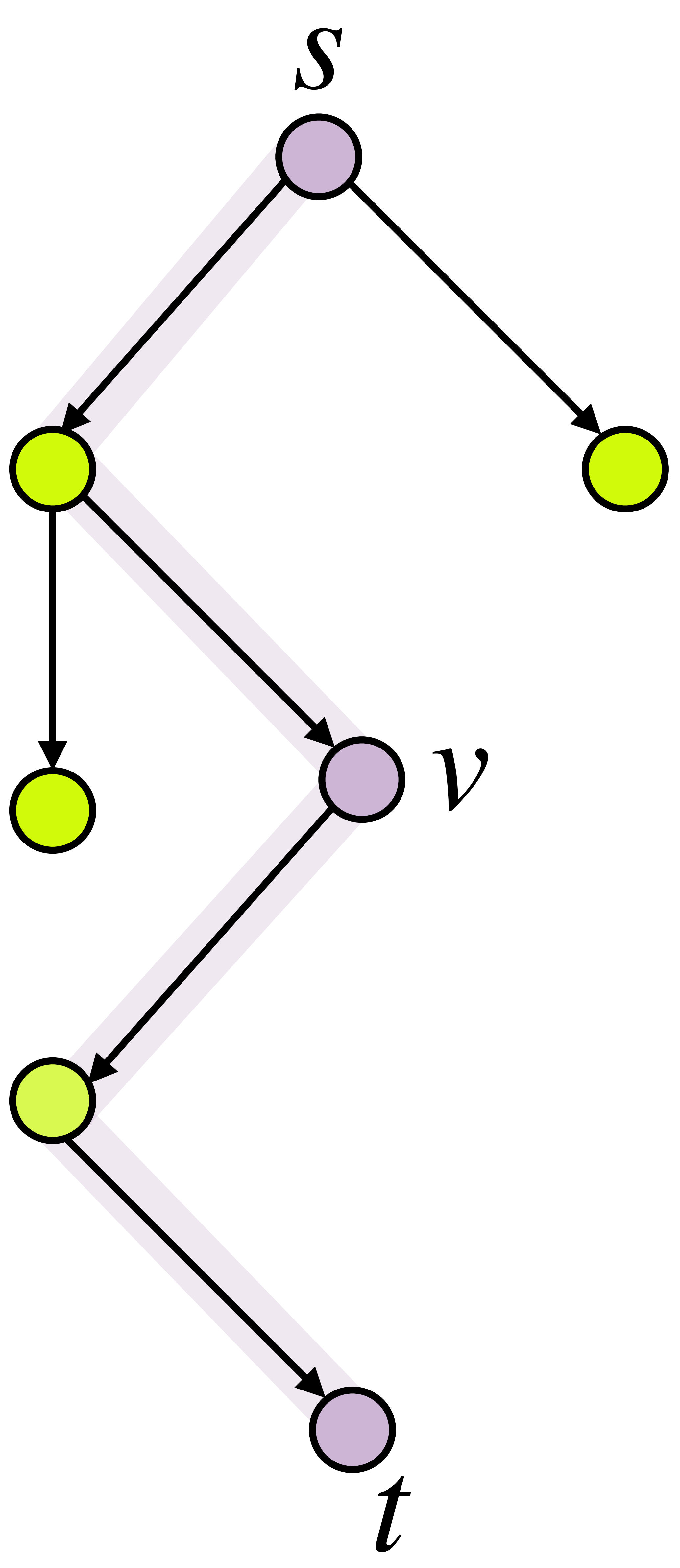
TIME $O\left((2n)^{\log n}\right)$

SPACE $O\left(\log^2 n\right)$

$|V| = n$

Can write down as a formula

$$Path(s,t,n) = \vee_{v \in V} \left(Path(s,v,n/2) \wedge Path(v,t,n/2)\right)$$



DIRECTED ST-CONNECTIVITY

$$Path(s, t, n) = \vee_{v \in V} \left(Path(s, v, n/2) \wedge Path(v, t, n/2)\right)$$

$|V| = n$

DIRECTED ST-CONNECTIVITY

$$Path(s,t,n) = \vee_{v \in V} \left(Path(s,v,n/2) \wedge Path(v,t,n/2)\right)$$

$$T(\underbrace{n}_{\text{path length}}, \underbrace{n}_{\text{graph size}}) = \sqrt{2n}\,T(n/2,n) + O(1)$$

$$|V| = n$$

$$f_{l,n} = \phi(\overbrace{f_{\frac{l}{b},n}, \ldots, f_{\frac{l}{b},n}}^{a}) \vee f_{aux,l,n}$$

$$T(l,n) \leq \sqrt{a}\,T(l/b,n) + T^{q}_{aux}$$

$$O(S^{Q}_{aux}(l,n) + \log T(l,n))$$

DIRECTED ST-CONNECTIVITY

$$Path(s,t,n) = \vee_{v \in V}\left(Path(s,v,n/2) \wedge Path(v,t,n/2)\right)$$

$$T(\underbrace{n}_{\text{path length}},\underbrace{n}_{\text{graph size}}) = \sqrt{2n}\,T(n/2,n) + O(1)$$

$$|V| = n$$

$$f_{l,n} = \phi(\overbrace{f_{\frac{l}{b},n},\ldots,f_{\frac{l}{b},n}}^{a}) \vee f_{aux,l,n}$$

$$T(l,n) \le \sqrt{a}\,T(l/b,n) + T^{q}_{aux}$$

$$O(S^{Q}_{aux}(l,n) + \log T(l,n))$$

	CLASSICAL	QUANTUM
TIME	$O\left((2n)^{\log n}\right)$	$\boxed{O\left(\sqrt{2n}^{\log n}\right)}$
SPACE	$O\left(\log^2 n\right)$	$O\left(\log^2 n\right)$

SUMMARY

- Classical divide and conquer + quantum primitives doesn't work

SUMMARY

- Classical divide and conquer + quantum primitives doesn't work
- We develop a time-efficient quantum divide and conquer framework

SUMMARY

- Classical divide and conquer + quantum primitives doesn’t work
- We develop a time-efficient quantum divide and conquer framework
- Quadratic speedup for DSTCON in the low-space regime

THANK YOU

for your attention